Early Release

RAW & UNEDITED

Detecting, Troubleshooting, and Preventing Congestion in Storage Networks

PARESH GUPTA, CCIE® NO. 36645
EDWARD MAZUREK, CCIE® NO. 6448

ciscopress.com

Detecting, Troubleshooting, and Preventing Congestion in Storage Networks

Paresh Gupta, Edward Mazurek

Contents

Table of Contents

Chapter 1. Introduction to Congestion in Storage Networks

This chapter provides a high-level overview of the types of storage, storage protocols, their transports, and networks in a data center. For those who have working experience, this chapter brings more clarity. For others, this chapter does not make you an expert but explains just enough to pave the way for the following chapters. This book assumes that you already know about storage networks and have some experience with them, up to the extent that you have witnessed at least one congestion event or are trying to be proactive in preventing them.

This chapter also covers some frequently asked questions and our responses to them.

Further, this chapter provides a high-level overview of the causes and sources of congestion in storage networks. These details are expanded further in the following chapters based on the network type.

This book uses “congestion management” as an overall term to refer to congestion detection, troubleshooting, and prevention. Essentially, a less wordy way of referring to everything that is needed to reduce or eliminate congestion in the networks.

Types of Storage in a Data Center

Storage in a data center can be classified based on its location (Local and Remote) and based on its access level (Block, File, and Object). Figure 1-1 shows a high-level overview of these storage types.

Storage Type — By Location

This section explains two types of storage based on their location—Local Storage and Remote Storage.

Local Storage

Storage within the physical enclosure of a device is called local storage. This is the type of storage that most people are familiar with because laptops, tablets, and even smartphones have an ample amount of local storage. A related term is direct-attached storage (DAS), but it is different from local storage because even an external storage device may be directly attached.

Local storage is accessed at the block level using block storage protocols such as SCSI or NVMe, explained shortly.

Remote Storage

Remote storage is outside the physical enclosure of a device, and it comes in different flavors. Many people are familiar with storage in the public cloud because of its easy accessibility, such as Amazon S3, Dropbox, Google Drive, and Microsoft OneDrive. But not so many people acquire working experience with on-premises remote storage devices until they get in a job managing these environments.

Remote storage can be in the form of large monolithic devices, such as Network Attached Storage (NAS) or Storage Area Network (SAN) storage arrays. An alternative approach is to use commodity servers with local storage and then an overlay Software-Defined Storage (SDS) solution or Distributed File Systems (DFS) to simplify the access to the local storage of the other servers.

As Figure 1-1 shows, accessing remote storage needs the following:

1. **Network**: A network for connecting the devices, such as Fibre Channel, Ethernet, or InfiniBand.
2. **Transport**: Transport protocols for end-to-end data delivery, such as Fibre Channel Protocol (FCP), Transmission Control Protocol (TCP), and Remote Direct Memory Access (RDMA) capable protocols.
3. **Storage Protocol**: Another layer above the transport protocol that understands the storage traffic, such as iSCSI (SCSI), NVMe/TCP (NVMe), NFS, and SMB.

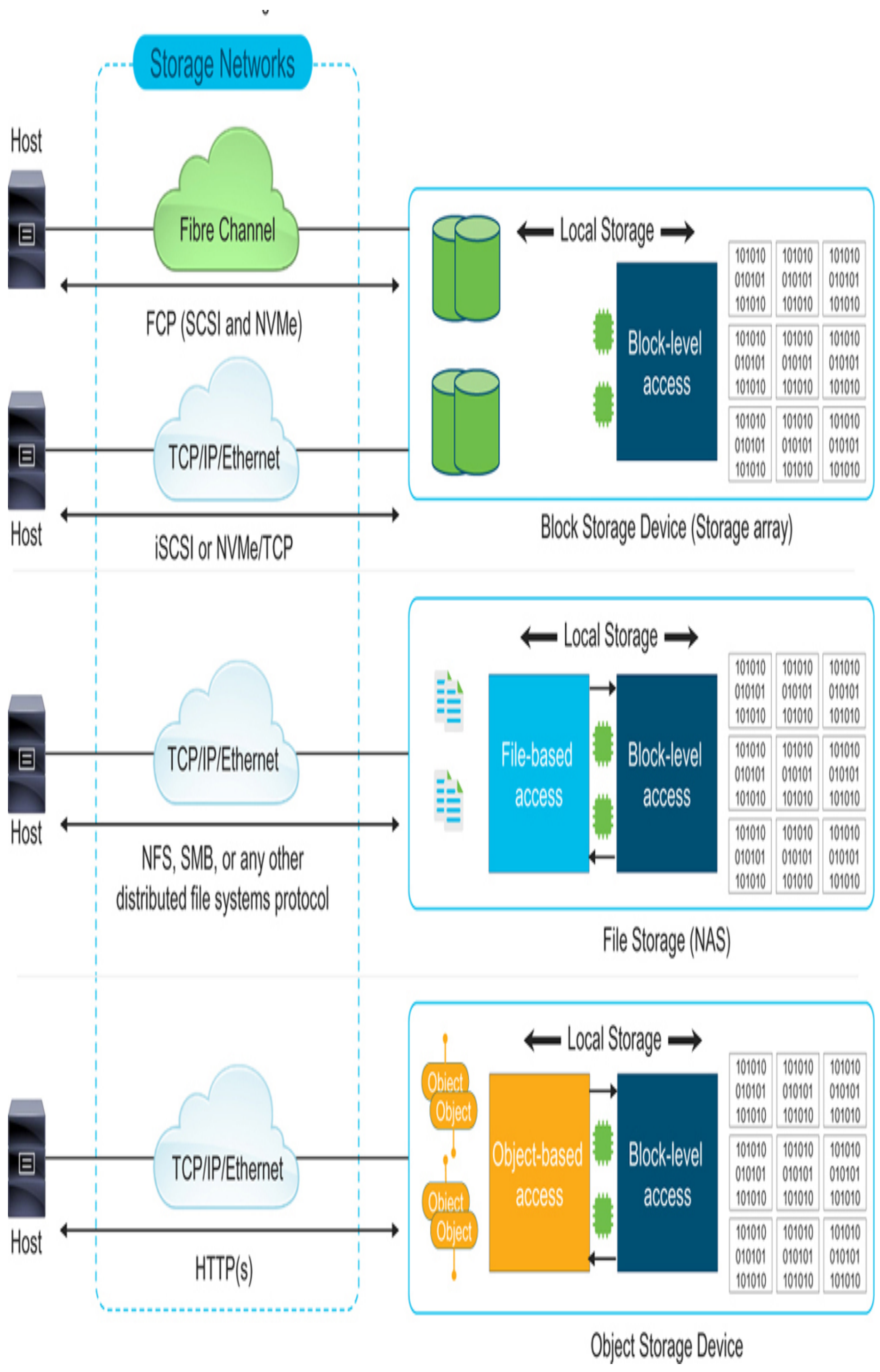


Figure 1-1 *A high-level overview of storage types based on their location and access level*

These are explained in the following sections.

Storage Type — By Access Level

The type of storage also depends on how it is presented to a host operating system. Local storage is presented as block storage to the operating system but may be presented as file storage or object storage to the application. Remote storage can be presented as block, file, or object storage.

Block Storage

A storage media (rotating disk or solid-state drive) is organized into blocks. Each block can store a predefined number of bytes of data, called the block size, such as 512 bytes, 1 KB, and 4 KB. When the SCSI protocol is being used, these blocks are organized into logical units (LU), and each logical unit is assigned a number, called a Logical Unit Number (LUN). An operating system uses a LUN and logical block address (LBA) to directly read and write data on a logical unit. This is called block-level access or simply, block storage.

The key protocols to access storage at the block level are SCSI and NVMe. Terms like LU and LUN are specific to SCSI, whereas NVMe uses different terminology such as Namespace (NS) and Namespace ID (NSID).

Regardless of local or remote block storage, a logical unit (SCSI) or namespace (NVMe) is presented to the host operating as a block storage volume. An application remains unaware if the storage is physically local or remote.

Fibre Channel is the most used network for carrying SCSI and NVMe I/O Operations between the hosts and the storage arrays to provide block-level access. iSCSI and NVMe/TCP offer the same function via an Ethernet network. Fibre Channel fabrics are typically referred to as Storage Area Networks (SANs), iSCSI and NVMe/TCP environments are typically referred to as IP SAN, and the storage devices are called SAN storage arrays.

Block storage is commonly used for transactional applications, which can read and write to each block directly, randomly, and rapidly with minimal overhead.

SCSI

SCSI (Small Computer System Interface) is a way of accessing computer peripherals. It defines hardware and software specifications, but for this book, think of SCSI as a software layer for block-based access to storage devices. It provides over a hundred commands, but the most important are the many variants of the read and write commands.

NVMe

NVMe (Non-Volatile Memory Express) is a relatively new way of block-based access to storage devices. It was needed because the recent storage devices, such as solid-state drives (SSD) allow electronic data access with high parallelism. This is different from the earlier hard-disk drives (HDD) that were limited in performance because of rotating media and mechanically moving heads. While the storage media is fast and is expected to become even faster in the future, even a minor software delay makes a difference. Therefore, NVMe was designed from the ground up to unleash the full potential of storage media. Unlike SCSI, NVMe has a lean stack. It offers only a handful of commands for I/O operations. The most important are those that read and write data.

File Storage

File-level access abstracts block-level details and presents storage to an application or operating system in the form of files and directories. As a result, files can be directly accessed without being aware of the logical address of data on the storage media.

The key protocols for accessing remote file storage are Network File System (NFS) and Server Message Block (SMB). These protocols use TCP or UDP transport over an Ethernet network for accessing network-attached storage (NAS).

A variant of SMB called SMB Direct uses an RDMA-capable protocol over a supported transport. Likewise, NFS over RDMA uses an RDMA-capable protocol for accessing a remote NAS device.

File storage is commonly used for document sharing, clustered databases, storing video and audio files, or anything else that classifies as a file.

Object Storage

Object storage is accessed by a handle, called an Object ID or a URL. All the other underlying details of the storage media are abstracted by an Object Storage Device (OSD). These objects are not changed in place. For making changes, a new version of the object is created instead. Because of the use of URLs, there is no need to mount a volume (as with block storage) or file system (as with file storage).

Object storage is accessed using Hypertext Transfer Protocol (HTTP), which further uses TCP transport over an Ethernet network, via a Wide-Area Network (WAN) link to remote sites, or an Internet connection to the public cloud.

The common use of object storage is in storing massive amounts of data that may not change often, such as for image hosting, video streaming services, and data backup.

Storage for Clustered and Distributed File Systems

Clustered File Systems (CFS) and Distributed File Systems (DFS) need special mention because of their use in handling a large amount of unstructured data, also known as big data. A DFS typically spans multiple commodity servers within the same building, remote sites, or even the public cloud. This is different from a centralized storage device such as SAN storage arrays and NAS devices, although DFS can span across these devices as well. Common implementations of DFS are Hadoop Distributed File System (HDFS), Ceph, and MapR file system, as well as NFS and SMB.

Based on the implementation, devices in a CFS or DFS cluster may use Remote Procedure Calls (RPCs) to communicate among themselves. RPC is an Open Systems Interconnection (OSI) layer 5 (sessions layer) protocol, and it assumes that some transport protocol, such as TCP exists. Additionally, API calls may use HTTP, which is an OSI layer 7 (application layer) protocol, and it uses TCP as its transport protocol.

Overall, most CFS and DFS implementations use TCP (and sometimes UDP) over an Ethernet network, via a WAN link to remote sites, or an Internet connection to the public cloud.

This book considers the network that connects the devices of a DFS as a storage network. Most details explained in Chapter 7, "Congestion Management in Ethernet Storage Networks" and Chapter 8, "Congestion Management in TCP Storage Networks" also apply to CFS and DFS environments.

SDS, HCI, and Everything Else

Software-Defined Storage (SDS) abstracts the lower-level details regardless of block, file, or object storage. Hyperconverged Infrastructure (HCI) combines compute, network, and storage in a single solution, and hence it includes a type of SDS solution with it.

A network that connects the devices of an HCI or SDS solution is considered a storage network in this book. Most SDS and HCI solutions have hooks to File storage using NFS or similar and block storage using iSCSI, NVMe/TCP, and Fibre Channel SAN. Most details explained in Chapter 7 and Chapter 8 also apply to SDS and HCI environments.

Storage Protocols, Transports, and Networks

This section provides a high-level overview of storage protocols, their transports, and networks. For the scope of this book, stay focused on how these protocols ultimately result in packets on the networks and how they are flow controlled.

Network Type — By Framing and Encoding

The most common types of storage networks are Ethernet, Fibre Channel, and InfiniBand. As Figure 1-2 shows, these networks have their dedicated physical and link layers. In other words, each defines a frame format as well as an encoding scheme for transmitting bits on the physical media (copper or fiber).

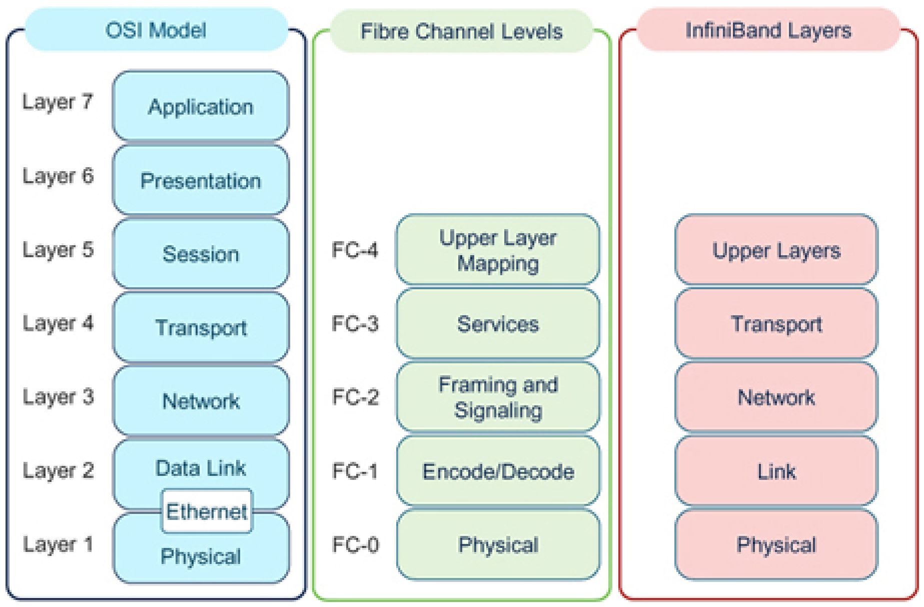


Figure 1-2 *OSI Model, Fibre Channel Levels, and InfiniBand Layers*

At a user level, there is nothing common between Fibre Channel, InfiniBand, and Ethernet networks except the end devices. However, at the technology level, they share many functions. For example, the Forward Error Correction (FEC) algorithm in Fibre Channel is the same as Ethernet has defined. Chapter 2 explains the FEC details for Fibre Channel. A similar explanation holds good for Ethernet also.

Ethernet

Ethernet is the most common network type across all kinds of data centers across the world. It belongs to layer 1 (physical layer) and layer 2 (link layer) in the OSI model (Figure 1-2). Ethernet makes the foundation for the upper layer protocols such as IP, UDP, and TCP, which further transport many types of storage protocols.

End devices connect to an Ethernet network using Network Interface Cards (NICs). This book uses GbE (Gigabit Ethernet) as a suffix to refer to various Ethernet speeds, such as 10 GbE and 25 GbE.

Fibre Channel (FC)

Fibre Channel is the most common type of network for transporting SCSI and NVMe protocols for accessing remote block storage. As Figure 1-2 shows, it has dedicated layers for physical interface (FC-0), encoding and decoding (FC-1), framing (FC-2), network services (FC-3), and transporting upper layer protocols (FC-4). While this entire stack is referred to as Fibre Channel, at FC-4 layer, Fibre Channel Protocol (FCP) provides mapping to the Upper Layer Protocols (ULP), such as SCSI and NVMe.

Essentially, FC is the network as well as the transport. SCSI and NVMe are the storage protocols.

Hosts connect to Fibre Channel fabrics using Host Bus Adapters (HBA). Storage arrays refer to their Fibre Channel ports as front-end ports, FA (Fibre Adapter) ports, or other names based on the product and vendor type.

This book uses GFC as a suffix to refer to various Fibre Channel speeds, such as 32GFC and 64GFC.

InfiniBand (IB)

InfiniBand is primarily used in High-Performance Computing (HPC) environments for providing direct memory access to a remote system, hence the term Remote Direct Memory Access (RDMA). Among its community, the InfiniBand operations are called RDMA verbs.

As Figure 1-2 shows, InfiniBand has a dedicated physical layer, link layer, network layer, and transport layer.

End devices connect to the InfiniBand networks using Host Channel Adapters (HCA).

In addition to transferring frames via an InfiniBand network, end devices can also exchange the same RDMA verbs via Ethernet networks using one of the RDMA-capable protocols. Each RDMA-capable protocol further defines the type of transport and network it uses. For example, RDMA over Converged Ethernet (RoCE) uses Ethernet networks, RoCE version 2 (RoCEv2) transfers UDP/IP packets via Ethernet networks, and iWARP transfers TCP/IP packets via Ethernet networks.

An RDMA layer within the end device translates the upper layer operations to RDMA verbs, such as RDMA_SEND, RDMA_READ, and RDMA_WRITE, which are encapsulated in lower layer headers based on the type of the network. This enhanced RDMA stack remains only within the end devices. The network remains unaware of these translations and transfers the packets to the destination using its standard forwarding mechanism.

Chapter 7 provides an example of RoCEv2 I/O operation. The network traffic pattern would be similar to any other RDMA-capable protocol. Further details of the RDMA stack within the end devices and InfiniBand networks are outside the scope of this book.

Network Type — By Use of Flow Control

Networks can be classified into three categories based on their use of flow control between directly connected devices—Lossy, Lossless, and Converged.

Lossy Networks

Networks that do not use flow control between directly connected devices are called lossy Networks.

Figure 1-3, Scenario-1 shows a speed-mismatch condition. Switch-1 connects to Target-1 at 10 GbE and Host-1 at 1 GbE. When Target-1 starts transmission at 10 Gbps, Switch-1 cannot transmit at the same rate on a 1 GbE link to Host-1. As a result, Switch-1 drops excess frames. Scenario 2 does not have a speed mismatch but rather has an aggregate bandwidth mismatch. When Target-1, Target-2, and Target-3 simultaneously start transmission at 10 Gbps, Switch-1 cannot transmit at 30 Gbps on a 10 GbE link to Host-1. Hence, even in this condition, Switch-1 drops excess frames.

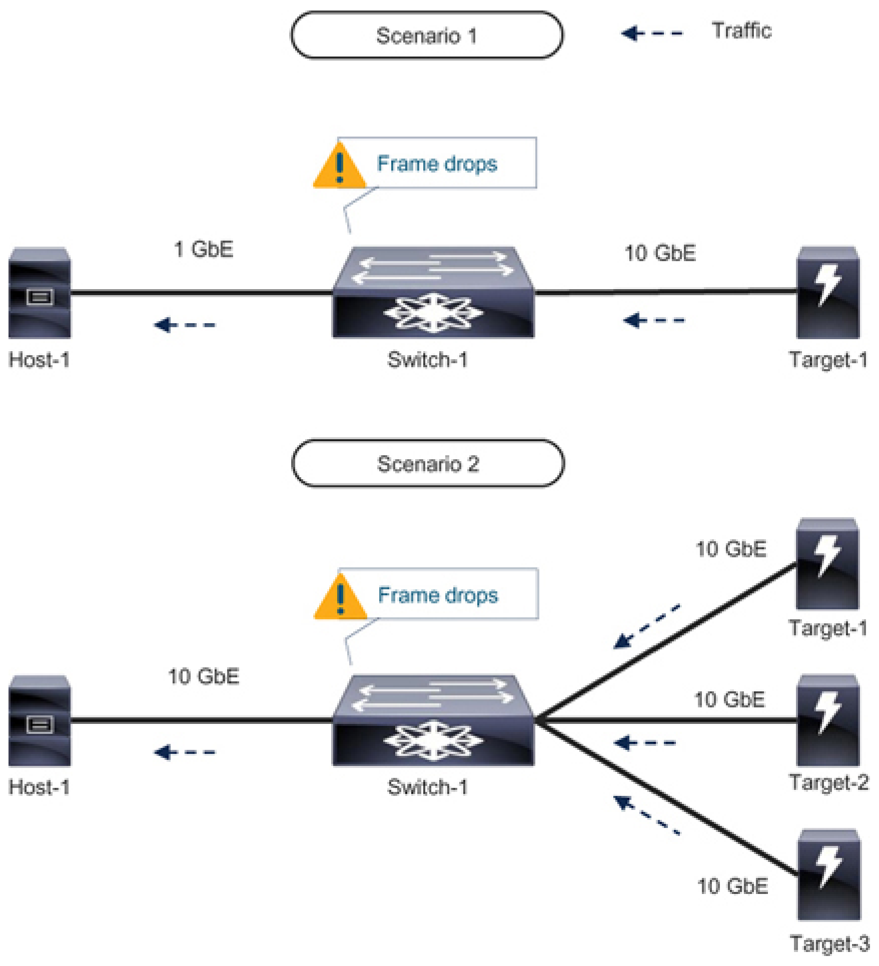


Figure 1-3 *Lossy networks drop frames during congestion*

A general-purpose Ethernet network that most people see around them does not use flow control to stop or slow down the sender from transmitting when the receiver's buffers are full, and hence, any new incoming frames are dropped by the receiver. When packets are dropped, another layer in the stack handles this condition. The most well-known example is TCP at layer 4 (transport layer) of the OSI model, which detects and retransmits the lost packets and adjusts its transmission rate to control network congestion.

But the key point to understand is that lossy networks handle congestion by dropping frames. They do not use flow control between directly connected devices to match the ingress traffic rate with the egress traffic rate.

Lossless Networks

Networks that use flow control between directly connected devices are called lossless Networks. This flow control results in applying backpressure to the directly connected sender to pace its traffic rate for avoiding buffer overrun on the receiver. By doing this traffic entering the network is equalized with the traffic exiting the network.

As Figure 1-4 scenario-1 shows, Switch-1 uses flow control to apply backpressure to slow down the ingress traffic from Target-1 to match the egress traffic rate (1 Gbps) to Host-1. Likewise, in scenario 2, Switch-1 applies backpressure to slow down the collective traffic rate from the three targets to match the egress traffic rate (10 Gbps) to Host-1.

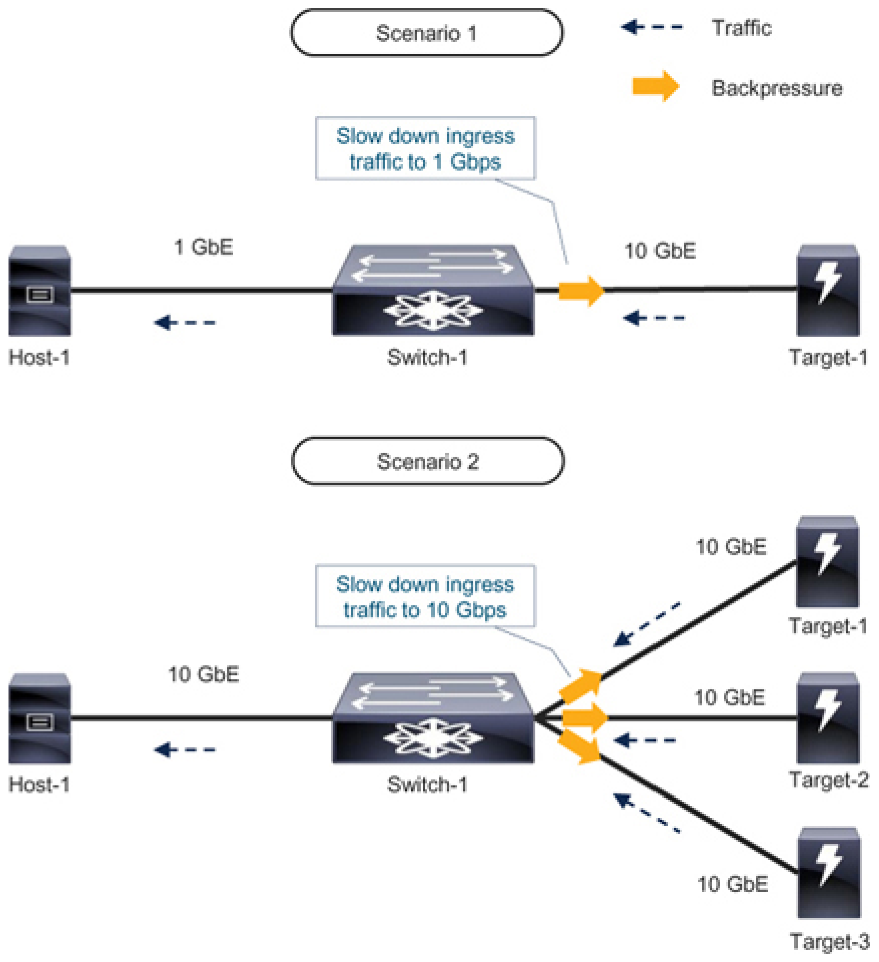


Figure 1-4 *Flow control in Lossless Networks*

In Figure 1-4, backpressure refers to a generic term for using flow control against traffic direction regardless of its implementation. Common examples of flow control mechanisms for achieving the function of a lossless network are Buffer-to-Buffer (B2B) flow control used by Fibre

Channel, a similar credit-based flow control mechanism used by InfiniBand, and Link-Level Flow Control (LLFC) and Priority-based Flow Control (PFC) used by Ethernet. B2B flow control uses a special bit sequence for flow control, called Receiver_Ready (R_RDY). LLFC and PFC use a dedicated Ethernet frame for flow control, called a Pause frame.

Flow control in lossless networks works only between the directly connected devices, such as between Host-1 and Switch-1, and between Switch-1 and Target-1, and hence it achieves hop-by-hop flow control. This should not be confused with the end-to-end flow control used by TCP, which aims to avoid buffer overrun of the endpoints of a TCP connection even via a lossy network.

A key point to understand is that these flow control mechanisms do not guarantee that traffic will not be dropped. Traffic is still dropped in some conditions, such as when frames are held up in the buffers for a long duration during severe congestion and when frames are corrupted due to bit errors. In other words, lossless networks do not provide a guarantee of lossless frame delivery, although they end up achieving it quite well.

Converged Ethernet Networks

Ethernet networks can flow control all the traffic on a link using Link-Level Flow Control (LLFC) or only selective traffic classes using Priority-based Flow Control (PFC). This means that PFC allows lossy and lossless functions on a shared Ethernet network. These are called Converged Ethernet Networks.

Note that the terms—Converged Networks or Converged Ethernet Networks—have been used in some literature to refer to the networks that carry storage and non-storage traffic regardless of lossless behavior. If your understanding is different, this book does not intend to change that. However, in this book, Converged Networks refer to networks that are configured to carry lossy and lossless traffic simultaneously.

Crossing the Boundaries of Network Types

Over the decades, many technologies have emerged for achieving the same results from Ethernet that Fibre Channel has been delivering in moving storage data and InfiniBand has been delivering in moving memory data across a network.

This section explains two such technologies. First, FCoE, which crosses the boundaries of the Fibre Channel levels and the OSI model at layer 2. Second, RoCE, which crosses the boundaries of the InfiniBand layers and the OSI model at layer 2.

Fibre Channel over Ethernet (FCoE)

FCoE retains the benefits of Fibre Channel upper layers on the Ethernet physical and link layers. It transports the same upper-layer protocols as Fibre Channel, such as SCSI and NVMe, for accessing remote block storage.

As Figure 1-5 shows, FCoE has a dedicated header. It encapsulates Fibre Channel frames, which are further encapsulated in the Ethernet frames. The Ethernet header uses an Ethertype of 0x8906 to indicate that the next header type is FCoE.

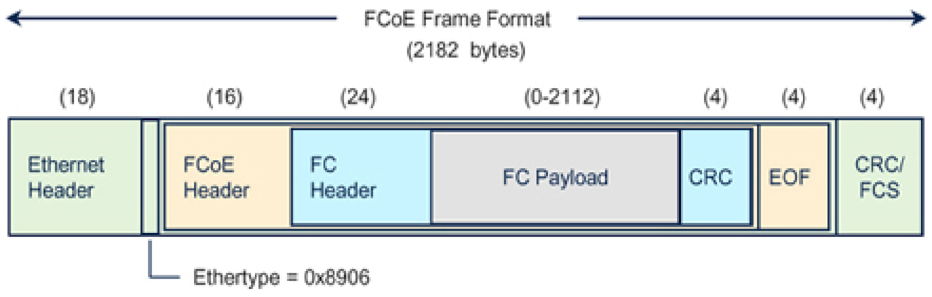


Figure 1-5 *FCoE Frame Format*

FCoE end devices connect to an Ethernet network using a Converged Network Adapter (CNA). Some documents refer to it as CAN, which is a typo resulting from a spelling check.

When Fibre Channel frames are encapsulated in Ethernet frames, the B2B flow control mechanism is not used because it works at a lower sub-layer than the framing layer. To retain the function of a lossless network, Ethernet uses its own Priority-based Flow Control (PFC) for FCoE traffic. A CNA "converges" lossy and lossless functions on the same link by classifying Fibre Channel traffic as lossless and other traffic as lossy.

RDMA over Converged Ethernet (RoCE)

RoCE (pronounced as rocky) retains the benefits of InfiniBand upper layers or RDMA on the Ethernet physical and link layers. It transports the same upper-layer protocols as InfiniBand. For example, NVMe/RoCE transports NVMe I/O operations using RDMA verbs via an Ethernet network.

What Fibre Channel did with FCoE, InfiniBand did with RoCE. But instead of calling it InfiniBand over Ethernet (IBoE), it was called RDMA over Converged Ethernet. Deriving its name based on what it does versus what it is was a good choice for increasing awareness because fewer people know about InfiniBand. Additionally, the ability to directly access the memory of a remote system seems impressive.

Unlike FCoE, RoCE does not have a dedicated header. While looking at its packet format, some new users keep looking for the RoCE header only to realize that it is just a name.

End devices with RoCE functions connect to an Ethernet network using an RDMA NIC (RNIC). These end devices encapsulate InfiniBand packets in Ethernet frames and use the EtherType of 0x8915 to indicate that the next header type is InfiniBand. These details of the RDMA stack are abstracted by the RNIC. The network is not aware of the presence of the InfiniBand layers and simply transports the Ethernet frames to their destinations using the standard forwarding functions.

RoCE generally uses a lossless network (with PFC), although in some environments it may use a lossy network. When RoCE uses a lossless network, its traits of network congestion are the same as seen for FCoE, hence, these are collectively explained in Chapter 7.

RoCE is revisited again in the following sections along with the other RDMA-capable protocols. Refer to Figure 1-10 for

RoCE frame format.

Climbing Up the Networking Layers

This section explains mechanisms to transport storage traffic by the higher layers in the OSI model. These are different from the mechanisms explained in the previous section that cross the network boundaries at the framing layer, which maps to layer 2 in the OSI model.

Internet Protocol

IPv4 and IPv6 are the de facto layer 3 (network layer) protocols in the OSI model. Adding an IP header on an Ethernet frame allows it to be routed between layer 2 domains.

In addition to delivering a packet to its destination, IP provides two key functions.

First, the 6-bit DSCP field in the IP header (Figure 1-6) allows classifying traffic into different classes, which can be used for providing quality of service to storage traffic, such as a minimum bandwidth guarantee.

Figure 1-6 shows the Ethernet frame format with TCP transport.

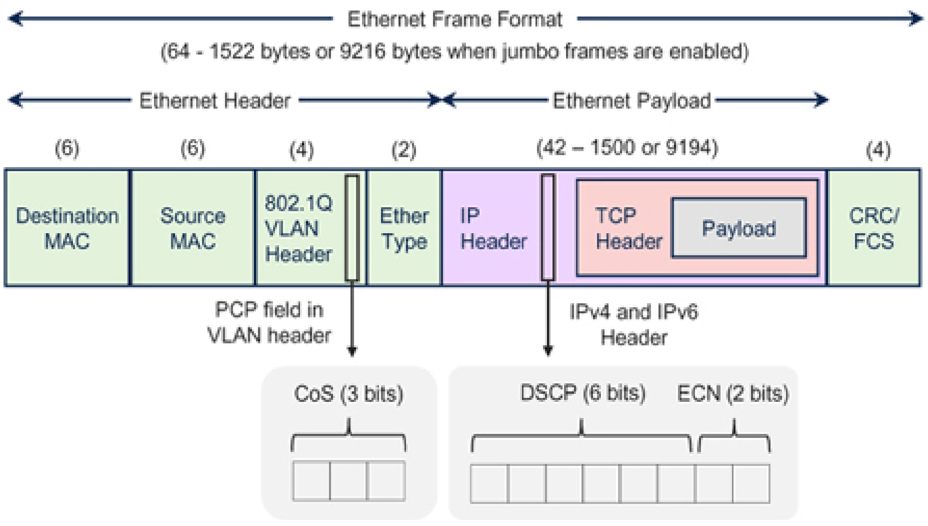


Figure 1-6 *Ethernet Frame Format with TCP transport*

Second, the 2-bit Explicit Congestion Notification (ECN) field in the IP header (Figure 1-6) can be used for notifying the end devices when the network is congested. The use of the ECN mechanism with TCP as the layer 4 (transport layer) protocol is explained in Chapter 8. The use of ECN for RoCEv2 Congestion Management (RCM) is explained in Chapter 7.

Transmission Control Protocol (TCP)

TCP is a layer 4 (transport layer) protocol in the OSI model. It is used as the “transport protocol” by many “storage protocols”. For example,

- Block storage protocols, like iSCSI and NVMe/TCP use TCP transport.
- iWARP, which is an RDMA-capable protocol uses TCP transport.
- Object Storage uses HTTP which further uses TCP transport.
- File Storage protocols like NFS and SMB use TCP transport.
- Distributed File Systems like HDFS, Ceph, and MapR may use one of the previously listed protocols or their custom implementations via TCP transport.

Generally, TCP runs on a lossy network because it already has robust mechanisms for reliable data delivery with built-in end-to-end flow control and congestion control. Some environments, however, may run TCP in lossless networks. Although this topic is not directly covered in this book, such environments may benefit from the details covered in Chapter 7 and Chapter 8.

User Datagram Protocol (UDP)

UDP is another layer 4 (transport layer) protocol in the OSI model that generally runs on lossy Ethernet networks. But unlike TCP, UDP does not provide reliable data delivery and end-to-end flow control, and hence, an upper layer or lower layer must provide these functions. For example, RoCEv2 adds UDP and IP headers on InfiniBand packets but may use a lossless Ethernet network. Some file storage protocols may also use UDP.

iSCSI

Internet SCSI (iSCSI) carries SCSI commands over TCP transport (Figure 1-7) for accessing remote block storage via a lossy or sometimes lossless network. It is an OSI layer 5 (sessions layer) protocol.

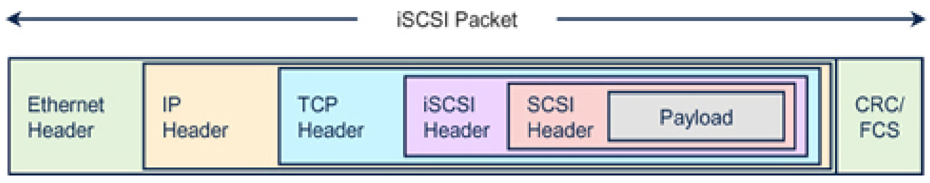


Figure 1-7 *iSCSI Packet Format*

NVMe/TCP

NVMe/TCP carries NVMe commands over TCP transport (Figure 1-8) for accessing remote block storage via a lossy or sometimes lossless network. It is an OSI layer 5 (sessions layer) protocol.

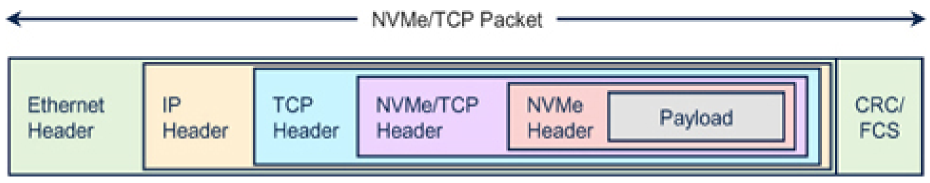


Figure 1-8 *NVMe/TCP Packet Format*

NFS

Network File System (NFS) provides access to remote file storage over TCP transport via a lossy network. It spans layer 5 (sessions layer), layer 6 (presentation layer), and layer 7 (application layer) of the OSI model.

SMB

Server Message Block (SMB) is another protocol for providing access to remote file storage over TCP transport via a lossy network. It spans layer 5 (sessions layer), layer 6 (presentation layer), and layer 7 (application layer) of the OSI model.

HTTP

Object storage uses HTTP(s) over TCP transport via a lossy network. HTTP is an OSI layer 7 (application layer) protocol.

Crossing the Boundaries of Network Types — Again

As the previous section explains, at the higher layers of the OSI model, for remote block storage, SCSI and NVMe directly map to the OSI model using iSCSI and NVMe/TCP protocols. This is a similar mapping that the Fibre Channel Protocol (FCP) achieves at the FC-4 layer for transporting SCSI and NVMe traffic via Fibre Channel fabrics. Hence, in general, there is no need to cross the boundaries of Fibre Channel layers and OSI model at the higher layers, except with a few exceptions such as FCIP.

Fibre Channel over IP (FCIP)

Fibre Channel frames are encapsulated within the FCIP header (Figure 1-9), which is transported using TCP/IP via a lossy network. FCIP is primarily used for data replication and backup over longer distances. Hence, its TCP connections typically pass through a WAN link.

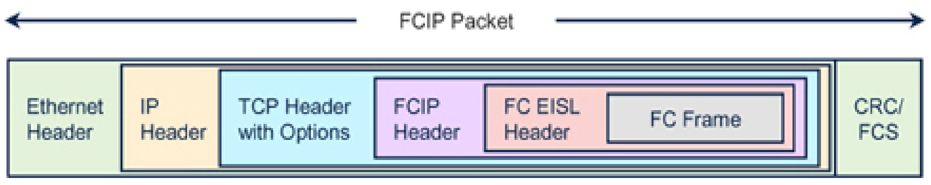


Figure 1-9 *FCIP Packet Format*

RDMA-capable Protocols

The interaction between the InfiniBand and OSI layers is not that straightforward at the higher layers because of many types of RDMA-capable protocols.

RDMA over Converged Ethernet (RoCE)

As explained earlier, RoCE end devices encapsulate InfiniBand packets in Ethernet frames (Figure 1-10). The network is not aware of the presence of the InfiniBand layers and simply transports the Ethernet frames to their destinations using lossless Ethernet or sometimes lossy Ethernet networks.

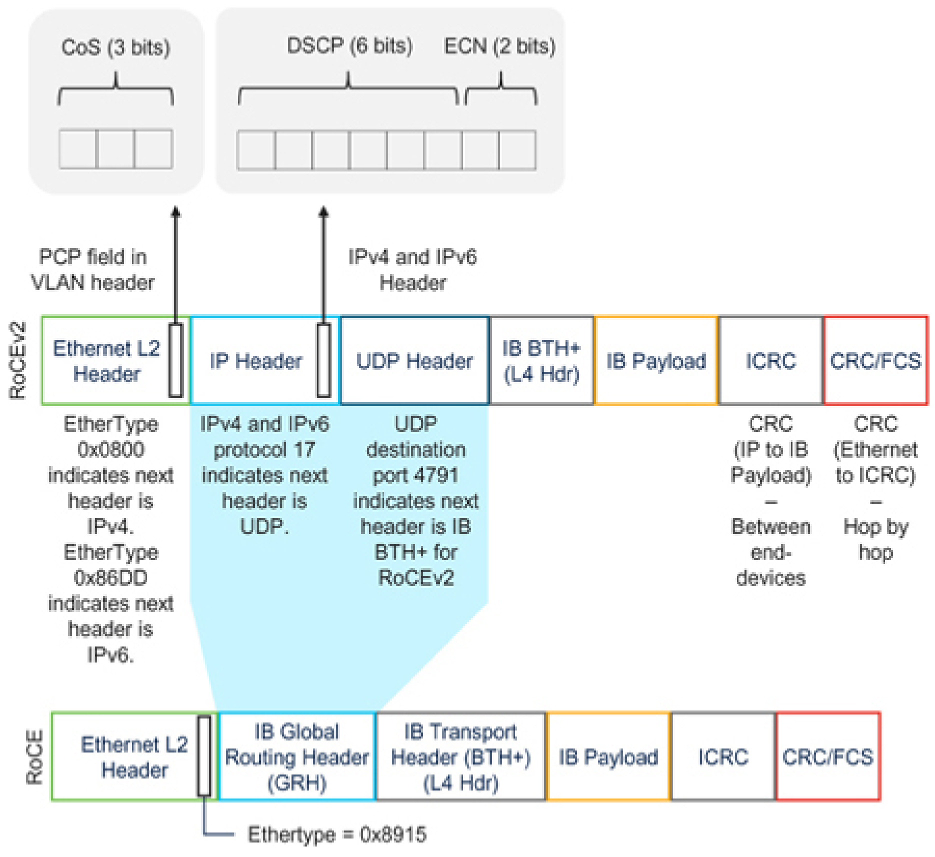

CoS (3 bits)
DSCP (6 bits)
ECN (2 bits)
PCP field in VLAN header
IPv4 and IPv6 Header
RoCEv2
Ethernet L2 Header
IP Header
UDP Header
IB BTH+ (L4 Hdr)
IB Payload
ICRC
CRC/FCS
EtherType 0x0800 indicates next header is IPv4. EtherType 0x86DD indicates next header is IPv6.
IPv4 and IPv6 protocol 17 indicates next header is UDP.
UDP destination port 4791 indicates next header is IB BTH+ for RoCEv2
CRC (IP to IB Payload) – Between end-devices
CRC (Ethernet to ICRC) – Hop by hop
RoCE
Ethernet L2 Header
IB Global Routing Header (GRH)
IB Transport Header (BTH+) (L4 Hdr)
IB Payload
ICRC
CRC/FCS
Ethertype = 0x8915

Figure 1-10 *RoCE and RoCEv2 Packet Format*

RDMA over Converged Ethernet (RoCE) Version 2

RoCEv2 is similar to RoCE except that in addition to the OSI layer 2 Ethernet header, it also adds OSI layer 3 IP and layer 4 UDP headers (Figure 1-10). Unlike RoCE (version 1) end devices that use EtherType of 0x8915 to indicate that the next header is the InfiniBand network layer, RoCEv2 end devices use EtherType of 0x0800 for IPv4 and 0x86DD for IPv6. The UDP port number 4791 indicates that the next header is the InfiniBand transport layer.

As previously explained, adding the IP header has three key benefits. First, it allows RoCEv2 end devices to be connected across an IP-routed network, therefore they need not be limited to the same layer 2 domain. For this reason, RoCEv2 is also known as Routable RoCE (RRoCE). Second, the DSCP bits in the IP header allow classifying RoCEv2 traffic for Quality of Service and PFC. Third, the ECN bits in the IP header allows RoCEv2 Congestion Management (RCM). Refer to Chapter 7 for further details.

Note that the RoCEv2 end devices remove the InfiniBand network header (different from the IP header in OSI layers) but retain the InfiniBand transport header (different from the TCP or UDP header in OSI layers). But these changes remain only within the end devices. The network is not aware of the presence of the InfiniBand layers and simply transfers the UDP/IP packets to their destinations using the standard forwarding function of a network.

Generally, RoCEv2 traffic runs on a converged Ethernet network while using its lossless behavior. Chapter 7 covers the details of congestion management in Ethernet storage networks.

Some environments, however, may use RoCEv2 with a lossy network and rely on other mechanisms for end-to-end reliable delivery. These details are implementation dependent and outside the scope of this book.

iWARP

iWARP (may be expanded to Internet Wide-area RDMA Protocol) carries RDMA operations over TCP transport, and therefore it gets all the benefits of TCP such as reliable and in-order delivery, end-to-end flow control, and congestion control.

The iWARP end devices remove the InfiniBand network as well as the transport headers (different from IP and TCP headers in OSI layers) and add a dedicated iWARP header. But these changes remain only within the end devices. The network is not aware of the presence of the InfiniBand layers and simply transfers the TCP/IP packets via a lossy Ethernet or sometimes lossless Ethernet network.

Important Details About RDMA-capable Protocols

Note the following points about RDMA-capable protocols:

1. Although FCoE and RoCE use lossless Ethernet, their networks are different. For RoCE, the network need not

be aware of the contents of the Ethernet frame. Likewise, for RoCEv2 the network need not be aware of the content of the UDP/IP packet. In contrast, FCoE networks must be aware of the content of the Ethernet frame for enforcing Fibre Channel zoning and providing exchange-based load-balancing because the exchanges are carried within the Fibre Channel frames, which are prepended with FCoE headers. But the focus of this book is on network congestion and the PFC still works the same for FCoE, RoCE, and RoCEv2 networks. Therefore, their congestion management is similar, which is covered together in Chapter 7.

2. Despite iWARP being an RDMA-capable protocol, its congestion management is the same as a TCP network, which may be simultaneously used for many other storage protocols, such as iSCSI, NVMe/TCP, NFS, and SMB. Chapter 8 explains congestion management in TCP Storage Networks.
3. There are no dedicated headers for RoCE and RoCEv2. These are just the names. In contrast, iWARP has a dedicated header.
4. RDMA verbs have a different mechanism to move data. For reading data, SCSI and NVMe use read commands, whereas RDMA uses the RDMA_WRITE verb in the reverse direction. Likewise, for writing data, SCSI and NVMe use the write command, whereas RDMA uses the RDMA_READ verb in the reverse direction. The SCSI and NVMe I/O Operations are explained in Chapter 5, and RoCEv2 I/O operations are explained in Chapter 7. Despite these differences, for the scope of this book, there is no need to be confused about these details. Ultimately, the applications define the traffic profile on the network regardless of the type of storage protocol and transport protocol.
5. Just because RDMA's name has memory in it, doesn't mean it artificially moves any data that the application doesn't want it to move. Storage networks move the same data regardless of the use of SCSI, NVMe, or RDMA-

capable protocol. Think of SCSI, NVMe, and RDMA as different ways of transferring the same data.

Storage Protocols That Use RDMA

End devices may use one of the "RDMA-capable protocols" explained in the previous section for exchanging data using the following "storage protocols".

- **SMB Direct**—Unlike SMB which uses TCP transport, its variant called SMB Direct uses one of the RDMA-capable protocols—RoCE, RoCEv2, or iWARP.
- **NFS over RDMA**—Like SMB Direct, NFSoRDMA is a variant of NFS that uses one of the RDMA-capable protocols—RoCE, RoCEv2, or iWARP.
- **iSCSI Extensions for RDMA (iSER)**—As its name suggests, iSER uses one of the RDMA-capable protocols to transfer the same data that iSCSI over TCP would transfer.
- **NVMe/RDMA**—NVMe/RDMA carries NVMe over one of the RDMA-capable protocols, which can be called NVMe/RoCE, NVMe/RoCEv2, or NVMe/iWARP based on the protocol type.

When using iWARP, the transport is still TCP, hence the network is unaware of the RDMA changes within the end device. But RoCE and RoCEv2 generally run on lossless (or converged) Ethernet network, which is not commonly used by the TCP transport.

Figure 1-11 shows a consolidated view of the storage protocols, transports, and networks that are explained in this section. The key takeaway from this figure is that for network congestion management, focus on the transport—TCP over Lossy Ethernet, Lossless Ethernet, Fibre Channel, or InfiniBand. The network is unaware of the storage protocols, which are encapsulated by the end devices in the transport before sending the frames to the network.

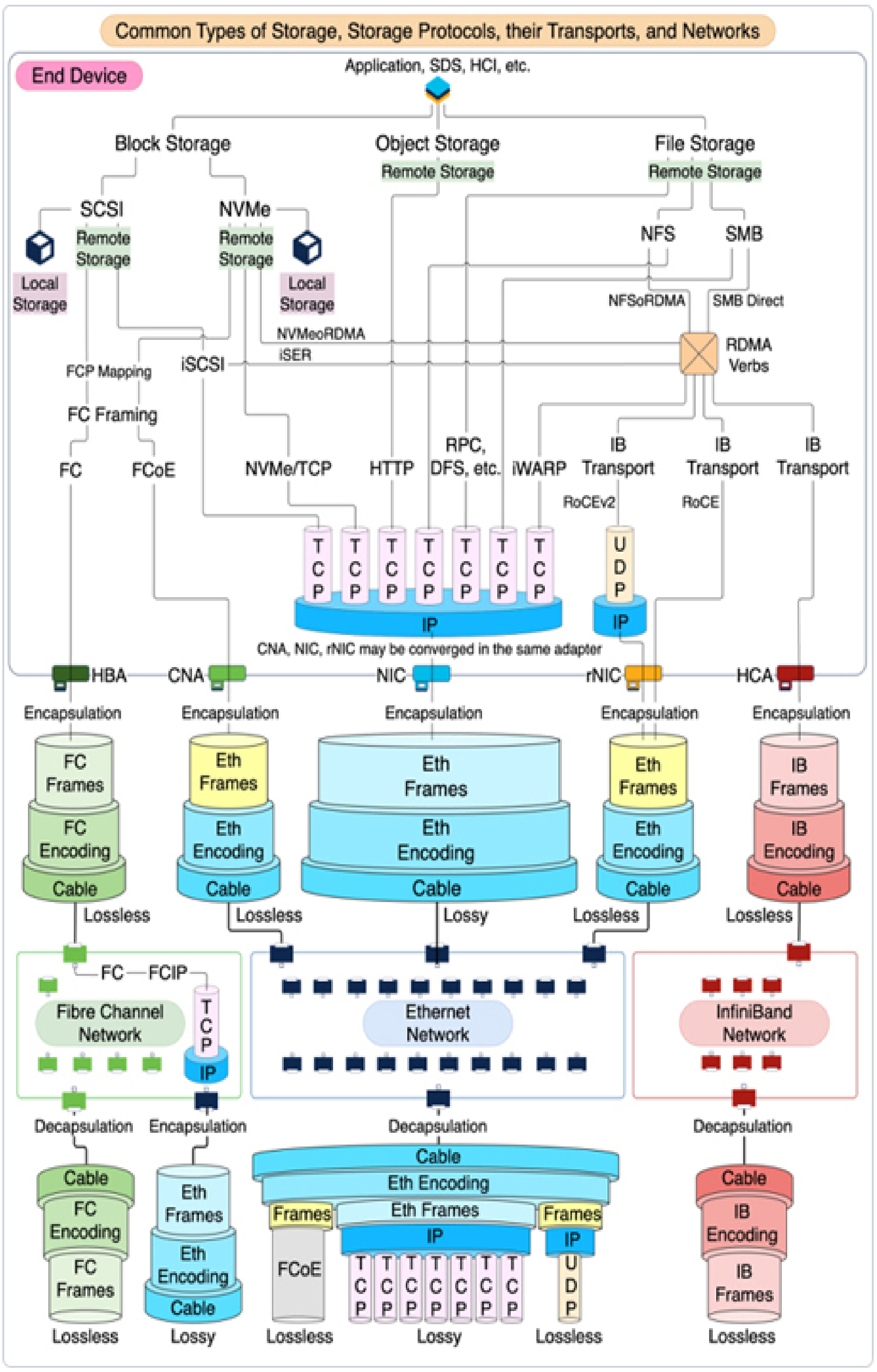


Figure 1-11 *Types of storage, storage protocols, their transports, and networks*

For Figure 1-11:

- Types of Storage as per their access level - Block, File, and Object
- Types of Storage as per physical location: Local and Remote

- Storage Protocols for Local Storage: SCSI and NVMe
- Storage Protocols for Remote Storage:
  - Block: iSCSI, NVMe/TCP, NVMe/FC, NVMe/RDMA
  - File: NFS, SMB, NFSoRDMA, SMB Direct, DFS implementations
  - Object: HTTP
- Networking Technologies: Fibre Channel (Lossless), InfiniBand (Lossless), and Ethernet (Lossy and Lossless)
- Transports: TCP over Lossy Ethernet, Lossless Ethernet, Fibre Channel, InfiniBand
- RDMA Transports: Lossless Ethernet (RoCE), UDP/IP over Lossless Ethernet (RoCEv2), iWARP over TCP Transport, InfiniBand over InfiniBand Transport

The following are the acronyms used in Figure 1-11:

- CNA: Converged Network Adapter
- DFS: Distributed File System
- FCoE: Fibre Channel over Ethernet
- FCIP: Fibre Channel over IP
- FCP: Fibre Channel Protocol
- HCI: Hyperconverged Infrastructure
- HCA: Host Channel Adapter
- HBA: Host Bus Adapter
- HTTP: Hypertext Transfer Protocol
- IP: Internet Protocol
- iSCSI: Internet SCSI
- iSER: iSCSI Extensions for RDMA
- iWARP: (Maybe) Internet Wide-area RDMA Protocol
- NIC: Network Interface Card
- NFS: Network File System
- NFSoRDMA: NFS over RDMA

- NVMe: Non-Volatile Memory Express
- NVMe-oF: NVMe over Fabrics
- NVMe/FC: NVMe over Fibre Channel
- NVMe/RDMA: NVMe over RDMA
- NVMe/TCP: NVMe over TCP
- RDMA: Remote Direct Memory Access
- RoCE: RDMA over Converged Ethernet
- RoCEv2: RoCE version 2 or Routable RoCE
- RNIC: RoCE NIC
- RPC: Remote Procedure Call
- SCSI: Small Computer System Interface
- SDS: Software Defined Storage
- SMB: Server Message Block
- SMB Direct: SMB over RDMA
- TCP: Transmission Control Protocol
- UDP: User Datagram Protocol

Storage Networks

A storage network provides access to a remote storage device using any of the block, file, or object storage protocols explained in the previous section.

A storage network contains the following entities:

- **End devices:** End devices transfer data through a network. Based on their function, the end devices can be of the following types:
  - **Hosts**: Devices that run the applications. These are also called Initiators because they initiate read and write I/O operations.
  - **Storage arrays**: Devices that store the data. These are also called Targets because they are the targets of the

I/O operations.

- **Switches**: Switches connect the end devices and other switches for providing control-plane functions (for example, Fibre Channel zoning, name server, and routing protocols) and data-plane functions (for example packet forwarding between the switchports and flow control in lossless networks).
- **Links**: Links connect the end devices and switches. They are of the following types.
  - **Inter-Switch Links (ISL):** The links that connect switches with each other. The ISL ports are also known as core ports.
  - **Edge Links:** The links that connect the switches with the end devices. When an edge link connects to a host, it's called a host link, and the switchport is called a host-connected switchport. Likewise, when an edge link connects to a storage array, it's called a storage link, and the switchport is called a storage-connected switchport.

Storage Network Designs

Figure 1-12 shows common designs for storage networks.

Single-Switch Design

A single-switch design connects initiators and targets to the same switch. The number of end devices in this design is limited by the maximum number of ports on a switch.

The traffic path between initiators and targets involves two edge links and a switch.

Edge-Core Design

An edge-core design allows for connecting many more end devices. Targets connect to a set of switches, called core switches. Likewise, initiators connect to a different set of switches, called (host-)edge switches. ISLs connect core and edge switches. Generally, ISLs do not exist between edge switches. Likewise, ISLs do not normally exist between core switches.

The traffic path between initiators and targets involves two edge links, two switches, and an ISL.

Edge-Core-Edge Design

An edge-core-edge design allows for connecting even more end devices. Storage-edge switches connect to the targets, whereas host-edge switches connect to the initiators. Core switches connect storage-edge switches with the host-edge switches. Generally, ISLs do not exist between storage-edge switches. Likewise, ISLs do not normally exist between host-edge switches, and between core switches.

The traffic path between initiators and targets involves two edge links, three switches, and two ISLs.

Mesh Design

Mesh designs connect all switches to each other. Sometimes, it may also be a partial-mesh design.

When initiators and targets are connected to the same switch, their traffic path involves two edge links and one switch (like the switch-single design). When initiators and targets connect to different switches, their traffic path involves two edge links, two switches, and an ISL (like edge-core design).

Spine-Leaf Design

The spine-leaf design, also known as the clos design, can scale for connecting thousands of end devices. Targets and initiators connect to the same or different leaf switches. Spine switches connect leaf switches with each other and typically do not connect directly to the end devices. ISLs do not exist between the spine switches. Likewise, ISLs do not exist between the leaf switches.

When initiators and targets are connected to the same leaf switch, their traffic path involves two edge links and one switch (like the switch-single design). When hosts and targets connect to different leaf switches, their traffic path involves two edge links, three switches, and two ISLs (like edge-core-edge design).

Spine-leaf design is common for Ethernet networks. It is like the edge-core-edge design of Fibre Channel fabrics.

As Figure 1-12 shows, In Fibre Channel and FCoE environments, hosts and targets connect via two physically separate fabrics. These are called SAN-A and SAN-B, Fabric-A and Fabric-B, or similar. Congestion in one fabric remains segregated only to that fabric, although in some cases, an end device may cause congestion in both fabrics.

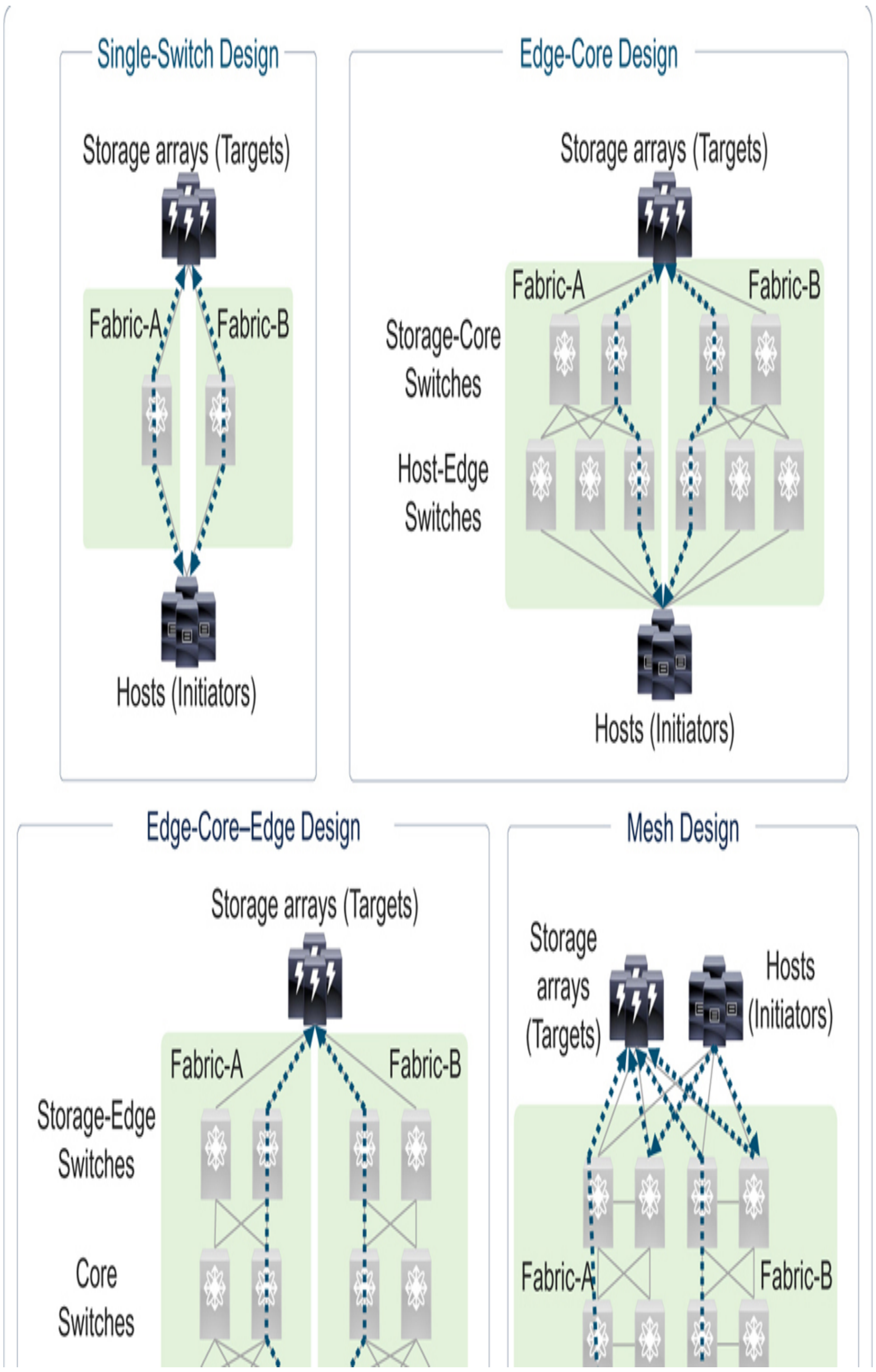


Figure 1-12 *Common storage network designs*

In Ethernet and TCP storage networks, when using spine-leaf design, an end device connects to two different leaf switches and a logical separation may exist between two paths, such as using VLANs. But physically, the network is shared. As a result, congestion does not remain segregated as compared to the design of two physically separate networks.

Terminology

iSCSI uses the terminology of initiator (or iSCSI client) and target (or iSCSI server). NVMe/TCP uses the terminology of host and controller respectively.

Fibre Channel uses the terminology of initiator, where each host may have more than one initiator. Likewise, each storage array may have more than one target.

This book uses initiator and host interchangeably based on the context regardless of the network and protocol type. Likewise, target and storage array are used interchangeably.

Fibre Channel and FCoE Terminology

Fibre Channel and FCoE use the following additional terminology (Figure 1-13):

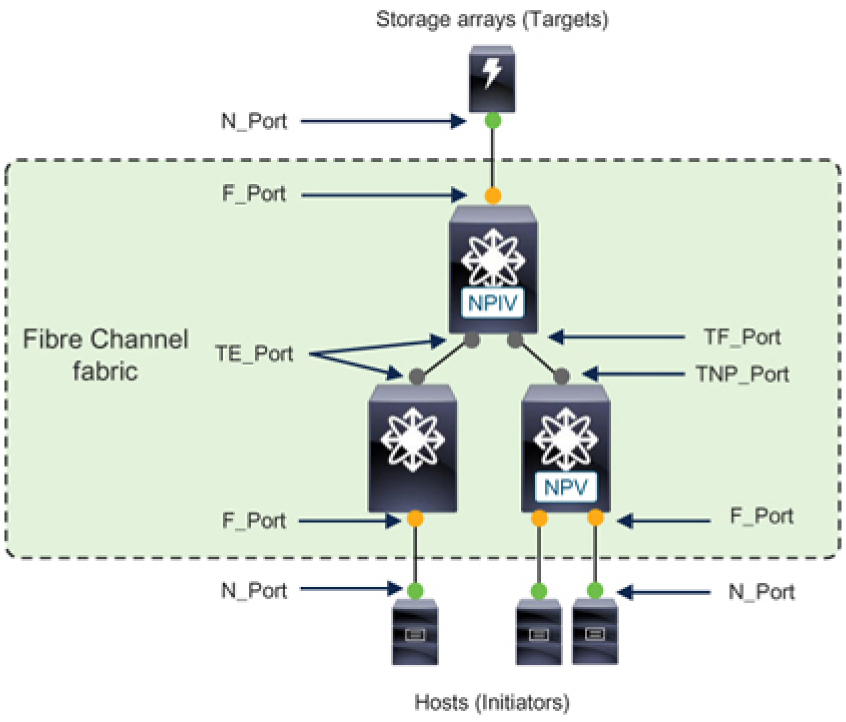


Figure 1-13 *Fibre Channel terminology*

- **N_Port:** N_Port is an end device port that connects to a Fibre Channel fabric. N refers to Node. FCoE calls this VN_Port (Virtual Node Port).
- **F_Port:** F_Port is a Switchport that connects to the end device (N_Port). F refers to Fabric. FCoE calls this

VF_Port.

- **E_Port:** E_Port is a Switchport that connects to other Fibre Channel switches. E refers to Expansion. FCoE calls this VE_Port (Virtual Expansion Port).
- **TE_Port:** On Cisco Fibre Channel and FCoE switches, the capability to multiplex traffic of more than one Virtual SANs (VSANs) on the same physical link is called trunking. Therefore, an E_Port that carries traffic for more than one VSAN operates as TE_Port. FCoE calls this VTE_Port.
- **NPIV:** N_Port Identifier Virtualization is a feature that allows more than one N_Port login via the same F_Port.
- **NPV:** A Fibre Channel switch operating in NPV mode aggregates multiple F_Ports (connected to N_Ports on end devices) to NP_Ports (connected to F_Port or TF_Port on NPIV switch). An NPV switch does not run Fibre Channel services such as zoning and name service. It simply forwards traffic between F_Ports and NP_Ports. A common example of a switch operating in NPV mode is the Cisco UCS Fabric Interconnect running in end-host mode.
- **NP_Port:** NP_Port is a switchport on an NPV switch that connects to the NPIV switch. NP refers to Node Proxy. FCoE calls this VNP_Port.
- **TNP_Port:** Like a TE_Port, a TNP_Port is a NP_Port that carries traffic for more than one VSANs.
- **TF_Port:** An F_Port on a switch with NPIV feature that connects to a TNP_Port on an NPV switch is called a TF_Port. This port carries traffic for more than one VSANs.

Choice of Storage

The choice of storage depends on the application requirements. But this book is not intended to help you choose the type of storage. Rather, the aim is to educate users about using each storage type in a better way.

A holistic understanding of different types of storage is important because an application may use all types of storage together. For example, an application may use SAN storage arrays connected via Fibre Channel fabrics for providing high performance and low latency for transactional tasks while using cloud-based object storage for daily backups. Essentially, the same application may then result in congestion in Fibre Channel SAN as well as its cloud connection. A related case study is explained in Chapter 6, “Preventing Congestion in Fibre Channel Fabrics,” the section on *Case study 1 — A Bank Avoided Congestion by Traffic Segregation*.

Choice of Storage Network

The choice of a storage network typically depends on the following factors:

1. Application requirements.
2. Consistent and predictable performance with minimal delay and high throughput.
3. Affordability.
4. Operational Simplicity.
5. Ability to scale while retaining all other benefits.
6. Security.
7. Team Expertise.
8. Diagnostics and Troubleshooting capabilities.
9. Visibility and Analytics.
10. User adoption and Interoperability.

Although these are the most common factors in our experience, the choice differs for different organizations based on their requirements and constraints. This book aims to educate users about using each network type in a better way. The focus of the following chapters is more on operational simplicity, diagnostics and troubleshooting capabilities, visibility, and analytics. The other factors are outside the scope of this book.

Dedicated Versus Shared Networks for Storage Traffic

Fibre Channel fabrics are dedicated only to SCSI and NVMe block-storage traffic. Likewise, InfiniBand networks are dedicated only to RDMA traffic. Typically, these networks are not used for general-purpose TCP/IP traffic.

In contrast, an Ethernet network can be used for all kinds of traffic. Because Ethernet networks “can” be shared by storage and non-storage traffic, the big question is, “should” Ethernet networks be shared by storage and non-storage traffic?

Another sub-category of a shared network is a converged network. As mentioned earlier, a network that is configured for lossy and lossless traffic on the same network is called a converged network in this book. FCoE, RoCE, and RoCEv2 are the primary use cases for lossless traffic. The same shared network can also carry other lossy storage or non-storage traffic, such as iSCSI, NVMe/TCP, NFS, and SMB

The most significant benefit of sharing the same Ethernet network for storage and non-storage traffic is its affordability. Not only the cost savings for buying fewer switches, adapters, cables, and SFPs, but also reduced operational cost for managing fewer components during the life of this network.

On the other hand, managing congestion in shared storage networks is more difficult, especially when different teams manage the network and the storage infrastructure. A traditional difference in the basic mindset of ok-to-drop-packets for network teams versus not-ok-to-drop-packets for storage teams is well-known. Additionally, there are lesser-known details, such as despite enabling QoS, traffic in other classes can still affect storage traffic and the complexity of traffic monitoring separately for storage traffic and responding to component failures.

Another consideration is network capacity planning. For example, a host connected to a converged network at 10 GbE can assign 5 Gbps for storage traffic and 5 Gbps for other traffic using QoS. But what happens when the storage traffic needs 8 Gbps while other traffic still needs 5 Gbps? If this is just a single host, you can probably add another link but if all the hosts have a similar traffic demand, then should you be increasing the capacity of the existing network or should you be creating a dedicated network for storage traffic?

In such cases, building a dedicated storage network is a valid alternative. The hosts can be connected to two networks. First, for storage traffic, and second, for other traffic. This model is like separate LAN and SAN in data centers. But the difference is that now both are Ethernet networks.

A dedicated storage network is a middle ground for users who have already decided to use Ethernet for carrying the storage traffic. Although there are many more considerations, for the scope of this book, a dedicated storage network helps in much faster and easier detection and resolution of congestion. When the network is dedicated, all its capacity is available to storage traffic, and hence other traffic is not affecting the storage traffic performance. When a link is highly utilized and about to be congested, you know for sure that it is all because of storage traffic.

To summarize, this section does not make a case for moving from Fibre Channel to Ethernet or vice-versa. That topic is outside the scope of this book. However, if you have already decided to move to Ethernet, this section highlights the benefits of managing congestion with dedicated networks for storage traffic.

Common Questions on Storage Networks

This section covers some questions that we have been frequently asked and our responses to those questions.

Q: What is the difference between a network and a fabric?

This book uses fabric and network interchangeably as per the context without losing any relevance.

For some people, Fabric conveys the use of services that stitch the network devices more closely. For example, Fibre Channel fabrics use distributed services like zoning, name service, and state change service. Some other people use Fabrics to convey the availability of a single management tool for all the devices even in IP and Ethernet networks. This book does not redefine or intend to change your understanding of these terms.

Q: What's the difference between a Storage Area Network (SAN) and Storage Network?

For most people,

- SAN means Fibre Channel fabrics for transporting block-storage (SCSI and NVMe) traffic.
- IP SAN means the use of iSCSI or NVMe/TCP via a lossy Ethernet network for transporting block-storage traffic.
- Ethernet SAN means the use of FCoE, RoCE, and RoCEv2 via a lossless Ethernet network.

Additionally, networks that transport file and object storage traffic using protocols like NFS and SMB are also a type of storage network. But most people do not call these networks a SAN.

For this book, a storage network refers to any network that transports storage traffic regardless of its type—Block, File, or Object and handles "reading data" or "writing data" to a remote storage device. In other words, a "storage network" is a superset of SAN, at least in the scope of this book.

Q: Do storage networks have a role in the Cloud?

Yes. SaaS offerings from many organizations are powered by SAN storage. Block-based storage is quite common among cloud-native applications. Even the Container Attached Storage (CAS) has hooks into block storage using iSCSI and NVMe/TCP. In summary, most applications need access to remote storage and a storage network provides the connectivity regardless of public, private, or hybrid cloud.

This book considers a network as a "storage network" if it transports any type of storage traffic regardless of its type—Block, File, or Object. Because all types of cloud offerings have these storage types, the role of storage networks in the cloud is as deeply rooted as other components.

Q: Do storage networks have a role in container storage?

This book considers a network as a “storage network” if it transports any type of storage traffic regardless of its type—Block, File, or Object. All containers use at least one of these services to read and write data across multiple nodes. Additionally, leading Container Attached Storage (CAS) offer the use of iSCSI and NVMe/TCP for attaching remote volumes. For example, refer to OpenEBS Architecture. https://openebs.io/docs/main/concepts/architecture.

Congestion in Storage Networks — An Overview

A network becomes congested when traffic that enters the network (called ingress traffic) is more than the traffic that exits the network (called egress traffic). Excess traffic may be temporarily stored in the buffers of the network devices. But buffers are a finite resource, and they eventually fill up if the ingress traffic rate does not become equal to or lower than the egress traffic rate. When the buffers are full, the only option for a network port is to drop new incoming traffic. This loss of packets severely affects storage performance because retransmitting the lost packets or reinitiating the I/O operations can have a big overhead.

To avoid packet drops during buffer overrun, data center networks have two common approaches. The first approach is to initially let the packets be dropped and later retransmit the lost packets. This approach is used in lossy networks, such as Ethernet. The second approach is to pace the ingress traffic rate so that buffer overrun can be avoided. This approach is used in lossless networks, such as Fibre Channel and lossless Ethernet.

Both lossy and lossless networks become congested. However, the effect of congestion is significantly different because only the lossless networks experience congestion spreading.

For a better understanding of congestion spreading, lossless networks can be compared to highways. Unlike packet drops in lossy networks, vehicles do not vanish from highways regardless of how severe the congestion is. In an imaginary world, implementing the idea of lossy networks to the highways would require a magical approach to removing the excess vehicles, such as airlifting them back to their starting locations. But in the real world, when a highway is congested, the backpressure builds against traffic direction and eventually spreads to nearby cities and highways.

Congestion Spreading

In lossless networks, to prevent a buffer overrun situation, a receiver applies backpressure to its directly connected neighbor. As a result, the neighbor reduces its transmission rate. But the neighbor's buffers start filling up and to avoid its own buffer overrun, the neighbor applies backpressure to its upstream directly connected neighbor. This backpressure ultimately spreads to the traffic source, and it is an expected behavior. But, as a side effect, the flow control slows down all traffic that flows through the same path. This is because the flow control mechanisms in lossless networks do not have a flow-level scope. Hence, all traffic passing through the flow-controlled links is affected. This condition is like highways, where congestion between two cities affects all vehicles regardless of their source and destination.

Figure 1-14 explains congestion spreading in lossless networks. Traffic flows from the two targets to the four hosts via the shown paths. When the link to Host-1 becomes congested, Switch-1 cannot transmit to Host-1 as fast as traffic is being delivered (received) for it. This leads to increased buffer utilization on Switch-1. To avoid buffer overrun, Switch-1 reduces the traffic rate from Switch-3 and Switch-4 by applying backpressure. Consequently, buffer utilization on Switch-3 and Switch-4 increases, and to avoid their own buffer overrun, they apply backpressure to Switch-5 and Switch-6. Finally, Switch-5 and Switch-6 apply backpressure to Target-1 and Target-2. In this condition, the effect of congestion is not only limited to its source (Host-1 link), but rather congestion has spread towards the traffic source via all the links that apply hop-by-hop flow control. This condition is called congestion spreading.

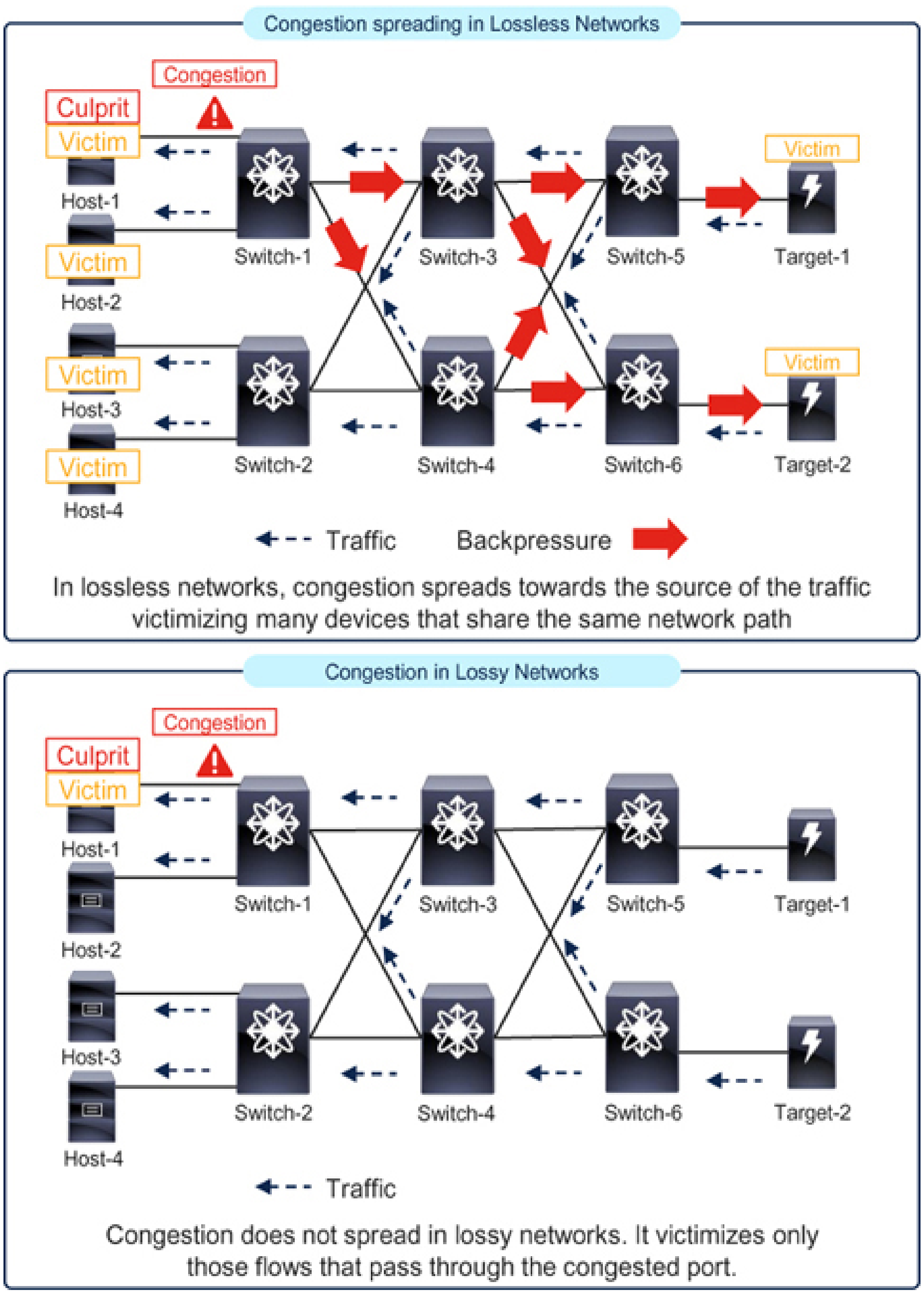

Congestion spreading in Lossless Networks
Congestion
Culprit
Victim
Host-1
Victim
Host-2
Victim
Host-3
Victim
Host-4
Switch-1
Switch-2
Switch-3
Switch-4
Switch-5
Switch-6
Victim
Target-1
Victim
Target-2
Traffic
Backpressure
In lossless networks, congestion spreads towards the source of the traffic victimizing many devices that share the same network path
Congestion in Lossy Networks
Congestion
Culprit
Victim
Host-1
Host-2
Host-3
Host-4
Switch-1
Switch-2
Switch-3
Switch-4
Switch-5
Switch-6
Target-1
Target-2
Traffic
Congestion does not spread in lossy networks. It victimizes only those flows that pass through the congested port.

Figure 1-14 *Congestion in lossless networks and lossy networks*

The effect of this congestion spreading results in victimizing Host-2, Host-3, and Host-4 because their paths from Target-1 and Target-2 are adversely affected.

Lossy networks do not experience congestion spreading because they drop the frames due to buffer overrun during congestion. As a result, the effect of congestion is limited only to the flows that pass through the congested port. In Figure 1-14, if Host-1 is congested, the effect is limited only to Host-1. Other hosts are not victimized because their paths are not affected by this congestion.

In summary, both lossy and lossless networks become congested. The difference is, however, only the lossless networks experience congestion spreading. Converged networks, which carry lossy and lossless traffic simultaneously, experience congestion spreading only for lossless traffic.

Therefore, congestion characteristics in Fibre Channel fabrics are similar to lossless (or converged) Ethernet networks. But TCP, which mostly runs via lossy networks, has different congestion characteristics.

This section provides a high-level overview of various causes and sources of congestion in storage networks. The following chapters further expand on these details based on the network type.

Causes of Congestion in Storage Networks

The cause of congestion in a storage network can be any of the following.

1. Congestion due to a slow or slow-drain end device.
2. Congestion due to over-utilization of a link.
3. Congestion due to bit errors.
4. Congestion due to the lack of buffers for the distance, frame size, and speed of a Link.

The following sub-sections provide a high-level overview of each of these.

Congestion Due to Slow End Devices

A slow end device has a slower processing rate as compared to the rate at which traffic is being delivered to it. For example, a host connected at 10 GbE and being sent traffic at 10 Gbps is a slow host if it can process only 5 Gbps.

- In lossy networks, this slow end device drops any excess traffic while the network remains unaware of this condition.
- In lossless networks, this slow end device uses hop-by-hop flow control to reduce the ingress traffic rate so that its buffers do not overrun, and frame drops are avoided. As mentioned, this condition results in congestion spreading to the traffic source victimizing many other end devices.

In Fibre Channel and FCoE fabrics, such a device is commonly known as a slow-drain device, and congestion caused by it is called slow-drain. “Credit stall” is a relatively newer term to refer to slow-drain, and it is also used in the Fibre Channel standards and Cisco MDS NX-OS. “PFC Storm” is another relatively newer term that has become famous among the users of lossless Ethernet networks, especially those that are used for RoCE. This book mostly uses “slow-drain”, and does not use “credit stall” and “PFC storm”.

Slow-drain Analogy — Traffic on a Highway

Congestion due to slow-drain in a lossless network can be compared to traffic congestion on a highway caused by internal congestion in a city. Not just the vehicles going to the congested city are affected, but the vehicles going to other non-congested cities are also slowed down because they share the same highway. In this analogy, the highway system is the network. The congested city is the slow-drain device (culprit). Vehicles going to other cities accessible via the same highway are the victims.

This analogy, however, does not apply to lossy networks because although packets are dropped during network congestion, vehicles do not vanish during highway congestion.

Reasons for Slow-Drain

An end device can become a slow-drain device because of a variety of reasons such as firmware bugs, adapter failure, performance problems, and overload.

In modern data centers, multiple virtual machines and containers share the same physical hosts and adapters. The behavior of the physical infrastructure depends on the application storage I/O profile. Moreover, although an application may not show any issues, multiple applications doing simultaneous I/O may result in a complex situation. Additionally, most workloads are dynamic. Applications may move to different hosts because of the dynamic scaling and high-availability policies. Under such cases, a host may become a slow(-drain) device only when a certain combination of applications runs on it or during specific hours of the day.

The following chapters bring more clarity on the reason for an end device to become a slow(-drain) device based on the type of the network and flow control. However, internal details of the end devices are outside the scope of the network and the scope of this book.

Congestion Due to Over-Utilization of a Link

Congestion due to over-utilization happens when a switchport is transmitting at the maximum speed yet it has more frames than can be transmitted on that link. For example, a switch receiving 40 Gbps of traffic on one or more ports to be transmitted on an egress port that is operating at 10 GbE.

- In lossy networks, any excess traffic is dropped on the switch.

- In lossless network, the switch invokes hop-by-hop flow control to reduce the collective traffic rate to match the egress traffic rate so that its buffers do not overrun, and packet drops are avoided. This type of congestion is frequently seen in Fibre Channel fabrics where hosts connect at slower speeds, such as 8GFC, and all-flash arrays connect at faster speeds, such as 32GFC. Like congestion due to slow-drain, this condition also results in congestion spreading to the traffic source victimizing many other devices. However, the source and the cause of congestion are different, as explained in detail in the following chapters.

Congestion due to over-utilization can happen anywhere in a network, such as host links, storage links, or ISLs.

This type of congestion is known by many names, such as:

- **Speed mismatch**: For example, when hosts are connected at slower speeds, such as 8GFC, and storage arrays are connected at faster speeds, such as 32GFC.
- **Bandwidth mismatch**: For example, when a host receives traffic from multiple storage ports simultaneously. There may not be a speed mismatch but aggregated traffic from multiple storage ports results in a bandwidth mismatch.
- **Over-subscription**: For example, when ISL (or the network core) is oversubscribed assuming that the end devices may not send traffic simultaneously. But there may be excessive over-subscription, for example, one ISL link for every 20 end devices instead of 5 end devices.
- **Lack of network capacity**: For example, when network links consistently report high utilization.

Congestion due to over-utilization Analogy — Traffic on a Highway

Congestion due to over-utilization can be compared to traffic congestion on a highway due to multiple lanes of traffic trying to go to a city that’s connected via a single lane. Although the destination city doesn’t have internal congestion, two lanes of traffic trying to enter this city via one lane leads to congestion back on the highway. As a result, not just the vehicles going to this city are affected, but the vehicles going to other cities are also slowed down because they share the same highway. In this analogy, the highway system is the network. The single lane of the destination city is an over-utilized link and vehicles going to other cities using the same highway are the victims.

Again, this analogy does not apply to lossy networks because although packets are dropped during network congestion, vehicles do not vanish during highway congestion.

Reasons for Over-utilization of Host Links

Congestion due to over-utilization of host links happens when hosts have requested more data from their targets than can be sent on the host link. This occurs with a combination of the following factors:

- Hosts connected at slower speeds, such as 8GFC or 10 GbE.
- Targets connected at faster speeds, such as 32GFC or 25 GbE.
- Faster storage technology of targets such as All-Flash arrays.
- Number of concurrent Read I/O operations.
- Spacing of concurrent Read I/O operations.
- Number of different targets the concurrent Read I/O operations are for.
- Read I/O operations requesting a large amount of data (large I/O sizes)

As long as one or more of these conditions are met, more data may be generated for the host than can be delivered to it.

Over-utilization of Host Links vs Storage Links

Congestion due to over-utilization is reported more commonly on the host links because of the refresh cycles and methodologies in data centers. While newer switches and All-

Flash arrays are deployed, the older hosts may not be refreshed at the same time. This leads to the following conditions:

- Faster link speeds of the All-Flash arrays, for example, 32GFC.
- The speeds of the hosts remain unchanged at slower speeds, for example, 8GFC.
- Faster response times of the All-Flash arrays.
- The ability of the All-Flash arrays to transmit frames at the full capacity of their links.
- A host continues to access volumes from many storage arrays resulting in many-to-one traffic patterns.

All these conditions collectively result in the over-utilization of the host links.

In contrast, congestion due to over-utilization of the storage links happens when many hosts, which may be connected at slower speeds, send traffic to a storage port at the same time. From the perspective of the network, over-utilization of the host link and storage link looks similar. The only difference is that the direction of congestion is reversed.

Role of All-Flash Arrays in Congestion Due to Over-utilization

Although congestion due to over-utilization is possible with non-flash and hybrid storage arrays, in Fibre Channel fabrics this issue was more commonly reported after deploying All-Flash arrays.

Multiple factors played together to make it a noticeable condition:

- **Faster link speed of All-Flash arrays**: While the storage media technology transitioned from spinning disks to all-flash, the Fibre Channel speeds transitioned to 16GFC and 32GFC, and thus, the newer All-Flash arrays provided faster front-end port speeds.
- **Fewer front-end ports on storage arrays**: Flash storage media has a smaller form-factor than spinning disks. As a

result, All-Flash Arrays have a compact form factor as compared to the earlier spinning-disk arrays. This resulted in fewer ports on storage arrays, which when combined with higher throughput resulted in increased per-port utilization.

- **Slower speeds of the hosts**: While newer All-Flash Arrays were deployed, the hosts were not upgraded at the same time, nor their link speed was increased. This inflated the speed mismatch in the fabric. Before the upgrade, hosts at 8GFC were doing read/write I/O with 8GFC storage ports. After the upgrade, the same hosts connected at 8GFC started doing I/O with 32GFC storage ports.
- **Improved internal architecture**: Many advancements in the internal architecture of the storage arrays, (e.g., the use of NVMe drives), made it possible to read and write faster to the back-end storage media within an array.

It’s not just one factor. As mentioned previously, a complex mix of multiple factors make congestion due to over-utilization more visible in production storage networks.

These factors collectively make it look like congestion due to over-utilization is related to All-Flash arrays only, but this issue was also seen with spinning-disk-based arrays, although it wasn’t as widely reported.

Consider an example to understand a condition in which congestion due to over-utilization was seen also with spinning disk arrays. Imagine that Host-1 connected at 8GFC initiates a 4-MB read I/O operation to Target-1, which is a storage array using spinning disks. The target starts reading data from the disks which involves a rotating motor and a mechanical head that must position itself to the sector on the disk where data is stored. Imagine that the initial 8 KB of data is written on adjacent sectors on the disk, but the rest of the data is written on sectors that are far or on a different disk. The mechanical movement of the head and the rotating motor involve a delay in positioning to the next sector. Meanwhile, the storage array may decide to send the already read 8 KB of data which results in approximately four full-size Fibre Channel frames.

As the target reads more data from the spinning disks, the next set of frames is sent without causing fabric-wide congestion.

The condition, however, would have been different had the data been stored on the spinning disks in sequential order. In such a case, the spinning motor and mechanical head don’t have to take a break to move to physically distant sectors. The data can be read as fast as the frame transmission rate on the front-end port. If this happens, the same host and the same target in the same fabric may cause congestion due to over-utilization.

History of Congestion Due to Link Over-utilization

Congestion in networks is as old as the networks themselves. It’s been a topic of interest within the research community since the early days. It is not surprising to observe congestion when a network port is expected to send more traffic than its capacity.

But in the production storage networks, this issue was not widely seen until the middle of the 2010 decade (around the year 2015). This is when the storage industry went through a historic transformation because of all-flash technology. When All-Flash Arrays were deployed in production storage networks, the storage links were utilized at their full capacity, a condition that was rare earlier. By this time, Fibre Channel had undergone multiple speed upgrades of 1GFC, 2GFC, 4GFC, 8GFC, and 16GFC. Most people followed the years of best practices that they had used while upgrading through many speed generations. But this time they saw a phenomenon that they had not seen earlier.

In the 2015-2016 timeframe, understanding and finding the reason for this type of congestion took a while. Although the effect on the fabric due to over-utilization of the edge links is similar to the congestion caused by a slow-drain device, both the detection of it and the reasons for it are different. Hence, both detecting and solving congestion due to over-utilization of the edge links requires a different approach.

Difference Between Over-Utilization and Over-Subscription

It’s important to understand that over-subscription doesn’t in itself cause congestion. Over-subscription is a network design concept, whereas over-utilization is the state of a link or port at an instant.

Most networks have some level of over-subscription to make them affordable. The basic assumption is that not all end devices send traffic at full capacity at the same time. Just because a link is over-subscribed, doesn’t necessarily mean that the link will be over-utilized. For example, at 6 AM, a host connected at 8GFC is sent traffic only at 1 Gbps by a 32GFC target port. However, at 9 AM, the same target may send traffic at 20 Gbps to that host, which results in congestion due to over-utilization of its 8GFC link. During this time, the over-subscription ratio (4:1) remains the same, but congestion is observed only at 9 AM. Because of this subtle difference, over-subscription doesn’t always cause congestion, although it is one of the contributing factors.

Defining Full Utilization vs High Utilization vs Over-Utilization

This section explains the theoretical definitions of full utilization, high utilization, and over-utilization.

A network port has a finite buffer space to temporarily store frames while it’s already transmitting another frame. This buffer space is managed by one or more queues. Frames are removed from these queues as the network port transmits the frames. At the same time, frames are added to these queues by other ports if this port is on a switch, and by upper layers if this port is on an end device.

First, let’s define the state of full utilization. While a frame is being transmitted, the port always transmits at full utilization. For example, when a 10 GbE port starts transmitting a 1500-byte frame, it transmits at full utilization for the next 1.2 microseconds, which is the time taken to transmit 12,000 bits of that frame at 10 Gbps. While a frame is being transmitted and the next frame is added to its queue, the port immediately starts transmitting the next frame. If this sequence repeats for a duration, such as 60 seconds, then the port achieves 100% or full utilization in those 60 seconds.

Next, let’s define the state of high utilization. If there is no frame in the queue after the transmission of the current frame and when adding the next frame to the queue is delayed, the port doesn’t transmit frames for a longer-than-minimum inter-frame gap, and therefore it operates at less than 100% utilization. Depending on this delay in adding the next frame to the queue (which can be thought of as idle time), the port may achieve low utilization, such as 10% or 20%, or high utilization such as 80% or 90%. The precise definition of high utilization depends on the admins of that environment. Some users consider 75% as high, whereas others wait until 90%. But overall, during the high utilization duration, such as 60 seconds, the port doesn’t transmit frames for some time while its queue is empty and until the next frame is added to the queue.

Finally, let’s define the state of over-utilization of a port. This is a condition in which a port is already transmitting at full utilization, yet more frames are added to its queue up to the extent that the queue becomes full. The port can’t transmit any faster because it’s already transmitting at 100% with a minimum inter-frame gap. Hence, when port utilization is measured for a duration of 60 seconds, the state of over-utilization looks the same as full utilization. However, only during over-utilization, the queue is full. As a result, new incoming frames are dropped in lossy networks and backpressure is applied in lossless networks. These results of over-utilization lead to performance degradation as explained in detail in the following chapters based on the type of the network—Fibre Channel, lossless Ethernet, and lossy Ethernet. So, over-utilization is really the combination of full utilization plus packet drops (lossy networks) or full utilization plus backpressure (lossless networks). This fact makes the basis for differentiating over-utilization from other congestion indications.

For now, the key point to understand is that a port in a state of full utilization without excessive frames waiting in its queue is an acceptable state because the port is achieving the full worth of its investment without causing congestion. However, practically, most storage networks don’t have a way to finely control the amount of traffic passing through a port. Therefore, although understanding the theoretical difference between these states of port utilization is important, more relevant is to understand their practical significance as explained next.

Monitoring Full Utilization vs High Utilization vs Over-Utilization

While monitoring utilization in production storage networks, when you find a port with high utilization, we recommend investigating it at the same severity as over-utilization. This section explains this recommendation.

Most devices report port utilization as an average value during an interval. Switches and the end devices don’t report the percent utilization directly. Rather, they report the accumulated value of bytes transferred in ingress and egress directions. To report port utilization in bytes per second (Bps) or bits per second (bps), the firmware of the device periodically polls the accumulated value at an interval, takes the difference from the previous value (delta), and divides it by the polling interval. For example, at 9:00 AM, a 10 GbE port reports bytes_rx as 500,666,777,888. At 9:01 AM, the bytes_rx become 560,666,777,888. The throughput of this port is calculated by ((560,666,777,888 – 500,666,777,888)/60), which is 1 GBps or 8 Gbps. For calculating percent utilization, the throughput is divided by the port speed. On a 10 GbE port, it results in 80% utilization. A remote monitoring platform involves similar steps but it’s a remote entity, and therefore often uses a longer polling interval.

This averaged value of port utilization misses traffic spikes during the polling interval. Even a 1-second polling interval, which is very aggressive and production environments at such a low granularity are almost non-existent, can easily miss multiple traffic spikes because NVMe and All-Flash Arrays have response times in microseconds. For example, consider a port that operates at 100% utilization for 100 ms and 0% utilization for the next 900 ms. The reported utilization during this 1-second interval is averaged to 10%, which is low and therefore may not raise any alarms. Yet, the 100% utilization during the initial 100 ms might be because of full utilization or even over-utilization states explained earlier. This is because a port transmits at 100% when its queue utilization is non-zero (full utilization) and even when the queue is full (over-utilization). Hence, this duration of 100% utilization might be the condition of over-utilization leading to packet drops in lossy networks or congestion spreading in lossless networks. From the perspective of detecting congestion, this would come out to be a non-trivial scenario to explain because the reported utilization is only 10% in 1 second, yet this port is the source of congestion. Monitoring and detecting this non-obvious source of congestion becomes even more challenging as the scale of the environment grows.

Because of this reason, for all practical purposes, we recommend treating any occurrence of high-utilization and full-utilization the same as over-utilization. In most cases that we have worked on, a highly utilized port, such as the one that reports more than 80% or 90% utilization over 10 seconds, when investigated, is found to be the source and cause of congestion. This is because storage traffic is bursty. A 10-second period of high utilization often involves a few seconds of over-utilization, a few seconds of high utilization, and even low utilization. The state of over-utilization can be verified by other counters, such as packet drops in lossy networks and TxWait (the duration for which a port could not transmit) in lossless networks. These counters are explained in the following chapters. But as mentioned, in most cases, we have found highly utilized ports to also have symptoms of over-utilization.

To summarize, theoretically, full utilization, high utilization, and over-utilization have a clear differentiation. However, practically in our experience, most occurrences of full utilization and high utilization have also shown some duration of over-utilization. If the devices in a network can differentiate between these states, use these features (such as tx-datarate-burst on Cisco MDS switches and microburst detection on Cisco Nexus switches) to their full potential. But start the investigation by treating the high and full utilization occurrences at the same severity as over-utilization unless proven otherwise. This approach saves time in detecting congestion and finding a solution.

Bit Errors on a Link

When bit stream is exchanged over a network, some bits may be altered resulting in bit errors. These bits are altered because of the following reasons:

- Accumulated dust on cable end points, SFP, or patch panels.
- Faulty cable.
- Faulty SFP.
- Loose cable connection.
- Loose SFP connection.
- Environmental issues interfering with data transmission such as temperature and humidity.

This is not an exhaustive list.

In lossy networks, the effect of bit errors is limited to the flows that pass through the link causing the bit errors.

In lossless networks, bit errors cause may additionally interfere with the flow control mechanism resulting in congestion or worsening existing congestion. The following chapters bring more clarity to this aspect based on the flow control mechanism such as B2B flow control and PFC. This congestion spreads to the traffic source victimizing many other devices that share the same network path.

Frames that are corrupted due to bit errors are dropped. Hence, besides interfering with the flow control mechanisms, bit errors lead to significant performance degradation for applications.

In Fibre Channel fabrics, if bit errors corrupt even one frame, that entire I/O operation (or Exchange) is reinitiated. This behavior is more evident as the amount of data transferred by an I/O operation (called I/O size) increases. Consider a 4-MB read I/O operation, which needs approximately 2000 full-size Fibre Channel frames to transfer data. A bit error in just one frame results in reinitiating this entire I/O operation. A 16GFC port, which can achieve 1600 MB/s of I/O throughput, can transfer 400 such 4-MB read I/O operations per second (1600/4). This means 400 bit errors per second can bring down the I/O throughput to zero. From the perspective of the 16GFC port, which has a bit rate of 14.025 Gbps, 400 bit errors corrupt only 0.0000028% bits per second. In this case, the network port reports throughput, but the application lacks the goodput.

These details of Fibre Channel speeds, bit rate, and data rate are explained in Chapter 2. For now, just remember that bit errors have two significant effects in Fibre Channel and FCoE fabrics. First, interference with the flow control resulting in congestion or worsening existing congestion. Second, a dramatic effect on performance due to reinitiating the entire I/O operations.

TCP can retransmit only the lost packet (lost due to bit error or other reasons), hence typically the entire I/O operation is not reinitiated. However, many lost packets result in a TCP connection to reduce its transmission rate. These details are explained in Chapter 8.

Lack of Buffers for the Distance, Frame Size, and Speed of a Link

In lossless networks, the required number of buffers on a port increase as the length of its link increases, the speed increases and the average frame size decreases.

In Fibre Channel fabrics, the lack of buffers reduces the amount of traffic that can be sent on a port. This results in lower link utilization and congestion spreading. In lossless Ethernet networks with PFC, lack of sufficient buffers may result in packet drops (which goes against the basic principle of a lossless network) or lower link utilization. The following chapters bring more clarity to this aspect based on the flow control mechanism such as B2B flow control and PFC.

Lossy networks do not spread congestion. In some cases, when traffic crosses from a lossless to a lossy network, congestion in the lossy network may spread to the lossless network, for example when transporting Fibre Channel traffic via FCIP. Refer to Chapter 8, section on *Fibre Channel over TCP/IP (FCIP)* for more details on this scenario.

TCP, that typically runs over lossy networks, still has dependency on enhanced buffering for long distance links, otherwise the throughput may not reach the expected rate. Refer to Chapter 8 for more details.

Source of Congestion in Storage Networks

All network entities (end devices, links, and switches) are susceptible to congestion, although their causes may be different. The previous section explains these causes (why congestion happens). This section explains the source of congestion (where congestion happens) in storage networks.

Congestion from End Devices

End devices can cause congestion due to

- Slow-drain.
- Over-utilization of their link.
- Bit Errors on their link.

In general, edge links are short, and thus, do not cause congestion due to their excessive length.

Congestion on ISLs

ISLs may get congested because of the following reasons:

- Over-utilization of an ISL: Multiple end devices sending traffic at the same time may cause over-utilization on an ISL.
- Bit errors on the ISL.
- Insufficient buffers for the length, speed, and frame size of the ISL.

In lossless networks, it is less common for ISLs to be the source of congestion. ISLs typically reflect congestion that spreads to them, but mostly the source is the end devices.

Congestion within Switches

In general, a switch is not a source of congestion. Switches may have hundreds of ports and they are expected to receive, process, and forward frames from any port to any port at a maximum data rate without any restrictions. In lossless networks, a switch typically reflects congestion from the egress port to the ingress port. In lossy networks, a switch drops frames when the egress port is congested.

In rare cases, if the switch itself causes congestion, it should be considered defective, and a solution should be worked with its manufacturer. These details are outside the scope of this book.

Common Questions About Congestion in Storage Networks

This section explains common questions that we have been asked about congestion in storage networks and our answers to them.

Q: What is backpressure?

Backpressure is a generic term that refers to flow control against traffic direction. It is the result of congestion. For example, Fibre Channel B2B flow control and Ethernet PFC apply backpressure towards traffic source on the link between directly connected devices. TCP end-to-end flow control applies backpressure from the traffic destination to the source.

Additionally, devices have internal backpressure mechanisms among different layers of a stack. Backpressure mechanisms also exist among ports of a switch to inform the ingress ports about congestion on egress ports. Such internal backpressure mechanisms are implementation dependent and may not be standardized.

Q: What are traffic burst and microburst?

A traffic burst is a condition when a network port observes high, full, or over-utilization for a duration. Microburst refers to a traffic burst that is sustained for only small durations, such as microseconds. Data center traffic is bursty, although different traffic types may have a different duration of the burst. For example, in Hadoop environments, multiple worker nodes send traffic to the primary node at the same time. This is known to cause traffic bursts that last a few seconds or even minutes. Another well-known example is backup jobs that move large data sets resulting in traffic bursts for hours. In contrast, all-flash storage arrays respond to SCSI and NVMe I/O operations in microseconds transmitting the frames at line rate, resulting in microbursts.

Q: Isn’t increasing network capacity the ultimate solution to network congestion?

Increasing network capacity is the solution when a lack of network capacity is the root cause of congestion. There are many more reasons for network congestion and in those cases, increasing network capacity does not solve this problem. For example, in lossless networks, such as Fibre Channel, congestion due to slow-drain happens even when network capacity is not a limiting factor.

Additionally, consider the following points.

1. Applying the solution involves practical challenges of production environments which may seem trivial until you experience them. First, finding the cause and the source of congestion may take a long time. This is important because you need to know where to add the capacity. Congestion detection and troubleshooting methodology explained in this book help with this. Then,

upgrading the network links may not be immediately possible because funds to buy new components may not be readily available or the shipment of the components may be delayed. Further, users must find a change window, which typically requires coordination among various teams. Especially, if the upgrade is needed on the hosts that run business-critical applications, approvals may be delayed by weeks or even months. During this long delay, allowing the network congestion to victimize the applications is unacceptable. Hence, the need for congestion prevention mechanisms in addition to an upgrade. This reality is not very different from the congested roads in large cities across the world. Adding more lanes or newer roads seems like a simple solution but this construction is expensive and takes a long time. Hence, many cities use other affordable ways to handle congestion immediately.

2. Another practical challenge is with legacy devices and applications that are still business-critical but running in maintenance mode with little support. If these devices do not come up after making the change, the result may be a loss of revenue for the organizations and extended work hours for the administrators to fix these problems.

3. Storage traffic is bursty. Traditional approaches to detecting network utilization may not capture all events of high or over-utilization. This phenomenon is explained earlier in the section on *Monitoring Full Utilization vs High Utilization vs Over-Utilization*. There have been many instances when the utilization wasn't high, yet the network was congested. For example, monitoring network utilization at 5-minute granularity does not capture short traffic bursts. Even monitoring the network utilization at 1-minute granularity may not capture all the events when the all-flash and NVMe storage devices can respond to the I/O operations within hundreds of microseconds. In other words, the network utilization seems low but in reality, the links are being over-utilized for shorter durations. This phenomenon often gets

undetected. Consequently, a problem that is not even known cannot be solved.

Q: I was told that RoCEv2 does not suffer from slow-drain like Fibre Channel. Is this correct?

All lossless networks have similar characteristics of congestion spreading because of their use of hop-by-hop flow control. If RoCEv2 uses a lossless Ethernet network, the effect of network congestion is the same as that of a Fibre Channel fabric. Typically, Fibre Channel users refer to congestion as slow-drain, whereas PFC storm has emerged as a common term among the RoCEv2 users. The names might be different, but the symptoms and their effect on the networks are the same.

Q: Is slow-drain the same as PFC Storm?

PFC Storm may cause slow-drain. Both terms refer to congestion in lossless networks. Slow-drain is common among Fibre Channel users, whereas PFC Storm has gained popularity among RoCE and RoCEv2 (lossless Ethernet) users.

Both are funny terms. Imagine the users who try to understand slow-drain by doing a web search, but instead of network congestion, they are presented with plumbing problems in their bathrooms and kitchens.

Both terms convey little details. It would be much better to call them "congestion due to over-utilization" or "congestion due to a slow device" instead of slow-drain or PFC storm.

Additionally, both terms are confusing. Congestion spreads quite "fast" with "slow"-drain. With PFC storm, "congestion severity" may be higher even when the "storm severity" is lower. Also, the term 'storm' does not convey reality. As explained in detail in Chapter 7, the rate of PFC Pause frames does not need to reach 'storm' levels to cause an almost complete throttling of traffic.

Chapter 2 defines the technical meaning of slow-drain in Fibre Channel fabrics. Chapter 7 re-defines slow-drain in lossless

Ethernet networks. This book does not use the term PFC storm except when necessary.

Q: Would moving to the cloud eliminate congestion in storage networks?

It depends on how you define the Cloud.

If moving an application to a public cloud, you just pay for it, and the cloud provider will detect, troubleshoot, and prevent congestion in their networks so that your application performance is acceptable.

If you want to offer a cloud-like experience using on-premises infrastructure (called private cloud), then it is your job (or someone in your organization) to detect, troubleshoot, and prevent congestion in your networks so that your application performance is acceptable.

A similar analogy is renting a car. The renter doesn’t do maintenance, although the car still needs maintenance, which is done by the rental car company. The renter just pays an all-inclusive price.

Q: Would moving to HCI or SDS eliminate congestion in storage networks?

It depends on the application and how much traffic it sends on the network. It would be inaccurate to assume that simply moving from a SAN or NAS architecture to HCI or SDS eliminates congestion. As explained earlier, all these architectures result in network traffic using different transports, and when there is traffic, there is a potential for congestion.

NVMe over Fabrics

As previously mentioned, NVMe provides efficient block-level access to the storage media for unleashing its full potential. The benefits of NVMe can also be extended over a network, collectively called NVMe over Fabrics (NVMe-oF).

Based on the network type, NVMe-oF has the following variants.

- NVMe over Fibre Channel (NVMe/FC)
- NVMe over TCP (NVMe/TCP)
- NVMe over RDMA-capable protocol, which can be NVMe/RoCE, NVMe/RoCEv2, NVMe/iWARP, and NVMe/InfiniBand.

For all these variants, the end devices abstract the NVMe-oF details before sending the packets on the network. Consequently, the network typically remains unaware that the packets are carrying NVMe operations, and hence, congestion detection, troubleshooting, and prevention remain the same regardless of the use of SCSI or NVMe.

NVMe over Fabrics is a relatively newer technology and production deployments are at an early stage at the time of this writing. This phase of technology often becomes the seeding ground for myths because the users have limited experience of their own and they are often misled by its possibilities versus its implementation on commercially available devices.

This section explains some common questions that we have been asked on this topic and our answers to them.

Q: I have heard NVMe supports 64K queues, each with 64K commands. How can I be ready for it?

Users do not need to do much about it. These details are abstracted by the product developers and vendors.

More importantly, just because NVMe specification supports 64K queues each with 64K commands, doesn't mean you are going to use them. NVMe has many forward-looking provisions to keep it relevant for the long-term future. One of them is the use of 64K queues, each with 64K outstanding commands. But these are theoretical limits. A typical implementation may have as many queue pairs as the number of CPU cores, which are nowhere close to 64K. Overall, be clear about the difference between a technology specification and its implementation on commercially available devices.

Q: Doesn't NVMe have mechanisms to control network congestion?

No.

NVMe relies on the network (Fibre Channel, TCP, or RDMA-capable protocol) for handling network congestion. NVMe defines a flow-control mechanism, but its purpose is to avoid the submission and completion queues becoming full on the host and the controller. This is different from network flow control and congestion control. Moreover, NVMe flow control is optional with NVMe over Fabrics, which means products in your environment may not even support or enable these mechanisms.

Q: I built a new environment with NVMe over Fabrics, but the network throughput did not increase as compared to earlier. Why?

Consider the following points.

1. Throughput depends on the application requirements. NVMe, NVMe over Fabrics, or the network itself does not generate extra traffic. If the applications remain the same, their throughput also remains the same regardless of how they access their storage.

2. NVMe and NVMe over Fabrics may result in increased network throughput only if the earlier stack (probably SCSI-based) was a bottleneck.

3. You should also consider how you measure the network throughput. If measured at a 5-minutes granularity, you are not seeing traffic burst at a microsecond scale. As mentioned, NVMe over fabrics may not generate more data but it can transfer the same data in a much shorter duration. For example, without NVMe-oF, the data may be transferred in 1 second. With NVMe-oF, the same amount of data may be transferred in 100 milliseconds. Hence, the throughput during this 100-millisecond duration increases and possibly also results in over-utilization of the links. But the “reported throughput” at 5-minute granularity is the same with or without NVMe, although the “real throughput” significantly changed, which could not be noticed.

Q: What effects does NVMe over Fabrics have on network congestion?

All environments are different; hence a general answer is not possible.

The high amount of parallelism of NVMe may increase network utilization and traffic bursts. As a result, in general, network congestion may remain the same or increase with NVMe over Fabrics.

Network congestion may remain the same in those environments where applications were already getting the maximum storage performance and were not pushing the limits of the storage networks and the various layers within the end devices.

In contrast, network congestion may increase in those environments where the storage devices can read data quickly and send the response at the maximum rate of their links. This leads to increased network utilization resulting in congestion due to over-utilization. Such conditions have been commonly reported over the last many years with all-flash arrays, which did not use NVMe. As the usage of NVMe storage arrays increases, the instances and severity of congestion issues may increase further.

Q: Someone told me that congestion in their networks vanished after upgrading to NVMe over Fabrics. Is that possible?

Possible but not necessarily expected. Even we know a few users who upgraded to NVMe over Fabrics by deploying all new infrastructure because its support is available only on newer devices. Using high-capacity hosts, faster storage arrays, and most importantly, higher network speeds eliminated many reasons for congestion, although we cannot determine if this success can be attributed to NVMe over Fabrics itself.

Q: Is building a dedicated network for NVMe over Fabrics better for congestion management?

Yes, if you can afford it. The decision of building a dedicated versus shared network for storage traffic depends on many factors, congestion management is just one key consideration.

Refer to the section on *Dedicated Versus Shared Networks for Storage Traffic* for additional details.

Quality of Service (QoS)

QoS is a set of mechanisms for providing different quality of service to different types of traffic to meet business requirements. For example, on the Internet, prioritizing an emergency voice call over a file download traffic and within a data center, providing a minimum bandwidth guarantee to storage traffic.

The core of QoS functions is applied on the network switches and these functions typically get activated only during congestion.

In summary, using and configuring QoS has the following steps:

1. **Classification**: The first step is to identify the traffic and then classify it into different traffic classes. For traffic classification, the Fibre Channel header has a 1-byte Priority/CS_CTL field, the Ethernet header has a 3-bit PCP/CoS field, and the IP header has a 6-bit DSCP field (Figure 1-6).
2. **Limit the excess ingress traffic (Policing)**: Drop any excess traffic using a mechanism called Policing.
3. **Re-classification**: If the classification based on the ingress traffic is not enough or not trustworthy, a switch can re-classify the traffic as per the business requirements.
4. **Queuing**: Traffic in different classes is assigned to different queues.
5. **Active Queue Management**: The packets in a queue are managed (dropped or marked) as per an Active Queue Management scheme such as Weighted Random Early Detection (WRED).
6. **Scheduling**: Packets from multiple queues are transmitted on a port using a scheduling mechanism, such

as Round Robin (RR), Weighted Round Robin (WRR), or Deficit Weighted Round Robin (DWRR).

7. **Limit the egress traffic (Shaping)**: Limit the egress traffic rate by buffering a limiting amount of excess traffic using a mechanism called Shaping.

A notable use of QoS is when an Ethernet network is shared (converged) for lossless (FCoE, RoCE, RoCEv2) and lossy traffic. The lossless traffic is classified using the Ethernet CoS value or IP DSCP value, assigned to a no-drop queue, and given a minimum bandwidth guarantee. These details are explained in Chapter 7.

Sources of Delay in a Network

The following are the sources of delay when packets are sent via a network.

Forwarding Delay

Forwarding delay is the time taken to forward a packet from an ingress port to an egress port on a switch. It can range from hundreds of nanoseconds to a few microseconds based on the type of switch. This variation in forwarding delay is because of different architectures that have their trade-offs. Refer to Chapter 2, the section on *Store-and-Forward vs Cut-Through Switching* for related details. Forwarding delay is often advertised as port-to-port switching latency by the vendors. It does not include queuing delay, which is explained shortly.

The important point to remember is that the forwarding delay is very small compared to the end-to-end completion times of the I/O operations that are exchanged over a network. Also, when a network is operational, the forwarding delay remains fixed and cannot be changed by the users.

Propagation Delay

Propagation delay is the time taken by a signal to travel the length of a link. It is defined by the speed of light, which is approximately 200,000 km/s in optical media (and 300,000 km/s in vacuum). A simple way to remember it is that a signal travels 1 meter in 5 nanoseconds. In other words, a 100-meter cable causes a delay of 500 nanoseconds.

In Fibre Channel networks, propagation delay is an important factor for calculating the required number of B2B credits. Refer to Chapter 2, the section on *Detailed Explanation for the Required B2B Credits on an FC port*. Likewise, in lossless Ethernet networks, propagation delay is used for calculating the pause thresholds and headroom. Refer to Chapter 7, the section on *Pause Thresholds for Long Distance Links* for more details.

In TCP/IP networks, the end devices account for the propagation delay to calculate Round-Trip Time (RTT), Smoothed Round-Trip Time (SRTT), and Retransmission Timeout (RTO) by default, and no user action is needed.

For most intra-data center links that are shorter than a few hundred meters, the propagation delay is very small as compared to the end-to-end completion times of the I/O operations that are exchanged over a network. It remains fixed and cannot be changed by the users.

Serialization Delay

Serialization delay is the time taken to transmit a packet on the wire by a network port. It depends on the link speed and packet size. For example, for transmitting a 1500-byte frame, a 10 GbE port takes approximately 1.2 microseconds ((1500 x 8) bits / 10 Gbps). A 100 GbE port can transmit the same packet in just 120 nanoseconds.

In general, you should always upgrade network links to the fastest possible speeds not just to reduce the serialization delay, but also to increase the link capacity (bandwidth). However, after a link is operational, the serialization delay remains fixed and cannot be changed by the users.

Queuing Delay

Queuing delay is the time spent by a packet in the queues of a network port waiting for its turn while the other packets ahead of this packet are transmitted. This is variable. Note that forwarding delay and queuing delay are caused within a switch, but they are separate. As previously mentioned, users cannot change forwarding delay, propagation delay, and serialization delay while a network is operational, whereas queuing delay can be eliminated or minimized. Hence, it needs special attention. Various mechanisms for minimizing queuing delay are explained in Chapter 8.

Common Questions on QoS in Storage Networks

An ample amount of educational content is already written on QoS, some of that is available in the references section. The focus on QoS in this book is limited only to the relevant details that are useful in congestion management in storage networks.

This section covers some frequently asked questions, especially for two types of users. First, those who have a background in Fibre Channel but may not have experience with QoS in IP/Ethernet networks. Second, those who have experience in QoS but may not have experience in handling storage traffic.

Q: Why do network devices need buffers?

Network devices need buffers to store packets momentarily. Imagine a coffee shop that can make ten coffees per minute. If five more customers arrive during that minute, they can be in the waiting area instead of standing outside or going away. That waiting area is like a buffer.

At least some buffers on a port are needed to maintain full utilization because the port can immediately start transmitting the next packet from the buffers.

More buffers can absorb traffic bursts. In lossless networks, this prevents the spreading of congestion that lasts only a small duration, such as microseconds. In lossy networks, this prevents packet drops, but may also increase latency and delayed action by other layers, such as TCP. These details are explained in the following chapters.

In lossless networks, additional buffers are needed to achieve full utilization for the speed, length, and average frame size on a link.

Q: What is the difference between buffers and queues?

Queues are created to manage the buffer space. In the same example of the coffee shop, the five customers can wait in a queue so that they get coffee in the same order as they arrived. If two counters are open, these customers can be served in two queues, or a customer-in-hurry can be served sooner via a priority queue.

Q: What is the difference between buffers, pause buffers, and B2B credits?

For most practical purposes, they are the same.

In Fibre Channel fabrics, buffers are managed by B2B credits.

In lossless Ethernet, they are called ingress buffers with pause and resume (or unpause) thresholds. The term ‘pause buffers’ is unhelpful or even misleading because pauses are not buffers.

Refer to Chapter 7 for differences between Pause frames and B2B credits.

Q: While referring to buffers, why the queue is a common term in IP/Ethernet networks but not in Fibre Channel fabrics?

Probably because most IP/Ethernet networks create multiple queues to provide QoS to different types of traffic. There are related concepts like queuing delay, queue scheduling, and Active Queue Management.

In Fibre Channel fabrics, because QoS is rarely used, most users do not deal with the same queueing concepts, hence the journey from B2B credits to queues never started, although internally the devices still use queues for frame forwarding. Fibre Channel switches, such as Cisco MDS, maintain per-destination queues in the ingress interface to prevent head-of-line blocking during periods of minor congestion. Refer to Chapter 2 for these architectural details.

Q: What are common myths about using QoS in Storage Networks?

The following are common myths about using QoS in storage networks.

1. **Myth** — Congestion in a traffic class does not affect traffic in other classes.

   Reality — It is a common misconception that congestion in a traffic class does not affect other classes. It depends on how a problem is approached. Refer to Chapter 7, the section on *Effect of Lossy Traffic on no-drop Class*, and the related case studies for more details.

2. **Myth** — QoS configuration is needed only in shared storage networks.

   Reality — QoS configuration is mandatory for enabling PFC in Ethernet networks. Even if the network is dedicated to storage traffic, notifying the end devices about congestion requires QoS configuration via Explicit Congestion Notification (ECN) in TCP storage networks and for RoCEv2 Congestion Management (RCM). Refer to Chapter 7 and Chapter 8 for more details.

3. **Myth** — Storage traffic should be prioritized

   Reality — This statement conveys different meanings to different users. The word—priority—has a specific use in QoS, which is a lot more than its dictionary definition. In QoS terms, priority traffic is assigned to a priority queue. Packets from this queue are transmitted before packets from other queues. However, the priority queue is also limited in bandwidth. This bandwidth limit may or may not be apt for storage traffic based on the application requirements.

4. **Myth** — Assigning bandwidth to storage traffic limits its throughput to the configured value

   Reality — The QoS **bandwidth** command typically assigns a minimum bandwidth guarantee to a traffic class in the presence of other traffic (under congestion). In the absence of other traffic, this traffic class can consume all the available capacity of the link. In other words, **bandwidth** command is a lower limit, not an upper limit.

5. **Myth** — No need to understand QoS configuration when using an automation platform or a Software-Defined Networking (SDN) solution.

   Reality — Detecting and troubleshooting congestion still requires a thorough understanding of QoS concepts and configuration (such as the Modular QoS CLI (MQC)), even though the initial configuration is simplified by an automation or SDN solution.

Q: Why is QoS not commonly used in Fibre Channel fabrics?

The answer is in the basics of how QoS works. Consider the following points:

1. **All traffic is priority traffic in Fibre Channel**: When QoS prioritizes a traffic class, it also means that another traffic class is de-prioritized. This approach works well in a campus network to prioritize a voice call and de-prioritize file transfer or even in a data center network to prioritize traffic for a business-critical application and de-prioritize traffic for an internal application. But this approach of traffic classification is unacceptable in Fibre Channel because it transports block-storage traffic for mission-critical applications and all this traffic is equally important.
2. **Packet Drops**: Many QoS mechanisms are based on dropping packets. For example,

   a. Policing limits the ingress traffic by dropping packets.

   b. Shaping leads to buffering a limited amount of traffic, but excess traffic is still dropped.

   c. The Active Queue Management mechanism starts dropping packets early and still drops new incoming packets when queues are full.

Dropping packets is generally unacceptable in Fibre channel fabrics and hence, these mechanisms are not used.

3. **QoS does not avoid congestion in lossless networks**: Because dropping packets is unacceptable, the only other option is to use a hop-by-hop flow control to slow down the directly connected sender. Even if QoS is used in Fibre Channel, congestion would still spread to the traffic source in each traffic class separately. This behavior is common in converged Ethernet networks where the congestion spreads to the source of traffic in the no-drop class.
4. **Dual-Fabric design**: Typically, the same end devices (hosts and storage arrays) connect via two redundant Fibre Channel fabrics. In addition to the benefit of high availability, the active-active fabrics provide ample network capacity. In other words, the network is over-provisioned and most QoS functions may not be invoked, even if configured.

Q: Which is better for storage traffic in Ethernet networks — Policing or Shaping?

Neither. Both limit traffic rates, although slightly differently. Policing drops excess traffic, which results in retransmitting the lost packets or re-initiating the entire operation based on the upper layers. In contrast, shaping leads to buffering a limited amount of excess traffic, which increases delay and variance in delay, and still drops excess traffic. Both lead to poor performance for storage traffic.

The aim should be to have enough network capacity such that neither policing nor shaping is needed.

Q: What is the difference between Priority and Bandwidth in the context of QoS?

While using multiple egress queues on a network port, packets are sent from the priority queue before the packets from other queues. This scheme works great for delay-sensitive traffic, but it can starve traffic in other queues. To avoid this issue, a priority queue is assigned a maximum bandwidth limit, such as 20% of the link capacity.

Other queues are assigned a minimum bandwidth guarantee, such as 80% of the link capacity, but this is not a maximum limit.

The important point to remember is that these limits are enforced only in the presence of other traffic (under congestion). When there is no traffic in the priority queue, the other queue can consume up to 100% capacity. Likewise, when there is no traffic in another queue, the priority queue can consume up to 100% capacity. When there is traffic in both queues, 20% of traffic in the priority queue is transmitted with low delay, but its bandwidth is limited to 20%. The 80% of traffic in other queues is transmitted with relatively higher delay, but it gets at least 80% capacity.

Summary

This chapter briefly explains the types of storage, storage protocols, their transports, and networks in a data center. Even experienced users may be overwhelmed by all these options. However, for the scope of this book, just remember that the details of the protocol stack are abstracted by the end devices before sending packets on the networks. These packets are sent to their destinations using the standard forwarding function of the network.

The network itself can be of two types. First, lossless networks with hop-by-hop flow control such as Fibre Channel and Lossless (or converged) Ethernet. Second, lossy Ethernet without hop-by-hop flow control using the transport of TCP with built-in flow control and congestion control. Congestion in lossless networks spreads toward the source resulting in slowing down all traffic regardless of its destination. Fibre Channel users refer to this type of congestion as slow-drain, whereas PFC storm is a common term among RoCE (lossless Ethernet) users. In contrast, TCP acts at a per-flow level by moving the congestion point to the traffic sender. Regardless of these different approaches, the aim should be to find the root cause of congestion and solve it.

This chapter also clarifies high-level concepts about NVMe over fabrics, Quality of Service (QoS), and congestion management in storage networks.

Finally, this chapter explains the commonly asked questions that we have been asked and our responses to those questions.

References

- NVMe, and NVMe over Fabrics Deep Dive - BRKDCN-2494 – Cisco Live San Diego 2019.
- The Networking Implications of NVMe over Fabrics (NVMe-oF) - BRKDCN-2729 – Cisco Live San Diego 2019.
- DCNM SAN Insights - Next Generation Network Visibility - BRKDCN-2271. Cisco Live 2019, San Diego.
- DCNM SAN Insights - Next Generation Network Visibility - BRKDCN-3645. Cisco Live 2022, Las Vegas.
- Detecting, Alerting, Identifying and Preventing, SAN Congestion - BRKDCN-3241, Cisco Live 2022, Las Vegas.
- SAN Congestion! Understanding, Troubleshooting, Mitigating in a Cisco Fabric - BRKSAN-3446, Cisco Live 2017, Las Vegas.
- IP Telephony Self-Study: Cisco QOS Exam Certification Guide, Second Edition. Cisco Press. ISBN: 1-58720-124-0
- End-to-End QoS Network Design: Quality of Service for Rich-Media & Cloud Networks (Networking Technology) 2nd Edition. Cisco Press. ISBN-13: 978-1587143694
- Internet Assigned Numbers Authority Service Name and Transport Protocol Port Number Registry:

https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt

- The Addition of Explicit Congestion Notification (ECN) to IP – RFC 3168
- INCITS 515 SCSI Architecture Model - 5 (SAM-5) Revision 11
- INCITS 502 SCSI Primary Commands - 5 (SPC-5) Revision 10
- INCITS 506 SCSI Block Commands - 4 (SBC-4) Revision 10
- INCITS 536 Zoned Block Commands (ZBC)
- INCITS 534 Serial Attached SCSI - 4 (SAS-4)
- INCITS 538 SAS Protocol Layer - 4 (SPL-4) Revision 08
- ANSI INCITS 481-2011 Fibre Channel Protocol for SCSI
- IEEE 803.2-2018, IEEE Standard for Ethernet
- INCITS 488-2016, FC-FS-4, Fibre Channel Framing and Signaling - 4
- INCITS 545-2018, FC-FS-5, Fibre Channel Framing and Signaling - 5
- INCITS 512-2015, FC-PI-6, Fibre Channel Physical Interfaces - 6
- INCITS 533-2016, FC-PI-6P, Fibre Channel Physical Interfaces - 6P
- INCITS 479-2011, FC-PI-5, Fibre Channel Physical Interfaces – 5
- NVM Express RDMA Transport Specification, Revision 1.0. Available from https://www.nvmexpress.org
- NVM Express TCP Transport Specification, Revision 1.0. Available from https://www.nvmexpress.org.
- INCITS 540 Fibre Channel – Non-Volatile Memory Express (FC-NVMe)

- NVM Express Base specification, Revision 2.0. Available from http://www.nvmexpress.org.
- 801.Qbb Priority-based Flow Control: http://www.ieee802.org/1/pages/802.1bb.html
- I/O Consolidation in the Data Center Cisco Press (ISBN-13: 978-1587058882)
- Supplement to InfiniBand^{TM} Architecture Specification Volume 1 Release 1.2.1 Annex A17: RoCEv2
- Comer, Douglas E. (2006). Internetworking with TCP/IP: Principles, Protocols, and Architecture. Vol. 1 (5th ed.). Prentice Hall. ISBN 978-0-13-187671-2.
- Transmission Control Protocol (TCP) - https://datatracker.ietf.org/doc/html/rfc9293
- TCP Congestion Control - https://datatracker.ietf.org/doc/rfc5681/
- TCP/IP Illustrated, Vol. 1: The Protocols (Addison-Wesley Professional Computing Series)
- TCP/IP Illustrated: The Implementation
- TCP/IP Illustrated: v. 3: TCP for Transactions, HTTP, NNTP, and the Unix Domain Protocols (Addison-Wesley Professional Computing Series)
- RFC 7143: Internet Small Computer System Interface (iSCSI) Protocol (Consolidated) - https://datatracker.ietf.org/doc/html/rfc7143
- RFC 4171: Internet Storage Name Service (iSNS) https://datatracker.ietf.org/doc/html/rfc4171
- RFC 3821: Fibre Channel Over TCP/IP (FCIP); https://www.rfc-editor.org/rfc/rfc3821.html
- Storage protocol stacks: https://brasstacksblog.typepad.com/brass-tacks/2016/07/storage-protocol-stacks.html
- Storage Protocol Stacks for NVMe: https://brasstacksblog.typepad.com/brass-tacks/2017/11/storage-protocol-stacks-for-nvme.html

- EMC Networked Storage Concepts and Protocols TechBook.

Chapter 2. Understanding Congestion in Fibre Channel Fabrics

This chapter covers the following topics:

- Fibre Channel Flow Control.
- Congestion Spreading in Fibre Channel Fabrics.
- Frame Flow within a Fibre Channel Switch.
- Effect of Bit Errors on Congestion.
- Credit Loss Recovery Mechanisms.

Fibre Channel is the most common type of network for transporting SCSI and NVMe protocols for accessing remote block storage. It uses B2B flow control between directly connected devices to build a lossless network. When congestion happens, it spreads to the traffic source affecting many other devices that share the same network paths. In the Fibre Channel community, congestion spreading is commonly known as slow-drain. However, in this book, slow-drain is specifically used for a type of congestion that is caused by an end device with a slower processing rate as compared to the rate at which frames are being delivered to it. Congestion may be caused by other reasons also, such as due to over-utilization of a link, bit errors, and lack of buffers for the length, speed, and frame size. These causes and sources of congestion in Fibre Channel fabrics are explained in this chapter.

Fibre Channel Flow Control

Fibre Channel defines two types of flow control mechanisms, as shown in Figure 2-1:

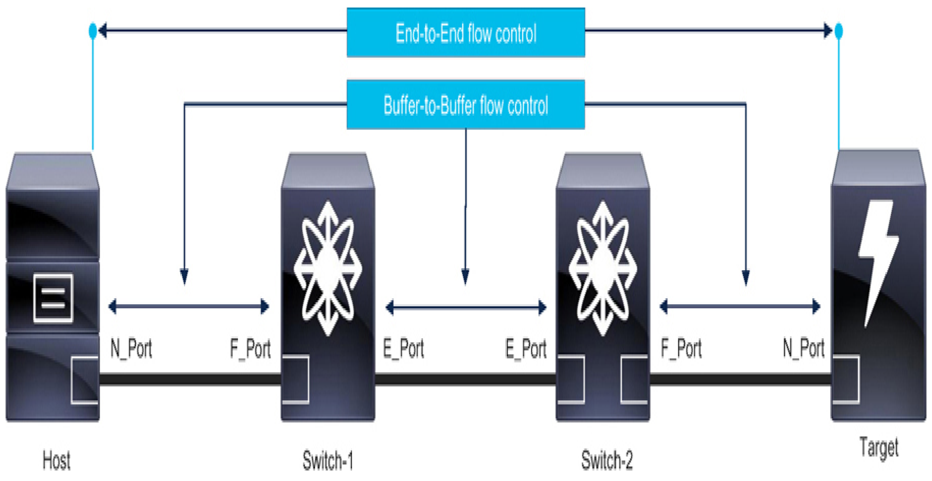


Figure 2-1 *Flow control mechanisms in Fibre Channel*

- Buffer-to-Buffer (B2B) flow control (between directly connected ports)
- End-to-End (E2E) flow control (between source and destination)

End-to-end flow control in Fibre Channel was never widely implemented. Like most real devices, the focus of this book is only on B2B flow control.

Each Fibre Channel link should be viewed as two unidirectional links. Each side of the link is both a frame-sender and a frame-receiver and B2B flow control works independently in each direction.

A frame-receiver must not be sent more frames than it can accommodate in its buffers. When a frame-receiver's buffers are full, it has no other option except to drop the new incoming frame. This goes against the basic principle of a lossless network. To prevent this buffer overrun, the devices on both sides of the link communicate credits during link initialization. The number of these credits is the same as the number of full-sized frames the frame-receiver can accommodate in its buffers. When the frame flow starts, the frame-sender decrements the remaining credits when it sends a frame. After the frame-receiver processes the frame and makes the buffer available for re-use, it returns a credit to the frame-sender.

What it means to process a frame depends on the type of the receiver. An end device processes the ingress frames by handing them off to an upper layer protocol (ULP), such as SCSI or NVMe. A switch processes the ingress frames by handing them off to an egress port. In all cases, processing a frame result in making the ingress buffer available for re-use.

The frame-sender continues to send frames as long as it has at least one remaining transmit credit, but stops transmitting frames if it has zero remaining transmit credits to avoid a buffer overrun situation with the receiver.

This is referred to as the Buffer-to-Buffer (B2B) flow control mechanism and its goal is to prevent buffer overflows between the directly connected Fibre Channel ports.

The B2B flow control can be divided into the following two phases:

1. The initial communication of B2B credits
2. Return of B2B credits during frame flow

Initial Communication of B2B Credits

When a link between two FC ports comes up, the neighbors communicate with each other about the number of receive buffers each has. These buffers equate to credits, and thus, receive buffers and receive credits refer to the same value, which is the maximum number of frames a receiver can accommodate in its buffers. They are also called B2B credits because of their use in the B2B flow control mechanism.

This communication of B2B credits must complete before starting to send frames on a link.

The procedure to communicate B2B credits between the neighboring FC ports is slightly different based on the port types (Figure 2-2). These port types are explained in Chapter 1, "Introduction to Congestion in Storage Network," the section on *Fibre Channel and FCoE Terminology*. A N_Port (such as a host port) connected to F_Port (such as a switchport) communicates B2B credit information through the Fabric Login (FLOGI) phase. An E_Port (such as a switchport) connected to another E_Port communicates B2B credit information through the Exchange Link Parameters (ELP) phase. Regardless of the exact procedure—FLOGI or ELP—the goal is to communicate the number of Rx B2B credits with its directly connected neighbor.

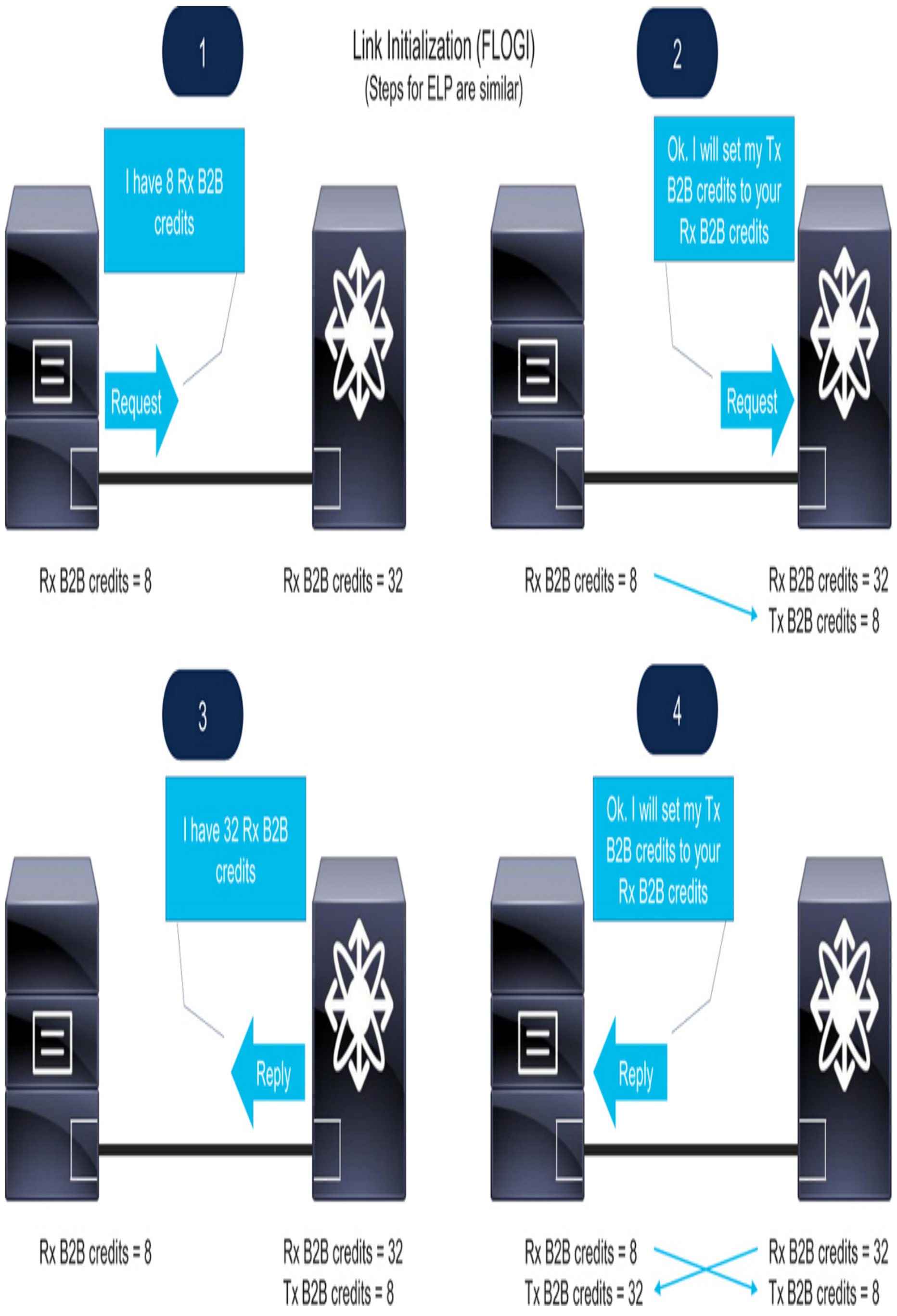


Figure 2-2 *Communication of B2B credits during link initialization*

After learning the Rx B2B credits of the neighbor, an FC port sets this value as its Tx B2B credits. This communication of credits is a bidirectional procedure.

The following are the important details about the initial communication of B2B credits:

- **B2B credits are not negotiated, just communicated:** Contrary to common belief, directly connected neighboring FC ports don't negotiate B2B credits. Rather, the number of Rx B2B credits is communicated or informed to the neighbor, which the neighbor uses as the Tx B2B credits. An FC port simply accepts whatever number is conveyed to it by its neighbor. It can't request to change this number, and thus, this process can't be called a negotiation. For example, an HBA port (N_Port) may communicate or inform to the switchport (F_Port) that it has 8 Rx B2B credits. The switchport uses this information to set its Tx B2B credits to 8. The switchport can't change its Tx B2B credits by itself. The only way to change the Tx B2B credits on this switchport is to change the number of Rx B2B credits on the HBA port and re-initialize the link for this change to take effect.
- **B2B credits are defined by the manufacturer:** The default number of Rx B2B credits on a port is defined by the manufacturer. Some products allow changing the number of Rx B2B credits, but the extent of this change is defined by the capability of the product. B2B credit count is the number of buffers, which is a finite resource decided during the design of a product (adapter or switch). For example, by default, a 64GFC port on Cisco MDS switches has 100 Rx B2B credits when it operates as F_Port. This number can be increased to 500 by a configuration change. Likewise, the same port when operating as E_Port has 1000 Rx B2B credits by default, which can be increased to 16000 by borrowing credits from other ports in the same port group. On the Cisco Nexus 93360YC-FX2 switch, the number of Rx B2B credits on an FC port is 32. This number is fixed, and can't be changed, extended, or borrowed from other ports in the same port group.

- **Changing the Tx B2B credits of an FC port requires changing the neighbor's Rx B2B credits followed by a link flap:** As explained, an FC Port communicates the number of Rx B2B credits to its neighbor during link initialization (FLOGI or ELP). After changing the number of Rx B2B credits, the port must flap or re-initialize to re-trigger the FLOGI or ELP phase so that the new value can be communicated to the neighbor. For example, on Cisco MDS switches, changing the Rx B2B credits of a port immediately flaps the port. Hence, be cautious before making this change.
- **The B2B credits of the neighbors need not be the same:** The B2B credits of the neighbors are independent of each other, and they need not be the same. For example, an HBA port (N_Port) may have 8 Rx B2B credits, whereas the connected switchport (F_Port) may have 32 Rx B2B credits. This difference in the number of buffers on the neighboring FC ports is common and shouldn't be a concern.

The most important point to remember about the initial communication of B2B credits is that the Rx B2B credits of an FC port are set as the Tx B2B credits of the neighbor and the neighbor's Rx B2B credits are set as the Tx B2B credits on the local FC port.

Return of B2B Credits During Frame Flow

The following steps and Figure 2-3 describe the return of B2B credits during frame flow.

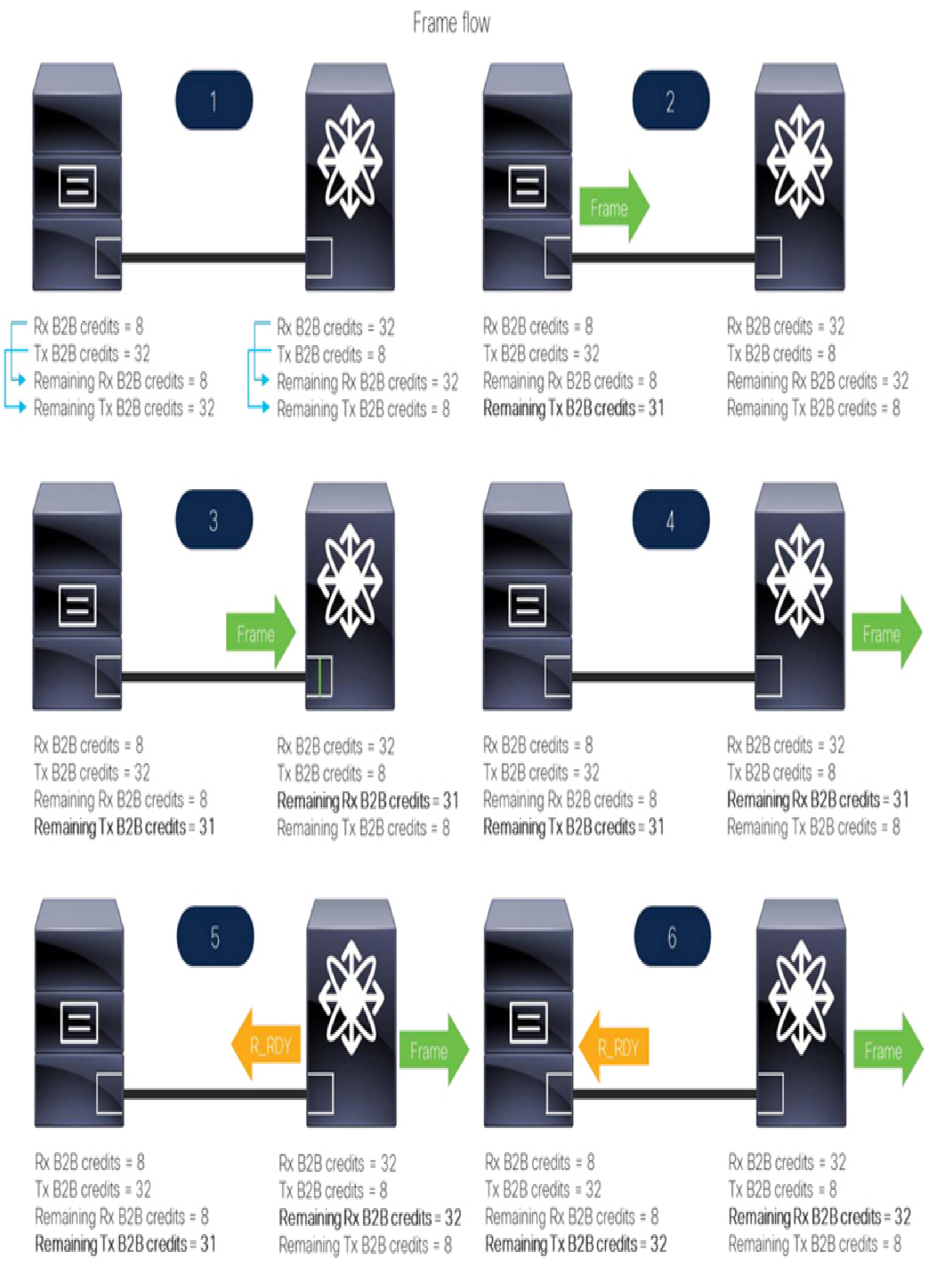

Frame flow
1
Rx B2B credits = 8
Tx B2B credits = 32
Remaining Rx B2B credits = 8
Remaining Tx B2B credits = 32
Rx B2B credits = 32
Tx B2B credits = 8
Remaining Rx B2B credits = 32
Remaining Tx B2B credits = 8
2
Frame
Rx B2B credits = 8
Tx B2B credits = 32
Remaining Rx B2B credits = 8
Remaining Tx B2B credits = 31
Rx B2B credits = 32
Tx B2B credits = 8
Remaining Rx B2B credits = 32
Remaining Tx B2B credits = 8
3
Frame
Rx B2B credits = 8
Tx B2B credits = 32
Remaining Rx B2B credits = 8
Remaining Tx B2B credits = 31
Rx B2B credits = 32
Tx B2B credits = 8
Remaining Rx B2B credits = 31
Remaining Tx B2B credits = 8
4
Frame
Rx B2B credits = 8
Tx B2B credits = 32
Remaining Rx B2B credits = 8
Remaining Tx B2B credits = 31
Rx B2B credits = 32
Tx B2B credits = 8
Remaining Rx B2B credits = 31
Remaining Tx B2B credits = 8
5
R_RDY
Frame
Rx B2B credits = 8
Tx B2B credits = 32
Remaining Rx B2B credits = 8
Remaining Tx B2B credits = 31
Rx B2B credits = 32
Tx B2B credits = 8
Remaining Rx B2B credits = 32
Remaining Tx B2B credits = 8
6
R_RDY
Frame
Rx B2B credits = 8
Tx B2B credits = 32
Remaining Rx B2B credits = 8
Remaining Tx B2B credits = 32
Rx B2B credits = 32
Tx B2B credits = 8
Remaining Rx B2B credits = 32
Remaining Tx B2B credits = 8

Figure 2-3 *Return of B2B credits during frame flow*

1. When the frame flow starts, the frame-sender maintains a remaining-Tx-B2B-credits counter which is initially set to the same value as the Tx B2B credits. Likewise, the frame-receiver maintains a remaining-Rx-B2B-credits counter which is set to the same value as the Rx B2B credits.
2. The frame-sender checks the remaining-Tx-B2B-credits and if it is zero the frame cannot be transmitted. If the remaining-Tx-B2B-credits are greater than zero, the sender decrements it by one and transmits the frame.

3. Similarly, the frame-receiver decrements the remaining-Rx-B2B-credits by one for every received frame.
4. The frame-receiver processes the frame (for example an ingress port on a switch sends the frame to an egress port) which frees up the buffer for re-use.
5. After the buffer is available for re-use, the frame-receiver increments the remaining-Rx-B2B-credits by one and returns a credit to the frame-sender by sending an R_RDY. An R_RDY (Receiver_Ready) is a Fibre Channel Primitive signal that an FC port sends to convey that it has a buffer available to receive one frame. Refer to the section *Special Functions — Delimiters, Primitive Signals, and Primitive Sequences* for more details.
6. The frame-sender increments the remaining-Tx-B2B-credits by one after receiving R_RDY.

These mechanisms happen for each frame on each side of the link.

Overall, the remaining-Tx-B2B-credits decrements as an FC port sends frames, whereas the remaining-Tx-B2B-credits increment as the port receives an R_RDY.

B2B Credit Counters

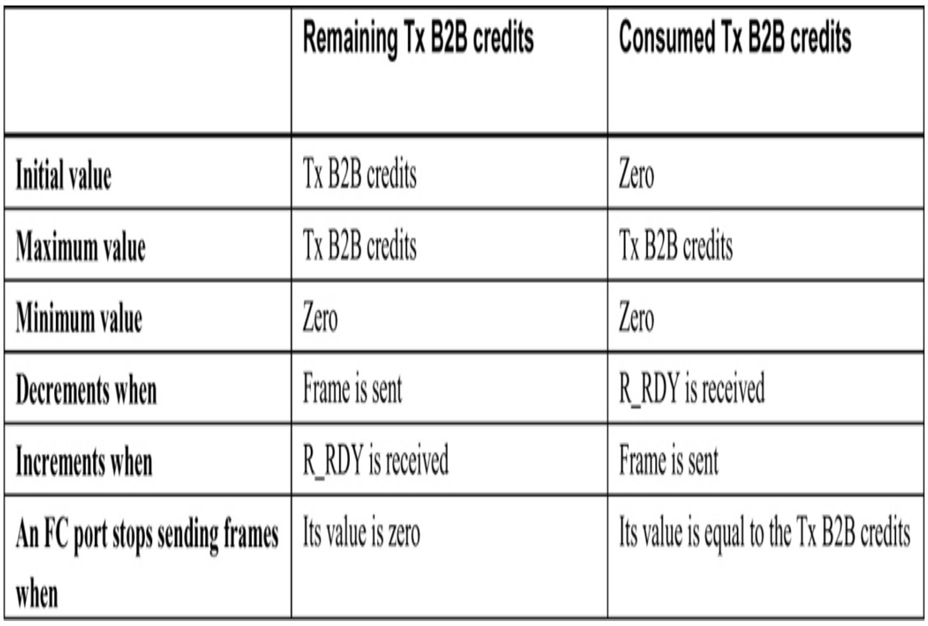
Figure 2-3 shows the following counters on an FC port:

1. **Rx B2B credits**: The number of receive buffers of an FC port. This value does not change while the port is up.
2. **Tx B2B credits**: The number of receive buffers of the neighbor FC port as informed during link initialization. This value does not change while the port is up.
3. **Remaining-Rx-B2B-credits**: The number of free receive buffers on an FC port. Its minimum value can be zero and the maximum value can be Rx B2B credits.
4. **Remaining-Tx-B2B-credits**: The number of free receive buffers on the neighbor FC port. Its minimum value can be zero and the maximum value can be Tx B2B credits.

FC devices that are available in the market may not show all these four counters. Some devices may show only the remaining Rx and Tx B2B credits ((3) and (4)) which are enough because their values are the same as (1) and (2) before traffic starts on a link. The key difference is that (1) and (2) don't change after a link comes up, whereas (3) and (4) are dynamic and their values change as frames and R_RDYs are sent or received.

Some FC devices may choose to show the consumed credits instead of the remaining credits. The difference is that the remaining-Tx-B2B-credit counter on an FC port tracks the number of free buffers on the neighbor. The goal is to avoid buffer overrun of the neighbor by stopping to send the frames when the remaining-Tx-B2B-credits are zero. The same goal can also be achieved by using an alternative approach that tracks the consumed buffers of the neighbor. This approach avoids buffer overrun of the neighbor by stopping to send the frames when the consumed-Tx-B2B-credits become equal to the Tx B2B credits. Table 2-1 compares these two implementations.

Table 2-1 *Comparison of two implementations to track free buffers on the frame-receiver*



	Remaining Tx B2B credits	Consumed Tx B2B credits
Initial value	Tx B2B credits	Zero
Maximum value	Tx B2B credits	Tx B2B credits
Minimum value	Zero	Zero
Decrements when	Frame is sent	R_RDY is received
Increments when	R_RDY is received	Frame is sent
An FC port stops sending frames when	Its value is zero	Its value is equal to the Tx B2B credits

A device may choose to implement only one of these approaches. For example, Cisco MDS switches implement the approach of remaining credits, not the approach of consumed credits. This book uses the approach of the remaining credits.

Example 2-1 shows the remaining-Rx-B2B-credits and remaining-Tx-B2B-credits on a Fibre Channel port of a Cisco MDS switch.

Example 2-1 *Remaining Rx B2B credits and remaining Tx B2B credits on Cisco MDS switches*

```
MDS9000# show interface fc1/2
<snip>
    32 receive B2B credit remaining
    5 transmit B2B credit remaining
<snip>
MDS9000#
```

Besides these, Fibre Channel ports may report many other counters, such as B2B credits transitions to zero (number of times remaining B2B credits fall to zero) and TxWait/RxWait (durations for which frames could not be transmitted or received because of zero remaining B2B credits). Refer to Chapter 3, "Detecting Congestion in Fibre Channel Fabrics," for more details on these counters.

Important Details about R_RDY and B2B Credits

The following are the important details about R_RDY and B2B credits.

- **Frame-receiver returns a B2B credit by sending an R_RDY:**. When an FC port receives an R_RDY, it increments its remaining-Tx-B2B-credits by one. Consequently, the following sentences have the same meaning.
  - The frame-receiver returned a credit.
  - The frame-receiver returned a B2B credit.
  - The frame-receiver returned an R_RDY.
- **R_RDY flows in the opposite direction of the frames:** A frame-sender sends frames to a frame-receiver,

whereas a frame-receiver sends R_RDYs to a frame-sender.

- **One B2B credit is for one frame:** The B2B flow control mechanism uses one credit for one frame irrespective of the size of the frame. In other words, the size of a frame is irrelevant to the B2B flow control mechanism. A 128-byte frame, 1024-byte frame, or a full-size Fibre Channel frame of 2148 bytes—all consume one B2B credit. Because of this reason, the size of buffers on FC ports is expressed as a counter, such as 8 or 32 credits. The size of buffers on FC ports is not expressed in bytes (B) or megabytes (MB), as it is typically done on Ethernet ports.
- **R_RDYs do not increase link utilization:** R_RDY is an FC primitive signal. It doesn't increase the link utilization because it's transmitted in the place of inter-frame fill words. Refer to the section on *Effect of Primitive Signals on Data Rate* for more details.
- **R_RDY is not linked to a specific frame:** R_RDY just conveys that the frame-receiver has a buffer available to receive one more frame. It doesn't convey which specific frame was processed by the receiver.
- **R_RDY doesn't convey the validity of a frame:** R_RDY doesn't convey if the frame was corrupted. A frame-receiver sends an R_RDY even if it detects CRC error with the frames. The exact handling of the CRC corrupted frame depends on its receiver. Refer to the sections on *Ability to detect and drop CRC corrupted frame* and *CRC* for more details. In some cases, a frame may get corrupted in a manner that the receiver doesn't even recognize that the ingress bits belong to a frame, and hence a credit is not returned for it because the receiver believes that it never received the frame. This condition may result in the loss of remaining-Tx-B2B-credits. Refer to the section on *B2B Credit Loss and Recovery* for further details.
- **Corruption of R_RDY**: An R_RDY may get corrupted due to bit errors, and thus its receiver (frame-sender) can't recognize it. This corruption of R_RDY results in

the loss of a remaining-Tx-B2B-credit for the frame-sender. Refer to the section on *B2B Credit Loss and Recovery* for further details.

- **R_RDYs remain local on a link:** Although R_RDY on different links may have the same format because they are FC primitive signals, these R_RDY are not directly related to each other (Figure 2-4).

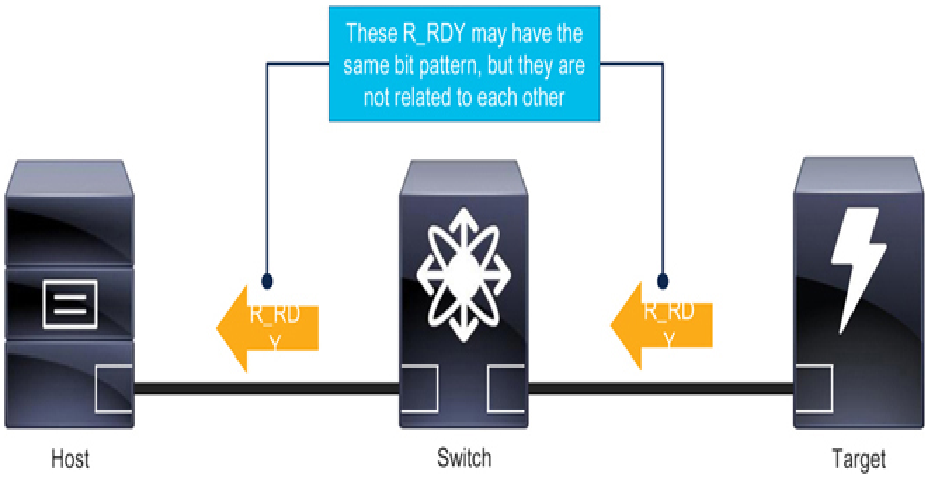

These R_RDY may have the same bit pattern, but they are not related to each other
R_RDY
R_RDY
Host
Switch
Target

Figure 2-4 *R_RDYs remain local to a link*

- **R_RDYs are not an acknowledgment or ACK**. Sometimes it’s stated that the R_RDY is an ACK but it shouldn’t be thought of that way. As mentioned, it doesn’t imply the frame was received intact and does not imply the frame ever really made it to its final destination.

B2B Flow Control in a Multi-hop Fabric

As explained, B2B flow control works only between the directly connected FC ports. But collectively it avoids buffer overrun on every port in a multi-hop fabric, and therefore achieves a lossless fabric. This section explains B2B flow control in a multi-hop fabric under two conditions. First, when the fabric is not congested. Second, when the fabric is congested.

B2B Flow Control in a Multi-hop Fabric — Without Congestion

Consider the example shown in Figure 2-5.

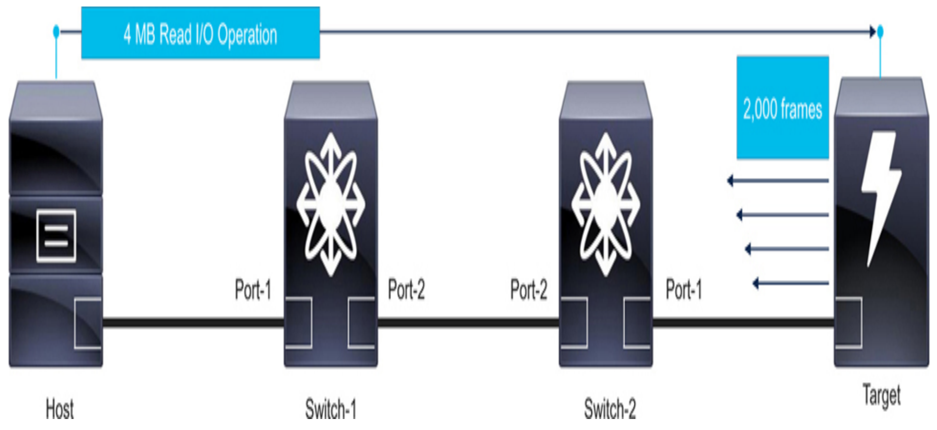


Figure 2-5 *B2B flow control in a multi-hop fabric*

The Host initiates a 4-MB read I/O operation to the Target. As a result, Target reads data from the back-end storage media and sends the data-carrying frames. A Fibre Channel frame can carry up to 2048 bytes of data (Refer to the section on Fibre Channel Frame Format for more details), and therefore, the 4-MB read operation results in approximately 2,000 frames. Refer to Chapter 5, "Solving Congestion by Storage I/O Performance Monitoring," for details on read and write I/O operations.

When the Target port has frames for transmission, it decrements the remaining-Tx-B2B-credits by one and sends a frame to Switch-2/Port-1.

Switch-2 does a lookup of the destination ID in the frame header. It finds that the frame destination is reachable via Port-2, and thus the frame is switched to Port-2. After Switch-2/Port-1 sends the frame to Switch-2/Port-2 and its ingress buffer is available for re-use, it sends an R_RDY to the Target. The Target port, after receiving the R_RDY, increments the remaining-Tx-B2B-credits by one.

Meanwhile, Switch-2/Port-2 decrements its remaining-Tx-B2B-credits by one and sends the frame to Switch-1/Port-2. Switch-1 follows a similar forwarding mechanism as Switch-2 which results in the switching the frame to Switch-1/Port-1. When the ingress buffer is available for re-use, Switch-1/Port-2 sends a R_RDY to Switch-2/Port-2. When Switch-2/Port-2 receives the R_RDY, it increments the remaining-Tx-B2B-credits by one.

Further, Switch-1/Port-1 decrements its remaining-Tx-B2B-credits by one and sends the frame to the Host. The Host port sends a R_RDY after handing off the frame to the upper layer, such as SCSI or NVMe, and making the ingress buffer available for re-use. Switch-1/Port-1, after receiving the R_RDY, increments the remaining-Tx-B2B-credits by one.

This sequence of events repeats for all the frames that are sent as part of this I/O operation.

This is how the B2B flow control mechanism works on every link in a multi-hop fabric, although its benefit is not evident until the fabric becomes congested, as explained next.

B2B Flow Control in a Multi-hop Fabric — With Congestion

Consider the same scenario as the Host initiates a 4-MB read operation. But this time, it becomes congested (slow-drain device), hence, it is unable to hand off the arriving frames to the upper layers, such as SCSI and NVMe. Also, consider a typical B2B credit count on the ports, as shown in Figure 2-6.

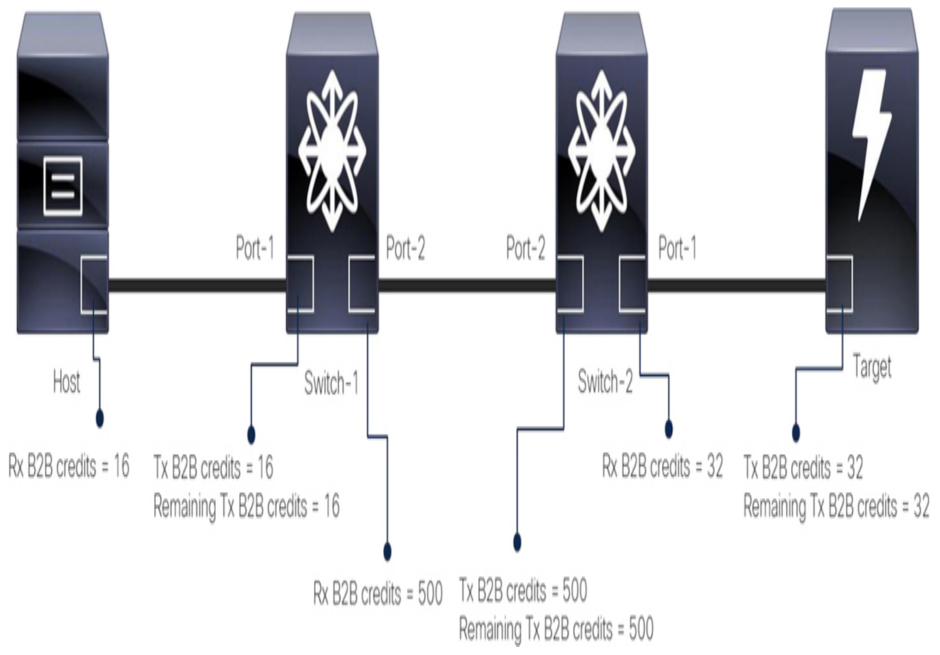


Figure 2-6 *B2B flow control in a multi-hop fabric*

- The Host port has 16 receive buffers (Rx B2B credits), which become the Tx B2B credits of Switch-1/Port-1.
- Switch-1/Port-2 has 500 receive buffers (Rx B2B credits), which become the Tx B2B credits of Switch-2/Port-2.
- Switch-2/Port-1 has 32 receive buffers (Rx B2B credits), which become Tx B2B credits of the Target port.

When the Host port is unable to hand off the frames to the upper layer, it still receives frames without any issues, but it doesn't send R_RDY to Switch-1/Port-1. This is because the frames remain in the ingress buffers without making them available for re-use.

Switch-1/Port-1 doesn't immediately stop sending frames to the Host because it has 16 Tx B2B credits. Not receiving an R_RDY from the Host port reduces the remaining-Tx-B2B-credits to 15, which means Switch-1/Port-1 can send 15 frames without risking the buffer overrun of the Host Port. If the Host continues to be congested and the host port doesn't send R_RDY while Switch-1/Port-1 sends 15 more frames, the remaining-Tx-B2B-credits on Switch-1/Port-1 fall to zero (Figure 2-7). As a result, it stops sending frames, which avoids buffer overrun on the host ingress port.

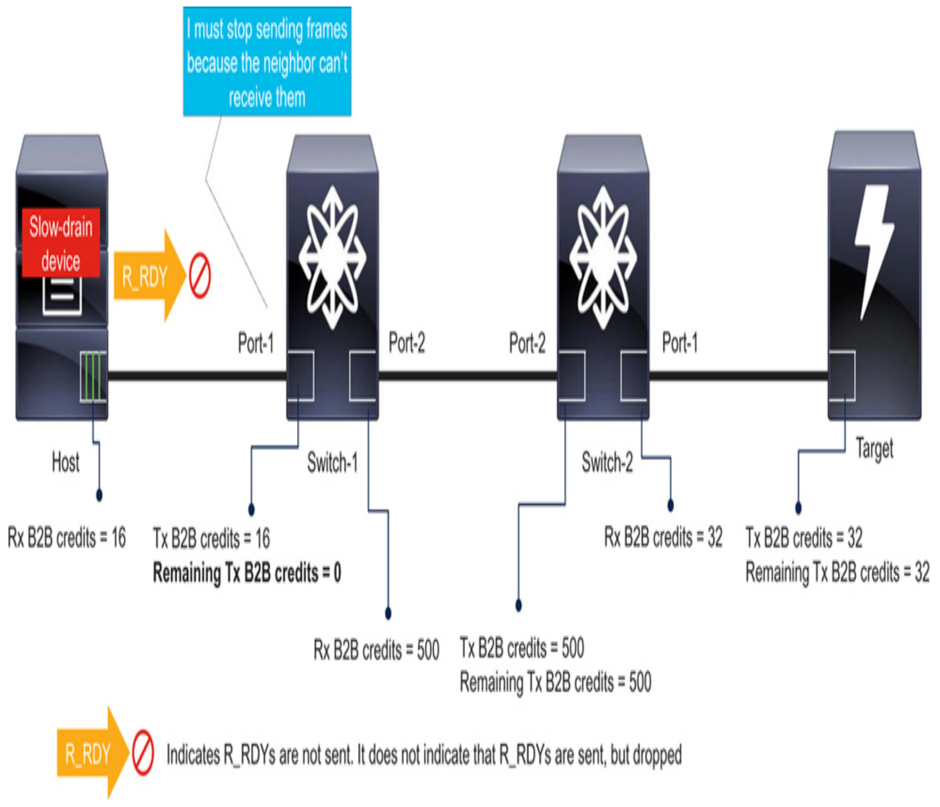
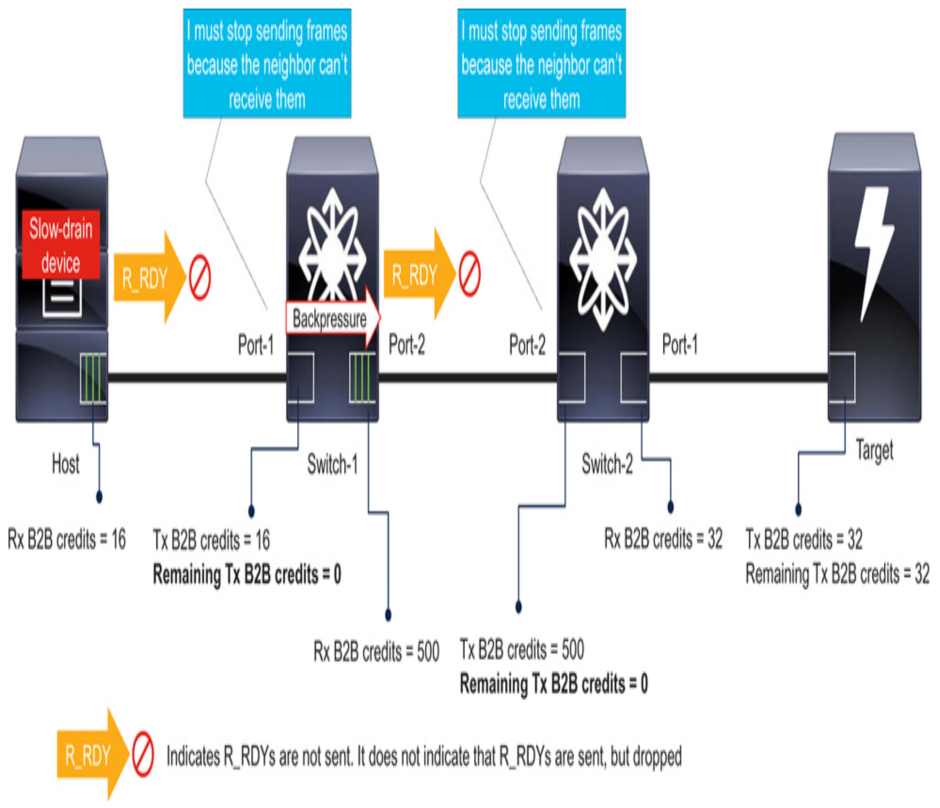

I must stop sending frames because the neighbor can't receive them
Slow-drain device
R_RDY
Port-1
Port-2
Port-2
Port-1
Host
Switch-1
Switch-2
Target
Rx B2B credits = 16
Tx B2B credits = 16
Remaining Tx B2B credits = 0
Rx B2B credits = 32
Tx B2B credits = 32
Remaining Tx B2B credits = 32
Rx B2B credits = 500
Tx B2B credits = 500
Remaining Tx B2B credits = 500
R_RDY
Indicates R_RDYs are not sent. It does not indicate that R_RDYs are sent, but dropped

Figure 2-7 *Switch-1 stops sending frames when its remaining-Tx-B2B-credits are zero*

Next, let's understand the events on the ISL ports when Switch-1/Port-1 has zero remaining-Tx-B2B-credits. Switch-1/Port-2 can't switch any more frames to Switch-1/Port-1 because of internal backpressure within Switch-1. The exact mechanism of the internal backpressure depends on the architecture of the switch. Refer to the section on *Frame flow within a Fibre Channel Switch* for more details.

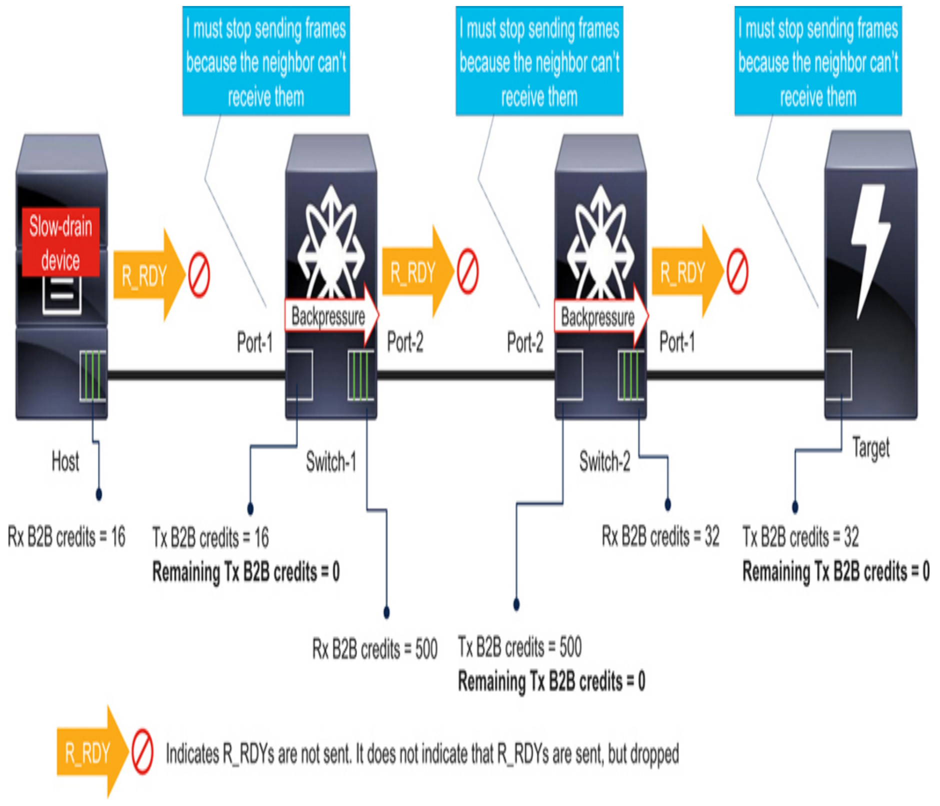
The ingress frames on Switch-1/Port-2 consume the receive buffers of Switch-1/Port-2. As a result, Switch-1/Port-2 doesn't send R_RDYs to Switch-2/Port-2. But Switch-2/Port-2 doesn't immediately stop sending frames because it has 500 Tx B2B credits. Not receiving an R_RDY makes the remaining-Tx-B2B-credits fall to 499, which means Switch-2/Port-2 can send 499 more frames without risking the buffer overrun of Switch-1/Port-2. When 499 frames are sent without receiving an R_RDY, the remaining-Tx-B2B-credits fall to zero (Figure 2-8). As a result, Switch-2/Port-2 stops sending frames, which avoids buffer overrun on Switch-1/Port-2.



I must stop sending frames because the neighbor can't receive them
I must stop sending frames because the neighbor can't receive them
Slow-drain device
R_RDY
R_RDY
Backpressure
Port-1
Port-2
Port-2
Port-1
Host
Switch-1
Switch-2
Target
Rx B2B credits = 16
Tx B2B credits = 16
Remaining Tx B2B credits = 0
Rx B2B credits = 32
Tx B2B credits = 32
Remaining Tx B2B credits = 32
Rx B2B credits = 500
Tx B2B credits = 500
Remaining Tx B2B credits = 0
R_RDY
Indicates R_RDYs are not sent. It does not indicate that R_RDYs are sent, but dropped

Figure 2-8 *Switch-2 stops sending frames when its remaining-Tx-B2B-credits are zero*

The same sequence of events continues between Switch-2/Port-1 and the Target port. Eventually, the remaining-Tx-B2B-credits on the Target port fall to zero. As a result, it stops sending any more frames to avoid buffer overrun on Switch-2/Port-1 (Figure 2-9).



I must stop sending frames because the neighbor can't receive them
I must stop sending frames because the neighbor can't receive them
I must stop sending frames because the neighbor can't receive them
Slow-drain device
R_RDY
R_RDY
R_RDY
Backpressure
Backpressure
Port-1
Port-2
Port-2
Port-1
Host
Switch-1
Switch-2
Target
Rx B2B credits = 16
Tx B2B credits = 16
Remaining Tx B2B credits = 0
Rx B2B credits = 32
Tx B2B credits = 32
Remaining Tx B2B credits = 0
Rx B2B credits = 500
Tx B2B credits = 500
Remaining Tx B2B credits = 0
R_RDY
Indicates R_RDYs are not sent. It does not indicate that R_RDYs are sent, but dropped

Figure 2-9 *Multi-hop lossless fabric using B2B flow control*

During the condition of Figure 2-9,

- 16 frames are waiting in the receive buffers of the Host port.
- 500 frames are waiting in the receive buffers of Switch-1/Port-2.
- 32 frames are waiting in the receive buffers of Switch-2/Port-1.

The target has 2000 frames to send for the 4 MB read operation, but it doesn't immediately send 1452 frames (2000

– 16 – 500 – 32 = 1452), and thus prevents the dropping of these frames in the network.

Further, let’s assume that after a few milliseconds, the host becomes ready to process the frames. It immediately starts handing off the frames that are waiting in its receive buffers to the upper layers. This frees up the buffers and makes them available for re-use. As a result, the host port immediately starts sending R_RDYs to Switch-1/Port-1 for every buffer that is now available. This releases the backpressure within Switch-1. As R_RDYs arrive at Switch-1/Port-1 it can transmit buffered (or queued) frames and can now accept additional frames from Port-2 and transmit them to the Host.

On Switch-1/Port-2, for every receive buffer that is available for re-use, it immediately sends an R_RDY to Switch-2/Port-2. This sequence of events repeats on Switch-2 as well. Eventually, the target port receives R_RDYs from Switch-2/Port-1, and thus resumes transmission of frames.

This is how B2B flow control on the FC links and internal backpressure of the Fibre Channel switches avoid buffer overrun through the end-to-end frame delivery from the source to the destination, and therefore achieves a lossless fabric.

This explanation simplifies some details such as the time required for serialization and propagation of frames and R_RDY. In reality, the transmission of the frames and R_RDYs happens very fast. Also, a receiver may start receiving the frame while the sender is still transmitting it because an FC frame can have up to 2148 bytes or 17,184 bits (These details are discussed in the section on Detailed Explanation for the Required Number of B2B Credits on an FC port). Because of these reasons, getting an exact count of remaining credits after the transmission of every frame is practically not feasible. By the time you get a snapshot of the remaining-Tx-B2B-credits on a port, the value may change.

Buffer Overrun Situation

What happens if a Fibre Channel port has zero remaining-Rx-B2B-credits, and it is still sent a frame by its directly connected neighbor? This condition is considered a severe error. The port takes two actions. First, it drops the incoming frame due to buffer overrun. Second, it initiates Link Reset Protocol, which results in a bidirectional traffic halt on the link until the Link Reset Protocol completes. For more details refer to the sections on *Link Initialization Counters* and *Credit Loss Recovery Using Link Reset Protocol*.

Frame Rate Equalization using B2B Flow Control

The illustration in Figure 2-9 assumes that the destination (Host) stops processing the frames completely. However, a more realistic scenario is when the receiver processes the frames slowly. In other words, the receiver is unable to process the frames as fast as the source (Target) is sending them. For example, Target sends 800 frames per second, but the Host can accept only 200 frames per second. Hence, the Host port starts pacing the rate of R_RDY. Because one R_RDY returns one credit which is used to send only one frame, sending 200 R_RDYs per second results in a frame rate of 200 per second.

Overall, the B2B flow control mechanism achieves the following two results:

- It equalizes the rate of ingress traffic with the rate of egress traffic.
- Because of this ingress/egress rate equalization, there are no frame drops, therefore achieving a lossless fabric.

Congestion Spreading in Fibre Channel Fabrics

As explained in Chapter 1, a network may get congested when the ingress traffic is more or faster than the egress traffic. To prevent a buffer overrun situation, the B2B flow control applies backpressure by slowing down sending of R_RDYs. Once the directly connected neighbor in the upstream traffic direction reaches zero remaining-Tx-B2B-credits it must stop sending frames. When it stops sending, it stops accepting frames from its upstream neighbor causing backpressure that spreads in the direction of the ultimate source of the frames.

This is intentional behavior and it is evident from Figure 2-9. But, as a side effect, it slows down other unrelated traffic. As a result, one culprit device may victimize many other devices sharing the same path or end devices.

The B2B flow control mechanism slows down all the traffic on a link because it has a link-level scope. It doesn't selectively slow down the frames destined for a specific device. In other words, it doesn't have a flow-level scope. This rule is slightly modified by creating multiple virtual links (VLs) or virtual circuits (VCs), dedicating credits to each VL/VC, and flow controlling them separately using ER_RDY or VC_RDY primitives. As a result, traffic in a VL does not consume credits of other VLs on the same port. More details on VLs/VCs and flow control using ER_RDY/VC_RDY are explained in Chapter 6, "Preventing Congestion in Fibre Channel Fabrics," the section on *Traffic Segregation Using Virtual Links*. But for now, for the sake of simplicity, it's enough to understand that B2B flow control mechanism applies to all the frames on a link regardless of their destination.

Congestion due to Slow-drain Devices

A slow-drain device has a slower processing rate as compared to the rate at which frames are being delivered to it. To avoid buffer overrun, this slow-drain device applies backpressure using B2B flow control, which results in congestion spreading.

A high-level overview of slow-drain and its reasons are explained in Chapter 1, the section on *Congestion Due to Slow End Devices*. This section expands on the same details as they apply to Fibre Channel fabrics.

Consider the example of Figure 2-10, which shows a part of a typical edge-core storage fabric. Hosts-1 and Host-2 connect to Switch-1, whereas Target-1 and Target-2 connect to Switch-2. Host-1 initiates a 4-MB read I/O operation to Target-1. At the same time, Host-2 also initiates a 4-MB read I/O operation to Target-2. As mentioned earlier, because a Fibre Channel frame can carry up to 2048 bytes of data, a 4-MB read operation results in approximately 2,000 frames. These frames flow through the shared ISL between Switch-1 and Switch-2 while going from Target-1 to Host-1 and Target-2 to Host-2.

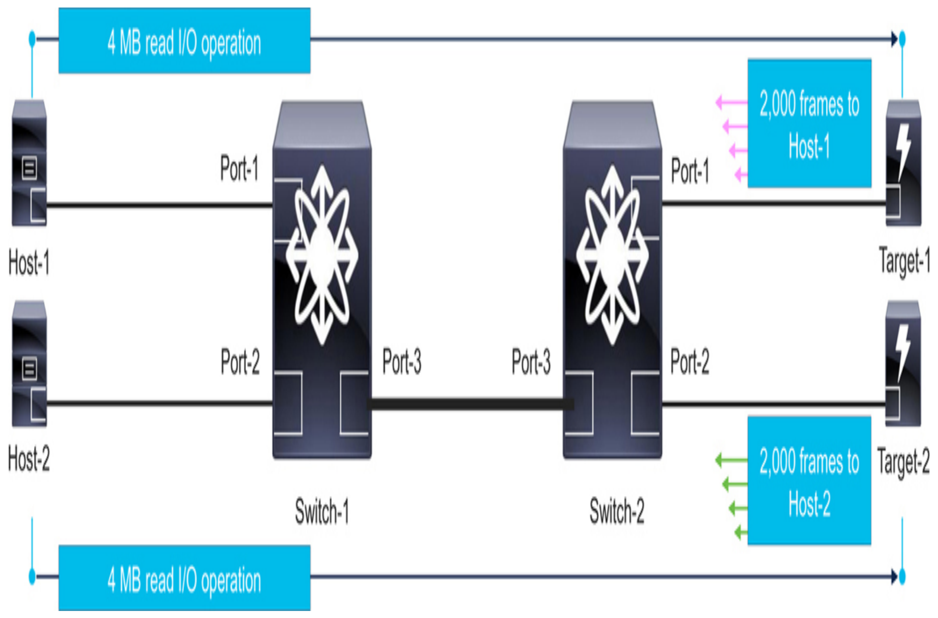


Figure 2-10 *A typical edge-core Fibre Channel fabric*

As explained earlier, when Host-1 doesn't return R_RDYs, the remaining-Tx-B2B-credits fall to zero on Switch-1/Port-1 which results in internal backpressure leading to slowing down the sending of R_RDY from Switch-1/Port-3 to Switch-2/Port-3 (Figure 2-11).

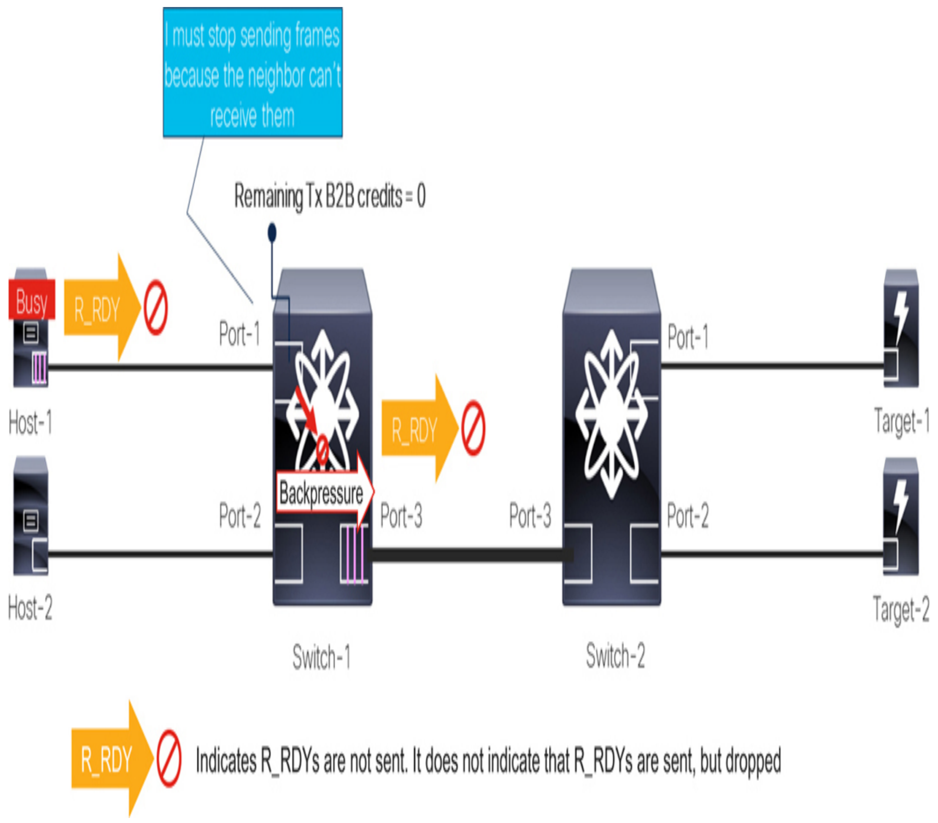

I must stop sending frames because the neighbor can't receive them
Remaining Tx B2B credits = 0
Busy
R_RDY
Port-1
Port-1
Host-1
R_RDY
Target-1
Backpressure
Port-2
Port-3
Port-3
Port-2
Host-2
Target-2
Switch-1
Switch-2
R_RDY
Indicates R_RDYs are not sent. It does not indicate that R_RDYs are sent, but dropped

Figure 2-11 *Congestion due to a slow-drain device*

With the slower rate of ingress R_RDYs, the remaining-Tx-B2B-credits on Switch-2/Port-3 eventually fall to zero. In this condition, Switch-2 reduces the frame sending rate to Switch-1, regardless of the destination of the frames. That means, frame rate from Target-1 to Host-1 slows down, and frame rate from Target-2 to Host-2 also slows down. Reduced frame rate from Target-1 to Host-1 is expected but slowing down the frame rate from Target-2 to Host-2 is an undesired side effect.

Eventually, congestion spreads to Switch-2 (Figure 2-12). Because Switch-2/Port-3 can't send frames, it exerts internal backpressure on both targets (Port-1 and Port-2).

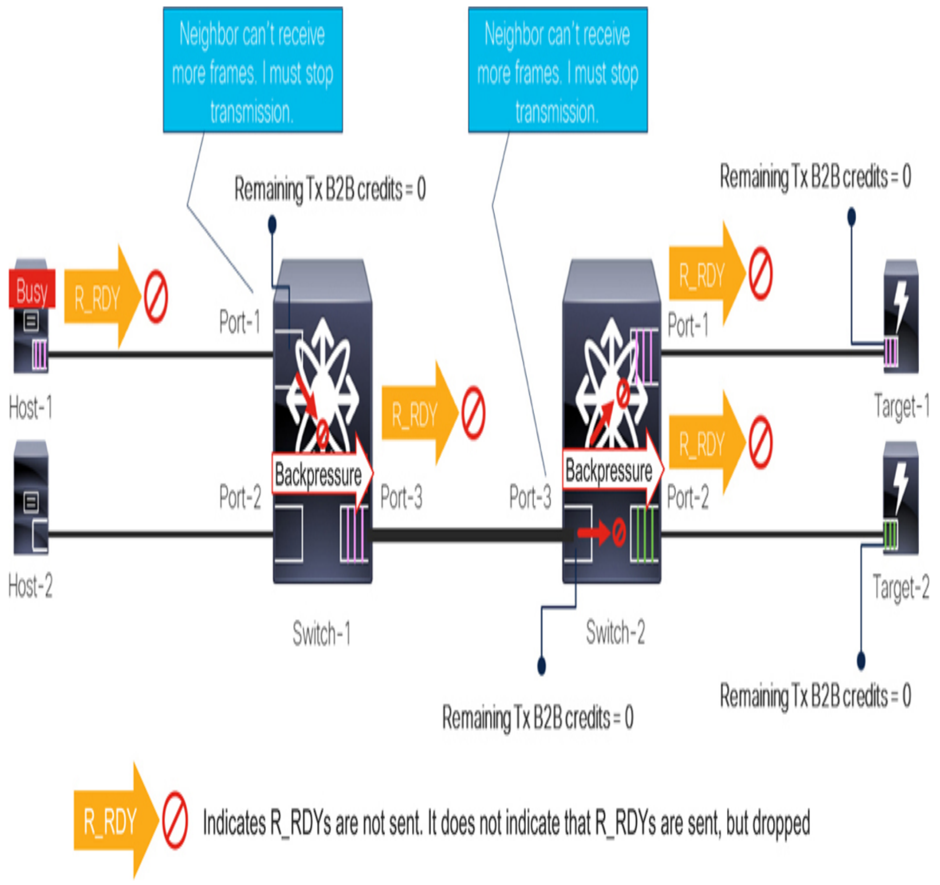

Neighbor can't receive more frames. I must stop transmission.
Neighbor can't receive more frames. I must stop transmission.
Remaining Tx B2B credits = 0
Remaining Tx B2B credits = 0
Busy
R_RDY
Port-1
Host-1
R_RDY
Backpressure
Port-2
Port-3
Host-2
Switch-1
Port-3
Backpressure
Switch-2
R_RDY
Port-1
R_RDY
Port-2
Target-1
Target-2
Remaining Tx B2B credits = 0
Remaining Tx B2B credits = 0
R_RDY
Indicates R_RDYs are not sent. It does not indicate that R_RDYs are sent, but dropped

Figure 2-12 *Congestion due to a slow-drain device*

On Switch-2, ingress frames on Port-1 and Port-2 quickly consume all the receive buffers because the frames can't be switched to Port-3. Because these buffers are not freed up for re-use, Port-1, and Port-2 reduce the rate of sending R_RDYs to Target-1 and Target-2 which ultimately results in zero remaining-Tx-B2B-credits on Target-1 and Target-2.

In the end, as shown in Figure 2-13, one culprit device (Host-1) potentially victimizes many other devices (Host-2, Target-1, and Target-2) connected to the same fabric because of congestion spreading.

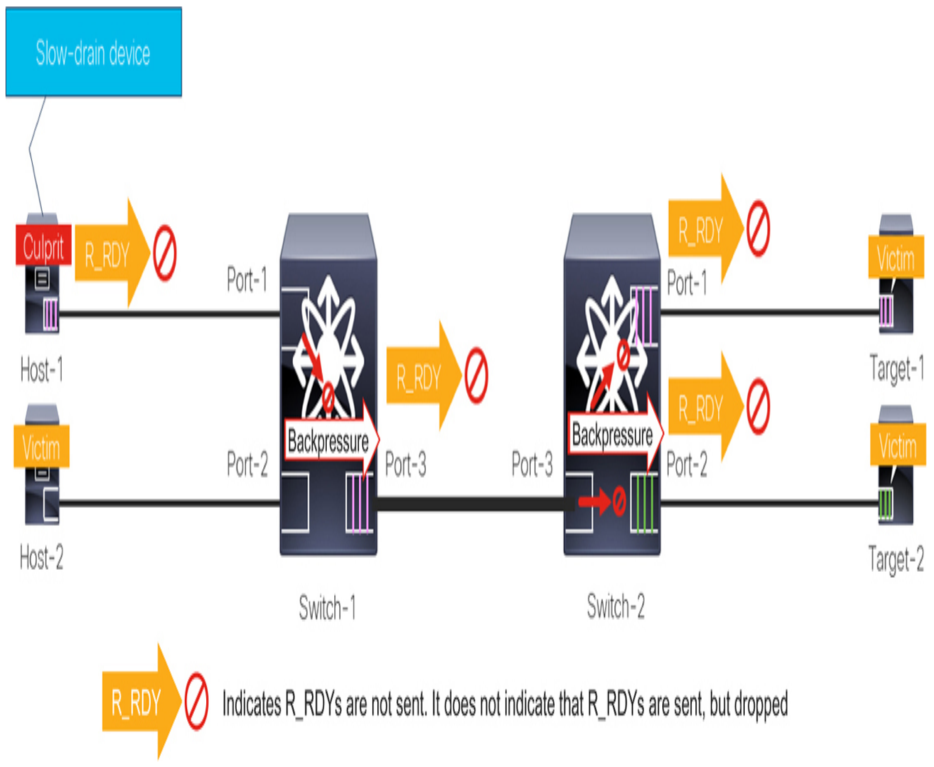


Figure 2-13 *Congestion due to a slow-drain device — Culprits and Victims*

Congestion due to Over-utilization

Congestion due to over-utilization happens when a switch receives more traffic destined for a specific link than the capacity of that link. The link transmits at its full capacity and its remaining-Tx-B2B-credits don't fall to zero. To equalize the ingress frame rate with the egress frame rate, the switch reduces the rate of sending the R_RDYs to the directly attached neighbor(s) in the direction of upstream traffic. Once the directly attached neighbor(s) has zero remaining-Tx-B2B-credits they must stop sending frames. When they stop sending, in turn, they stop accepting frames from their upstream neighbor(s) causing backpressure that spreads in the direction of the ultimate source of the frames. Although the end-to-end backpressure to equalize the traffic rate is expected behavior, it adversely affects other traffic that shares the same network path. This effect is similar as caused by slow-drain.

A high-level overview of over-utilization and its reasons are explained in Chapter 1, the section on *Congestion Due to Over-Utilization of a Link*. This section expands on the same details with a focus on host-edge links because this is the most widely reported condition in production Fibre Channel fabrics, although congestion due to over-utilization can happen on any link in a fabric. Over-utilization on an ISL is explained in a later section.

Congestion due to Over-Utilization on Host-edge Links

Congestion due to over-utilization on host links happens when hosts have requested more data from their targets than can be sent on the host-edge link.

Consider the example of Figure 2-14, which shows a part of an edge-core Fibre Channel fabric. Hosts-1 and Host-2 connect to Switch-1, whereas Target-1 and Target-2 connect to Switch-2. During a recent refresh, the targets were upgraded to all-flash arrays with 32GFC connections. The ISLs were upgraded to 64GFC. The hosts, however, haven't been upgraded yet and their links still operate at 8GFC.

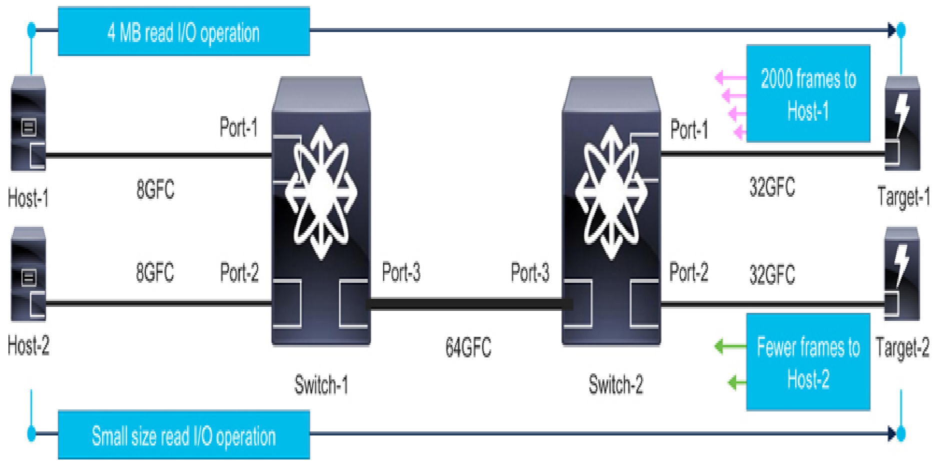


Figure 2-14 *A typical edge-core storage network*

Host-1 and Host-2 perform (read and write) I/O operations with Target-1 and Target-2 respectively. These I/O operations are of small size with enough inter-I/O gap that doesn't result in congestion.

Suddenly, Host-1 starts initiating large-size (such as 4-MB) read I/O operations to Target-1 while Host-2 continues with smaller-size read I/O operations to Target-2. As explained earlier, a 4-MB read I/O operation results in approximately 2,000 frames.

Both the targets have extremely fast response times because they are All-Flash arrays. They read data from the backend storage media and start sending frames to Host-1 and Host-2 at the speed of the directly connected links, which is 32GFC. On Switch-2, Port-1 and Port-2 switch the ingress frames to Port-3 without any issues because the egress port's speed (64GFC of Port-3) is double that of the ingress ports' speed (32GFC of Port-1 and Port-2). In this example, the ISL has enough capacity even if both the targets transmit at full capacity at the same time.

If the ISL's speed was slower, such as 32GFC, the collective ingress rate of Port-1 and Port-2 might have been faster than the maximum possible rate on Port-3 (32GFC). In this condition, Switch-2 must equalize the ingress frame rate with the egress frame rate by reducing the rate of sending R_RDYs on Port-1 and Port-2 to the two targets. This condition would result in congestion due to over-utilization on the ISL, which is explained in more detail in the section on *Over-utilization of ISL*. This section focuses on the over-utilization of the host-edge links so for this example, let's assume that ISL is operating at 64GFC and it's not a bottleneck.

As Figure 2-15 shows, the ISL carries the frames at the same rate as sent by the targets. Switch-1, however, can send frames to the destination only at 8GFC. In other words, Switch-1 receives frames four times faster than it can send them. Because of this difference in the ingress and the egress rate, Switch-1/Port-3's ingress buffers are quickly consumed, and thus, it reduces the rate of sending the R_RDYs to Switch-2/Port-3.

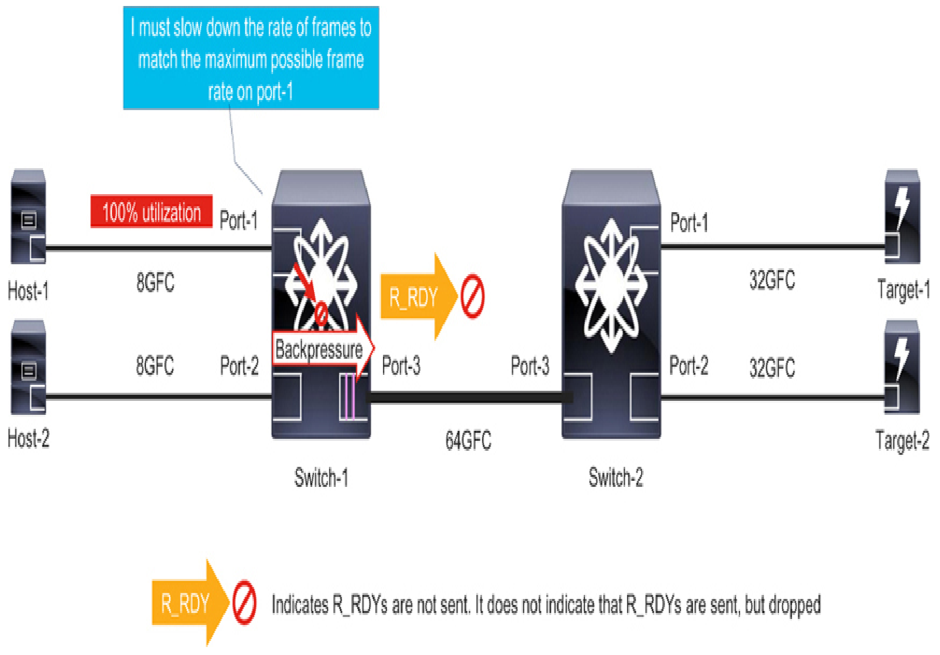

I must slow down the rate of frames to match the maximum possible frame rate on port-1
100% utilization
Port-1
Host-1
8GFC
R_RDY
Backpressure
Port-2
8GFC
Host-2
Port-3
Port-3
64GFC
Switch-1
Switch-2
Port-1
32GFC
Target-1
Port-2
32GFC
Target-2
R_RDY
Indicates R_RDYs are not sent. It does not indicate that R_RDYs are sent, but dropped

Figure 2-15 *Internal backpressure within a switch because of speed mismatch*

In this condition, Switch-2 reduces the frame transmission rate to Switch-1, regardless of the destination of the frame. That means, frame rate from Target-1 to Host-1 reduces, and frame rate from Target-2 to Host-2 also reduces. Reduced frame rate from Target-1 to Host-1 is expected because Host-1's large size read I/O commands result in more frames than can be sent to it on its directly connected link. But reduced frame rate from Target-2 to Host-2 is an undesired side effect while Host-2's read I/O commands don't result in as many frames.

The same sequence of events repeats on Switch-2 as well. Because Port-3 can only accept frames from Port-1 and Port-2 at the rate it can transmit them, it applies an internal backpressure to Port-1 and Port-2 to achieve frame rate equalization, as shown in Figure 2-16.

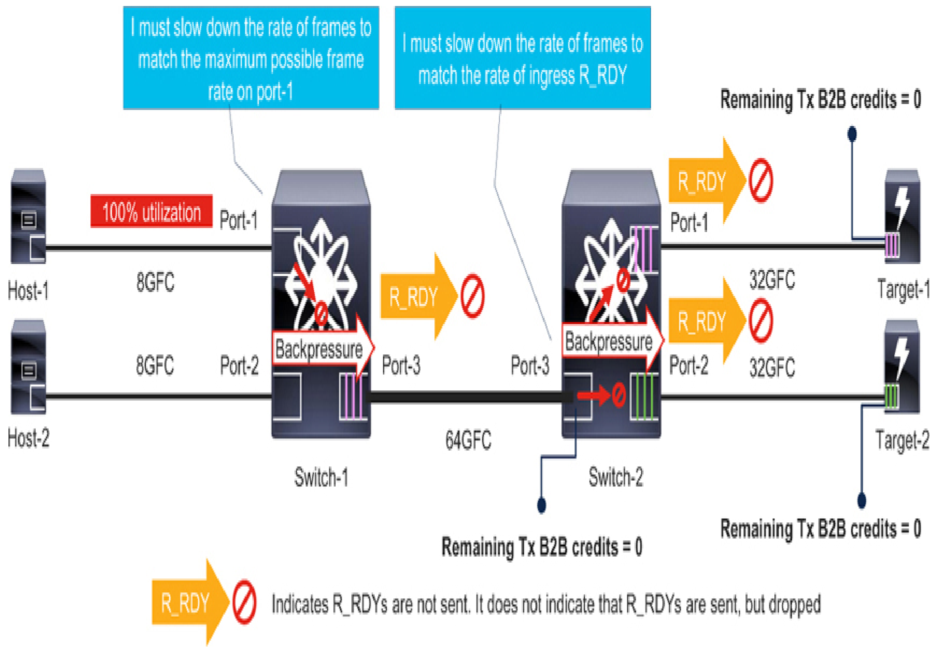

I must slow down the rate of frames to match the maximum possible frame rate on port-1
I must slow down the rate of frames to match the rate of ingress R_RDY
Remaining Tx B2B credits = 0
100% utilization
Port-1
Host-1
8GFC
Host-2
8GFC
Port-2
Backpressure
R_RDY
Port-3
64GFC
Port-3
Backpressure
Switch-1
Switch-2
R_RDY
Port-1
32GFC
R_RDY
Port-2
32GFC
Target-1
Target-2
Remaining Tx B2B credits = 0
Remaining Tx B2B credits = 0
R_RDY
Indicates R_RDYs are not sent. It does not indicate that R_RDYs are sent, but dropped

Figure 2-16 *Congestion due to over-utilization of the host links*

On Switch-2, ingress frames on Port-1 and Port-2 consume the ingress buffers because the frames can’t be switched to Port-3 fast enough. Because these buffers are not freed up for re-use, Port-1, and Port-2 reduce the rate of sending R_RDY to Target-1 and Target-2 which ultimately results in zero remaining-Tx-B2B-credits on Target-1 and Target-2.

In the end, as shown in Figure 2-17, over-utilization of one host link (Switch-1/Port-1) may victimize many other devices (Host-2, Target-1, and Target-2) connected to the same fabric. This situation is known as congestion due to over-utilization and its effects are very similar to those of slow-drain.

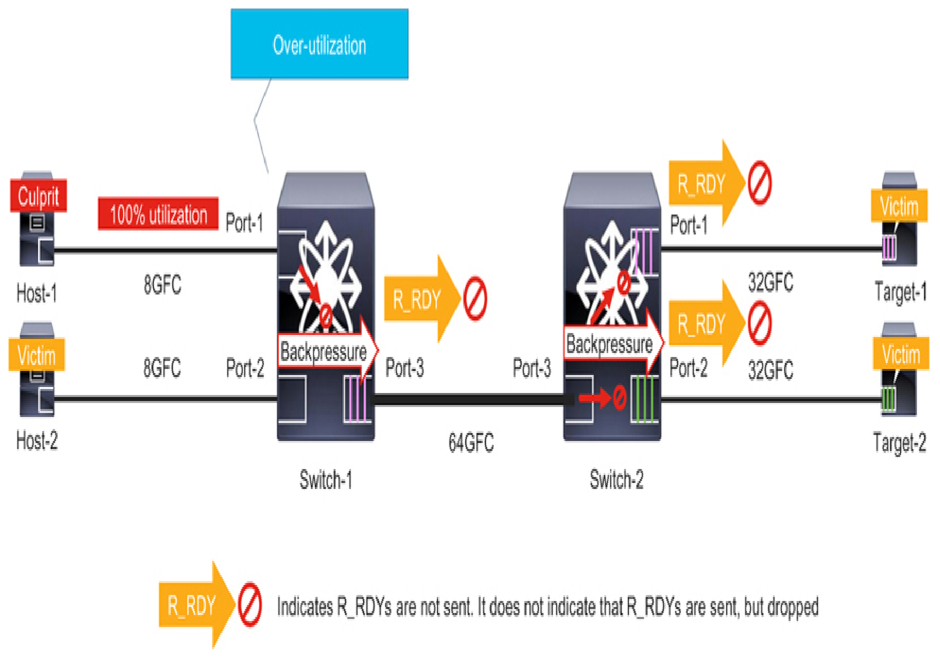


Figure 2-17 *Congestion due to over utilization of the edge links — Culprit and Victims*

The Culprit Host

In this case, is Host-1 really a culprit? The answer depends on whether to look from Host-1's perspective or fabric's perspective. If looking from Host-1's perspective, calling it a culprit may not be entirely correct because it's not doing anything it shouldn't be doing. It connects to the switch at 8GFC and it receives traffic at full capacity. Also, the link between Host-1 and Switch-1/Port-1 is not congested because it has enough credits to send the frames at full capacity. However, if Host-1 was initiating several large read I/O operations to multiple targets closely spaced together then perhaps the case could be made that the host is requesting more data than it possibly could receive on its 8GFC link. It's the responsibility of a device to control how much traffic it sends or receives. In this case, the device is asking for more data than can be sent on its directly connected link. Because of this reason, calling Host-1 a culprit is justified.

Comparing Congestion due to Slow-drain and Over-utilization

Many people use slow-drain as a generic term to refer to congestion in a Fibre Channel fabric. But the reality is that congestion due to slow-drain and congestion due to over-utilization have different root causes, although they look similar.

This section explains the effect on a host, host link, and the fabric when a host causes congestion due to slow-drain and over-utilization.

Effect on the Culprit Host

When a host acts as a slow-drain device, its exact behavior depends upon the severity of the congestion it causes. Under mild congestion, the host continues to perform read and write I/O operations and the hosted applications continue to function with reduced performance. Under moderate congestion, the host capacity to perform read and write I/O is adversely affected, and thus the hosted applications are significantly affected. Under severe congestion, the host observes issues at an operating system level, for example, crashes and unresponsiveness. Because of the vast variety of applications, operating systems, and production environments, a general categorization is not possible. At a minimum, the host has some issues making it slow in processing the incoming frames. In other words, the host isn't healthy.

On the contrary, during congestion due to over-utilization, the host is healthy. It's getting the full worth of its resources because it's receiving traffic at the full capacity of the directly connected link. The performance of the application on this host doesn't worsen or remain the same as compared to the period of no congestion.

Effect on the Culprit's Port and Its Connected Switchport

The port on a slow-drain device reduces the rate or even stops sending of R_RDYs to the connected switchport. As a result, the switchport has fewer remaining-Tx-B2B-credits leading to reduced egress utilization of the switchport.

In contrast, during congestion due to over-utilization, the end device port doesn't reduce the rate of sending of R_RDY, and hence, the connected switchport has enough remaining-Tx-B2B-credits to maintain full utilization of the link in the egress direction.

Overall, during slow-drain, the switchport is frequently at zero remaining-Tx-B2B-credits and the port is not highly utilized. On the contrary, during overutilization, the switchport is rarely at zero remaining-Tx-B2B-credits, and it is highly utilized. This difference makes the foundation for detecting the two types of congestion in production fabrics. Refer to Chapter 3 for more details.

Effect on the Fabric

As Figure 2-18 shows, the effect on the fabric is the same during congestion due to slow-drain and congestion due to over-utilization. In both cases, the ISL switchport (Switch-1/Port-3) reduces the rate of sending R_RDYs to the upstream port (Switch-2/Port-3). In both cases, congestion spreads toward the traffic source. Hence, both types of congestion look similar if observed from the upstream port or other links in the upstream traffic path (Switch-2/Port-3 or the target ports). Because of this reason finding the source of congestion is necessary to find the cause of congestion.

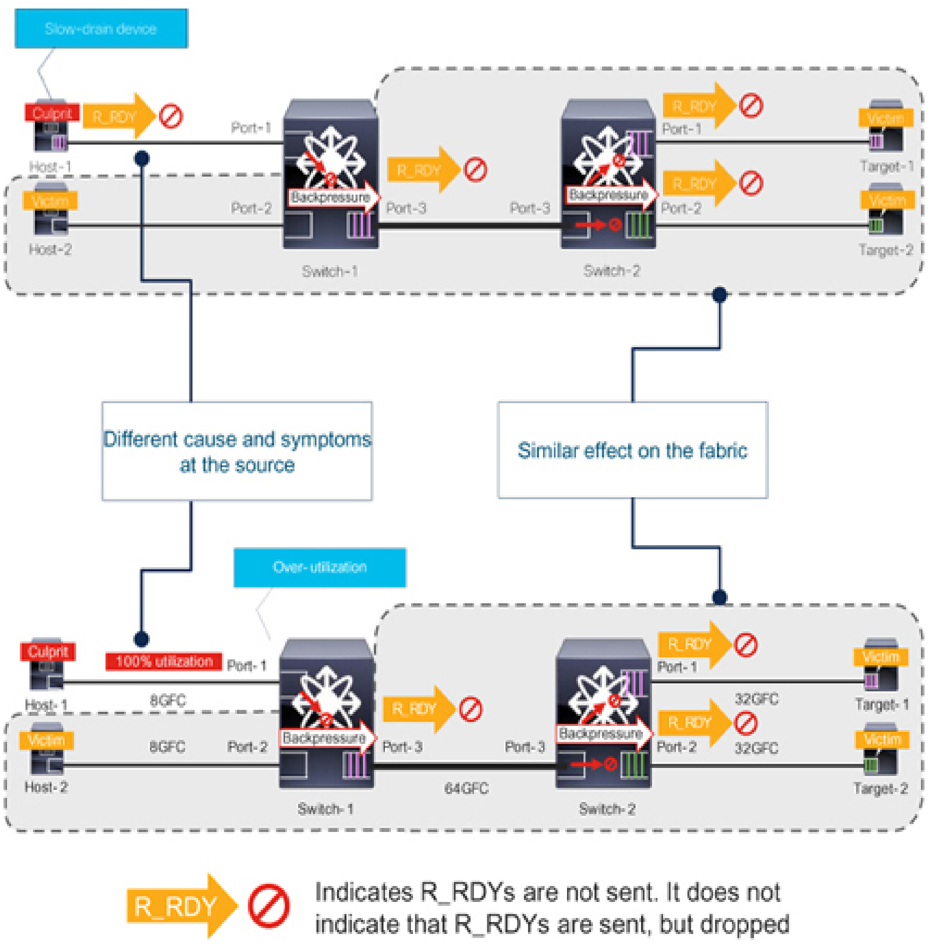


Figure 2-18 *Comparing the effect of congestion due to slow-drain and over-utilization in a Fibre Channel fabric.*

Although Figure 2-18 shows only 4 end devices, production networks may have hundreds or thousands of devices. The devices that are adversely affected by the culprit are known an victims. These victims can be further categorized as direct victims, indirect victims, and same-path victims. The use of this categorization will become clear in Chapter 5.

Congestion in Single-switch Fabrics

Contrary to common belief, congestion affects single switch Fibre Channel fabrics also, although the number of affected devices in a single-switch fabric is fewer as compared to multi-switch fabric (edge-core and edge-core-edge fabrics) where thousands of end-devices share the network paths.

Consider the example in Figure 2-19 to understand the effect of congestion in a single switch fabric. Host-1 receives traffic from Target-1, while Host-2 receives traffic from Target-2.

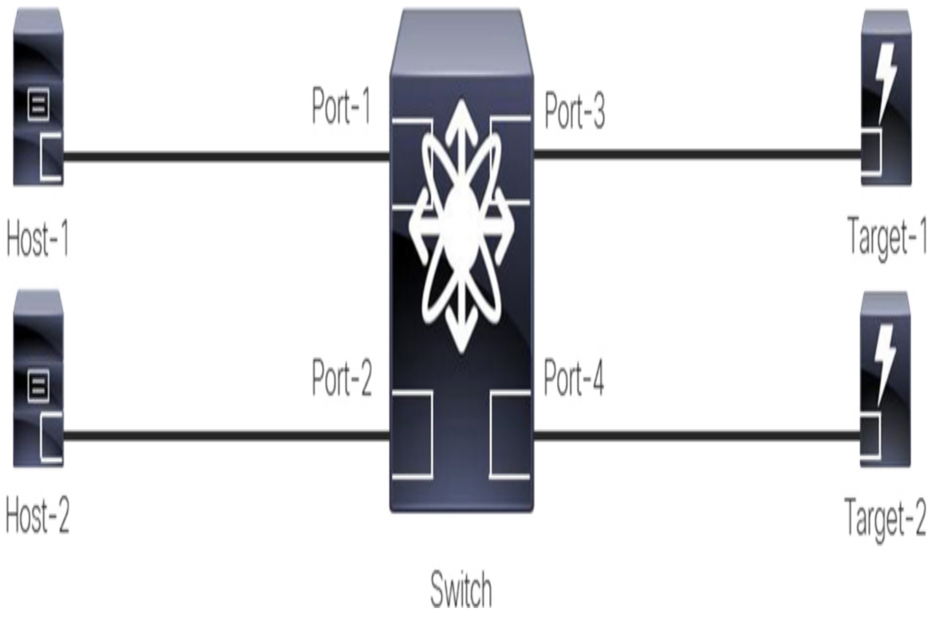


Figure 2-19 *Congestion in a single-switch fabric*

If Host-1 becomes a slow-drain device (reduces the rate of sending of R_RDY), the remaining-Tx-B2B-credits fall to zero on Port-1, and therefore Port-3 can’t switch frames to Port-1. Consequently, Port-3 reduces the rate of sending of R_RDYs to Target-1. Overall, the traffic from Port-1 to Host-1 and Target-1 to Port-3 is affected. But the traffic from Port-2 to Host-2 and Target-2 to Port-4 is unaffected. This is how a single switch fabric reduces the spread of congestion.

Consider the same example under congestion due to over-utilization. If Target-1 sends frames to Host-1 faster than the link speed of Port-1, Port-3 applies backpressure to Target-1 by reducing the rate of R_RDY. In this case also, traffic from Target-2 to Host-2 continues to flow normally.

A production storage network, however, rarely has a one-to-one traffic pattern. Figure 2-19 helps in understanding the basics, but it is far from reality. Based on the number of hosts that receive traffic from Target-1, the effect of congestion is seen in all of them because Port-3 slows down all the traffic from Target-1 regardless of destination. With just two hosts, as shown in Figure 2-19, if both hosts receive traffic from both targets, Host-2 also gets victimized. Hence, the exact spread of congestion depends on the traffic pattern—less spread with one-to-one traffic and more spread with many-to-many traffic.

Overall, even single switch fabrics are affected by congestion, but the spread is less compared to edge-core and edge-core-edge fabrics.

Congestion in an ISL

An ISL can be the source of congestion in a Fibre Channel fabric, although such conditions are less common and relatively easier to detect and solve. This is because an ISL is just an interconnect in the network. It's not the source or the destination of the frames. Also, ISLs are in greater control of the people who manage the networks, which reduces organizational silos thus making the detection and resolution of issues faster.

The common reasons for ISLs to cause congestion in a Fibre Channel fabric are the following:

- The existence of congestion due to an edge device, such as slow-drain or over-utilization — This isn't really a cause of congestion by ISL.
- Over-utilization of an ISL.
- Bit errors on ISL.
- Lack of B2B credits for the distance, frame size, and speed of an ISL.

Congestion Spreading due to Edge Devices

As seen in the previous examples, congestion spreads to ISLs because of a slow-drain device or over-utilization of an edge link. In such cases, although ISLs are not the source of congestion, the effect is seen by the ISLs, which often leads to misinterpretation. For example, in Figure 2-17 and Figure 2-13, Port-3 on Switch-2 doesn't have enough remaining-Tx-B2B-credits. But the source of this condition is Host-1, which is a slow-drain device or the cause of over-utilization of its edge link. Finding the root cause of congestion in this example is easy. But finding the root cause in a production environment with thousands of end devices may take hours. Because of this reason, while troubleshooting congestion issues, be aware that the most common reason for a congested ISL is not the ISL itself. Rather, it’s the existence of a congested end device. Suspect this reason before other reasons that are explained next.

Over-utilization of ISL

Over utilization of an FC ISL causes backpressure towards the source of the traffic. This type of congestion is the same as the congestion due to over-utilization of the edge links (Refer to the section on Congestion due to Over-utilization of the Edge Links). The only difference now is that the over-utilized link is an ISL and the source of congestion.

Following are the two common conditions when an ISL in a Fibre Channel fabric gets over-utilized:

- When an ISL is highly utilized.
- When an ISL is not the fastest link in a fabric.

High Utilization of an FC Link

For practical purposes, treat high and full link utilization occurrences at the same severity as over-utilization unless proven otherwise. To know why, refer to Chapter 1, the sections on *Defining Full Utilization vs High Utilization vs Over-Utilization* and *Monitoring Full Utilization vs High Utilization vs Over-Utilization*.

An ISL Not Being the Fastest Link

Over-utilization may occur when the speed of the ISL is slower than the speed of the edge links to end devices that exchange traffic via this ISL. Although the collective bandwidth after aggregating multiple ISLs may be higher than the bandwidth of an edge link, the slower operational speed of a member of an aggregated link (for example, port-channel) than the speed of the edge links causes the problem. Remember, speed and bandwidth are different. What matters here is the speed of a link.

Refer to the example in Figure 2-20. The ISL has a bandwidth of 64 Gbps because of aggregating four physical links operating at 16GFC. Target-1 is an all-flash array that connects via a 32GFC link, and thus it sends frames at 32GFC speed. But Switch-2/Port-3 can send frames belonging to an I/O operation (mapped to a Fibre Channel Exchange) only via a single 16GFC link. This happens when Switch-2 uses a load-balancing scheme based on Source ID, Destination ID, and Exchange ID (the default setting on Cisco Fibre Channel switches). If there's another I/O operation, frames belonging to it may take the other physical links in the aggregated link, but the frames belonging to an Exchange must traverse through the same physical link.

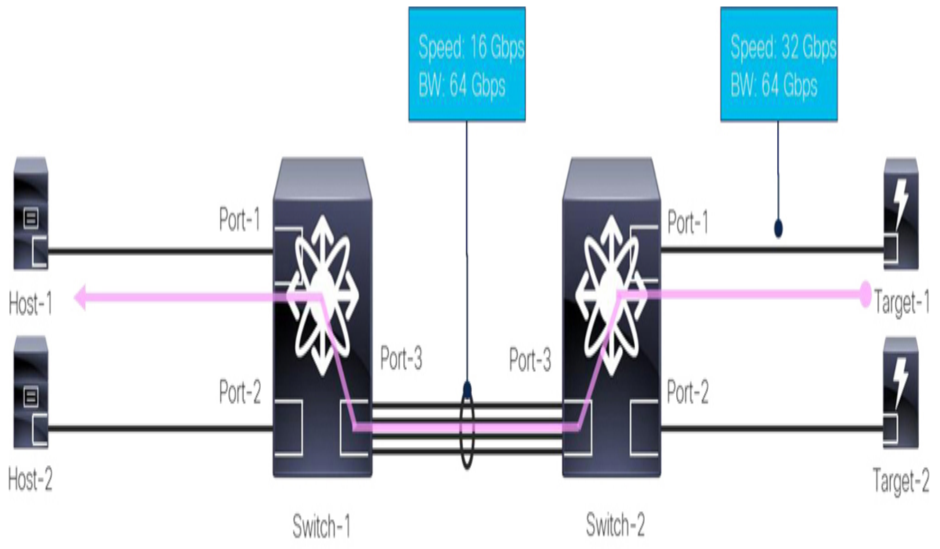


Figure 2-20 *An example of congestion caused by an ISL that's operating at a slower speed than the edge links.*

This speed-mismatch condition between the edge links and an ISL may make the ISL a source of congestion. This condition often goes unnoticed because most production networks monitor only the aggregated bandwidth of network links. As a result, the ISL of Figure 2-20 doesn't show high utilization, yet it becomes a source of congestion because of the slower speeds of the individual physical links. This congestion spreads to the traffic source (Targets) and hence can be detected there similar to as when the source of congestion is a slow-drain device or an over-utilized edge link.

The only solution to this type of congestion is to ensure that the individual members of an aggregated link must be as fast or faster than the fastest edge links in the fabric. This should be a best practice during the design phase or when upgrading the end devices. Be proactive by avoiding running into this type of congestion.

Lack of B2B Credits for the Distance, Speed, and Frame Size of ISL

When an ISL port has zero remaining-Tx-B2B-credits, the port is no longer able to transmit frames. Because of this, it is no longer able to accept frames from other ports in the switch. This results in other ports' ingress buffers being consumed. Once those ports' ingress buffers are totally consumed the ingress ports must stop sending R_RDYs to their upstream neighbor, which results in congestion spreading. If there's no other reason found for this congestion and the ISL traverses a long distance, it's possible that the connecting ports just don't have enough B2B credits.

If the credits for a link are not enough, its utilization is lower than expected. The remaining-Tx-B2B-credits on the port fall to zero momentarily until the next R_RDY arrives. The low arrival rate of R_RDY leads to lower frame transmission rate. But the frames are not dropped, and the lossless behavior of Fibre Channel is maintained.

In this type of congestion, the lack of sufficient remaining-Tx-B2B-credits is because of the lower than required Tx B2B credits that this port learned from the neighbor during link initialization. Refer to the section on *Return of B2B Credits During Frame Flow* to understand the difference between Tx B2B credits and remaining-Tx-B2B-credits.

To solve this type of congestion, increase the number of B2B credits on both ports of an ISL or move the cables to ports that have a higher number of B2B credits. Be aware that changing the number of B2B credits on a port is disruptive because the port must flap for the change to take effect. For example, changing the number of B2B credits on a port of Cisco MDS switch immediately flaps the port. Be cautious. Find the solution but implement it carefully only during a maintenance window.

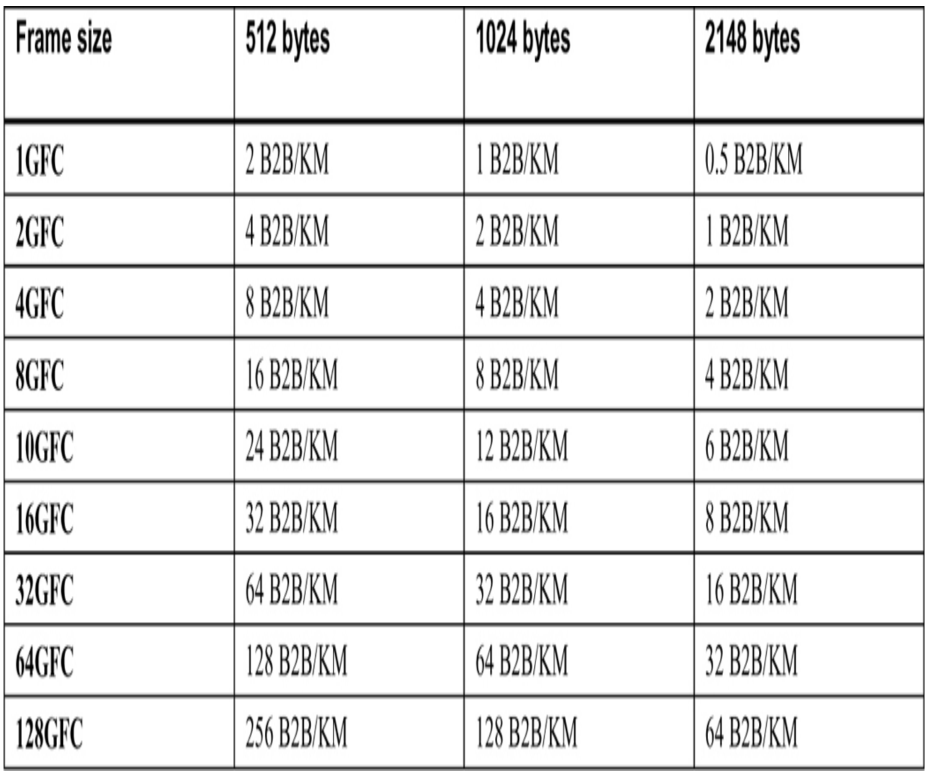
Required Number of B2B Credits on an FC Port

The requirement of B2B credits on the ports of a Fibre Channel link increases with:

- Increase in distance.
- Increase in speed.
- Decrease in frame size.

Table 2-2 provides the minimum numbers of B2B credits required for the per-kilometer length of ISL at different speeds and frame sizes to maintain full utilization.

Table 2-2 *Minimum B2B credit requirements to maintain full utilization of a Fibre Channel link*



Frame size	512 bytes	1024 bytes	2148 bytes
1GFC	2 B2B/KM	1 B2B/KM	0.5 B2B/KM
2GFC	4 B2B/KM	2 B2B/KM	1 B2B/KM
4GFC	8 B2B/KM	4 B2B/KM	2 B2B/KM
8GFC	16 B2B/KM	8 B2B/KM	4 B2B/KM
10GFC	24 B2B/KM	12 B2B/KM	6 B2B/KM
16GFC	32 B2B/KM	16 B2B/KM	8 B2B/KM
32GFC	64 B2B/KM	32 B2B/KM	16 B2B/KM
64GFC	128 B2B/KM	64 B2B/KM	32 B2B/KM
128GFC	256 B2B/KM	128 B2B/KM	64 B2B/KM

For most practical purposes changing the number of B2B credits on a link is not needed for intra-data-center links that are used for high-speed I/O transfer between hosts and storage arrays. Most links are short, often limited by the distance of short-reach transceivers, such as SW SFP+, and the cables, such as OM4 and OM5. The interoperability certifications from vendors ensure that the default number of B2B credits is enough for most intra-datacenter scenarios.

Changing the number of B2B credits is generally required for long-distance FC links that are used for replication and backup traffic between two data centers. For such cases, Table 2-2 provides a quick reference chart for the minimum numbers of B2B credits. But these are theoretical numbers that do not account for any R_RDY delay caused by even minor congestion. To account for practical variations, such as a longer cable than the claimed length, smaller frame sizes, and minor transient delay, adding an extra number of B2B credits is recommended. Some people add 25% extra to the numbers shown in Table 2-2. Also, seek recommendations from the vendor of the Fibre Channel switches and the applications that exchange traffic on these long-distance ISLs.

Detailed Explanation for the Required B2B Credits on an FC port

An FC port can send a frame as long as it has at least one remaining-Tx-B2B-credit. Hence, to keep sending frames at full capacity, it must have enough Tx B2B credits for a round trip on the link, which includes the following events:

- A Frame is completely transmitted on a link including serialization.
- The frame reaches the receiver from the sender.
- The receiver processes the frame.
- The receiver transmits an R_RDY.
- The R_RDY arrives back at the sender.

During this time, the remaining-Tx-B2B-credits must not fall to zero. Table 2-2 provides the number of B2B credits required to maintain full utilization on a link. As mentioned, the rest of this section provides a detailed explanation of how the values in Table 2-2 are calculated, which is a good academic exercise. You can skip the details without missing any practical significance.

Before transmitting data on optical media such as single-mode and multi-mode fiber cables, the sender port transceiver (E.g.,: SFP) converts the electrical signals into optical signals. These signals travel 200,000 km in one second within a fiber cable and 300,000 km/s in vacuum.

After changing the units from km/s to m/ns, the speed of light is 0.2 m/ns in fiber cable. This means that a signal can reach 0.2 m in one nanosecond, or 1 m in 5 ns.

This is the speed of an optical signal. But data transmission is measured in bits. That leads to the next question—what's carried within a signal? If one bit is carried by a signal, the signaling rate is the same as the bit rate. However, if two bits are carried by a signal, the bit rate is double the signaling rate.

The relationship between the signaling rate and bit rate is explained by baud rate, which is measured in symbols per second. Essentially, every signal carries a symbol. A symbol, however, may carry one or more bits. The number of bits carried by a symbol depends on the modulation code.

In Fibre Channel, 32GFC and slower speeds use the NRZ (non-return-to-zero) modulation that carries one bit per symbol. 64GFC and 128GFC use PAM-4 (Pulse Amplitude Modulation 4-level) modulation that carries two bits per symbol. Using the example of 32GFC that uses a baud rate of 28.05 Gigabaud (GBd), because one symbol carries one bit, the bit rate becomes 28.05 Gigabits per second (Gbps).

The next step is to calculate the time taken to transmit a full-size Fibre Channel frame on a 32GFC link. The size of an FC frame is 2,148 bytes or 17,184 bits. An FC link using the B2B flow control mechanism inserts at least 6 primitive signals (such as fill word) between two FC frames. The size of a fill word is 4 bytes. Therefore, six fill words add 24 bytes or 192 bits resulting in 17,376 bits for the size of a frame and adjacent fill words.

At the transmission rate of 28.05 Gbps, a 32GFC link takes 619.46 nanoseconds to transmit a full-sized FC frame.

> 17,376 bits / (28.05 x 10^9 bits per second) = 619.46 x 10^{-9} second

Let’s round this off to 620 ns.

For the sake of simplicity, this section doesn’t account for the overhead because of the encoding and FEC. The encoding scheme at 32GFC is 256b/257b which has an overhead of 1 bit for 256 bits (0.39%). Likewise, the FEC scheme at 32GFC is RS (528,514) which has an overhead of 140 bits for 5140 bits (2.72%). The section on *Data Transmission on Fibre Channel Media* explains this topic in more detail.

Although the transmission rate is defined by the sender, how fast the bits travel on the media is defined by the speed of the light, which is 1 m in 5 ns. As per the transmission rate, a 32GFC link takes approximately 620 ns to transmit an FC frame. Consequently, while the sender transmits the last bit of a frame, the first bit of the frame travels 124 m on the fiber cable.

(620 ns) / (5 ns / 1 m) = 124 m

In other words, the length of a full-size FC frame on a 32GFC link is 124 m. For the sake of simplicity, let’s round this off to 125 m.

Now is the time to bring the B2B flow control into this conversation. A sender sends a frame and decrements its remaining-Tx-B2B-credits by one. It can send the next frame only if the remaining-Tx-B2B-credits are not zero. If the sender and the destination are 1 km apart, the sender sends a frame, which reaches approximately 125 m away, when it’s ready to send the next frame. To transmit at full capacity, the sender must have more than one Tx B2B credit. With 125 m per frame, a kilometer of cable can accommodate eight frames. While the first bit of the first frame enters the receiver, the last bit of the eighth frame leaves the sender. For this duration, to maintain the full utilization of the link, the sender must have eight Tx B2B credits.

Further, the first frame enters the receiver. Assuming the receiver is extremely quick in handling the frames and in making its buffer available for reuse, it sends a R_RDY the next nanosecond when the first frame is received.

The R_RDY travels at the speed of light, which is 1 m in 5 ns in the optical media. By the time it travels the 1 km back to the sender, the sender would have sent 8 more frames which need eight additional Tx B2B credits.

Overall, a 1 km long 32GFC link needs 8 + 8 = 16 Tx B2B credits for continuous transmission at full capacity. That is why Table 2-2 indicates the need of a minimum 16 B2B credits per km.

This is the theoretical best-case number. Practically, more credits are needed because the receiver has a delay between receiving a frame and sending a R_RDY for it and any minor congestion.

Next, let's understand the factors that change the required number of B2B credits on a link.

The first factor is the operational speed, which has a linear relationship with the required number of B2B credits on a link. For example, a 16GFC port takes approximately double the time to transmit a full-size FC frame compared to a 32GFC port. As a result, the length of a frame is 250 m, which is twice the length of a frame at 32GFC. Therefore, a 1 km link can have 4 frames on the cable. Using the earlier explanation, the sender needs 4 + 4 = 8 B2B credits to maintain full utilization of a 16GFC link. Overall, lowering the speed by half reduces the B2B credits requirements by half, whereas doubling the speed increases the B2B credit requirements by two.

The length of the link is another factor with a linear relationship to the required number of B2B credits. For example, a 32GFC link can accommodate a full-size FC frame in 125m. A 1 km long link can accommodate 8 frames. Likewise, a 2 km long link can accommodate 16 frames, and so on. Overall, doubling the length of a link doubles the number of B2B credits required to maintain full utilization.

These two factors—the speed of a link and the length of a link—are relatively simple to understand. They are normally known before a link comes up and their values don't change dynamically during the operation of the link.

The third factor to calculate the required number of B2B credits on a link—Size of the frames—is relatively more difficult to know because the frame size can change dynamically depending on the application.

The frame size on a link is inversely proportional to the number of B2B credits required on a link to maintain full utilization. For example, a 1 km 32GFC link can accommodate eight full-size FC frames. If, however, the frame size is half (approximately 1024 bytes instead of 2148 bytes), the same link can accommodate 16 frames which inflates the required number of B2B credits on this link to 16 + 16 = 32. If the frames are even smaller, for example, 512 bytes, the same link can now accommodate 32 frames resulting in the required number of B2B credits to 64. Overall, the smaller the frame size, the higher the requirement for B2B credits to maintain full utilization of the link.

A Fibre Channel link never carries only one-size frames. The frames that initiate I/O operations and carry the response are closer to 100 bytes. In contrast, data-carrying frames are large and closer to full-sized. Refer to Chapter 5 for more details. The frame size also depends on the amount of data requested by the I/O operations. For example, an I/O operation of 4 KB can result in two full-size frames, each with 2 KB of data. But if the application request only 1 KB of data per I/O, the frames will carry only 1 KB resulting in their size of approximately 1 KB.

The frame sizes can be dynamic, and hence, the calculation of the required number of B2B credits on a link should be based on the average frame size, which leads to the next question - How to calculate the average frame sizes?

Different products offer different approaches to calculating the average frame size on a link. The exact approach should be obtained from the vendor documentation. On a high level, the following approaches may be available:

- An FC port may report bytes/sec and frames/sec. Dividing the first counter by the second gives the average frame size on that link.

- An FC port may report the number of 4-byte words/second and frames/sec. This approach is like the previous approach except that the word must be converted to bytes by multiplying by 4.
- An FC port may report the number of frames belonging to a size bucket. For example, the number of frames between 1-500 bytes, 501-1000 bytes, and so on. This approach provides an improved average frame size calculation.
- Features like SAN Analytics report the amount of data transferred by I/O operations, called I/O size. It can be used to estimate the frame sizes. Refer to Chapter 5 for more details.

That concludes a detailed explanation for calculating the required number of B2B credits on a Fibre Channel link to maintain full utilization.

Buffering and Ability to Absorb Congestion

The ability to absorb congestion increases as the number of buffers (credits) increases. When more buffers are available, a traffic burst consumes many buffers, but may still leave free buffers. When buffers are free, a port in a lossless network does not invoke hop-by-hop flow control to pace ingress traffic, and hence, there is no congestion spreading.

To understand this further, consider the example from Figure 2-9. ISL port on Switch-1 has 500 remaining-Rx-B2B-credits. This port is not returning any credits because of downstream congestion caused by the Host. But Switch-2 does not stop transmission until those 500 credits are used. Recall that one credit is for one frame. If the port is operating at 32GFC, a full-sized FC frame of 2148 bytes takes approximately 600 nanoseconds for transmission. Therefore, consuming 500 credits by 500 frames takes approximately 300 microseconds (600 x 500). In other words, congestion up to 300 microseconds can be absorbed by 500 credits. If Switch-1 had only 50 credits, it could absorb congestion only up to 30 microseconds (time to transmit 50 full-sized frames). If the traffic has a microburst pattern that causes congestion for small durations or if the host becomes only momentarily busy (such as 200 microseconds), 50 credits will spread this congestion to Switch-2 forcing it to stop transmission until a credit is returned, whereas 500 credits will absorb the microburst on Switch-1, and Switch-2 will not stop transmission.

The relationship between credits (buffers) and their ability to absorb congestion which is explained in this section is different from the required number of credits for a long-distance FC link which is explained in the previous section. An increased number of credits for a long-distance FC link is mandatory for sending traffic at its full capacity. These credits are used even without congestion. However, excess credits in general for long or short links are used only during congestion. In the absence of congestion, these excess buffers remain unused.

Dependency on Traffic Pattern

A higher number of buffers can absorb congestion only for specific traffic patterns when the burst of frames can be completely absorbed in the buffers. For other traffic patterns, when the burst is big consisting of more frames than can be stored in the buffers, congestion spreading is only delayed. Production environments generally have a mix of traffic patterns, and hence the higher number of buffers absorbs congestion for at least some traffic patterns, not all.

Effect on Latency

When frames consume a higher number of switch buffers, it results in increased delay for a frame that must wait for other frames to be transmitted ahead of it. In the earlier example, the 500th frame must wait for at least 300 microseconds in the switch buffer. This delay (also known as queuing delay) increases the I/O completion time resulting in application performance degradation. In lossless networks, such as Fibre Channel and lossless Ethernet (covered in Chapter 7, "Congestion Management in Ethernet Storage Networks"), this delay in traffic to some devices is better than spreading the congestion, which would increase the delay in traffic to many more devices.

In lossy networks, the availability of extra buffers (or deep buffers) is debatable. These extra buffers absorb congestion, but they also increase latency. On the other hand, having fewer buffers can have merit because excess frames can be dropped using an Active Queue Management (AQM) mechanism and then TCP can quickly retransmit the lost frames. These details are explained further in Chapter 8, "Congestion Management in TCP Storage Networks."

When compared with lossless networks where frames are not dropped due to buffer overrun, absorbing congestion in a higher number of buffers is better because it limits the spread of congestion, thus avoiding many other devices from being victimized.

Amount of Buffers

Buffers can absorb only micro congestion events. For example, when a device exhibits slow-drain behavior temporarily, such as for a few microseconds or occasional large-size I/O operations resulting in a traffic burst. However, if these conditions sustain longer, congestion eventually spreads. As previously mentioned, 500 frames at 32GFC can be transmitted in approximately 300 microseconds. Hence, congestion of up to 300 microseconds can be absorbed, but congestion still spreads if it sustains longer for a period.

Also, buffers are a finite resource, which is defined when designing a product. Some switches allow changing the default buffers, but there are upper limits. For example, 64GFC ports on Cisco MDS switches when operating as E_Port has 1,000 B2B credits by default, which can be increased to 16,000 by borrowing credits from other ports. In contrast, FC ports on Nexus 93360YC-FX2 have 32 B2B credits, and this number can't be changed, extended, or borrowed from other ports.

Finally, increasing buffers on a product increases its cost. This increased cost is often passed to the buyers (users) or makes the products less competitive to sell in the market. Hence, device manufacturers try to keep a balance between increased buffers and cost.

User Action

Because of the ability to absorb micro congestion, some users have gone as far as increasing the number of credits on the Fibre Channel switchports up to the maximum permissible limits. In lossy Ethernet networks with TCP transport, Active Queue Management (AQM) is quite common, as explained in Chapter 8.

Changing the number of credits on Fibre Channel ports for the purpose of absorbing congestion, if you choose to make this change, must be done after a thorough understanding of the product architecture because there may be additional implementation details, such as the number of buffers, how they are shared among the ports and any side effects of borrowing the credits.

Frame Flow within a Fibre Channel Switch

A Fibre Channel switch uses two types of flow control.

- **B2B flow control**: This is used externally to the switch between the switchports and their directly connected devices. As explained earlier, it paces traffic rate from the directly connected sender to avoid buffer overrun on the receiver. B2B flow control is a standard-based approach.
- **Internal backpressure**: This is used internally to the switch between the egress and the ingress switchports. When the egress switchport is congested, an internal backpressure paces the ingress frame rate on the ingress port to match the egress frame rate on the egress port. Consequently, the ingress switchport applies backpressure to its directly connected sender using B2B flow control. The implementation of the internal

backpressure within a Fibre Channel switch depends on its architecture.

This section explains the frame flow path and internal backpressure mechanism using the example of the Cisco MDS switches. Understanding these details are useful for tracing the source of congestion from ingress switchports to egress switchports.

Frame Switching within a Cisco MDS Switch

A central arbiter in a Cisco MDS switch ensures centralized-coordinated forwarding among all the ports. When an egress port doesn't have remaining-Tx-B2B-credits to send a frame, the central arbiter doesn't give a grant to any ingress port to send frames to the egress port. This achieves the internal backpressure mechanism. The frame forwarding, however, is achieved by the crossbars that provide a dedicated, non-blocking, and non-oversubscribed path between the switchports. For easier understanding, central arbiter can be compared to traffic lights on an intersection, whereas the crossbar is like the actual path that the vehicles drive through.

A Fibre Channel frame goes through the following steps from an ingress port to an egress port on a Cisco MDS switch.

1. Fibre Channel frames enter a switchport through an optical transceiver (Figure 2-21).

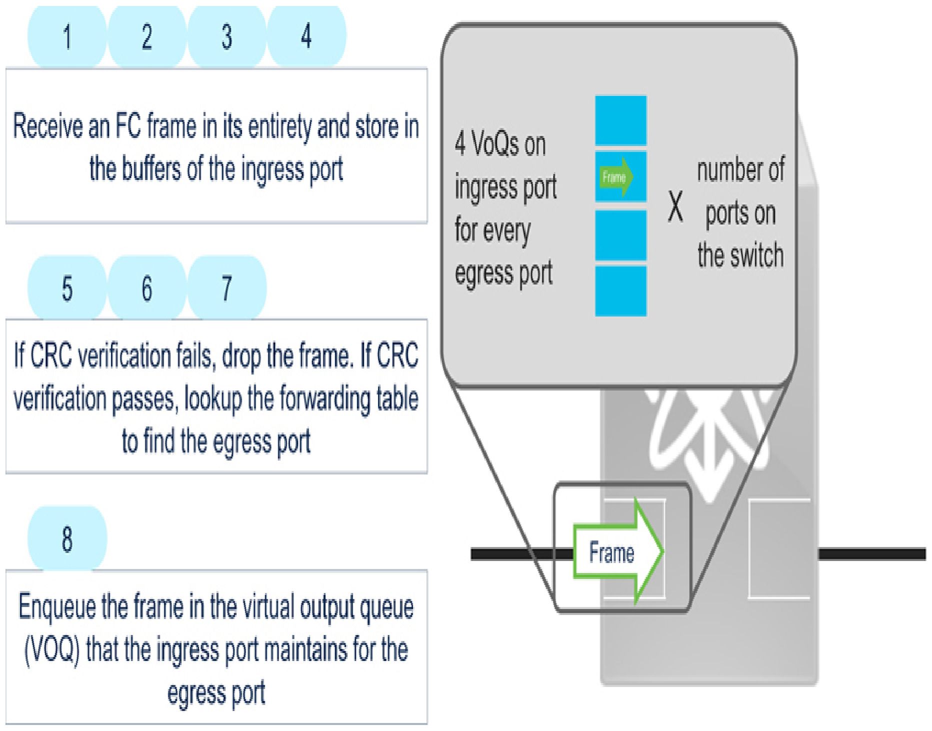

1
2
3
4
Receive an FC frame in its entirety and store in the buffers of the ingress port
5
6
7
If CRC verification fails, drop the frame. If CRC verification passes, lookup the forwarding table to find the egress port
8
Enqueue the frame in the virtual output queue (VOQ) that the ingress port maintains for the egress port
4 VoQs on ingress port for every egress port
Frame
X
number of ports on the switch
Frame

Figure 2-21 *Frame switching path within Cisco MDS switch — Ingress port*

2. A switchport has an associated physical (PHY) layer and a media access controller (MAC) layer. The PHY layer converts the received optical signals into electrical signals and sends the electrical stream of bits into the MAC layer (Figure 2-21).
3. The MAC layer deciphers Fibre Channel frames from the incoming bit stream by identifying start-of-frame (SOF) and end-of-frame (EOF) delimiters, as well as other Fibre Channel primitives (Figure 2-21).
4. The frame is received in its entirety and stored in the ingress buffers of the port (Figure 2-21).
5. The MAC verifies the CRC field in the Fibre Channel frame by calculating the Frame Check Sequence (FCS) (Figure 2-21).
6. If the CRC verification fails, the frame is dropped which immediately frees up the buffer for re-use. As a result, R_RDY is sent on the ingress FC port (Figure 2-21).
7. If the frame passes the CRC verification, the ingress port determines the egress port for this frame by doing a

lookup in the forwarding tables (Figure 2-21).

8. The ingress port queues the frames in the virtual output queue (VOQ). A switchport has four dedicated VOQs for every egress port in the switch. The four VOQs per egress port support four Quality of Service (QoS) levels. A frame is queued in a VOQ based on the egress port and the QoS for that frame (Figure 2-21).
9. The ingress port sends a request to the central arbiter, seeking permission to transmit the frame to the egress port via the crossbar (Figure 2-22).

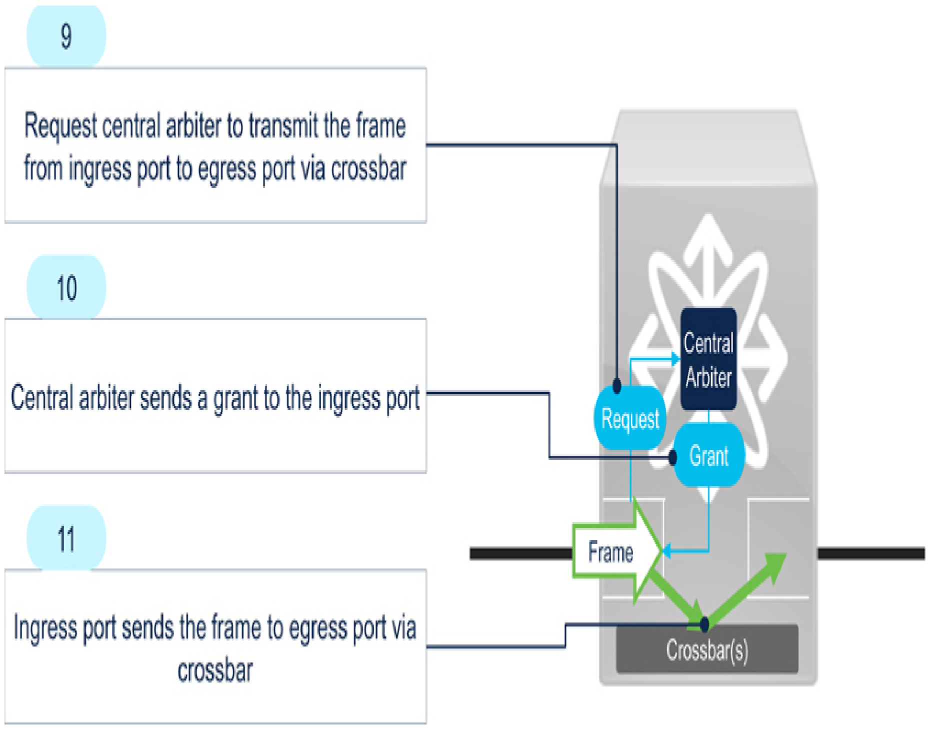

9
Request central arbiter to transmit the frame from ingress port to egress port via crossbar
10
Central arbiter sends a grant to the ingress port
11
Ingress port sends the frame to egress port via crossbar
Central Arbiter
Request
Grant
Frame
Crossbar(s)

Figure 2-22 *Frame switching path within Cisco MDS switch — Ingress port to Egress port*

10. The central arbiter sends a grant if the egress port has at least one remaining-Tx-B2B-credits. But, the arbiter may not immediately grant a request to the ingress port if the egress port has zero remaining-Tx-B2B-credits. The frame keeps waiting in the ingress port buffer until the arbiter grants the request to transmit the frame to the egress port (Figure 2-22). As a result, the grant requests may be retried.
11. After receiving a grant, the ingress port transmits the frame to the egress port via one of the crossbar links

(Figure 2-22).

12. After the frame is sent to the egress port in its entirety and the buffer is available for re-use, an R_RDY is sent on the ingress FC port (Figure 2-23).

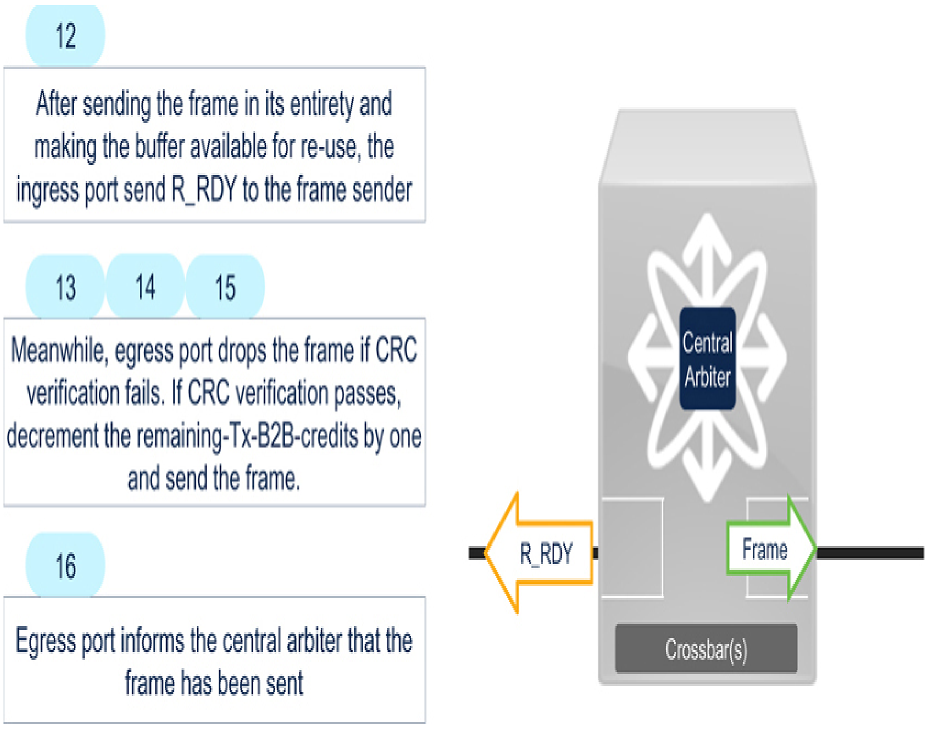

12
After sending the frame in its entirety and making the buffer available for re-use, the ingress port send R_RDY to the frame sender
13
14
15
Meanwhile, egress port drops the frame if CRC verification fails. If CRC verification passes, decrement the remaining-Tx-B2B-credits by one and send the frame.
16
Egress port informs the central arbiter that the frame has been sent
Central Arbiter
R_RDY
Frame
Crossbar(s)

Figure 2-23 *Frame switching path within Cisco MDS switch — Egress port*

13. The egress port verifies the validity of the frame by doing another CRC calculation (Figure 2-23). If the CRC verification fails, the frame is dropped. If the CRC verification passes, the egress port MAC encodes the frame by inserting the SOF, EOF, and other Fibre Channel primitives (Figure 2-23).

14. The egress port checks its remaining-Tx-B2B-credits if they are greater than zero. If yes, the egress port decrements its remaining-Tx-B2B-credits by one (Figure 2-23).

15. PHY converts the electrical signals into optical signals and transmits the frame on the cable (Figure 2-23).

16. The egress port informs the central arbiter that the frame has been transmitted (Figure 2-23).

Frame Switching Architecture of a Fibre Channel Switch

In addition to the lossless frame switching among the ports, a Fibre Channel switch has additional architectural details, which are important to understand for effective troubleshooting of congestion or performance issues.

Location of Buffers — Ingress, Egress, or Both

A Fibre Channel switch may buffer the frames on the ingress port, egress port, or both. For example, Cisco MDS switches buffer most frames on ingress ports but do have a small number of egress buffers. When the egress port has zero remaining-Tx-B2B-credits, the central arbiter doesn't grant a request to the ingress port to send frames to the egress port via the crossbar. As a result, the frame keeps waiting in the buffers of the ingress ports. In contrast, Cisco Nexus 9000 switches buffer frames on egress ports. But for lossless traffic, such as Fibre Channel and FCoE, buffers are reserved for ingress ports. These details are explained further in Chapter 7.

Number of Buffers

Some Fibre Channel switches have a higher number of buffers, which can help absorb traffic microbursts rather than let microbursts spread the congestion to the source of the traffic. A higher number of buffers, however, may not help much in the condition of sustained congestion. This topic is explained in detail in the section on *Buffering and Ability to Absorb Congestion*.

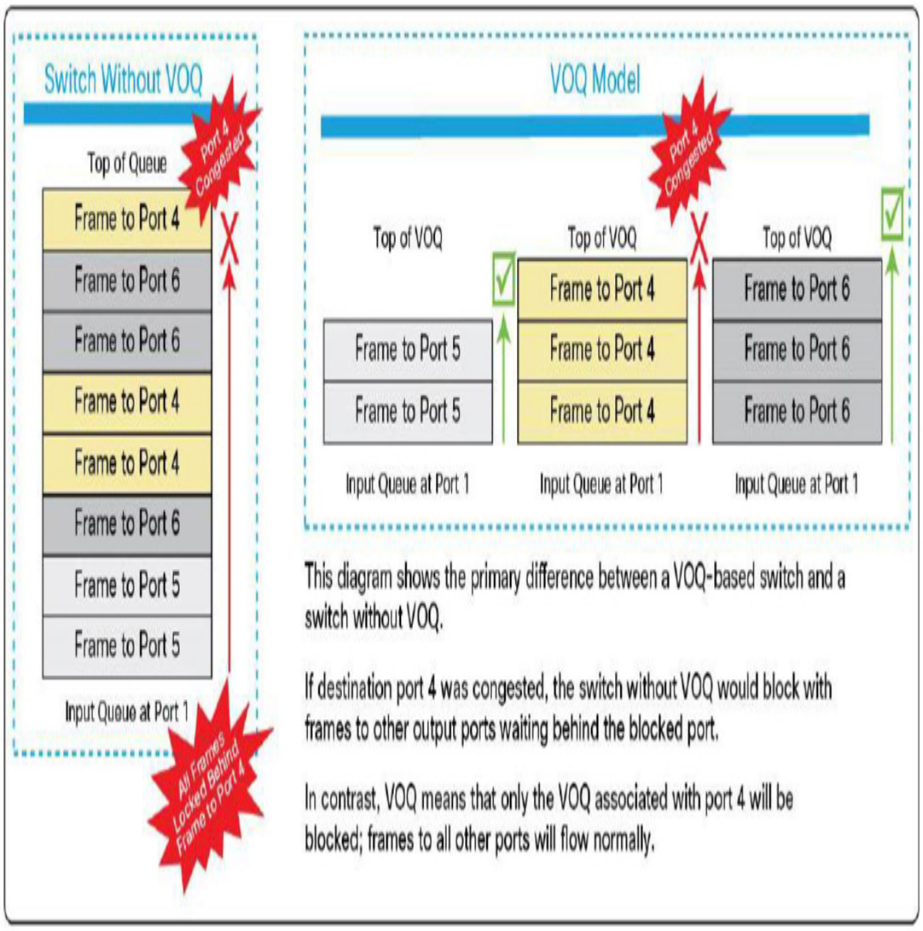
A higher number of buffers are required to maintain full utilization on long-distance FC links. Refer to the section on *Required number of B2B Credits on an FC Port* for more details.

Buffers on switches are a finite resource. Although some switches allow borrowing buffers from the adjacent ports, buffers can't be dynamically increased beyond the capability of that switch. Consider this factor before making a buying decision for a switch.

Preventing Head-of-Line Blocking

Head-of-line blocking occurs when the frame at the head of the queue cannot be sent because of congestion at its output port, while the frames behind this frame are blocked from being sent to their destinations, even though their respective output ports are not congested.

Cisco MDS switch prevents head-of-line blocking using virtual output queues (VOQs). Instead of a single queue, separate VOQs are maintained at all ingress ports. Frames destined for different ports are queued to separate VOQs. Individual VOQs can be blocked, but traffic queued for different (nonblocked) destinations can continue to flow without being delayed behind frames waiting for the blocking to clear on a congested egress port (Figure 2-24). Cisco MDS switches supports up to 4096 VOQs, allowing to address up to 1024 destination ports per chassis, with 4 QoS levels.



This diagram shows the primary difference between a VOQ-based switch and a switch without VOQ.

If destination port 4 was congested, the switch without VOQ would block with frames to other output ports waiting behind the blocked port.

In contrast, VOQ means that only the VOQ associated with port 4 will be blocked; frames to all other ports will flow normally.

Figure 2-24 *Virtual Output queues to prevent head-of-line blocking*

It may be a point of confusion that if VOQs prevent head-of-line blocking, why does an FC port still apply B2B flow control towards the traffic source? This is because the number of buffers on a port is a finite resource and when buffers are not available, the B2B flow control mechanism slows down all the ingress traffic on that port. In this condition, the VOQs for non-congested egress ports are empty, whereas the VOQs for the congested egress ports have frames waiting in them, consuming the total number of buffers on the ingress port. When the ingress port is ready to receive the next frame, and if the next frame's egress port is not congested, having a dedicated VOQ helps it to reach the egress port while the frames for the congested port keep waiting in their VOQ.

Remember, queuing in VOQ is different from buffering. A free buffer (credit) is required to receive a frame when using B2B flow control. Only then, the frame can be queued to a VOQ and benefit from it.

To clear up this confusion about the benefit of VOQs, consider a switch that doesn't use VOQs but just a single ingress queue. If egress port-1 is congested as soon as a frame arrives for that destination port, it is queued and remains queued. Then if another frame is received for a different destination port (say port-2) it is queued behind the frames for port-1. So, even though port 2 is uncongested it is affected by the congestion at port 1. VOQs prevent this.

Store-and-Forward vs Cut-Through Switching

A switch using store-and-forward architecture receives the whole frame before starting to send the frame through the switch to the egress port. In contrast, a switch using cut-through architecture starts transmitting a frame on the egress port before the whole frame is received.

Both architectures have pros and cons. A switch using store-and-forward architecture can detect and drop CRC-corrupted frames and prevent their propagation to the egress port. However, as a trade-off, waiting to receive the whole frame marginally increases the port-to-port switching latency. This marginal latency increase may not be visible to applications because there are many other elements with a much bigger latency contribution like storage media, adapters, and host software stack. In contrast, a switch using cut-through architecture has a lower (better) port-to-port switching latency, but it propagates CRC-corrupted frames to the egress ports. In addition, when using cut-through mode, some features like encryption or compression can increase latency, an impact that is not seen in store-and-forward switches.

Cisco MDS switches operate in store-and-forward mode, whereas Cisco Nexus switches even with Fibre Channel ports operate in cut-through mode.

Ability to Detect and Drop CRC Corrupted Frames

As mentioned, a store-and-forward switch can detect and drop CRC-corrupted frames because the frame is received in its entirety before starting to transmit it on an egress port. This allows inspection of the CRC field, which is present at the end of a frame. Dropping the CRC corrupted frames where they are detected has the following benefits:

- Saving the egress port bandwidth by not forwarding corrupt frames.
- Saving the B2B credits by not forwarding corrupt frames.
- Sending only good frames to the destination.
- Accurate and faster troubleshooting (Frames with CRC errors are seen only on interfaces where actual corruption is occurring on the link).

These benefits are not available with switches that use cut-through architecture because the switch starts sending the frame before seeing the end of it, the CRC field can be verified but the frames can’t be dropped on the ingress port or even on the egress port.

Corrupt frames should be a fairly rare occurrence, and hence, among these differences, the most important factor in production networks is the ability for accurate and faster troubleshooting. For example, discards (or drops) on a port of a switch using store-and-forward architecture include CRC corrupted frames. An increasing CRC counter on a switchport suggests bit errors on the local link. When these are found there is certainly some physical problem on that link so checking/replacing components like the transceivers, cables, and patch panels rectifies the problem.

But, in a network of cut-through switches, corrupt frames may be forwarded across multiple links. This greatly complicates troubleshooting because corrupt frames received on a link may be forwarded to multiple different links and switches to their different destinations. These destination interfaces will also detect CRC errors, but they are not a result of bit errors on their local links. Determining the link where the initial corruption took place is topology-dependent and can be quite complex.

This complexity of cut-through switches in detecting the exact source of CRC errors can be reduced up to an extent if the switches encode special bits in a CRC-corrupted frame. In Fibre Channel, these special bits are referred to as EOFni. A similar concept also exists in Ethernet. Refer to Chapter 7, the section on *Stomped CRC Counters* for further details.

Because the troubleshooting methodology is drastically different, it’s important to understand the architecture of the switches that build your storage networks.

Load-balancing Scheme on ISLs

Cisco MDS switches offer two ways to load balance traffic on ISLs between a pair of switches.

The first load-balancing scheme is based on the source (Source ID or SID) and destination (Destination ID or DID) of a frame. In other words, a sender sends all the frames between a pair of SID and DID on the same link. For example, if a host initiates 100 read I/O operations to a storage port, all the frames for the 100 operations take the same link.

The second load-balancing scheme is based on SID, DID, and exchange ID (OXID, a unique identifier to identify an exchange) of a frame. In Fibre Channel, a single I/O (read or write) operation maps to a single exchange (Refer to the *Data Transmission on Fibre Channel Media* section for more details). This means a sender sends frames belonging to different exchanges between a pair of SID and DID on different links. For example, if a host initiates 100 read I/O operations to a storage port, all the 100 operations uniformly distribute across the available links. Cisco MDS switches use a load-balancing scheme based on SID/DID/OXID, by default.

Knowing the default load-balancing scheme of the switches is important for faster troubleshooting and finding the root cause of the problems. Refer to the section on *Over-utilization of ISL* to understand a type of congestion that depends upon the load-balancing scheme on an ISL.

Congestion Management Features

Different switches provide different capabilities for congestion management. For example, Cisco MDS and Nexus switches have similar performance, but congestion detection, troubleshooting, and prevention capabilities on Fibre Channel ports are superior on the MDS switches. Knowing these capabilities is crucial for effectively detecting and troubleshooting congestion.

Effect of Bit Errors on Congestion

This section explains how bit errors may worsen existing congestion or cause new congestion in Fibre Channel fabrics. This section also answers other related questions such as, what if the bit errors corrupt an R_RDY? What if the bit errors corrupt a frame resulting in a CRC error? What if the bit errors corrupt a frame in a manner that the receiver can’t even recognize it as a frame? To find the answers to these questions, it’s important to have a refresher on the basics of data transmission on Fibre Channel media. Besides the basics, this section goes into the details of relatively newer additions to Fibre Channel, such as Forward Error Correction (FEC) because not much content is written on these topics from Fibre Channel perspective. This section also explains various counters on Fibre Channel ports and their use in detecting congestion.

Before proceeding, refer to Chapter 1, the section *Bit Errors on a Link* for a high-level overview of bit errors and their potential for significant performance degradation in Fibre Channel fabrics. Those details are not repeated here for the sake of brevity.

Fibre Channel Frame Format

The size of a Fibre Channel frame can be up to 2,148 bytes carrying up to 2,112 bytes of payload or data, as shown in Figure 2-25.

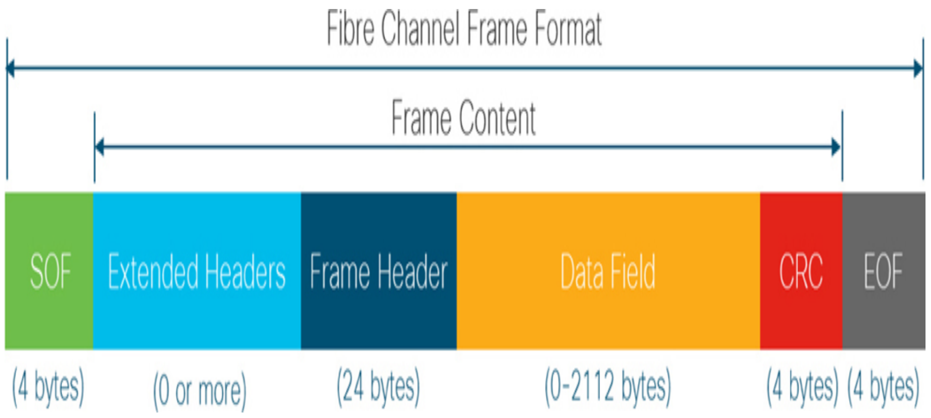
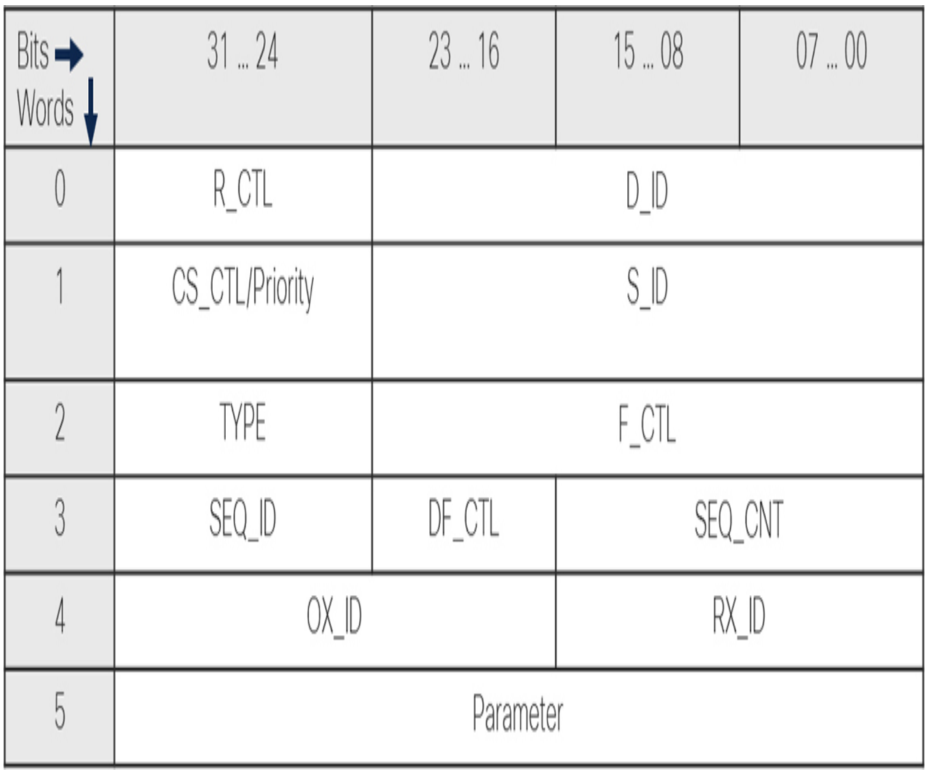


Figure 2-25 *Fibre Channel frame format*

A Fibre Channel frame is composed of the following:

- A Start-of-Frame (SOF) delimiter (4 bytes): SOF is a type of ordered set (bit sequence with special function) that immediately precedes the frame content and follows fill words. It conveys the beginning of a frame.
- An End-of-Frame (EOF) delimiter (4 bytes): EOF is a type of ordered set that immediately follows the frame content and is followed by fill words. It conveys the end of a frame.
- Frame content is composed of the following:
  - Zero or more Extended headers: Extended headers extend the functionality of the frame header, such as the Cisco Virtual SAN (VSAN) technology.

- Frame header (24 bytes): The frame header (Figure 2-26) contains information for the handling of the frame.



Bits → Words ↓	31 ... 24	23 ... 16	15 ... 08	07 ... 00
0	R_CTL	D_ID		
1	CS_CTL/Priority	S_ID		
2	TYPE	F_CTL		
3	SEQ_ID	DF_CTL	SEQ_CNT	
4	OX_ID		RX_ID	
5	Parameter			

Figure 2-26 *Fibre Channel frame header*

- Data field (0 – 2112 bytes): Data field contains the payload of the frame from the source to the destination.
- CRC (4 bytes): The Cyclic Redundancy Check (CRC) field is calculated by using the Frame Check Sequence (FCS) polynomial on the frame content (Extended headers, frame header, and data field). EOF and EOF delimiters are not within the scope of CRC.

The size of the FC frame header is 24 bytes. Figure 2-26 shows its contents.

Fibre Channel header is composed of the following:

- **R_CTL:** Routing control is a 1-byte field that contains information to categorize the frame functions, such as Extended Link Service and Extended Header.
- **CS_CTL/Priority:** A 1-byte field to carry class-specific control information or priority information.
- **D_ID:** Destination FC ID of the frame (3 bytes).

- **S_ID**: Source FC ID of the frame (3 bytes).
- **TYPE:** It conveys the upper layer protocol (ULP) carried in the payload of the frame. TYPE is 08h for SCSI and NVMe frames that belong to I/O operations because both use Fibre Channel Protocol (FCP). However, for Link Service Request and Responses, TYPE is set to 28h only for NVMe over Fibre Channel.
- **F_CTL:** Frame control is a 3-byte field containing control information relating to the frame content.
- **SEQ_ID:** Sequence Identifier is a 1-byte field. All the frames of a sequence carry the same SEQ_ID.
- **DF_CTL:** Data Field Control is a 1-byte field that specifies the presence of optional headers at the beginning of the data field.
- **SEQ_CNT:** Sequence count is a 2-byte field that identifies the order of the transmission of the frames within a sequence. SEQ_CNT of the first data frame in a sequence is zero. The SEQ_CNT of each subsequent data frame in the sequence increments by one.
- **OX_ID:** Originator Exchange Identifier is a 2-byte field assigned by the originator of exchange to identify it.
- **RX_ID:** Responder Exchange Identifier is a 2-byte field assigned by the responder of exchange to identify it.
- **Parameter:** Parameter is a 4-byte field carrying information dependent on the frame type.

Fibre Channel Levels

Upper Layer Protocols (ULP), such as SCSI and NVMe, initiate I/O transactions, which go through multiple Fibre Channel levels (Figure 2-27) before being transmitted on a link. Fibre channel levels have the following functions.

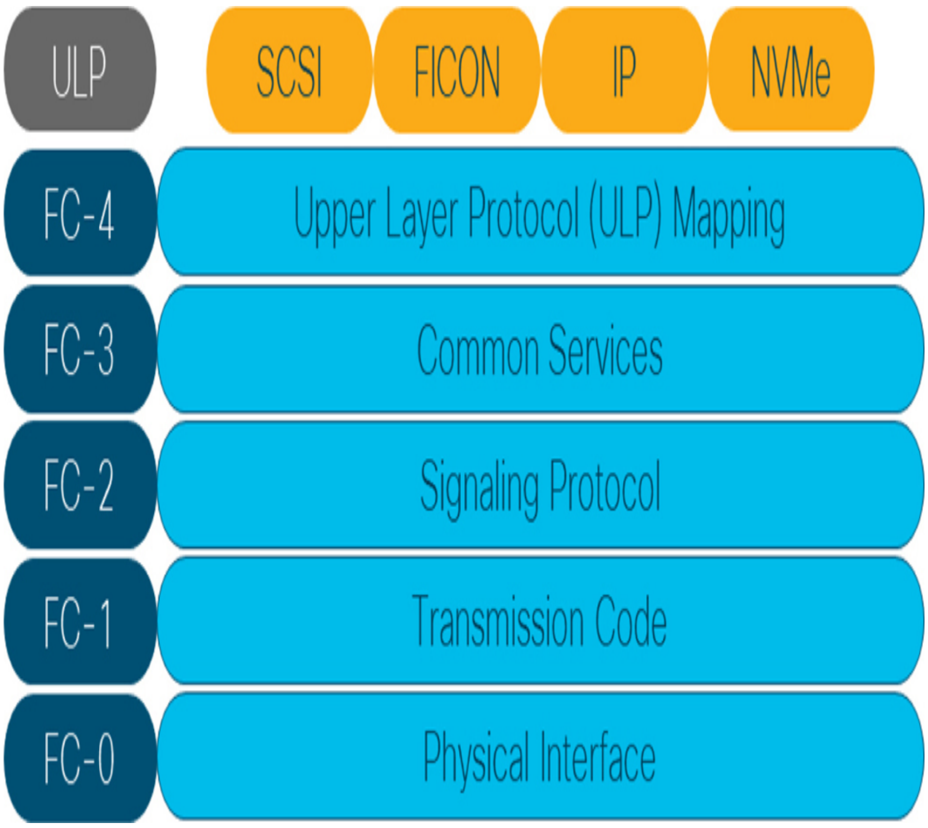


Figure 2-27 *Fibre Channel levels*

- **FC-4:** Defines the mapping of ULP to the lower levels.
- **FC-3:** Defines a set of common services, such as Extended Link Services.
- **FC-2:** Defines how the information is transported, such as frames, sequences, and exchanges.
- **FC-1:** Defines how data is encoded and decoded for transmissions, and error control, such as Forward Error Correction (FEC).
- **FC-0:** Defines the physical interface characteristics, such as transmission media, transmitters, and receivers and their interfaces.

Data Transmission on Fibre Channel Media

A Fibre Channel link goes through a series of steps before it’s ready for data transfer.

When two neighboring FC ports are enabled and connected, the first step is that they detect light (optical) signals from each other. After that, they may undergo a speed negotiation, agree on a common FEC mechanism, try to synchronize the bit and Transmission Word boundaries, and so on. Next, the FC ports exchange control-plane states based on their port types. For example, an end-device N_Port does a Fabric Login (FLOGI), Port Login (PLOGI), Process Login (PRLI), and so on. Finally, the upper-layer protocols, such as SCSI and NVMe, can initiate read or write I/O operations via these FC ports.

Transforming an I/O Operation to FC frames

A read or write I/O operation (Figure 2-28) between an initiator and a target undergoes a series of transformations before being transmitted on a Fibre Channel link.

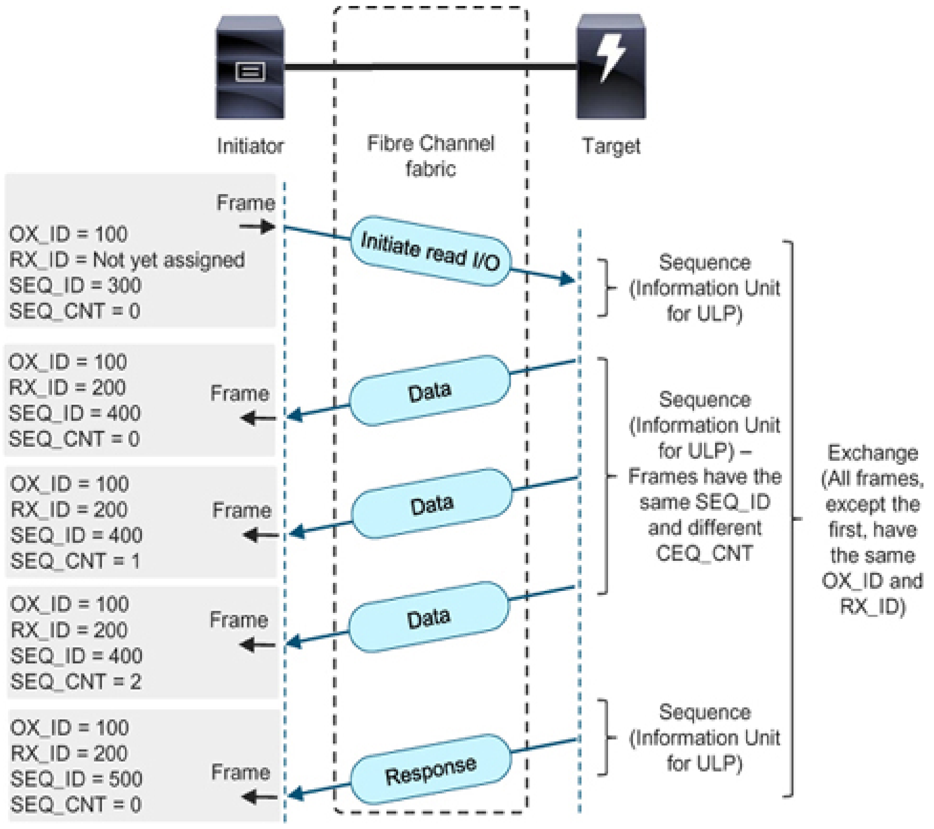


Figure 2-28 *An I/O operation in Fibre Channel*

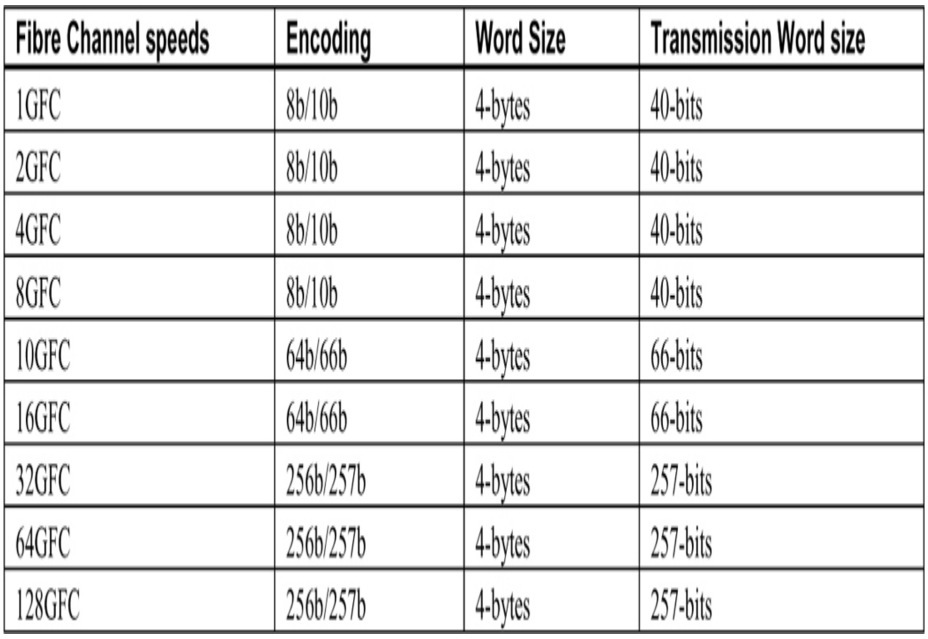
An I/O operation is mapped to a Fibre Channel Exchange. The initiator assigns and identifies an exchange using a unique OX_ID. Likewise, the target assigns and identifies the exchange using a unique RX_ID. Upper levels handoff data to the FC-2 level in the form of information units (IU). At FC-2 level, an IU is mapped to a sequence that is identified by a unique Sequence ID (SEQ_ID). A sequence, based on its size, is divided into one or more Data Fields that are identified by a unique Sequence Count (SEQ_CNT). A Data Field can carry up to 2112 bytes which is the maximum payload size of a single Fibre Channel frame (Figure 2-25).

Each Data Field is encapsulated within an optional extended header, mandatory header, and CRC to make a Fibre Channel frame. Then, the SOF and EOF delimiters are added to clearly define the frame boundaries. Further, inter-frame fill words are added in the form of primitive signals. Finally, a stream of bits is ready to be transmitted on the wire.

Encoding the Frames and Special Functions

The FC-2 level presents bits to the FC-1 level as a stream of words. A word is a series of four bytes. The FC-1 layer encodes the words to Transmission Words using a pre-defined encoding scheme. Different Fibre Channel speeds use a different encoding scheme, as described in Table 2-3. Encoding helps to synchronize the clocks between the sender and the receiver, increases the likelihood of detection of bit errors, and helps in maintaining an alignment of Transmission Words.

Table 2-3 *Fibre Channel speeds, encodings, word size, and Transmission Word size*



Fibre Channel speeds	Encoding	Word Size	Transmission Word size
1GFC	8b/10b	4-bytes	40-bits
2GFC	8b/10b	4-bytes	40-bits
4GFC	8b/10b	4-bytes	40-bits
8GFC	8b/10b	4-bytes	40-bits
10GFC	64b/66b	4-bytes	66-bits
16GFC	64b/66b	4-bytes	66-bits
32GFC	256b/257b	4-bytes	257-bits
64GFC	256b/257b	4-bytes	257-bits
128GFC	256b/257b	4-bytes	257-bits

The following are the types of encodings:

- **8b/10b encoding:** The 8b/10b encoding maps 8 bits to 10 bits of Transmission Characters and four contiguous Transmission Characters compose a 40-bit Transmission Word. The 8b/10b encoding adds 2 additional bits for every 8 bits, resulting in 25% overhead.
- **64b/66b encoding:** The 64b/66b is more efficient because 64 bits are encoded to a 66-bit Transmission Word, resulting in only 2 overhead bits for every 64 bits (3.125% overhead).
- **256b/257b encoding:** The 256b/257b encoding operates on four consecutive 64b/66b Transmission Words, each group being transcoded to a 257-bit Transmission Word (0.39% overhead).

Transmission Words are of two types:

- **Data Transmission Words:** As the name suggests, a Data Transmission Word carries frames (with Data Field in it).
- **Control Transmission Words:** A control Transmission Word carries special functions discussed next.

Special Functions — Delimiters, Primitive Signals, and Primitive Sequences

Special functions are link-control operations, such as:

- Link initialization.
- Frame delimiting.
- Inter-frame fill.

Special functions are communicated by Ordered Sets rather than by frame content. As mentioned earlier, an Ordered Set is a bit sequence with a special function. All Ordered Sets remain local to a link.

Based on their function, Ordered Sets can be classified into the following categories:

- **Delimiters:** Ordered Sets that immediately precede or follow the content of a Fibre Channel frame. For example, Start-of-Frame (SOF) Ordered set precedes the frame content and End-of-Frame (EOF) Ordered Set follows the frame content. The size of SOF and EOF delimiters is 4 bytes.
- **Primitive Signals:** Ordered Sets that convey special meaning. The most common Primitive Signals are the following:
  - **R_RDY:** A frame-receiver sends a Receiver_Ready (R_RDY) to indicate that it has buffer available to receive one frame. The size of R_RDY is 4 bytes.
  - **Fill word:** They are used as inter-frame fill and transmitted when frames and other Primitive Signals or Primitive Sequences are not being transmitted. Common fill words are Idle and ARBff. The size of a fill word primitive signal is 4 bytes.
  - **BB_SCs:** The B2B State Change SOF (BB_SCs) primitive signal is used for recovering lost credits using the B2B state change mechanism. Refer to the *Credit Loss Recovery Using B2B State Change Mechanism* section for more details.
  - **BB_SCr:** The B2B State Change R_RDY (BB_SCr) primitive signal is used for recovering lost credits using the B2B state change mechanism. Refer to the

    Credit Loss Recovery Using B2B State Change Mechanism section for more details.

    - **Congestion Signals:** These primitive signals are used to indicate congestion between two directly attached FC ports. It can be of type warning (ARB(F1)) or alarm (ARB(F7)). Refer to Chapter 6, the section *Notifications and Signals in Fibre Channel Fabrics* for more details.

- **Primitive Sequences:** Ordered Sets that are transmitted repeatedly and continuously until a special response is received (Figure 2-32). For example:

    - **Not Operational Sequence (NOS):** A port sends a NOS to indicate that it has detected a link failure.

    - **Offline Sequence (OLS):** A port sends OLS to indicate that it's initiating the Link Initialization Protocol.

    - **Link Reset Sequence (LR):** Transmitted to indicate the initiation of the Link Reset Protocol.

    - **Link Reset Response Sequence (LRR):** Transmitted to indicate that an LR has been received and the B2B credits are set to their full, agreed-to, amounts on the directly attached FC ports.

Transmitting Bits on the Media

A stream of Transmission Words on a link may be further transformed to provide Forward Error Correction (FEC). When FEC is enabled, a sender adds a few parity (redundant) bits before data transmission. The parity bits allow a receiver to detect a limited number of errors and may correct these errors without re-transmission.

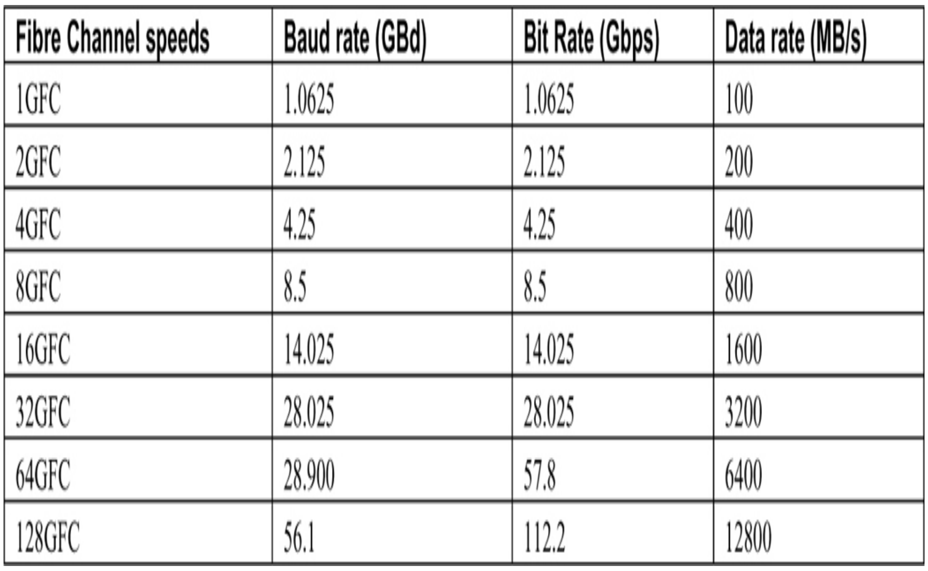
Finally, the bits are transmitted to the media.

To summarize, bits are grouped into words that are encoded into Transmission Words. Transmission Words have parity words to support FEC. The Transmission Words can carry data or convey a special function in the form of an Ordered Set. Data from upper-layer protocols are carried in the Data Field of Fibre Channel frames which are encoded in Data

Transmission Words whereas ordered sets such as SOF, EOF, Idle, and R_RDY are encoded in Control Transmission Words.

Next, let’s understand the Fibre Channel baud rate, bit rate, and data rate as shown in Table 2-4.

Table 2-4 *Fibre channel speeds, baud rate, bit rate, and data rate*



Fibre Channel speeds	Baud rate (GBd)	Bit Rate (Gbps)	Data rate (MB/s)
1GFC	1.0625	1.0625	100
2GFC	2.125	2.125	200
4GFC	4.25	4.25	400
8GFC	8.5	8.5	800
16GFC	14.025	14.025	1600
32GFC	28.025	28.025	3200
64GFC	28.900	57.8	6400
128GFC	56.1	112.2	12800

Fibre Channel Baud Rate

The baud rate is the number of symbols transmitted per second. For example, a 32GFC port transmits 28.025 x 10^9 symbols per second (28.025 Gigabaud (GBd)).

Fibre Channel Bit Rate

The bit rate is the number of bits transmitted per second. It can be calculated from the baud rate based on the modulation scheme. 32GFC and slower speeds use the NRZ (non-return-to-zero) modulation that carries one bit per symbol. 64GFC and 128GFC use PAM-4 (Pulse Amplitude Modulation 4-level) modulation that carries two bits per symbol. Because of this reason, FC speeds lower than 64GFC have bit rates the same as the baud rate. However, 64GFC and 128GFC have a bit rate double that of the baud rate.

Fibre Channel Data Rate

The data rate shown in Table 2-4 is the amount of data that the upper-layer protocols on the end devices exchange per second. These values account only for the data that's carried by the Data field of Fibre channel frames (Refer to the *Fibre Channel Frame Format* section for more details). The Data Field in an FC header carries the data that an I/O operation aims to transfer and the headers of the upper-layer protocol, such as SCSI or NVMe headers.

The data rate and the bit rate in Table 2-4 don't match directly. For example, a 32GFC link has a bit rate of 28.025 Gbps, whereas the data rate is 3200 MB/s or 25,600 Mbps. This difference is because of the following factors:

1. **Frame encapsulation:** As previously mentioned, the data rate accounts only for the Data Field. An encapsulated frame carries the following additional fields:

   a. Frame Header – 24 bytes

   b. CRC – 4 bytes

   c. Frame Delimiters

      i. SOF – 4 bytes

      ii. EOF – 4 bytes

   A frame may also carry optional Extended Headers. None of these are accounted for the data rate.

2. **Inter-frame fill:** Two adjacent FC frames are never transmitted next to each other. An FC port must insert a minimum of six primitive signals (each of 4 bytes) as inter-frame fill words to maintain frame boundaries and clock synchronization. Out of these six primitive signals:

   a. Two primitive signals must be the fill words that immediately follow a frame (after EOF delimiter)

   b. Two primitive signals must be the fill words that immediately precede a frame (before SOF delimiter)

   c. Two can be any primitive signals such as R_RDY, BB_SCr, BB_SCn, and Congestion Signals or they can be fill words.

None of these are accounted for the data rate.

3. **Encoding:** Refer to Table 2-3 for different types of encodings and their overhead. This is not accounted for the data rate.
4. **FEC:** FEC adds a small amount of overhead. Refer to the *FEC* section for more details. This is not accounted for the data rate.

Difference Between Fibre Channel Speed and Bit Rate

An FC link doesn't transmit bits exactly at the advertised speed. For example, a 32GFC link doesn't transmit at 32 Gbps. Its actual bit rate is 28.025 Gbps. Likewise, a 16GFC link transmits at 14.025 Gbps, not 16 Gbps.

This difference is because the aim of Fibre Channel is to achieve a defined data rate for the upper-layer protocols at a speed. Baud rate, bit rate, encoding, and other factors are decided accordingly. For example, 1GFC provides a data rate of 100 MB/s. At each successive faster speed, the data rate is doubled, for example, 200 MB/s at 2GFC, 400 MB/s at 4GFC, and so on. Likewise, a 32GFC port provides data rate of 3200 MB/s while operating at a bit rate of 28.05 Gbps, and a 64GFC port provides a data rate of 6400 MB/s while operating at a bit rate of 57.8 Gbps.

This is a significant detail to understand while monitoring and calculating the percent utilization of FC ports to avoid being misled. A 32GFC port transmitting at 28 Gbps leads to 100% utilization. However, assuming 32 Gbps as the bit rate of a 32GFC link leads to only 87.5% utilization (28/32), which is wrong.

Think of the notation of Fibre Channel only as advertised speed, such as 32GFC. The real speed, however, is the bit rate as shown in Table 2-4.

Effect of Primitive Signals on Data Rate

Primitive signals, such as R_RDY, do not contribute to the data rate of an FC link, and hence, they do not increase link utilization. Primitive Signals, however, are accounted in the bit rate. As explained earlier, the difference between the bit rate and data rate leaves enough space for Primitive Signals to be transmitted without contributing to the link utilization. Let's understand it in detail.

An FC port is ready to send frames when it reaches an active state. But it sends frames only when an upper-layer protocol hands off data to the Fibre Channel levels. If there is no data from the upper layers, the FC port continuously transmits fill words (4 bytes), for example, Idle, at a bit rate (Figure 2-29) defined by the port speed. These fill words maintain synchronization between the sender and the receiver and keep the link ready for frame transmission.

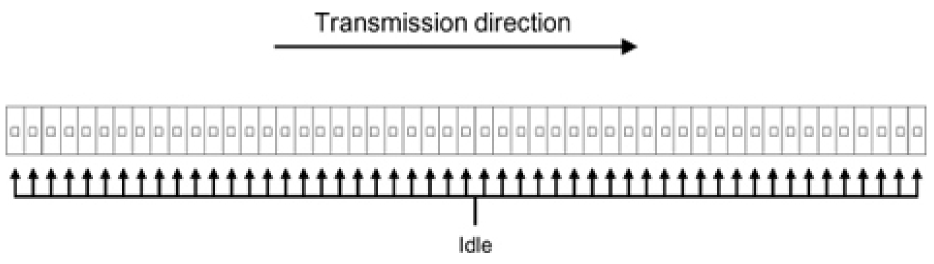


Figure 2-29 *Fill words on a Fibre Channel link with no traffic*

When an upper layer hands off data to Fibre Channel levels and as a result the FC port has frames for transmission, it replaces multiple fill words with the frame. But fill words are never completely eliminated because they make the inter-frame fill (Figure 2-30). Only a frame, which carries the Data Field, adds to the data rate on a link.

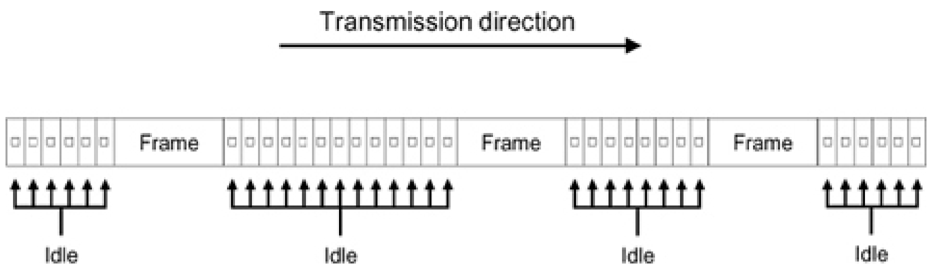


Figure 2-30 *Fill word primitive signals and frames on a Fibre Channel link with some traffic*

As previously explained, even at the maximum frame transmission rate (Figure 2-31) when the data rate is 100% such as 3200 MB/s on a 32GFC link, the sender inserts fill words between two adjacent frames.

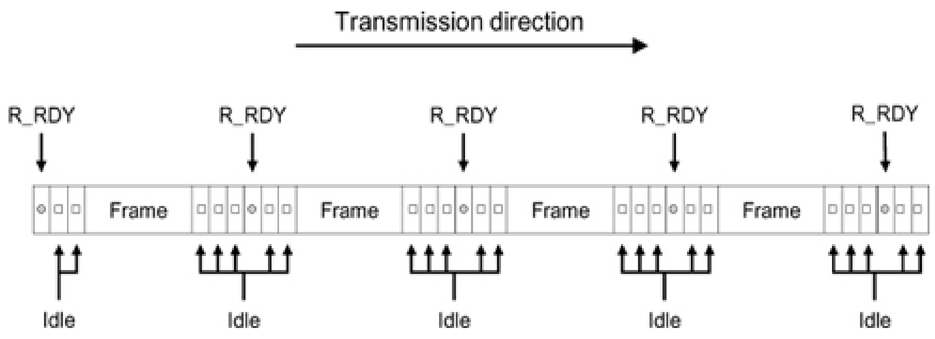


Figure 2-31 *Fill words, frames, and R_RDY on a Fibre channel link with 100% utilization*

When an FC port wants to send a different primitive signal (such as R_RDY), it simply replaces an optional fill word with it. Replacing fill words with another type of primitive signal doesn't change the data rate because fill words always flow on a link without contributing to the data rate and hence, replacing them with other primitive signals of the same size doesn't make a difference to the bit rate and data rate.

Counters on Fibre Channel Ports

This section explains various counters that a Fibre channel port may report for link initialization, flow control, utilization, and errors.

Link Initialization Counters

Fibre Channel uses Primitive Sequences for link initialization procedures. As Figure 2-32 shows, these procedures are Link Failure Protocol, Link Initialization Protocol, and Link Reset Protocol. The Primitive Sequences (NOS, OLS, LR, and LRR) are already explained earlier.

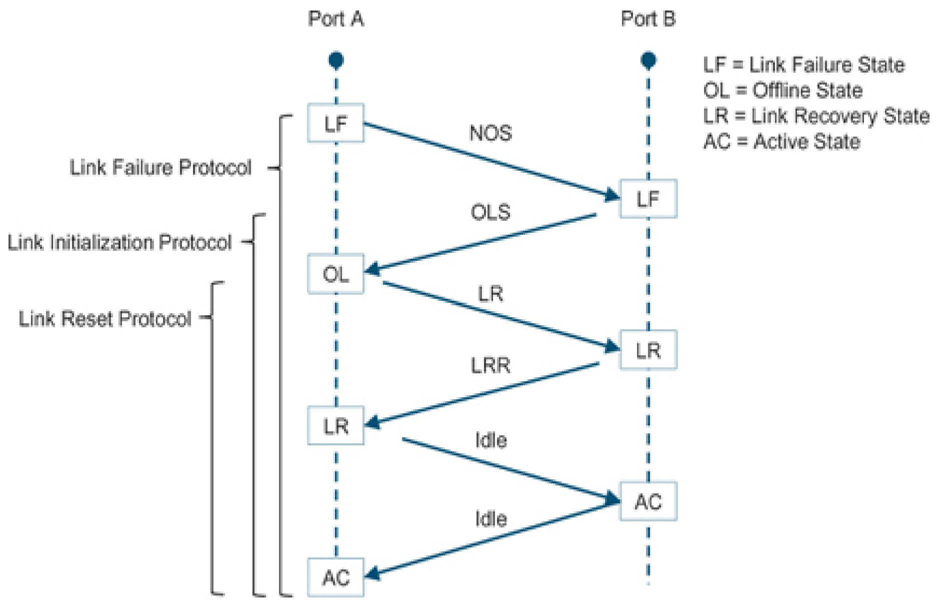


Figure 2-32 *Fibre Channel Primitive Sequences and errors.*

For the scope of this book, remember that a port starts in a link failure state. It may also enter this state due to link errors. When in this state it sends or receives these primitive sequences to reach the active state. An increase in these primitive sequence counters is normal when an FC link is being established, for example, when a host port connects to a switchport. But after the link reaches an active state and frames start flowing, these counters shouldn't increment. If these counters keep incrementing, verify the health of the link.

Invalid Transmission Words

A receiver may treat a Transmission Word as invalid if it doesn't meet the encoding rules and other checks of the Fibre Channel data transmission. Such Transmission Words are reported as Invalid Transmission Words (ITW) by the receiver. Note that ITWs are detected only on the receiving side of the link. The transmitting side of the link never knowingly transmits an ITW unless it is faulty. A valid Transmission Word may become invalid due to bit errors.

ITWs may result in other errors, for example, invalid CRCs when they occur in FC frames. They may also cause a loss of synchronization when they occur in large numbers. Depending upon what specifically is corrupted, the effect of ITW may also go unnoticed at upper levels.

CRC

A sender, before sending a frame, calculates the Frame Check Sequence (FCS) polynomial on the frame content (Extended headers, frame header, Data Field) and places the calculated output in the CRC field of the frame. The receiver, after receiving the frame, calculates the same FCS polynomial on the frame content. If the calculated output matches the content of the CRC field, the receiver knows that bit errors don't exist within the frame. If, however, the calculated output doesn't match the content of the CRC field, the frame is known to fail the CRC verification and it is called a CRC corrupted frame or Invalid CRC frame. Consequently, the receiver increments the input CRC error counter.

The exact handling of a CRC corrupted frame depends upon the type of receiver. An end-device always drops a CRC corrupted frame. A switch operating in cut-through mode forwards a CRC-corrupted frame towards its destination, whereas a switch operating in store-and-forward mode drops a CRC-corrupted frame. Refer to the section on *Ability to Detect and Drop CRC Corrupted Frame* for more details.

Although CRC errors indicate a degraded link quality, it doesn't show the entire picture. Consider the following two statements:

> "The link is unhealthy because the port reports CRC errors"—Correct
>
> "The link is healthy because the port doesn't report CRC errors"—Not correct

This is because the scope of CRC includes only the extended headers if any, frame header, and the Data Field. If the corrupted bits are outside the scope of the CRC, the CRC error counter doesn't increment, although the integrity of the link is compromised.

Some common examples of bit errors outside the scope of CRC are:

- The most obvious example is when the corrupted bits are outside the scope of the CRC. For example, SOF and EOF delimiters, fill words, R_RDY, and other primitive signals. Such bit errors can be detected by ITW and FEC counters.
- Frames not flowing on the link. A Fibre Channel link in an active state may not carry Fibre Channel frames if the upper-layer protocol doesn't initiate read or write I/O operations. During this state, FC ports still send and receive bits (belonging to fill words) to keep the link synchronized. During this state, the FC ports may report lower-level counters (such as ITW and FEC counters) that indicate bit errors on a link, but it doesn't report the CRC counter.
- The corrupted bits were recovered by FEC, which should be reported by the FEC corrected blocks counter. Refer to the *FEC* section for more details.

Refer to the *Case Study — An Online Retailer* for a real example when bit errors were seen in a large number, yet CRC counters did not increment.

Forward Error Correction

When FEC is enabled, a sender encodes multiple Transmission Words into an FEC block and adds a few additional parity bits with it. The receiver can use the parity bits to detect and recover a limited number of errors in the FEC block.

This section provides an academic explanation of FEC before explaining what matters in a production network. This is because:

- FEC is relatively new to Fibre Channel. It was added in 16GFC, but it was optional. FEC is mandatory and enabled by default at 32GFC and faster speeds.
- Because it's new, some people have misunderstood its benefits. Others are not monitoring it or not doing so correctly.
- Not much education exists on FEC.

If you are not interested in the academic details, you may skip the next section and jump directly to what matters in a production network.

FEC — The Details

With 16GFC, which uses 64b/66b encoding, the FEC encoder takes 32 65-bit Transmission Words (32x65 bits=2080 bits) for generating a 32-bit parity word. Then, it sends out the 32 65-bit Transmission Words followed by the 32-bit parity word, collectively known as an FEC block of size 2112 bits (32 x 65 bits + 32 bits = 2112 bits). The reason for using 65 bits instead of 66 bits is that the FEC layer compresses 2 additional bits (which are the result of the 64b/66b encoding) to 1 bit. The receiver can correct an error burst of up to 11 bits per FEC block. Overall, 32 bits of overhead with 2080 bits of payload can correct up to 11 bits. Consequently, the 16GFC FEC encoder has an overhead of 1.53% (32/2080) and a correction rate of up to 0.53% (11/2080). The FEC code used by 16GFC is represented as (2112, 2080), and is known FireCode FEC.

With 32GFC, which uses 256b/257b encoding, the FEC encoder uses a Reed-Solomon (RS) code which operates slightly differently. It operates on a symbol size of 10 bits. The FEC encoder takes 20 257-bit Transmission Words carrying 5140 bits (20 x 257) or 514 10-bit symbols and generates 14 parity 10-bit symbols. This results in an FEC block size of 5280 bits (5140 + 140). The receiver can correct any combination of up to 7 symbol errors in an FEC block. Overall, 140 bits of overhead with 5140 bits of payload can correct up to 70 bits. Consequently, the 32GFC FEC encoder has an overhead of 2.72% (140/5140 bits or 14/514 10-bit symbols) and a correction rate of up to 1.36% (70/5140 bits or 7/514 10-bit symbols). The Reed-Solomon FEC code used by 32GFC is represented as RS(528, 514).

With 64GFC, which uses 256b/257b encoding, the FEC encoder uses a more powerful Reed-Solomon code. Like 32GFC, the 64GFC FEC encoder operates on a 10-bit symbol. It takes 20 257-bit Transmission Words carrying 5140 bits (20 x 257) or 514 10-bit symbols. But unlike the 32GFC, the

64GFC FEC encoder generates 30 parity 10-bit symbols resulting in the FEC block size of 5440 bits (5140 + 30 x 10). The receiver can use the 30 parity symbols to correct any combination of up to 15 symbol errors in an FEC block. Overall, 300 bits of overhead with 5140 bits of payload can correct up to 150 bits. In other words, the 64GFC FEC code has an overhead of 5.83% (300/5140 bits of 30/514 symbols) and a correction rate of up to 2.91% (150/5140 bits or 15/514 symbols). The Reed-Solomon FEC code used by 64GFC is represented as RS(544,514).

The Reed-Solomon codes—RS(528, 514) used by 32GFC and RS(544,514) used by 64GFC—can correct half the number of parity symbols. RS(528, 514) has 14 parity symbols and it can correct up to 7 symbols. RS(544,514) has 30 parity symbols and it can correct 15 symbols. In other words, if *t* is the number of symbols that can be corrected, *2t* is the number of parity symbols.

The Reed-Solomon codes used by 32GFC and 64GFC have the following variables.

- **k:** The number of payload symbols used as input to the FEC encoder. The value of k for RS(528, 514) and RS(544,514) is 514.
- **2t:** The number of parity or redundant symbols generated from k symbols. The value of 2t for RS(528, 514) is 14, whereas the value of 2t for RS(544,514) is 30.
- **n:** The number of symbols in an FEC block.

The following equation represents the relationship between these variables.

Relationship between the variables of Reed-Solomon FEC code

$n=k+2t$

As mentioned, these FEC codes are presented by RS(n.k) notation.

For RS(528, 514), n = 528, k = 514, t = 7.

For RS(544,514), n = 544, k = 514, t = 15.

The Fibre Channel standards don’t define FEC codes. Instead, they reuse the FEC code defined in the IEEE 803.2 standard for Ethernet. Because of this reason, the FEC concepts are the same between Fibre Channel and Ethernet. The terminology, however, may be different. Ethernet references as “blocks” what Fibre Channel references as “Transmission Words”. Likewise, FEC block, FEC frame, and FEC codeword refer to the same entity carrying the FEC payload and parity bits. For more details on FEC, refer to the IEEE 803.2-2018 specification, especially section 74.7 and section 91.5.

Although this section briefly explains FEC operations and refers to the IEEE 803.2-2018 specification for more details, this understanding is mostly good for academic purposes or for equipment manufacturers. Refer to the next section to know what matters more in a production environment.

FEC — What Matters in a Production Network

While operating production storage networks, knowing the internal details of FEC isn’t needed when trying to troubleshoot a congestion issue or application slowdown. What’s important to remember is the following:

- FEC may be able to correct a limited number of errors. The exact correction rate depends on the FEC code and the extent of corruption.
- FEC is not magic. As mentioned, it can’t recover all the errors.
- CRC, ITW, and other errors may still be seen with FEC. If FEC can’t correct an error, it simply reports it as an “FEC uncorrected block’ and passes the bit-stream to the higher layers.
- FEC counters can provide predictive insights into the health of a network. Refer to the section on *Case Study — An Online Retailer* for more details.

An FEC-enabled port reports the following counters.

- **FEC corrected blocks:** The number of blocks that a receiver could detect with errors and could correct the errors. As a result, the upper level (for example FC-2 level) doesn’t notice any errors.

- **FEC uncorrected blocks:** The number of blocks that a receiver could detect with errors but couldn’t correct errors. The FEC blocks are decoded, and the Transmission Words are handed off to the upper level (for example FC-2 level). The exact effect of the corrupted bits depends on the location and type of error. It may result in a CRC error, an Invalid Transmission Word error, or other kinds of errors explained in this section.

While monitoring the FEC counters on an FC port, be aware that:

- Both types of FEC counters—FEC corrected blocks and FEC uncorrected blocks—indicate degraded quality of a link or a faulty sender. The only difference with the FEC corrected blocks counter is that these errors could be corrected, whereas the FEC uncorrected blocks could not be corrected. For all practical purposes, both these counters should be monitored at the same severity level because both indicate bit errors.
- The FEC counters are reported in the number of FEC blocks, which are different from the number of Transmission Words, the number of bits, the number of bytes, and the number of frames. The previous section explains the relationship between these entities but for all practical purposes, this subtle difference can be ignored. Anytime the FEC counters increment, it indicates bit errors.

Be aware that a nominal or occasional increase in the FEC counters should not result in replacing the components without an investigation. At faster speeds, bit error thresholds are low, and hence FEC is mandated to recover these errors. This means a nominal and occasional increase in FEC counters is acceptable. However, ports that report higher FEC count than other ports or report an increasing trend must be added to a watchlist for further investigation.

FEC counter, especially, the FEC corrected block counter can make an excellent predictive mechanism about the health of the cables and transceivers in a network. The most common cause of bit errors on network links is faulty cables, transceivers, loose connections, and dust. Mostly, these issues are detected after they cause damage. For example, a CRC error results in frame drops, which may cause a catastrophic effect on the application performance. The increasing FEC corrected block counter indicates that an error was corrected which could have resulted in CRC error. Because of the random nature, bit errors on the same link may exceed the FEC threshold and may not be corrected. Instead of waiting for the CRC error, use the FEC counters to make operations more proactive. Take the same steps with FEC counters, as you would take with CRC, ITW, and similar error counters. The only difference with FEC corrected block counter is that the damage was corrected.

To understand the predictive possibilities of the FEC counters, consider a case study of a Fibre Channel port in the production storage network of an online retailer company.

Case Study — An Online Retailer

An online retailer provides block storage to their key applications using the storage arrays connected via Fibre Channel fabrics. During the peak shopping season, an FC port, which provides connectivity to their tier-1 application, started alerting on Invalid Transmission Words. As per their operational practices, they investigated the issue but were not allowed to make corrective changes, such as replacing SFP or cables, due to a no-change policy enforced during the peak season. Hence, they moved the application to another host with healthy links. This change was expensive because instead of one host, they had to provision (and were billed for) two hosts with enough resources to run the tier-1 application, while one host remained unused due to bit errors on its link.

The online retailer uses a robust monitoring infrastructure. They wanted to know if there was something they could have done better to prevent this expensive change during the peak business days. For example, had they known about this condition earlier, they could have replaced the faulty component that caused the bit errors before the no-change period, and therefore could have avoided being billed for two hosts instead of one.

Observations

The operations team of the online retailer company monitors all the network ports continuously. Their monitoring infrastructure collects counters and plots them on a graph periodically. The FC port under investigation was no different.

Figures 2-33 to 2-36 show the FEC corrected blocks, FEC uncorrected blocks, Invalid Transmission Words, and CRC errors on the investigated port from October end till mid-December.

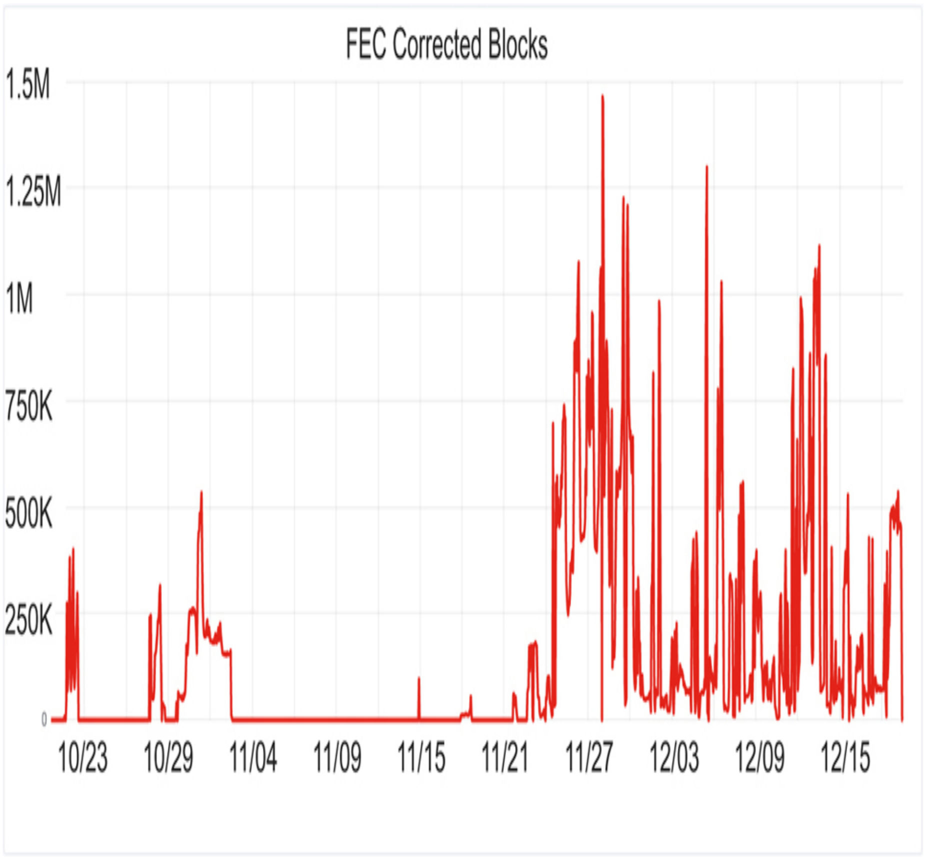


Figure 2-33 *Time-series graph of FEC Corrected Blocks from October till December*

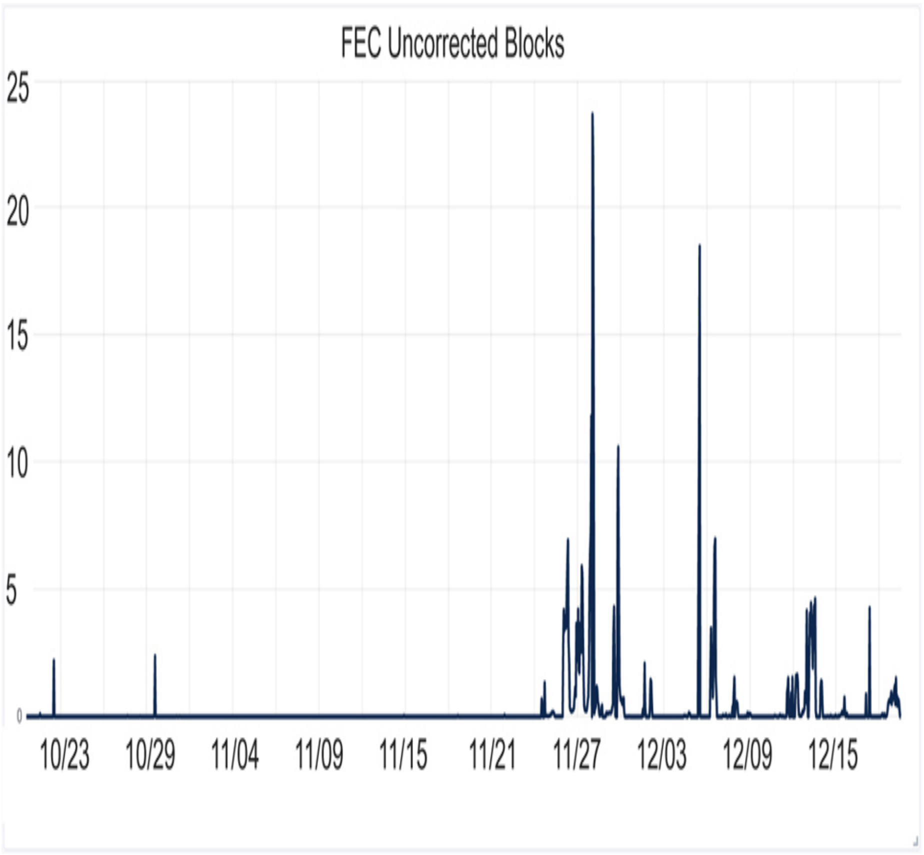


Figure 2-34 *Time-series graph of FEC Uncorrected Blocks from October till December*

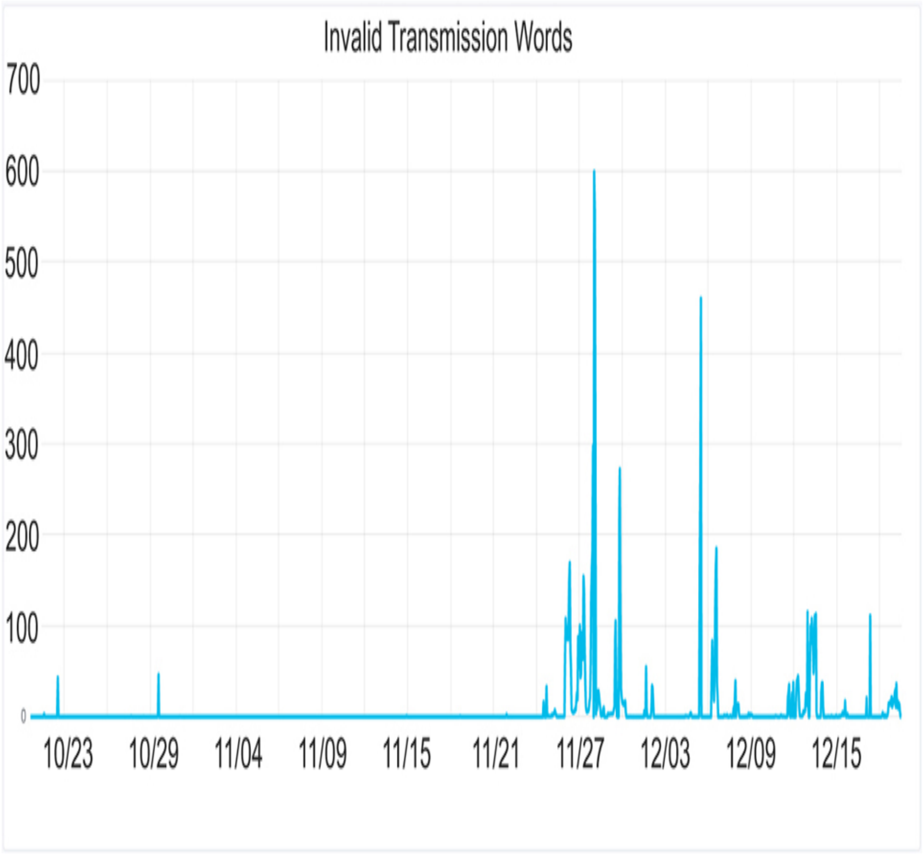


Figure 2-35 *Time-series graph of Invalid Transmission Words from October till December*

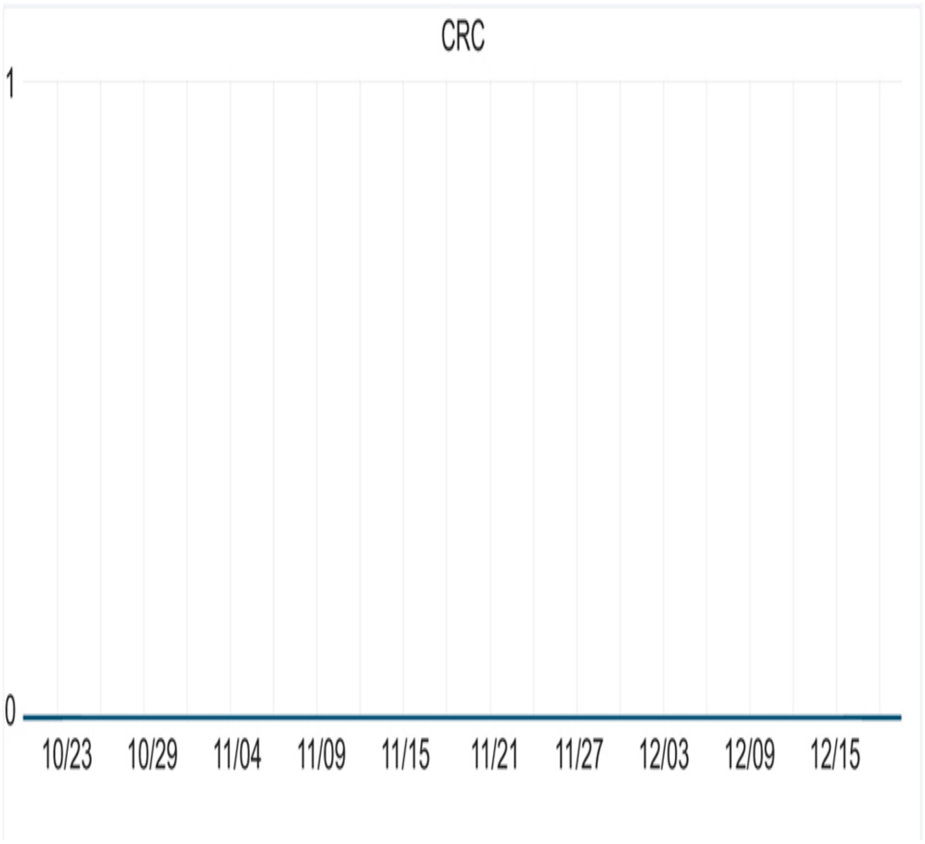


Figure 2-36 *Time-series graph of CRC errors from October till December*

The following are the observations from this data:

1. The link observed bit errors as evident from a large number of FEC corrected blocks (Figure 2-33).
2. FEC was able to prevent many ITWs and potential CRC errors. For example, on 28 November and 6 December, FEC corrected more than 1.25 million errored blocks (Figure 2-33).
3. But FEC could not correct everything. For example, on 22 October, FEC could not correct 2 errored blocks (Figure 2-34) while it corrected more than 250,000 errored blocks (Figure 2-33).
4. When FEC could not correct the errored blocks, the port reported Invalid Transmission Words. The time-series pattern of FEC uncorrected blocks (Figure 2-34) closely

matches that of Invalid Transmission Words (Figure 2-35).

5. No CRC errors were seen (Figure 2-36), which means that bit errors were outside the scope of the CRC. These bit errors might have corrupted SOF or EOF delimiters, fill words, or R_RDYs.

After analyzing the data, let's understand how the operations team could have predicted this issue much earlier rather than waiting till the peak business days which resulted in the high cost of making the change.

Conclusions

FEC uncorrected blocks and ITWs were reported on 22nd October and 30th October, but these values were low and therefore did not exceed the alerting threshold. The operations team should have moved this port to a watch list after seeing the first or second spike of FEC corrected blocks on the 21st and 22nd October. The port was fine for a few days, but it reported the FEC corrected blocks again on 28th October (3rd occurrence) and continuously from 30th October till 3rd November (4th occurrence). By this time, the operations team should have investigated the bit errors and taken corrective action, such as replacing the faulty SFP.

But no action was taken. As a result, on 15th and 18th November, the port reported the FEC corrected blocks again. By 24th November, the port started seeing severe bit errors as evidenced by a large number of FEC corrected blocks. Starting around November 26th many bit errors escaped the FEC layer as evidenced by the spikes in FEC uncorrected errors. This resulted in the detection of Invalid Transmission Words. On 28th November, the Invalid Transmission Words show a spike (Figure 2-35) which must have triggered the alerts during the peak season.

Lessons Learned

The cost of bit errors would have been much lower had the operations team used the predictive insights of the FEC counters and resolved the root cause earlier in October-end or the beginning of November.

The operations team learned a lesson because of these detailed investigations. They were already monitoring the FEC counters. They made their operations more proactive by enabling alerting on FEC counters and treating FEC counters at the severity as the Invalid Transmission Words and CRC errors. As mentioned earlier, this does not mean replacing the components on the first occurrence of FEC counters. However, they started populating a watch list of the ports that show spikes, increasing trends, and higher counts of the FEC counter than other ports.

Finally, in this case study, data wasn't conclusive to prove that these errors resulted in congestion. However, the Invalid Transmission Words might have resulted in the loss of credits because of corruption of R_RDY or frames. Refer to the section on *Loss of Credits due to Bit Errors* for more details. Overall, the lessons learned from this case study reinforce the importance of proactive monitoring.

Effect of Bit Errors on Congestion — Summary

The effect of bit errors on congestion depends upon the location of the errored bits and the success of the defense mechanisms.

FEC is the first level of defense against bit errors. It can recover a limited number of errored bits in an FEC block. If FEC is successful in correcting an errored block, higher levels don't know that the bits were corrupted. FEC decodes its blocks and passes on the Transmission Words to the next level.

If FEC can't correct an errored block, it still decodes this block and passes on the Transmission Words (with errors) to the next level which may result in Invalid Transmission Words.

If these Invalid Transmission Words are Control Transmission Words, the bit errors may corrupt frame delimiters, primitive signals, or primitive sequences. For example, if bit errors corrupt an R_RDY, the receiver (frame-sender) may not recognize it. Hence, it doesn't increment its remaining-Tx-B2B-credits resulting in the loss of a credit. If bit errors corrupt SOF primitive in a manner that the frame-receiver can't recognize it as the beginning of a frame, an R_RDY isn't sent for this frame resulting in a loss of a credit for the frame-sender. Refer to the section on *B2B Credit Loss and Recovery* for more details on such cases.

If these Invalid Transmission Words are Data Transmission Words, the bit errors may corrupt the content of a frame which results in a CRC corrupted frame. Regardless of the valid or corrupt frame, the frame-receiver sends a R_RDY to the frame-sender. A corrupt frame is dropped which results in reinitiating the entire I/O operation. This results in significant performance degradation because the application goodput lacks even though network throughput may exist.

Irrespective of the location of the bit errors, what's important to understand is that the plan of action in a production environment should remain the same—Detect the bit errors, find their root cause, and make the corrective change as soon as possible.

B2B Credit Loss and Recovery

The remaining-Tx B2B-credits on an FC port fall to zero when it doesn't receive credits (by not receiving R_RDYs) for the frames it had sent earlier. This may happen under the following conditions:

1. **Bit errors:** Bit errors may corrupt R_RDYs and frames leading to loss of Tx B2B credits on the frame sender. This can cause a partial or complete loss of Tx B2B credits.
2. **Severe congestion:** Under congestion, the frame-receiver purposefully stops sending R_RDYs with the goal of reducing the rate of incoming traffic. When this congestion becomes severe, the remaining-Tx-B2B-credits on the frame-sender stay at zero for an extended

duration. This condition is not exactly a “loss of credits” because the credits are not technically lost but withheld.

Fibre Channel has mechanisms to handle both these cases to restore the remaining-Tx-B2B-credits on the frame sender.

Loss of Tx B2B Credits due to Bit Errors

Certain types of bit errors can cause a loss of Tx B2B credits leading to frame-sender’s remaining-Tx-B2B-credits and frame-receiver’s remaining-Rx-B2B-credits going out of sync. In this condition, the number of Tx B2B credits lost can be partial or all.

When it is a partial loss of Tx B2B credits, neither the frame receiver nor the frame-sender is aware that the loss has occurred. If it were possible to view the credits on both sides of a link simultaneously, with no traffic, the frame-sender’s remaining-Tx-B2B-credits would be a smaller number than the full agreed-to number (Tx B2B credits) during link initialization. But the frame-receiver’s remaining-Rx-B2B-credits would be the same as the full agreed to number (Rx B2B credits) during link initiation. Consequently, the frame-receiver doesn’t believe it “owes” any credits to the frame-sender. Refer to the section about *B2B Credit Counters* to understand the difference between Tx B2B credits and remaining-Tx-B2B-credits.

If the number of lost credits does not result in frame-sender’s remaining-Tx-B2B-credits ever going to zero (no incrementing of the Tx-credit-transition-to-zero counter), then it doesn’t even impact performance. However, if an FC port continues to lose Tx credits, its remaining-Tx-B2B-credits eventually fall to zero leading to a complete traffic halt.

Consider an FC port with 8 remaining-Tx-B2B-credits. It sends 256 frames but receives only 250 R_RDYs, resulting in a current value of 2 remaining-Tx-B2B-credits. The frame-receiver, however, returned credits for all the received frames. As a result, it has 8 remaining-Rx-B2B-credits and it’s ready to receive 8 frames, but the sender shows only 2 remaining-Tx-B2B-credits and can only send 2 frames because it believes that the receiver only has 2 available buffers. This out-of-sync condition between frame-sender and frame-receiver may happen because of two cases of bit errors.

1. **Corruption of R_RDY:** Refer to Figure 2-37. The frame-receiver sends a R_RDY but the R_RDY gets corrupted on the way, and thus the frame-sender can't recognize it. As a result, the frame-receiver increments its remaining-Rx-B2B-credits, whereas the frame-sender doesn't increment its remaining-Tx-B2B-credits because it believes that it didn't receive an R_RDY. This corruption of an R_RDY results in the loss of a Tx B2B credit for the frame-sender, which makes the frame-sender and the frame-receiver go out of sync.

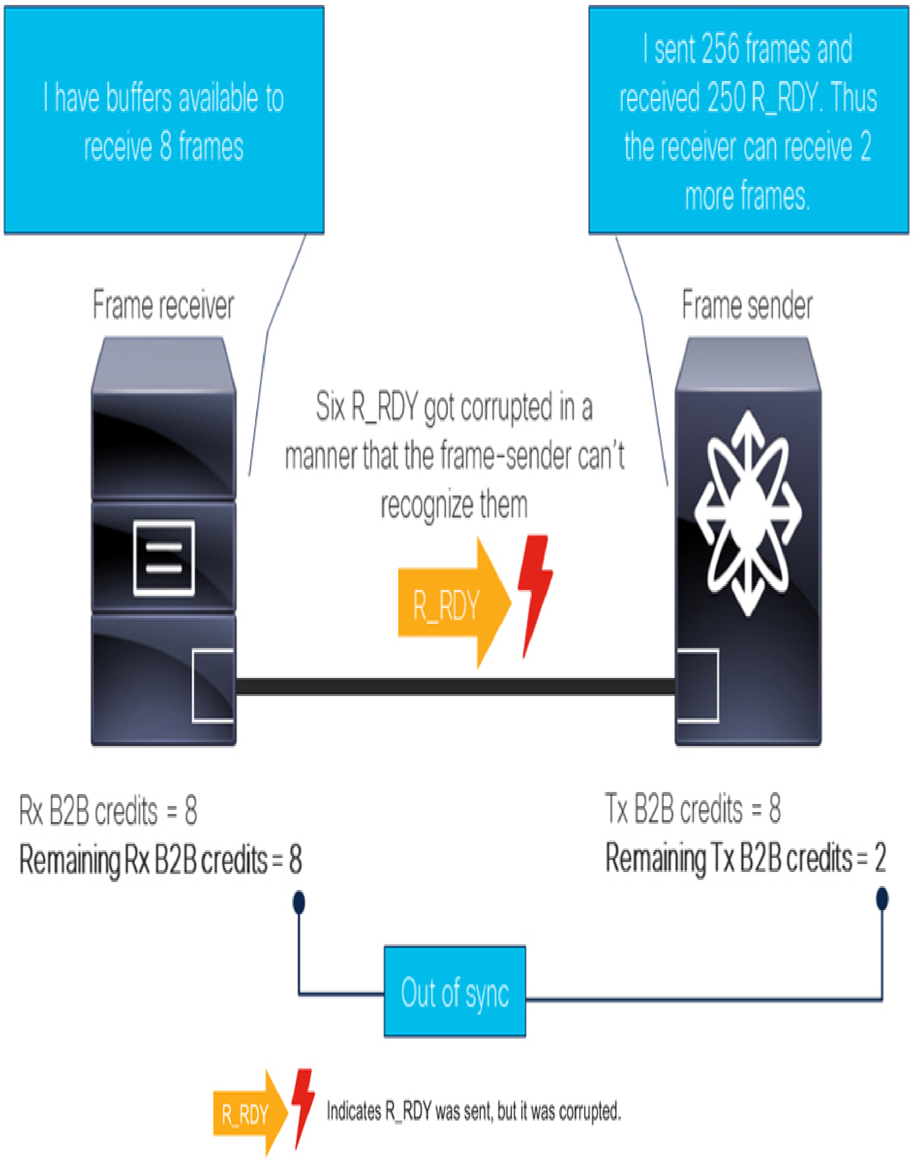

I have buffers available to receive 8 frames
I sent 256 frames and received 250 R_RDY. Thus the receiver can receive 2 more frames.
Frame receiver
Frame sender
Six R_RDY got corrupted in a manner that the frame-sender can't recognize them
R_RDY
Rx B2B credits = 8
Remaining Rx B2B credits = 8
Tx B2B credits = 8
Remaining Tx B2B credits = 2
Out of sync
R_RDY
Indicates R_RDY was sent, but it was corrupted.

Figure 2-37 *Loss of B2B credits due to corruption of R_RDY*

2. **Corruption of a frame:** Refer to Figure 2-38. The frame-sender sends a frame and decrements the remaining-Tx-B2B-credits. But the frame gets corrupted in a specific manner (such as SOF is corrupted) so that the frame-receiver can't recognize that the ingress bits belong to a frame. Because the frame-receiver believes that it never received a frame, there's no reason for it to send an R_RDY to the frame-sender. As a result, the frame-sender's remaining-Tx-B2B-credits don't increment. This corruption of the frame in a specific manner that makes it unrecognizable as a frame, results in a loss of Tx B2B credit for the frame-sender and makes the frame-sender and the frame-receiver go out of sync.

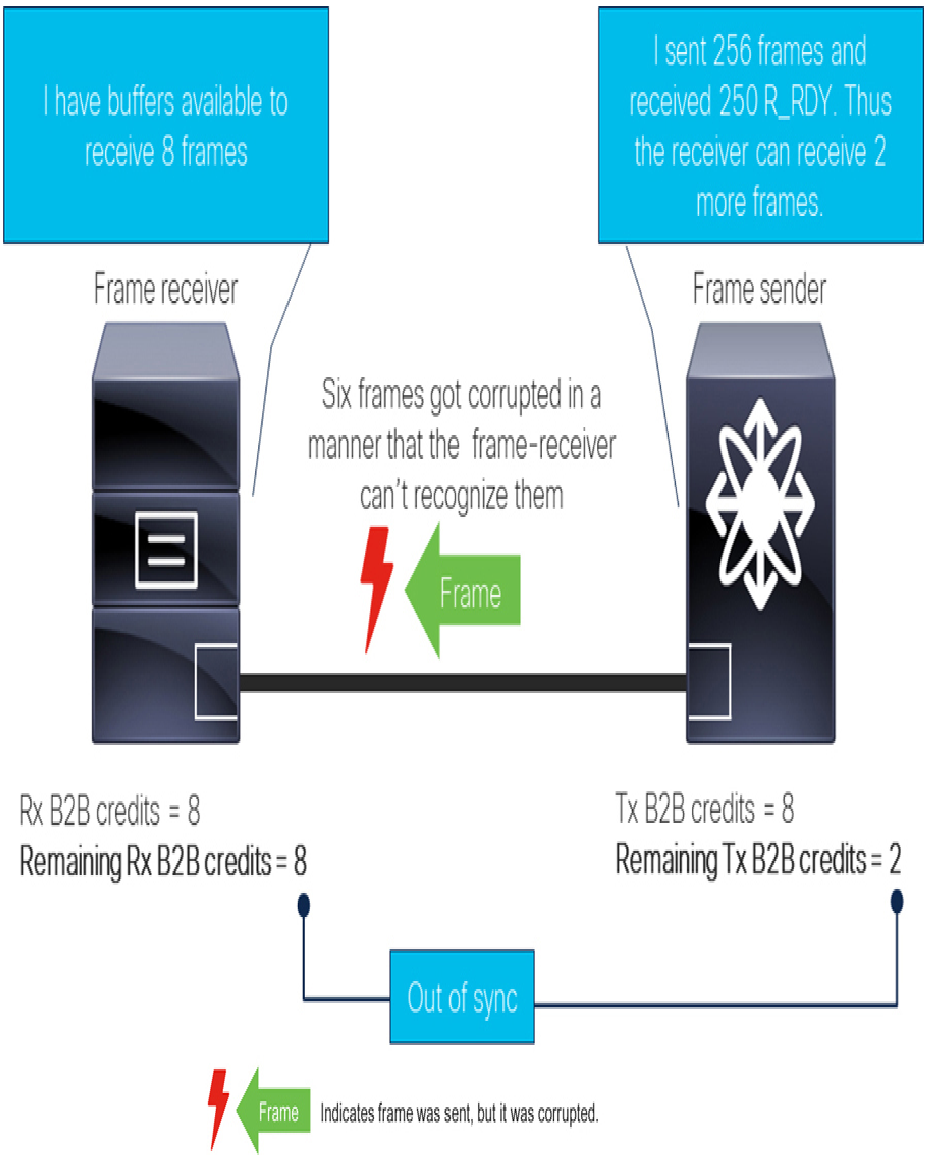

I have buffers available to receive 8 frames
I sent 256 frames and received 250 R_RDY. Thus the receiver can receive 2 more frames.
Frame receiver
Frame sender
Six frames got corrupted in a manner that the frame-receiver can't recognize them
Frame
Rx B2B credits = 8
Remaining Rx B2B credits = 8
Tx B2B credits = 8
Remaining Tx B2B credits = 2
Out of sync
Frame
Indicates frame was sent, but it was corrupted.

Figure 2-38 *Loss of B2B credits due to a corrupted frame*

A common misconception is that a corrupted frame always results in a CRC error. As previously explained, a frame may get corrupted in the following two distinct ways:

- A receiver detects a valid frame, but the CRC verification fails. This results in incrementing the ingress frame counter and the CRC error counter. The frame-receiver sends a R_RDY to the frame-sender regardless of the CRC validity of a frame. This type of frame corruption doesn't result in the loss of a credit for the frame-sender.

- The frame is corrupted in a specific manner that the receiver can't recognize that the ingress bits belong to a frame. Because it's not even a valid frame, the receiver doesn't run a CRC calculation on it, and thus the ingress frame counter and the CRC error counter don't increment. The port, however, shows lower-level error counters such as Invalid Transmission Words and FEC uncorrected blocks. This type of frame corruption results in the loss of a credit for the frame sender.

In production environments, detecting this kind of out-of-sync condition is extremely difficult. The illustration in Figure 2-37 and Figure 2-38 assume that a frame-sender stops sending frames after sending 256 frames. In production networks, a port continuously sends frames while it receives R_RDY. By the time the remaining-Tx-B2B-credits counter is monitored, its value may change. To know for sure if there has been credit loss, traffic flow must stop which is not possible in a production network. Because of these reasons, it's important to proactively monitor bit errors.

Also, viewing the credits simultaneously on both sides of a link is practically not possible, so the complete loss of all Tx B2B credits is not easily distinguishable from the situation of zero remaining-Tx-B2B-credits for an extended duration discussed next.

Zero Tx B2B Credits for an Extended Duration

An FC port momentarily reaching zero remaining-Tx-B2B-credits is a common condition (it results in incrementing the credit-transition-to-zero counters. Refer to Chapter 3 for more details). It just means that the B2B flow control mechanism is working, equalizing the ingress traffic rate to the egress traffic rate, and preventing loss of frames due to buffer overrun. The port can resume sending frames as soon as it receives a credit (by receiving a R_RDY).

But if the port is at zero remaining-Tx-B2B-credits for an extended duration (Figure 2-39), it doesn't wait forever. It invokes a special procedure to reinitialize the B2B credits in both directions as explained in the section on *Credit Loss Recovery Using Link Reset Protocol*.

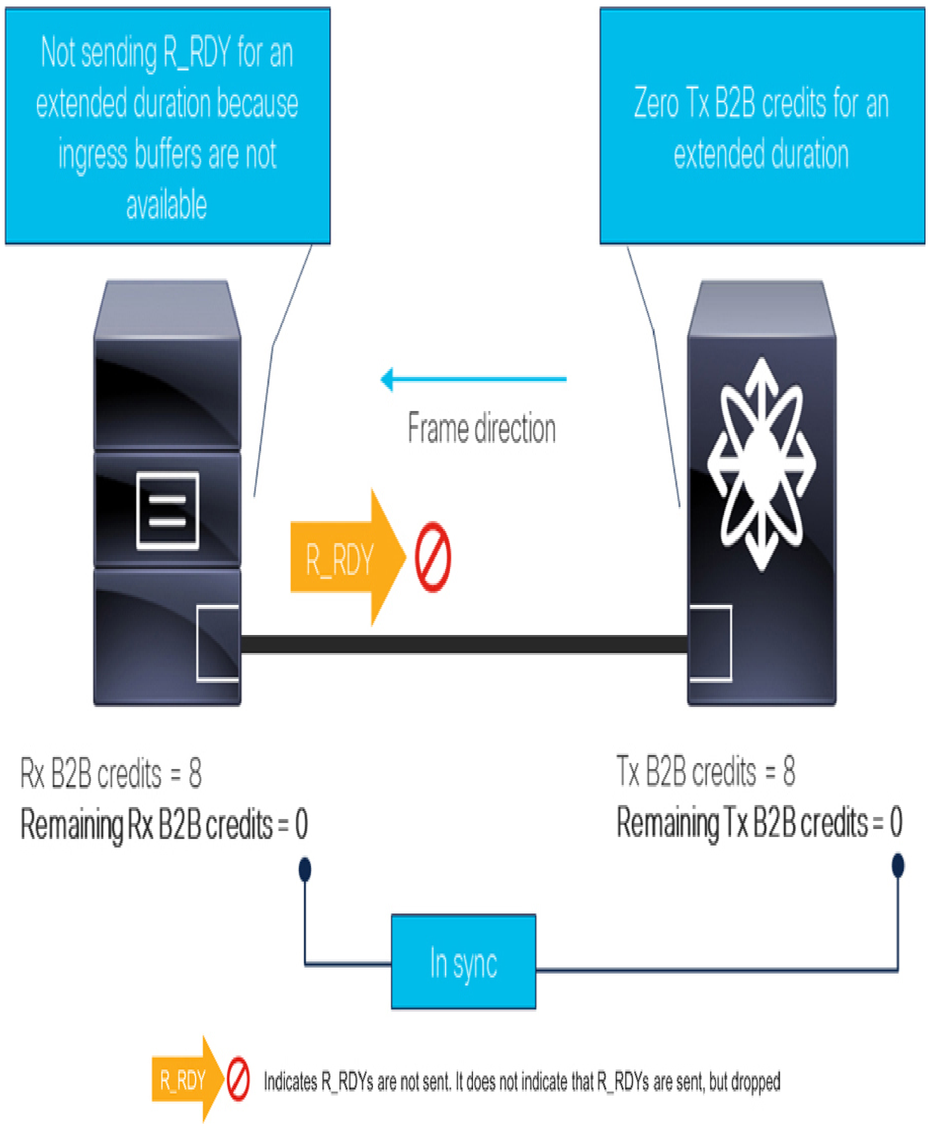

Not sending R_RDY for an extended duration because ingress buffers are not available
Zero Tx B2B credits for an extended duration
Frame direction
R_RDY
Rx B2B credits = 8
Remaining Rx B2B credits = 0
Tx B2B credits = 8
Remaining Tx B2B credits = 0
In sync
R_RDY
Indicates R_RDYs are not sent. It does not indicate that R_RDYs are sent, but dropped

Figure 2-39 *Zero remaining Tx B2B credits for an extended duration due to severe congestion*

Whether the frame-sender's remaining-Tx-B2B-credits and the frame-receiver remaining-Rx-B2B-credits are in sync or not, depends on the reasons that led to zero remaining-Tx-B2B-credits for an extended duration on a port. If the reason is bit errors, the neighbors are not in sync. However, if the reason is severe congestion, the neighbors may be in sync.

Credit Loss Recovery Using B2B State Change Mechanism

The B2B state change mechanism recovers from a condition when the frame-sender’s remaining-Tx-B2B-credits and the frame-receiver’s remaining-Rx-B2B-credits go out of sync.

As explained earlier, this out-of-sync condition may be a result of bit errors which may corrupt R_RDYs, and thus the frame-sender can’t recognize them. Alternatively, a frame may get corrupted in a specific manner so that the receiver is unable to recognize it as a frame resulting in not sending a R_RDY. In both these conditions, the frame-sender loses a Tx B2B credit; therefore, it has a lower number of remaining-Tx-B2B-credits than frame-receiver’s remaining-Rx-B2B-credits.

The B2B state change mechanism between neighboring FC ports resolves this out-of-sync condition by informing each other about the number of frames sent and the number of R_RDY sent. Sending a pre-determined number of frames and R_RDYs followed by informing the neighbor about that number gives enough information to the neighbor to detect loss of credits and activate a credit recovery mechanism.

The B2B state change mechanism has two phases:

1. Initially, the neighbors agree on using the B2B state change mechanism and the number to be used.
2. As the frames and R_RDYs flow, the neighbors inform each other after sending the agreed number of frames and R_RDYs. This happens independently in both directions. If the neighbors detect a loss of credits, they activate the recovery mechanism.

Negotiation at Link Initialization

The B2B state change mechanism is an optional feature. It can recover the lost credits only if the neighbors agree on using it during the link initialization phase. Just like the exchange of B2B credits, this capability is agreed upon through the FLOGI phase between a N_Port and F_Port and through the ELP phase between two E_Ports.

During FLOGI or ELP phase, two neighboring FC ports inform each other of a non-zero B2B state change number (BB_SC_N), as shown in Figure 2-40. The neighbors may inform different values of BB_SC_N to each other. They agree on using the greater of the two numbers.

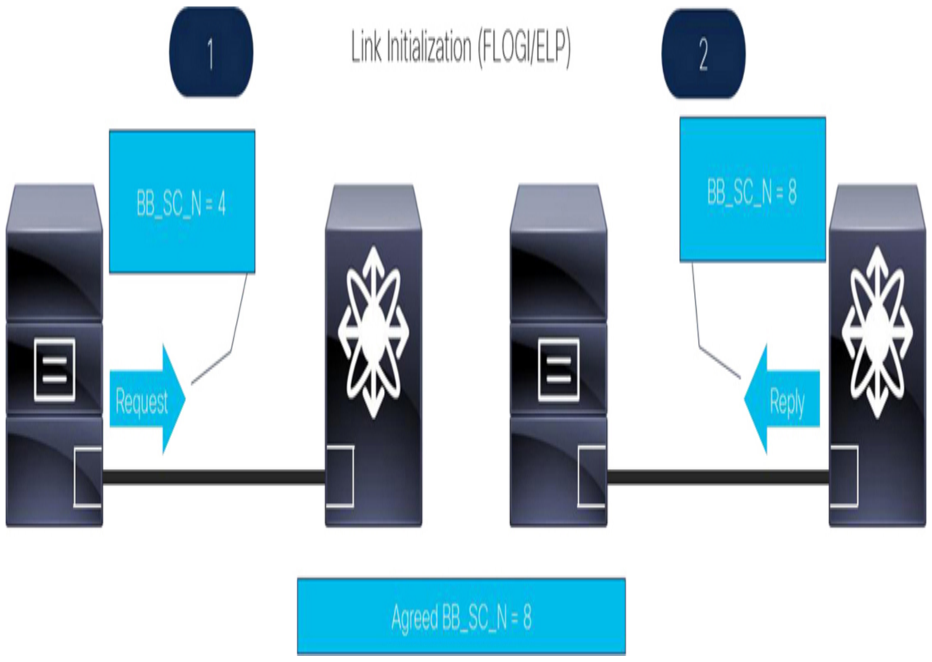


Figure 2-40 *Initial agreement on using the B2B state change mechanism*

The B2B state change mechanism is disabled if either neighbor informs the value of BB_SC_N as zero.

Periodic Detection and Recovery of Credit Loss

As previously explained, a frame-sender may lose credits because of the corruption of R_RDY or the corruption of frames in a specific manner. These two conditions have different recovery mechanisms.

Credit Loss Recovery When R_RDYs Are Corrupted

After agreeing on a BB_SC_N number, each side of the link starts keeping track of the number of frames sent, frames received, R_RDYs sent, and R_RDYs received. A frame-receiver informs the frame-sender that it has sent $2^{BB\_SC\_N}$ number of R_RDYs by sending a BB_SCr (B2B State Change

R_RDY) primitive signal. This process repeats after every $2^{BB\_SC\_N}$ number of R_RDYs are sent. For example, if BB_SC_N is 8, a frame-receiver sends 256 (2^8) R_RDYs between two consecutive BB_SCr (Figure 2-41).

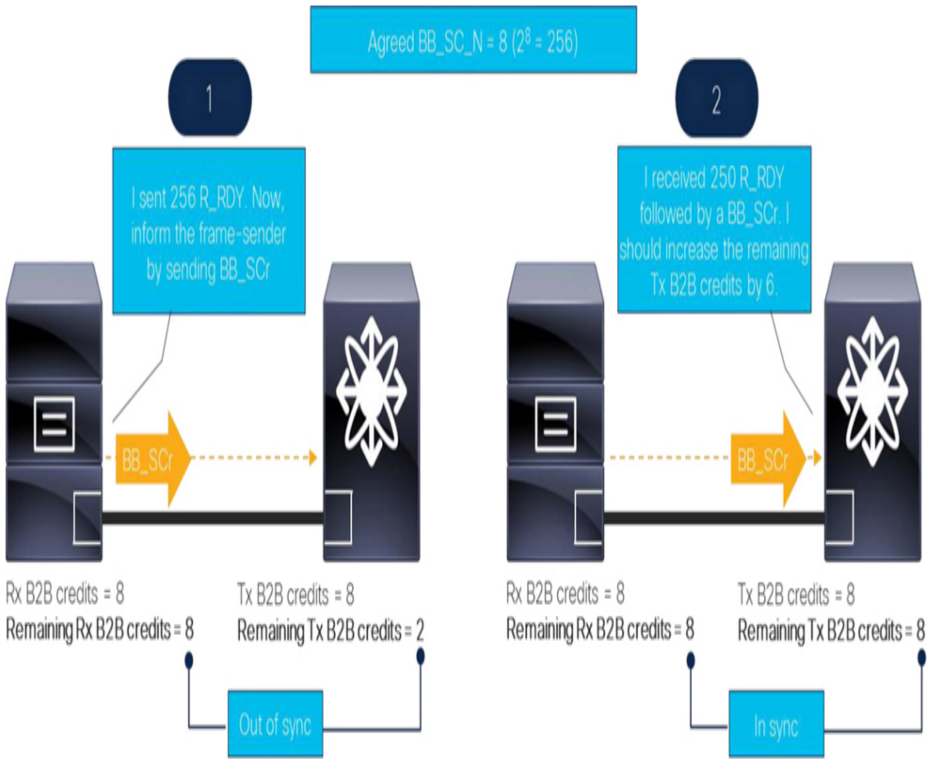

Agreed BB_SC_N = 8 (2⁸ = 256)
1
I sent 256 R_RDY. Now, inform the frame-sender by sending BB_SCr
2
I received 250 R_RDY followed by a BB_SCr. I should increase the remaining Tx B2B credits by 6.
BB_SCr
BB_SCr
Rx B2B credits = 8
Remaining Rx B2B credits = 8
Tx B2B credits = 8
Remaining Tx B2B credits = 2
Out of sync
Rx B2B credits = 8
Remaining Rx B2B credits = 8
Tx B2B credits = 8
Remaining Tx B2B credits = 8
In sync

Figure 2-41 *Credit recovery using B2B state change mechanism when a R_RDY is lost*

The frame-sender already keeps a count of the number of received R_RDY. It detects a loss of R_RDY if fewer R_RDY are received than $2^{BB\_SC\_N}$ between two consecutive BB_SCr primitives signals. For example, if BB_SC_N is 8, the frame-sender expects to receive 256 R_RDY between two consecutive BB_SCr primitives. But if it counts only 250 R_RDYs, it has detected a loss of 6 Tx B2B credits.

The frame-sender recovers the lost credits by simply incrementing its remaining-Tx-B2B-credits by the lost number of credits. For example, if the frame-sender detects a loss of 6 credits, it increments its remaining-Tx-B2B-credits by 6.

Credit Loss Recovery When Frames Are Corrupted

After agreeing on a BB_SC_N number, a frame-sender informs the frame-receiver that it has sent $2^{BB\_SC\_N}$ frames by sending a BB_SCs (B2B State Change SOF) primitive signal. This process repeats after sending $2^{BB\_SC\_N}$ frames. For example, if BB_SC_N is 8, a frame-sender sends 256 (2^8) frames between two consecutive BB_SCs primitives (Figure 2-42, Step-1).

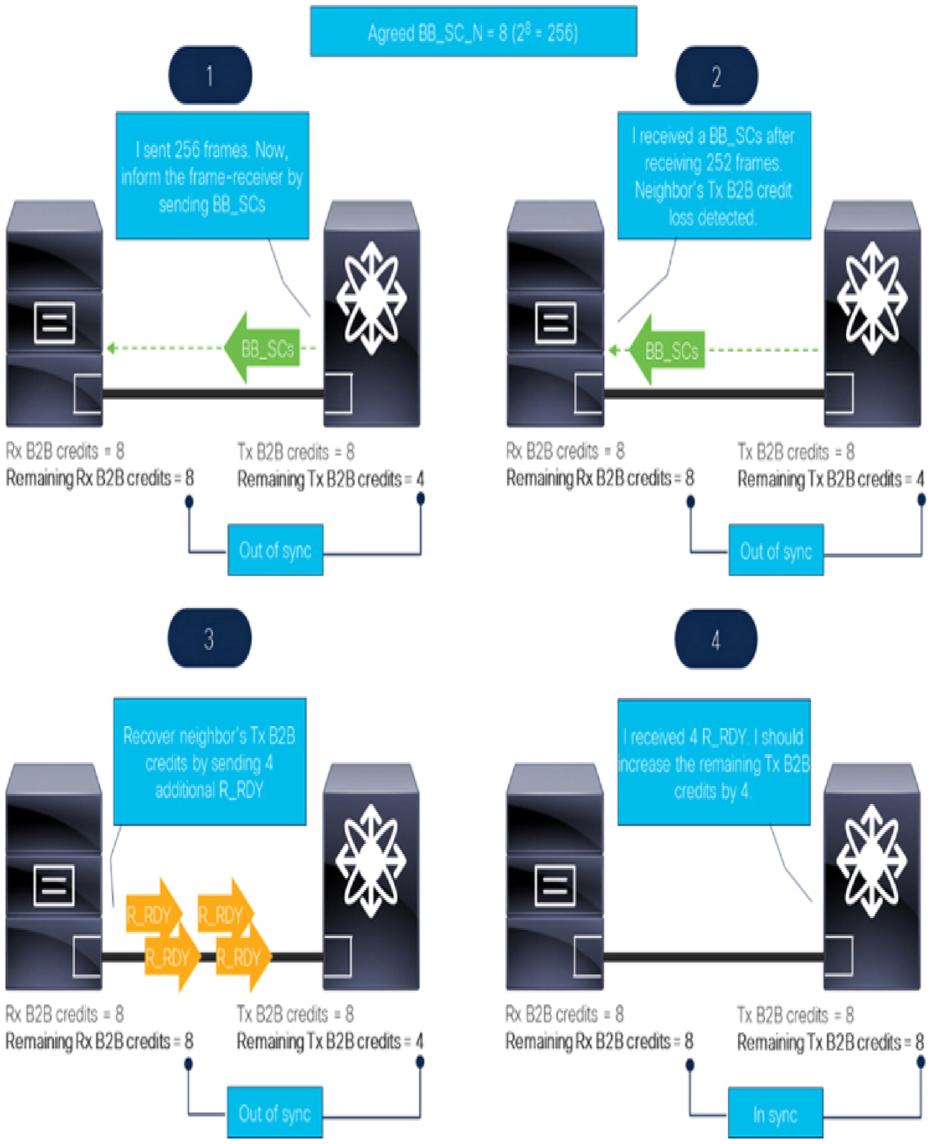

Agreed BB_SC_N = 8 (2⁸ = 256)
1
I sent 256 frames. Now, inform the frame-receiver by sending BB_SCs
BB_SCs
Rx B2B credits = 8
Remaining Rx B2B credits = 8
Tx B2B credits = 8
Remaining Tx B2B credits = 4
Out of sync
2
I received a BB_SCs after receiving 252 frames. Neighbor's Tx B2B credit loss detected.
BB_SCs
Rx B2B credits = 8
Remaining Rx B2B credits = 8
Tx B2B credits = 8
Remaining Tx B2B credits = 4
Out of sync
3
Recover neighbor's Tx B2B credits by sending 4 additional R_RDY
R_RDY
R_RDY
R_RDY
R_RDY
Rx B2B credits = 8
Remaining Rx B2B credits = 8
Tx B2B credits = 8
Remaining Tx B2B credits = 4
Out of sync
4
I received 4 R_RDY. I should increase the remaining Tx B2B credits by 4.
Rx B2B credits = 8
Remaining Rx B2B credits = 8
Tx B2B credits = 8
Remaining Tx B2B credits = 8
In sync

Figure 2-42 *Credit recovery using B2B state change mechanism when a frame is lost*

As mentioned, the frame-receiver keeps a count of the number of received frames. It detects a loss of frames if fewer frames are received than $2^{BB\_SC\_N}$ between two consecutive BB_SCs primitives. For example, if BB_SC_N is 8, it expects to receive 256 (2^8) frames between two consecutive BB_SCs primitives. But if it counts only 252 frames, it has detected a loss of 4 frames from the frame-sender (Figure 2-42, Step-2).

The frame-receiver recovers the frame-sender's lost credits by sending additional R_RDYs. For example, if the frame-receiver detects a loss of 4 frames, it sends 4 additional R_RDYs to the frame-sender (Figure 2-42, Step-3). The frame-sender increments its remaining-Tx-B2B-credits as a normal process without even realizing that these were extra credits sent (Figure 2-42, Step-4).

Important Details About the B2B State Change Mechanism

There are a few more details about the B2B state change mechanism that is worth knowing.

1. Like the R_RDY primitive, the BB_SCr and BB_SCs primitives don't consume the data rate of a link. They are inserted between two frames in place of inter-frame fill words. Refer to the section on *Effect of Primitive Signals on Data Rate* for more details.
2. Like the R_RDY primitive, the BB_SCr and BB_SCs primitives remain local to a link because the B2B state change mechanism operates locally on a link between two directly connected FC ports that have agreed on using it.
3. Like the R_RDY primitive, the BB_SCr and BB_SCs primitives are not sent reliably. They may get corrupted because of bit errors and their receiver may not detect them. However, losing BB_SCr and BB_SCs occasionally doesn't affect the recovery mechanism because these primitive signals are sent periodically after sending $2^{BB\_SC\_N}$ R_RDYs or frames. If a BB_SCr or BB_SCs is lost, their receiver can detect the lost credits when BB_SCr or BB_SCs are received the next time. This is possible because of using the modulo operator to calculate the difference in the actual and the expected number of received R_RDY or frames. For example, with BB_SC_N being 8, a frame-receiver sends 256 R_RDY between two consecutive BB_SCr primitives. The frame-sender received only 250 R_RDY resulting in a loss of 6 Tx B2B credits. The BB_SCr also gets corrupted, and

thus the frame-sender doesn't recognize it. Consequently, it doesn't activate the recovery. As the frames flow, the frame-receiver sends the next BB_SCr after sending the next 256 R_RDY. Assuming R_RDY and BB_SCr are not corrupted now, the frame-sender uses the following formula to detect credit loss.

Number of lost credits = ($2^{BB\_SC\_N}$ – Number of R_RDY received since last BB_SCr) modulo $2^{BB\_SC\_N}$

Applying this formula to the example

Number of lost credits = (256 – 506) modulo 256 = 6

As illustrated, although the BB_SCr and BB_SCs primitive signals are not sent reliably, losing them doesn't affect the credit recovery mechanism.

4. The B2B state change mechanism can recover a few lost credits on a link. But it can't recover if all the credits are lost because at that point no further frames or R_RDYs are sent and the BB_SC_N count is not reached. When this occurs credit loss recovery using the Link Reset Protocol occurs.

5. The B2B state change mechanism is an optional feature and it's operational only when neighboring FC ports agree on using it.

While operating production storage networks, what's important to remember is the following:

1. Ensure that the FC port supports the B2B state change mechanism. Not just the readiness, ensure that the firmware supports it, and it is operational between the neighbors.

2. When the frame sender increments its remaining Tx B2B credits to recover from lost or corrupted R_RDYs the neighbor doesn't know this occurred, Likewise, when the frame-receiver sends additional R_RDYs, the frame-sender doesn't know these R_RDYs are additional. It treats them as normal R_RDYs. The counts of both cases are maintained independently on each side of the link. Because of this, to determine if this is occurring, the

interface statistics on both sides of the link must be proactively monitored and investigated.

For example, Cisco MDS switches report the following counters that detect if the B2B state change mechanism was activated:

- The number of times when a port detects the loss of its Tx B2B credit after receiving a BB_SCr primitive and recovering credits by incrementing the remaining-Tx-B2B-credits.
- The number of times when a port detects the loss of its neighbor's Tx B2B credits after receiving a BB_SCs primitive and recovering the neighbor's Tx credits by sending additional R_RDYs.

Note the above don't indicate how many credits were recovered, just that the mechanism detected the out of sync condition and recover at least one credit.

Any such counter indicates a degraded quality of the link resulting from bit errors or a similar problem. Watch for other error counters such as FEC corrected blocks, FEC uncorrected blocks, and Invalid Transmission Words. Although the B2B state change counters indicate a successful recovery of the lost credits, it doesn't resolve the real problem that resulted in the loss of credits in the first place. This problem must be investigated and resolved to prevent congestion and maintain a healthy network.

Credit Loss Recovery Using Link Reset Protocol

An FC port invokes the Link Reset protocol (Figure 2-43) to attempt to recover the credits when it has zero remaining-Tx-B2B-credits for an extended duration. The Fibre Channel standard refers to this duration as Error Detect Timeout Value (E_D_TOV). This duration must be long enough for a port to give up on the normal mechanism of the return of B2B credits. Cisco MDS switches set E_D_TOV to 2 seconds by default.

However, for initiating Link Reset Protocol MDS switches use the default values of 1.5 seconds for an E_Port and 1 second for a F_Port.

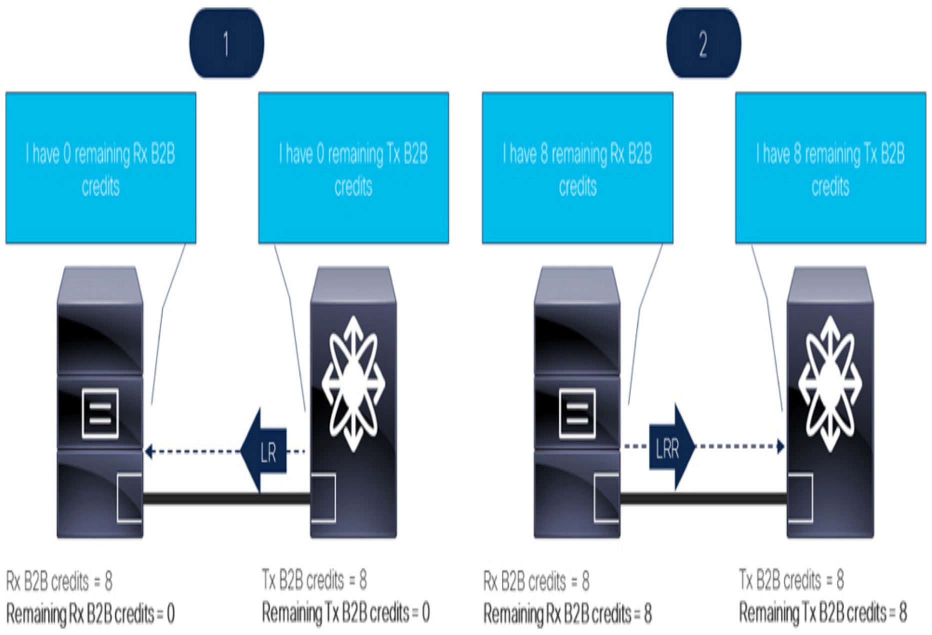

1
2
I have 0 remaining Rx B2B credits
I have 0 remaining Tx B2B credits
I have 8 remaining Rx B2B credits
I have 8 remaining Tx B2B credits
LR
LRR
Rx B2B credits = 8
Remaining Rx B2B credits = 0
Tx B2B credits = 8
Remaining Tx B2B credits = 0
Rx B2B credits = 8
Remaining Rx B2B credits = 8
Tx B2B credits = 8
Remaining Tx B2B credits = 8

Figure 2-43 *Credit loss recovery by Link Reset protocol*

Note:

Cisco MDS switches call this ‘Credit Loss Recovery’.

The Link Reset Protocol starts when an FC port sends a Link Reset (LR) primitive sequence. The neighbor, after receiving LR, returns a Link Reset Response (LRR) primitive sequence. Before returning LRR, the neighbor processes the frames in its ingress buffers to make the buffers available for re-use. If, however, it’s unable to process the frames, the neighbor may drop the frames before sending LRR or not send an LRR at all resulting in a link failure.

When Link Reset Protocol completes, both FC ports have reset their remaining Tx and Rx B2B credits to the same value as they had learned from each other during link initialization (FLOGI or ELP).

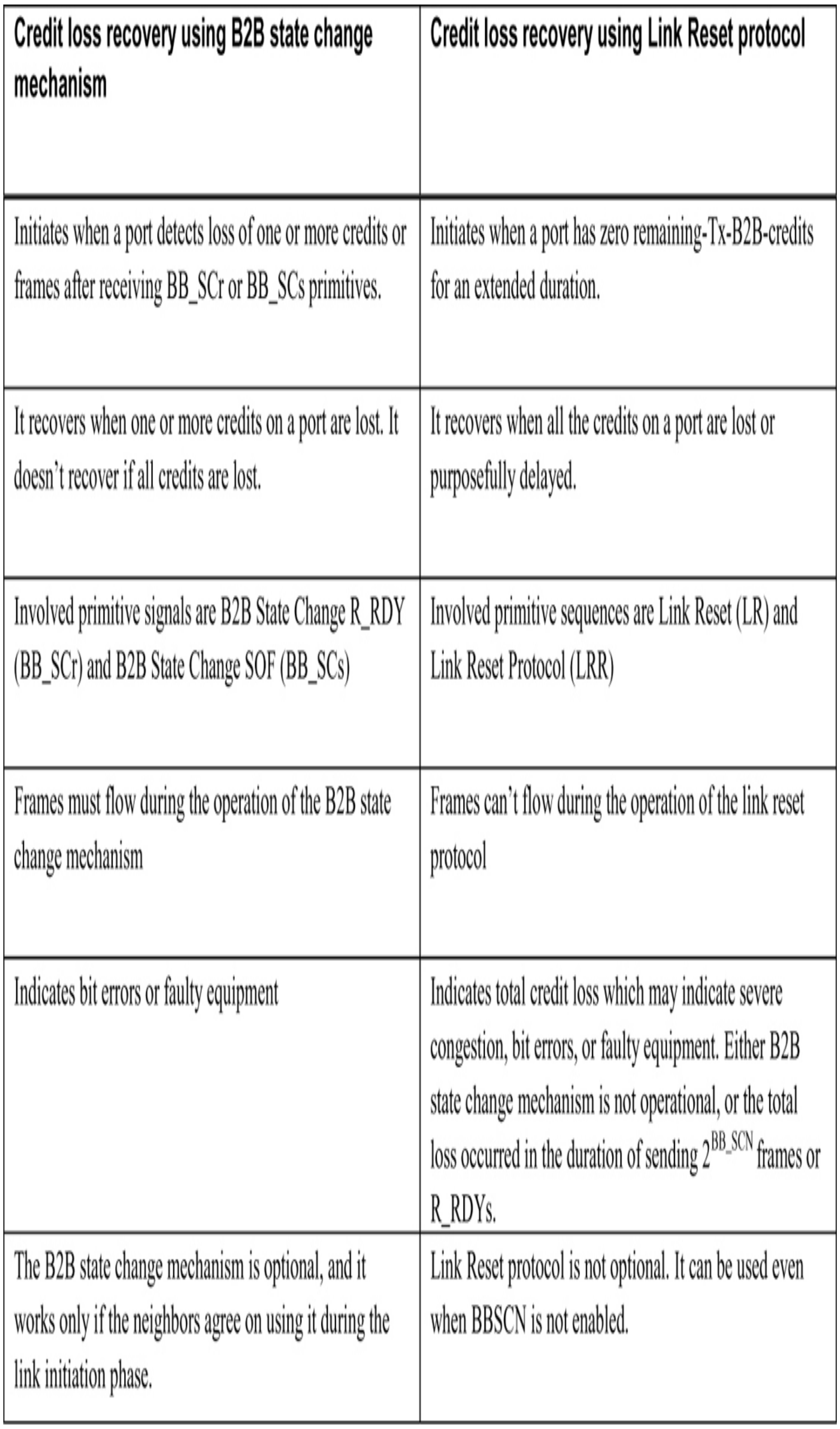
Despite the name—Link Reset—the Link Reset Protocol is really a link credit reset and doesn’t reset the port or link itself. The exchange of an LR and an LRR on a Fibre Channel port is different from a port flap. Credit loss recovery doesn’t involve loss of signal or loss of sync between the FC ports.

Frames are not sent bidirectionally on a port during the Link Reset Protocol. After the Link Reset Protocol, frames can be sent on the link using the normal mechanism. Although the Link Reset protocol recovers from the credit loss condition, it doesn’t resolve the original reason that would have caused it. If the original issue is not solved, the port may lose its Tx B2B credits for an extended duration again and the Link Reset Protocol repeats.

Comparison of B2B State Change Mechanism and Link Reset Protocol

Although B2B state change mechanism and Link Reset Protocol recover lost credits, their operations are different. Table 2-5 compares both approaches.

Table 2-5 *B2B State Change mechanism compared with Link Reset Protocol*



Credit loss recovery using B2B state change mechanism	Credit loss recovery using Link Reset protocol
Initiates when a port detects loss of one or more credits or frames after receiving BB_SCr or BB_SCs primitives.	Initiates when a port has zero remaining-Tx-B2B-credits for an extended duration.
It recovers when one or more credits on a port are lost. It doesn't recover if all credits are lost.	It recovers when all the credits on a port are lost or purposefully delayed.
Involved primitive signals are B2B State Change R_RDY (BB_SCr) and B2B State Change SOF (BB_SCs)	Involved primitive sequences are Link Reset (LR) and Link Reset Protocol (LRR)
Frames must flow during the operation of the B2B state change mechanism	Frames can't flow during the operation of the link reset protocol
Indicates bit errors or faulty equipment	Indicates total credit loss which may indicate severe congestion, bit errors, or faulty equipment. Either B2B state change mechanism is not operational, or the total loss occurred in the duration of sending 2^{BB_SCN} frames or R_RDYs.
The B2B state change mechanism is optional, and it works only if the neighbors agree on using it during the link initiation phase.	Link Reset protocol is not optional. It can be used even when BBSCN is not enabled.

Fibre Channel Counters — Summary

Table 2-6 summarizes the Fibre Channel error counters explained in this chapter.

Table 2-6 *Fibre Channel error counters*

Counter name	Direction	Description
FEC corrected blocks	Rx	The number of FEC blocks that could be corrected using FEC. It indicates degraded link quality or faulty sender. Refer to the *Forward Error Correction* section for more details.
FEC uncorrected blocks	Rx	The number of FEC blocks that couldn't be corrected using FEC. It indicates degraded link quality or faulty sender. When this counter increments, ITW should also increment. Refer to the *Forward Error Correction* section for more details.
Invalid Transmission Words (ITW)	Rx	The number of Transmission Words that don't pass the validity checks by the receiver. It indicates degraded link quality or faulty sender. May cause a loss of B2B credits or frames with invalid CRC errors. Refer to the *Invalid Transmission Words* section for more details.
Not Operational Sequence (NOS)	Rx and Tx	Number of NOS Primitive Sequences. It indicates link failure. Refer to the *Link Initialization Counters* section for more details.
Offline Sequence (OLS)	Rx and Tx	Number of OLS Primitive Sequences. It indicates ongoing Link Initialization Protocol. It is sent in response to NOS. Refer to the *Link Initialization Counters* section for more details.
Link Reset (LR)	Rx and Tx	Number of LR Primitive Sequences. If these are approximately the same number as NOS and OLS then it indicates nothing more than normal link initialization. If the number of LRs are more than NOS and OLS then it indicates there has been a total loss of B2B credits on the link. LR is transmitted after an extended duration of continuous zero remaining Tx B2B credits. Once the LR is sent, its sender expects an LRR within an interval (such as 100 ms) to complete the Link Reset Protocol. This bidirectionally initializes or recovers the B2B credits. Refer to the *Link Initialization Counters* and *Credit Loss Recovery Using Link Reset Protocol* sections for more details.
Link Reset Response	Rx and Tx	Number of LRR Primitive Sequences sent in response to LR.

Link Reset Response (LRR)	Rx and Tx	Number of LRR Primitive Sequences sent in Response to LR. Refer to the *Link Initialization Counters* and *Credit Loss Recovery Using Link Reset Protocol* sections for more details.
Link Failure (LF)	Rx	Number of link failure events detected on an FC port. A common example is when no light is present, or cable is not connected. Refer to the *Link Initialization Counters* section for more details.
Sync Loss	Rx	Number of times when synchronization was lost between the FC ports. Synchronization is mandatory for data transmission on a Fibre Channel link. Sync loss counter during the initial bring-up of a link is normal but it shouldn't increment after a link comes up. Continuous increase in Sync loss indicates lower-level errors that are preventing the sender and receiver to achieve bit and Transmission Word synchronization. Refer to the *Link Initialization Counters* section for more details.
Signal Loss	Rx	Number of times when loss of signal was detected. Signal loss during the initial bring-up of a link is normal but it shouldn't increment after a link comes up. Continuous increase of Signal loss counter indicates a lower-level error. Refer to the *Link Initialization Counters* section for more details.
Primitive Sequence Protocol Error (PE)	Rx	The number of Primitive Sequences that were errored and the errors can't be specified by other codes. Refer to the *Link Initialization Counters* section for more details.
Cyclic Redundancy Check (CRC) errors	Rx	The number of frames that fail to match the value of the CRC field with the value of the locally calculated Frame Check Sequence (FCS) on the extended headers, if any, the frame headers, and the data field in the frame. CRC errors indicate corrupted frames because of degraded quality of the link or faulty sender. Refer to the *CRC* section for more details.
BB_SCs credit resend actions	Rx (Frame receiver	Number of times when the B2B state change mechanism detected loss of credits on the neighbor because of loss of frames and it sent

actions	(Frame-receiver takes the action)	loss of credits on the neighbor because of loss of frames and it sent additional R_RDYs to the neighbor to help it recover the credits. This counter indicates bit errors. Refer to the *Credit Loss Recovery Using B2B State Change Mechanism* section for more details.

Example 2-2 shows the error counters on a port on Cisco MDS switch.

Example 2-2 *Error counters on a Fibre Channel port on a Cisco MDS switch.*

```
MDS9000# show interface fc1/2 counters detailed
<snip>
Link Stats:
  Rx Link failures:
0
  Rx Sync losses:
0
  Rx Signal losses:
0
  Rx Primitive sequence protocol errors:
0
  Rx Invalid transmission words:
0
  Rx Invalid CRCs:
0
  Rx Delimiter errors:
0
  Rx fragmented frames:
0
  Rx frames with EOF aborts:
0
  Rx unknown class frames:
0
  Rx Runt frames:
0
  Rx Jabber frames:
0
  Rx too long:
0
```

```
  Rx too short:
0
  Rx FEC corrected blocks:
2
  Rx FEC uncorrected blocks:
0
  Rx Link Reset(LR) while link is active:
0
  Tx Link Reset(LR) while link is active:
0
  Rx Link Reset Responses(LRR):
0
  Tx Link Reset Responses(LRR):
0
  Rx Offline Sequences(OLS):
0
  Tx Offline Sequences(OLS):
0
  Rx Non-Operational Sequences(NOS):
0
  Tx Non-Operational Sequences(NOS):
0



Congestion Stats:
  Tx Timeout discards:
0
  Tx Credit loss:
0
  BB_SCs credit resend actions:
0
  BB_SCr Tx credit increment actions:
0
  TxWait 2.5us due to lack of transmit credits:
0
  Percentage TxWait not available for last
1s/1m/1h/72h:           0%/0%/0%/0%
  Rx B2B credit remaining:
500
  Tx B2B credit remaining:
500
  Tx Low Priority B2B credit remaining:
```

```
500
  Rx B2B credit transitions to zero:
0
  Tx B2B credit transitions to zero:
0
<snip>
MDS9000#
```

Summary

A Fibre Channel fabric connects end devices, such as hosts and storage arrays, using switches and links. Although the source of congestion may be within any of these devices, congestion due to end devices, especially hosts, is the most common.

These end devices cause congestion due to two major reasons. The first reason, called slow-drain, happens when a device is unable to process frames as fast as the ingress rate. The second reason, called over-utilization of the edge links, happen when a switch tries to send more data to an end device than the capacity of the link. In both types, congestion spreads from the destination to the source of the frames, affecting traffic between other devices that share the same network path.

Next, this chapter explains the frame forwarding architecture of a Fibre Channel switch using the example of Cisco MDS switch.

The chapter concludes with an explanation of the exchange of read and write I/O operations on Fibre Channel media which helps in understanding the effect of congestion due to bit errors.

Overall, this chapter provides an in-depth explanation of the types of congestion commonly seen in the production Fibre Channel fabrics. A thorough understanding of these topics is important for detection, troubleshooting, and preventing congestion, which is explained in the following chapters.

References In This Chapter

- INCITS 488-2016, FC-FS-4, Fibre Channel Framing and Signaling - 4
- INCITS 545-2018, FC-FS-5, Fibre Channel Framing and Signaling - 5
- INCITS 512-2015, FC-PI-6, Fibre Channel Physical Interfaces - 6
- INCITS 533-2016, FC-PI-6P, Fibre Channel Physical Interfaces - 6P
- INCITS 479-2011, FC-PI-5, Fibre Channel Physical Interfaces – 5
- IEEE 803.2-2018, IEEE Standard for Ethernet
- Cisco Slow-Drain Device Detection, Troubleshooting, and Automatic Recovery White Paper
- Cisco MDS 9000 Family Switch Architecture White Paper
- Cisco MDS 9000 NX-OS Software Configuration Guides

Chapter 3. Detecting Congestion in Fibre Channel Fabrics

This chapter covers the following topics:

- Congestion detection workflow.
- Congestion detection metrics.
- Congestion detection metrics and commands on Cisco MDS switches.
- Automatic Alerting.
- Detecting congestion using remote monitoring platforms.
- Detecting congestion on long-distance FC links.

Congestion Detection Workflow

The first step for detecting congestion is to understand it, which Chapter 2 explains in detail. The next step is to understand the following factors.

- What to detect.
  - What is the effect of congestion? In other words, how severe is it?
  - What is the cause of congestion?
  - Where is the source of congestion (culprits)?
  - Where is the spread of congestion (victims)?
  - When did congestion happen?

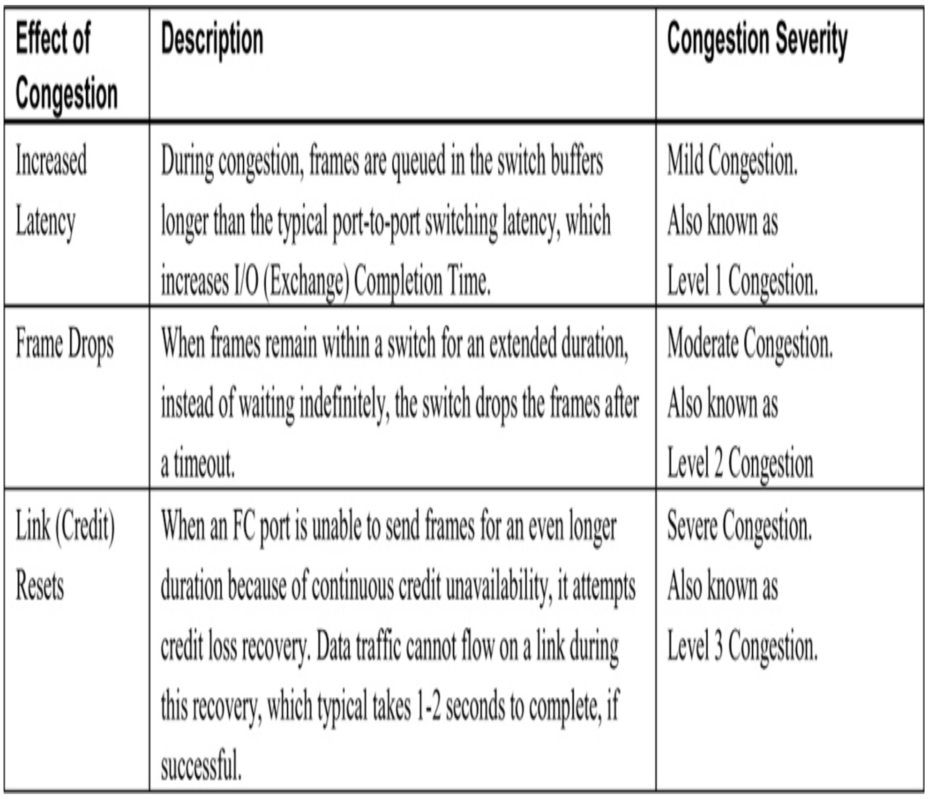
- How to detect:
  - Reactive approach: Detect and troubleshoot congestion events after they happen.
  - Proactive: Detect congestion events in real time.
  - Predictive: Predict congestion events before they happen.
- Where to Detect
  - Natively on the devices, such as switches, hosts/servers, or storage arrays.
  - Using remote monitoring platforms.

These factors are explained in the following sections.

Effect of Congestion (Congestion Severity)

Congestion in a Fibre Channel fabric can lead to increased latency, frames drops, and link credit resets. Table 3-1 explains them further.

Table 3-1 *Effect of Congestion in Fibre Channel fabrics and severity*



Effect of Congestion	Description	Congestion Severity
Increased Latency	During congestion, frames are queued in the switch buffers longer than the typical port-to-port switching latency, which increases I/O (Exchange) Completion Time.	Mild Congestion. Also known as Level 1 Congestion.
Frame Drops	When frames remain within a switch for an extended duration, instead of waiting indefinitely, the switch drops the frames after a timeout.	Moderate Congestion. Also known as Level 2 Congestion
Link (Credit) Resets	When an FC port is unable to send frames for an even longer duration because of continuous credit unavailability, it attempts credit loss recovery. Data traffic cannot flow on a link during this recovery, which typical takes 1-2 seconds to complete, if successful.	Severe Congestion. Also known as Level 3 Congestion.

Moderate congestion includes all symptoms of mild congestion. Likewise, severe congestion includes all symptoms of moderate and mild congestion. In short:

1. **Mild Congestion (Level 1)**: Increased Latency but no frame drops and no link (credit) resets.
2. **Moderate Congestion (Level 2)**: Increased Latency and Frame drops but no link (credit) resets.
3. **Severe Congestion (Level-3)**: Increased Latency, Frame drops, and Link (credit) resets

Each of these severities may have its sub-categories (Minor or Major). For example, during mild congestion, when latency increases, smaller latency degradation can be sub-categorized as a Minor event (Level-1), whereas excessive latency degradation (but no frame drops) can be sub-categorized as a Major event (Level 1.5).

Congestion severities and sub-categories are not standardized. Their sole purpose is to help a user in developing a workflow for detecting and troubleshooting congestion quickly and accurately. Chapter 4 explains the practical usage of these severity levels as part of a congestion troubleshooting methodology that we have developed over the years while troubleshooting congestion issues in thousands of production environments.

Cause of Congestion

As Chapter 1 and Chapter 2 explain, the following can be the cause of congestion in a Fibre Channel fabric.

1. **Slow-drain device**: A slow-drain device has a slower processing rate as compared to the rate at which the frames are being delivered to it. Hence, it slows down sending the credits (R_RDY) resulting in credit unavailability on the connected switchport.
2. **Over-utilization of a link**: A condition when a port (or link) is already transmitting at its maximum capacity, yet it has more frames than can be transmitted on it.
3. **Bit errors on a link**: Bit errors may interfere with the B2B flow control mechanism resulting in congestion or worsening existing congestion.
4. **Long distance links**: A Fibre Channel link causes backpressure towards the source when it does not have enough credits for the distance, speed, and size of the frames flowing on the link. An FCIP link causes backpressure towards the source if the egress IP packets cannot be sent as fast as the ingress rate on the FC links.

Detecting the cause of congestion is the foundational step for solving it accurately.

Source of Congestion (Culprits)

Like detecting the cause of congestion, detecting the source of congestion is equally important. For example,

- Which device caused slow-drain.
- Which link is highly utilized?
- The cables, transceivers, patch panels, or similar faulty equipment causing bit errors.

- The long-distance FC links or FCIP links.

Spread of Congestion (Victims)

Congestion spreading in lossless networks may victimize many healthy devices and their users may experience significant problems just like the users on a culprit device. Differentiating victims from culprits is important so that steps can be taken to not penalize the victims. For differentiating culprits from victims, the source and cause of congestion must be known. These details are explained in this and the next chapter. Refer to Chapter 4, Case Study 2 where the problem was reported on a host (victim), but the source of congestion was a different host (culprit).

Time of Congestion Events

Detecting the time of congestion helps in correlating it with other events such as a scheduled backup job on a virtualized server, increased activity during peak business hours, or changes to application behavior. Refer to Chapter 4, the section on <> for related details. Congestion events that are date and time-stamped are extremely important for effective troubleshooting.

How to Detect Congestion

The following are high-level approaches for detecting congestion.

Reactive Approaches

Reactive approaches include troubleshooting congestion issues after they happen, typically while working with the support teams or doing detailed investigations of network design and traffic flow.

Proactive Approaches

Proactive approaches detect congestion in real time. They result in alerting the admins before the applications complain about performance degradation and allow taking automated preventive actions, such as disconnecting a culprit device or enabling Dynamic Ingress Rate Limiting (DIRL). Refer to Chapter 6 for more details on congestion prevention.

Predictive Approaches

These approaches help in predicting the source and cause of congestion before it victimizes the other devices in the network. These predictive approaches can be as simple as analyzing the throughput requirements of the applications on a host and accordingly allocating the network capacity, or these can be more advanced mechanisms like using machine learning to analyze traffic patterns for forecasting congestion. Storage I/O performance monitoring also helps in predicting congestion. For more details, refer to Chapter 5, especially the section on *Case Study 1 — A Trading Company Predicted Congestion Issues using SAN Analytics*.

Reactive, Proactive, Predictive, or All?

Reactive, Proactive, and Predictive approaches solve different use cases, hence one does not eliminate the need for another. The best solution is to use them together. For example, troubleshooting congestion is mostly a reactive exercise. Likewise, analyzing the traffic pattern over the last weeks or months is also reactive. But this analysis helps in finding baselines, which can be used as thresholds for generating automatic alerts to make congestion detection proactive. This history of traffic patterns can also be fed to machine learning algorithms for predicting future congestion events or to know when a network path will need more capacity.

Another key aspect is to understand that the reactive approaches make the foundation of proactive and predictive approaches. In other words, the priority should be to have a robust reactive workflow in place, then proactive approaches, and only then deploy the predictive approaches. For example, a robust troubleshooting methodology should be ready so that when congestion alerts are received, the next step would be to use that methodology for a swift and accurate investigation. But reversing this sequence, as in, receiving alerts without being aware of the next troubleshooting steps, does not lead to a timely and accurate resolution of the problem. So avoid running into the situation—“I received the alert, but did not know what to do with it”—by following the correct sequence, which is reactive, followed by proactive, and finally predictive.

Where to Detect Congestion

Congestion can be detected natively on network devices and by remote monitoring platforms.

Detecting Congestion on Network Devices

Because congestion is a data-plane phenomenon, it is best detected by the network switches and the end devices. Not just the collection of the metrics and logs by these devices, but also making them available in a simple format is equally important. Also, automated actions, such as alerting or even congestion prevention mechanisms can be triggered at a low granularity when congestion is detected natively on the devices.

Detecting Congestion on Remote Monitoring Platforms

Remote monitoring platforms (such as Cisco Nexus Dashboard Fabric Controller (NDFC) and the MDS Traffic Monitoring (MTM) App explained later) can monitor thousands of devices simultaneously to provide single-pane-of-glass visibility. Such platforms can continuously poll the metrics from the network ports and present them in intuitive graphs that help in finding trends and seasonality. Some advanced platforms can show topologies, correlate the metrics, generate alerts, and even use machine learning for predicting congestion events.

Remote monitoring platforms gather metrics and logs from the network devices and enhance them further. But they are generally not the source of the symptoms of congestion. These symptoms must be detected by the network devices first.

Additionally, remote monitoring platforms do not eliminate the need for monitoring congestion natively on the network devices and vice versa. Both these options should be used together as per the use cases. For example, Cisco MDS switches show real-time information, but the command-line interface (CLI) is not as intuitive. In contrast, visualizing the same information on Cisco NDFC is slightly delayed, but the graphical user interface (GUI) significantly increases data interpretation using various forms of charts. Another example is that the CLI provides access to a single switch at a time. In contrast, NDFC provides a fabric-wide or even a global view.

Because of these reasons, we recommend using both—detecting congestion natively on the network devices and using remote monitoring platforms—in congestion detection workflow.

Congestion Direction — Ingress or Egress

Congestion is directional. That is, each link is comprised of two unidirectional links bundled together. Congestion usually occurs in one direction, while the reverse direction may be uncongested. Consequently, understanding the direction of the congestion is important. Attempting to follow the congestion in the wrong direction does not lead to finding the source and the cause of it.

For example, Figure 3-1 shows that traffic flows from hosts to targets and from targets to hosts (bi-directional), but only the traffic from targets to hosts is affected by congestion if Host-1 is a slow-drain device. Based on the port location, the congestion can be in the ingress direction or egress direction. But the overall congestion direction is against the traffic flow—from the destination to the source of traffic.

Egress Congestion

Egress congestion indicates congestion toward the neighbor on that port. For example, if the neighbor is a host (initiator), then this initiator is throttling the switchport, thus slowing down the frame transmission from the switch to the initiator.

As Figure 3-1 shows, congestion on an egress switchport results in congestion on the ingress port, which causes egress congestion on the upstream ports along the traffic path towards the source. In other words, egress congestion causes ingress congestion. Ingress congestion does not cause egress congestion. Because of this reason, congestion detection workflow and metrics in the egress direction are more important for identifying the source of congestion.

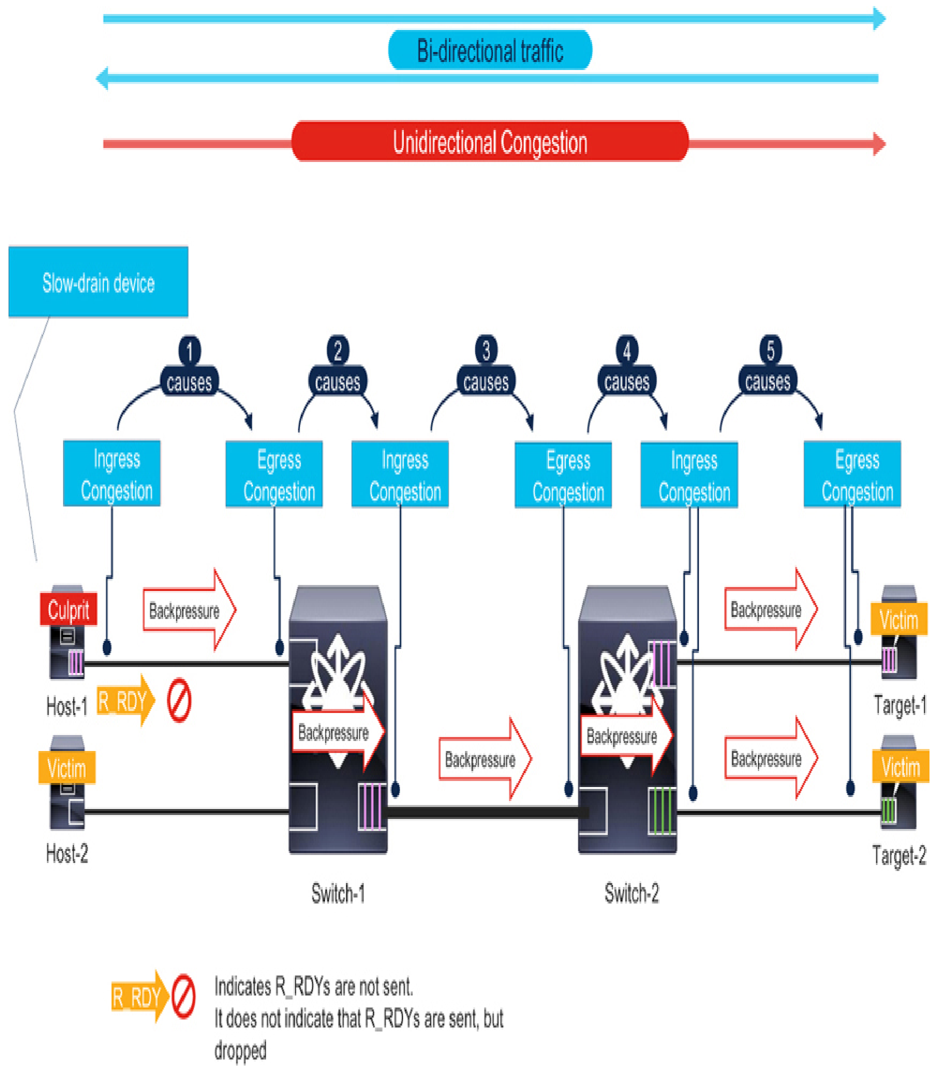


Figure 3-1 *End-to-end traffic and congestion direction at various locations.*

Ingress Congestion

Ingress congestion indicates that the switch port is preventing or slowing the directly connected neighbor from sending frames into the switch. In other words, the switch is throttling the adjacent device. When ingress congestion is seen the problem is not on the port itself or the device connected to the port, but in one or more ports where the received frames are being forwarded to. Ingress congestion typically indicates devices that are negatively affected by congestion (Victims). Ingress congestion can be normal when there are speed mismatches between devices that are in the traffic path. If the ingress switchport has a faster link speed than the egress switchport, then the switch must throttle the ingress traffic to equalize the ingress and egress data rates.

Congestion Detection Metrics

The following is a high-level overview of the metrics that can help in detecting congestion in Fibre Channel fabrics. Later sections in this chapter explain these metrics in detail and then the next chapter explains the usage of these metrics.

- Metrics that detect the (un) availability of B2B credits
  - Duration of credit unavailability — How long a port is at zero remaining-Tx-B2B-credits or zero remaining-Rx-B2B-credits. Different metrics may report this duration at various granularities. For example, on Cisco MDS switches, TxWait reports credit unavailability every 2.5 microseconds, whereas slowport-monitor detects credit unavailability at 1-millisecond granularity.
  - The number of times when credits were unavailable, for example, Tx/Rx B2B transitions to/from zero detects the number of times when a port was at zero remaining-Tx/Rx-B2B-credits. These metrics do not detect the duration at that state.

  - The instantaneous value of available credits, which are the remaining-Tx-B2B-credits and remaining-Rx-B2B-credits at an instance.
- Metrics that detect frame drops, such as the number of frames that are dropped because they could not be sent out of a switch within a timeout duration.
- Metrics that detect credit recovery attempts.
  - Number of credit loss recovery attempts: The number of times when a port invoked Link Reset Protocol because it was at zero remaining-Tx-B2B-credits for an extended duration (typically 1 second for F_Ports and 1.5 seconds for E_Ports on Cisco MDS switches)
  - Number of Link Reset (LR) primitives sent and received: An FC port initiates credit loss recovery by sending a Link Reset (LR) primitive. This counter reports the number of LR sent and received.
  - Number of Link Reset Response (LRR) primitives sent and received: An FC port may respond to a received LR primitive by sending a Link Reset Response (LRR) primitive. This counter reports the number of LRRs sent and received.
- Metrics that detect instances of the B2B state change mechanism occurring. Refer to Chapter 2, section on *Credit Loss Recovery Using B2B State Change Mechanism*, for more details.
  - The number of times BB_SCs credit resend actions detected lost frames.
  - The number of times BB_SCr Tx credit increment actions detected lost R_RDYs.
- Metrics that detect link errors, such as CRC corrupted frames and Invalid Transmission words. Refer to Chapter 2, section on *Counters on Fibre Channel Ports* for more details.
- Metrics that detect link utilization, such as the number and the size of frames on FC ports.

- Metrics that detect the application I/O profile, such as the timing, size, type, and rate of I/O operations that are carried within the frames. Refer to Chapter 5 for more details.

The availability of these metrics is implementation dependent and not all devices may report these metrics.

If an FC port does not report enough metrics, a workaround is to use the metrics from the neighboring port in the reverse direction. For example,

- Ingress utilization of a port is the same as the egress utilization of its neighbor and vice-versa. If the local port does not support automatic alerting, the workaround is to enable automatic alerting on the neighbor.
- In most cases, if the local port does not report the duration of Rx credit unavailability, the same information can be detected using the neighbor's Tx credit unavailability duration. The minor difference between the two neighboring ports is usually negligible.

Such correlations across neighboring devices are better done on a remote monitoring platform with end-to-end visibility.

Congestion Detection Metrics on Cisco MDS Switches

Cisco MDS switches have a large variety of metrics designed to identify the source and the cause of congestion. Some of these metrics are 'real time' or instantaneous displays. However, the most important metrics are logged on the switches with time and date stamps as they occur or periodically (slightly delayed), which allows for investigating the events that occurred in the recent past as well as the time and date correlation between the metrics in different switches in the data path. Additionally, the same set of metrics can trigger automated alerts and send Syslog messages and/or SNMP traps to remote monitoring platforms.

Besides explaining the metrics, this section also covers the output of commands from NX-OS operating system that runs on Cisco MDS switches. Even Cisco Nexus switches run NX-OS, but in this section, NX-OS commands refer to the releases that run only on MDS switches, unless specifically mentioned otherwise.

The initial goals of these congestion metrics are twofold:

1. To determine the device(s) causing the congestion (culprits), for example, Host-1 in Figure 3-1. This is the main goal. This often involves following the egress congestion to the actual source of the congestion. Once the source of the congestion is identified then preventive steps can be implemented.
2. To understand the spread of congestion for finding the affected devices (victims), for example, Host-2 in Figure 3-1. This is a secondary yet important goal because these devices may also report errors. Typically, these are the devices that share the data path between the culprit device and the devices it is in communication with.

Tx Credit Unavailability in Microseconds — TxWait

Direction — Egress Congestion

Severity — Mild Congestion (Sub-Category: Minor to Major)

TxWait increments when a switchport has zero remaining-Tx-B2B-credits for 2.5 µs (microseconds) and it has at least one frame queued for transmission. Consequently, if there are no frames queued for transmission and the port has zero remaining-Tx-B2B credits, then TxWait does not increment.

The raw TxWait value can be converted to seconds by multiplying it by 2.5 and then dividing it by 1,000,000.

TxWait in seconds = (TxWait in 2.5 µs ticks x 2.5) / 1,000,000

TxWait in seconds can then be used to calculate the percentage TxWait during a period.

Percentage TxWait = (TxWait in seconds/ Monitoring duration in seconds) x 100

For example, consider a switchport that shows raw TxWait of 436587435. After 1 second, TxWait changes to 436707435. In other words, TxWait incremented by 120,000 in the last one second.

TxWait in seconds = (120,000 x 2.5) / 1,000,000 = 0.3 seconds.

Percentage TxWait = (0.3/1) x 100 = 30%

In simple terms, this switchport was unable to transmit frames for 300 ms in a 1-second monitoring duration due to zero remaining-Tx-B2B-credits. Essentially, the port has 30% congestion.

For most practical purposes, using percentage TxWait instead of the raw TxWait counter is preferred because it is more meaningful, simpler, and it is used for automated alerting via Port-Monitor (explained later in the section — *Port-Monitor on Cisco MDS Switches*). Cisco MDS switches provide ready-made percentage TxWait for all switchports. If the raw TxWait is exported from the switch to a remote monitoring platform, use this explanation for converting to percentage TxWait.

TxWait can be exported to remote monitoring platforms via SNMP, NX-API, and streaming telemetry. TxWait is available as fcIfTxWaitCount SNMP variable.

TxWait is one of the most important metrics for detecting SAN congestion because of the following reasons.

- TxWait not only reports the unavailability of the credits, but it also reports the duration for which Tx B2B credits were not available. In other words, TxWait helps in detecting congestion and the severity of congestion because the longer the TxWait, the more severe the congestion.
- The underlying granularity of TxWait is in nanoseconds. It is reported in 2.5 µs ticks. This means TxWait also captures the durations that are smaller than 2.5 µs. This is useful for detecting micro congestion events.

- TxWait is natively implemented in the Fibre Channel port-ASICs used in Cisco MDS Switches. Hence, it does not cause additional load on the control-plane CPU of the switch.
- TxWait is enabled by default. It is always operational for all FC ports on Cisco MDS Switches.
- On FC ports that are in ER_RDY mode with multiple Virtual Links (VLs), TxWait is recorded per VL. Refer to Chapter 6 for more details on VLs.

Cisco MDS switches make TxWait easy to interpret by enriching the raw metric and presenting it in simpler formats. For example:

- Absolute TxWait count since the counters were last cleared.
- Percentage TxWait over the last 1 second, 1 minute, 1 hour, and 72 hours.
- TxWait history graph for the last 60 seconds, 1 hour, and 72 hours.
- TxWait delta and congestion percentage every 20 seconds with date and time stamps.

Raw TxWait

Use the NX-OS commands **show interface, show interface counters**, or **show interface counters detailed** to display the raw TxWait metric. Refer to Example 3-1. The “TxWait 2.5us due to lack of transmit credits” counter contains the total amount of TxWait on the interface since the counters were last cleared. As mentioned, this value is in 2.5 µs increments so a value of 4 equals 10 µs. In Example 3-1, interface fc2/15 was unable to transmit frames for 15.63 seconds ((6252650 x 2.5) / 1,000,000) since the counters were last cleared.

A common practice is to clear the counters (using the NX-OS command **clear counters**) followed by displaying the TxWait value multiple times. But this approach is not intuitive because a user must manually take the time difference, then multiply the raw TxWait value by 2.5 and divide by 1,000,000.

Using the raw TxWait counter directly has limited usefulness since it contains just the raw total (cumulative) value. However, the raw value is the basis for all other forms. It can be made much more intuitive and meaningful by converting to other forms, as explained next.

Percentage TxWait

Example 3-1 also shows, the “TxWait for the last 1 second, 1 minute, 1 hour, and 72 hours” values. These are the percentage TxWait in the four specific time intervals indicated and they are calculated as explained earlier in this section. This is useful for a quick check of interface-level (switchport) transmit congestion.

Example 3-1 *Raw and Percentage TxWait in show interface counters detailed*

TxWait History Graphs

Cisco MDS switches also display TxWait in graphical format. This has a distinctive advantage over raw numbers due to better visualization for finding spikes and dips over time.

The TxWait-History graph shows three ASCII graphs for TxWait.

- Last one minute — Each column shows the cumulative TxWait in each second (Example 3-2)
- Last one hour — Each column shows the cumulative TxWait in each minute (Example 3-3)
- Last 72 hours — Each column shows the cumulative TxWait in each hour (Example 3-4)

For each of these graphs, the most current time is the leftmost column, which is the time when the command was issued. The numerical value of the column's TxWait is contained in the top rows. It should be read vertically from top to bottom. The middle 10 rows show a plot of the TxWait value for the column using the pound/hash symbol (#). The bottom two rows show a timeline as an x-axis. Example 3-2 shows that 20 seconds prior to this command being executed, TxWait on fc9/17 was 527 ms in 1 second monitoring period.

This command can be useful for real-time congestion troubleshooting on a switch.

Example 3-2 *show interface txwait-history — 60-second graph*

Every 60 seconds the amount of TxWait is summed and rolled into the 60-minute graph where each column now shows TxWait accumulated in 1 minute (Example 3-3). The amount of TxWait in that 1 minute is contained in the top 4 rows in an ss.t (seconds and tenths of a second) vertical format. Example 3-3 shows that 45 minutes prior to the command being executed, TxWait on this interface was 54.8 seconds in a 1-minute monitoring period.

Example 3-3 *show interface txwait-history – 60-minute graph*

Every 60 minutes the amount of TxWait is summed and rolled into the 72-hour graph. Each column now shows 1-hour accumulated TxWait (Example 3-4). The top 4 rows contain the amount of TxWait that accumulated in that 1-hour interval. Example 3-4 shows a TxWait spike 24 hours earlier that lasted for approximately 3 hours with a peak TxWait of 957 seconds in a 1-hour monitoring period.

Example 3-4 *show interface txwait-history – 72-hour graph*

TxWait History in OBFL

Cisco MDS switches log TxWait history in the Onboard Failure Logging (OBFL) buffer when TxWait increases by 100 ms or more in a 20-second interval. Refer to Example 3-5, which shows the following:

- The interface and virtual link, if enabled.
- The raw TxWait increase (the delta TxWait) in the 20-second sampling period.
- Raw TxWait converted to seconds.
- Percentage Congestion (Percentage TxWait).
- Date and time.

The last row of Example 3-5 shows that fc9/17 was unable to transmit frames due to lack of Tx B2B credits for 14 seconds in the 20-second monitoring period ending on Apr 21, 2022, at 13:09:27. This is a congestion percentage of 72%. Note that the delta TxWait (5786677) in this 20-second interval is in 2.5 µs ticks. This is actually 14.46 seconds, which is used to calculate the congestion percentage.

Example 3-5 *TxWait history in show logging onboard txwait*

Collection of TxWait history in OBFL is enabled by default.

OBFL is a persistent memory on Cisco MDS switches. Information stored in OBFL is available even after rebooting a switch. The space on OBFL is limited. Older logs are overwritten to make space for newer logs.

TxWait history in OBFL has become a powerful tool for detailed investigations because it shows when the problem happened, where the problem happened, and how severe it was. It also helps in optimizing the thresholds for automatic alerting because it captures the events that might have stayed below the configured thresholds for automatic alerting and might have gone unnoticed. Another merit of OBFL is that it provides a view into the past, meaning that it makes possible to investigate congestion events that happened in the past.

Rx Credit Unavailability in Microseconds — RxWait

Direction — Ingress Congestion

Severity — Mild Congestion (Sub-Category: Minor to Major)

RxWait increments when a switchport has zero remaining-Rx-B2B-credits for 2.5 µs (microseconds). When RxWait increments, it is an indication of ingress congestion, where the switchport is preventing the adjacent device from transmitting to it. This normally indicates that the adjacent device is affected by congestion, not causing the congestion. In other words, the adjacent device is a victim, not a culprit.

RxWait is similar to TxWait in the reverse direction. The calculation of converting the raw RxWait to seconds and percentage values is exactly the same as TxWait. Like TxWait, RxWait is also implemented in the Fibre Channel port-ASICs used in Cisco MDS Switches. Hence, it does not cause additional load on the control-plane CPU of the switch.

RxWait is measured by default on 64GFC ports on Cisco MDS switches, and it is available in NX-OS 9.3(2) onwards. The port does not need to be operating at 64GFC speed for RxWait to function. No configuration is required to use it. RxWait is not available on 32GFC and slower ports on MDS switches.

Refer to Example 3-6 for raw RxWait and percentage RxWait.

Example 3-6 *Raw and percentage RxWait*

Refer to Example 3-7 for RxWait History in OBFL. Read this output similar to the TxWait history as explained earlier in *TxWait History in OBFL* section.

Example 3-7 *RxWait history in show logging onboard rxwait*

Continuous Tx Credit Unavailability in Milliseconds — Slowport-monitor

Direction — Egress Congestion

Severity — Mild congestion (Sub-Category: Minor to Major)

Slowport-monitor is a feature of Cisco MDS switches that measures the continuous duration when a switchport has zero remaining-Tx-B2B-credits. The following are more details about Slowport-monitor:

- Slowport-monitor reports only the continuous duration when the remaining-Tx-B2B-credits on a switchport were zero.
- Slowport-monitor is natively implemented in the Fibre Channel port ASICs in Cisco MDS Switches. Thus, enabling Slowport-monitor does not cause additional load on the control-plane CPU.
- Slowport-monitor reports a minimum value of 1 ms with a granularity of 1 ms increments.
- Slowport-monitor is not enabled by default on Cisco MDS Switches at the time of this writing. Refer to the later section on *When to Enable Slowport-monitor?* for further details.

As Example 3-8 shows, slowport-monitor can be enabled by the NX-OS config command **system timeout slowport-monitor**.

Example 3-8 *Enabling slowport-monitor on Cisco MDS Switches*

The configured timeout value of slowport-monitor is called ‘admin delay’ and the actual duration of zero remaining-Tx-B2B-credits that slowport-monitor detects is called ‘oper delay’ (operational delay).

The slowport-monitor admin delay can be configured from 1 to 500 ms with 1 ms increments and separate admin delay values can be configured differently for logical type core and edge ports.

Note:

The slowport-monitor admin delay must be lower than the no-credit-drop timeout. Refer to Chapter 6, section on *Details on no-credit-drop timeout* for details.

MDS switches capture ‘oper delay’ every 100ms and only if it is longer or the same as the configured ‘admin delay’. If configured with the default option, the default admin delay is 50 ms. So, with the default configuration (if you don’t specify the admin delay value) slowport-monitor does not capture an event when a port has zero remaining-Tx-B2B-credits for 49 continuous ms or less. Because of this reason, when enabling slowport-monitor, consider using a smaller non-default admin delay, such as 10 ms. However, lower admin delay may result in many events being generated and will cause older events to be overwritten sooner. If this situation happens, increase the admin delay value to reduce the rate of events to be able to determine the devices that are withholding credits for a long duration. Also, note that since the events are captured every 100ms, it is possible that more than one slowport event occurred during that period. When there are more than one events, the ‘Oper Delay’ value is the average across all events.

Normally, TxWait is sufficient for even low-level congestion troubleshooting since OBFL TxWait will record entries when there is a total of just 100ms of total TxWait in a 20-second interval. However, if that is insufficient for troubleshooting the congestion problem at hand, then slowport-monitor should be enabled to detect even minor congestion. Refer to the section on *When to Enable Slowport-monitor?* for more details.

Slowport-monitor Events in Real Time

Use NX-OS command **show process creditmon slowport-monitor-events** to display the last 10 events per port. Although the events are captured by the port ASIC at 1 ms granularity, a window of 100 ms is used to display the output. Example 3-9 shows:

- Admin delay of 1 ms configured using the NX-OS command **system timeout slowport-monitor**.
- Slowport detection count is the number of times when remaining-Tx-B2B-credits were zero for a duration longer than the configured admin delay value. It is an absolute counter, not a delta counter. Increments in the last 100 ms can be obtained by subtracting the value displayed in the row below.
- The oper delay is the actual duration for which remaining-Tx-B2B-credits were zero at the displayed time stamp. If there was more than one event in the 100 ms period, then it is the average of all events.

For example, on 04/27/22 at 12:12:08.238, Tx B2B credits were unavailable continuously for 24 ms. This event occurred two times (785 –783). So for 48 ms (24 ms x 2) in the 100 ms interval, fc1/13 had zero remaining-Tx-B2B-credits.

Example 3-9 *Slowport-monitor events in show process creditmon slowport-monitor-events*

Slowport-monitor History in OBFL

Slowport-monitor history is captured in OBFL and can be displayed using NX-OS command **show logging onboard slowport-monitor-events** (Example 3-10). Interpret this output the same way as the **show process creditmon slowport-monitor-events** command, explained in Example 3-9.

Example 3-10 *Slowport-monitor history in show logging onboard slowport-monitor-events*

Continuous Tx Credit Unavailability for 100 ms — Tx-credit-not-available

Direction — Egress Congestion

Severity — Mild Congestion (Sub-Category: Major)

This counter increments when a switchport has zero remaining-Tx-B2B-credits for a continuous 100 ms. This can augment the TxWait counter to indicate larger 100ms blocks of time where there are zero Tx credits.

A software process called creditmon polls all the switchports every 100 ms. If creditmon detects that a switchport has zero remaining-Tx-B2B-credits continuously between two consecutive cycles, it increments this counter by one. In later MDS switches, this counter is calculated in hardware but the functionality remains the same.

In addition to generating log entries, this counter can also generate alerts via Port-Monitor.

Tx-credit-not-available in Real Time

Use the NX-OS command **show system internal snmp credit-not-available** to display the percentage of a 1-second interval when the remaining-Tx-B2B-credits were zero continuously. Refer to Example 3-11.

- On Tuesday, May 19, 2015, at 16:13:19, fc1/1 had zero remaining-Tx-B2B-credits continuously for 100 ms (which is 10 percent of 1 second).
- On Tuesday, May 19, 2015, at 16:14:06, fc1/1 had zero remaining-Tx-B2B-credits continuously for 100ms twice in the last 1-second.

- On Tuesday, May 19, 2015, at 16:15:12, fc1/1 had zero remaining-Tx-B2B-credits continuously for 200ms in the last 1-second.

Example 3-11 *show system internal snmp credit-not-available*

The interval(s) column displays the number of times the threshold of 10 percent or 0 percent crossed during the 1-second interval. Refer to Figure 3-2, which shows remaining-Tx-B2B-credit on a switchport in the last 1-second. The creditmon process polls the switchport every 100ms starting at 0ms. At the beginning of the second, the remaining-Tx-B2B-credits fall to zero. At the 150th millisecond, R_RDY is returned, resulting in nonzero remaining-Tx-B2B-credits. At the 500th millisecond, the remaining-Tx-B2B-credits again fall to zero. At the 600th millisecond, R_RDY is returned, resulting in nonzero remaining-Tx-B2B-credits. This sequence of events would result in two intervals of 10% duration under the NX-OS command **show system internal snmp credit-not-available**, which is similar to the output of Tuesday, May 19, 2015, at 16:14:06.

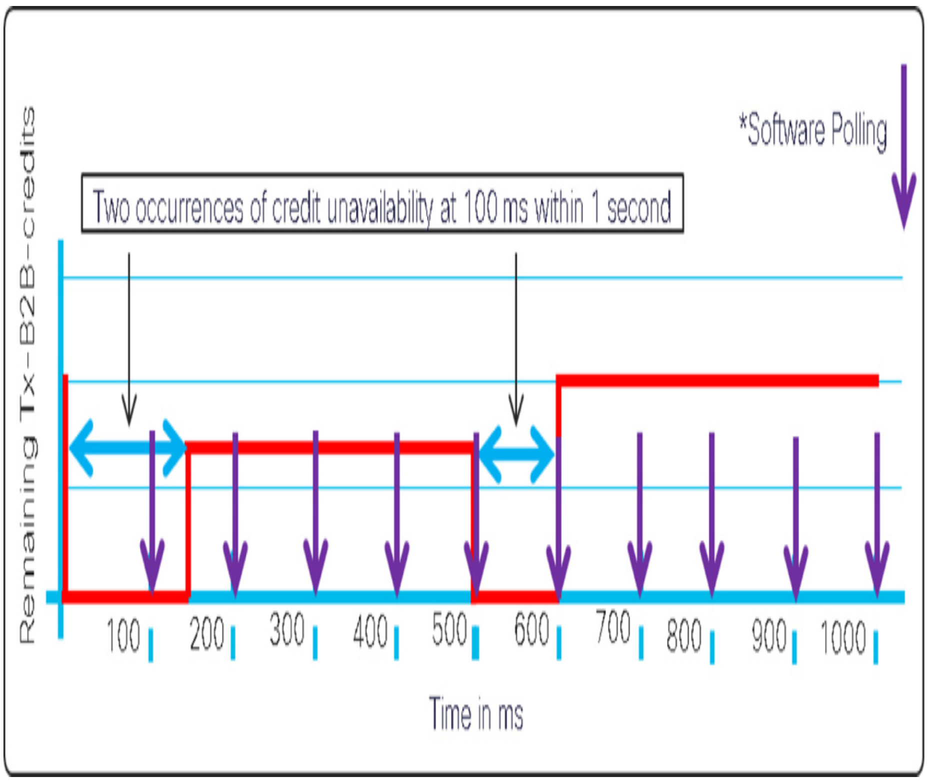


Figure 3-2 *Detecting Tx-credit-not-available at 100ms using creditmon process*

Tx-credit-not-available History in OBFL

Tx-credit-not-available is also logged in OBFL via the TX_WT_AVG_B2B_ZERO counter. The exact name of this counter is different for various hardware generations. Further details are explained in Chapter 4. For all of them, the most important detail to remember is that this counter shows the count of occurrences of 100ms continuous zero remaining-Tx-B2B-credits.

Refer to Example 3-12. Watch for CNTR_TX_WT_AVG_B2B_ZERO. The count column shows the absolute number of times this counter incremented. FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO incremented from 407 to 420 in the 20-second duration from 12:54:35 to 12:54:55 on 04/27/22. In simple terms, port fc1/13 had 13 instances when it had zero remaining-Tx-B2B-credits continuously for 100 ms in the 20-second interval ending at the timestamp.

Example 3-12 *Tx-credit-not-available for 100ms in show logging onboard error-stats*

Differences Between TxWait, Slowport-monitor, and Tx-credit-not-available

TxWait, slowport-monitor, and Tx-credit-not-available seem similar because they indicate egress congestion by measuring the time at zero remaining-Tx-B2B-credits. However, they each have slightly different meanings with different granularities. TxWait increments every 2.5 µs when at zero remaining-Tx-B2B-credits (not configurable). Slowport-monitor indicates a continuous time at zero remaining-Tx-B2B-credits starting at 1 ms up to 500ms (configurable). Tx-credit-not-available shows 100 ms of continuous zero remaining-Tx-B2B-credits duration (not configurable).

Consider Figure 3-3, which shows the remaining-Tx-B2B-credits on a switchport over the last 1-second. At ~250ms, the remaining-Tx-B2B-credits fall to zero and recover to a non-zero value at 410 ms. The reported value of these counters is as follows:

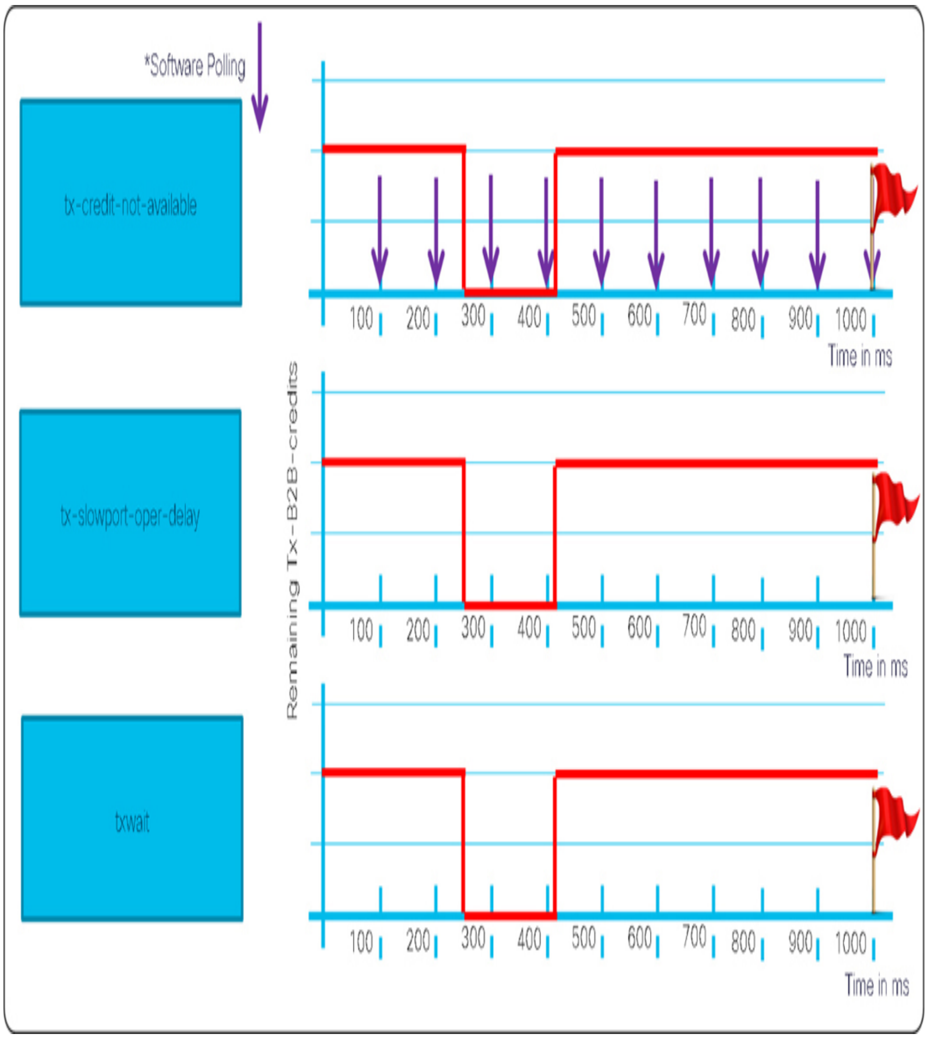


Figure 3-3 *TxWait, slowport-monitor, and tx-credit-not-available at 160ms of continuous credit unavailability*

- Tx-credit-not-available: 100 ms because of the use of the creditmon process that polls every 100 ms and reports only when the port was at zero remaining-Tx-B2B-credits between two subsequent polls — continuous.
- Slowport-monitor oper delay: If slow-port-monitor was configured to a value less than or equal to 160ms then it would show 160ms — continuous. If was configured to a value higher, say, 200ms then it would not show any occurrence.
- TxWait: 160ms — not necessarily continuous.

If automatic alerting is enabled at a threshold of 100 ms, all three counters detect this condition.

Next, consider Figure 3-4. This time the remaining-Tx-B2B-credits fall to zero at 250 ms and recover to a non-zero value at 390ms. The overall duration is still more than 100ms but tx-credit-not-available does not flag this event because the software polling takes place every 100 ms and there were no two consecutive polls when the remaining-Tx-B2B-credits were zero. In contrast, the hardware-based implementation of tx-slowport-oper-delay and txwait detect this condition. The reported value of these counters is as follows:

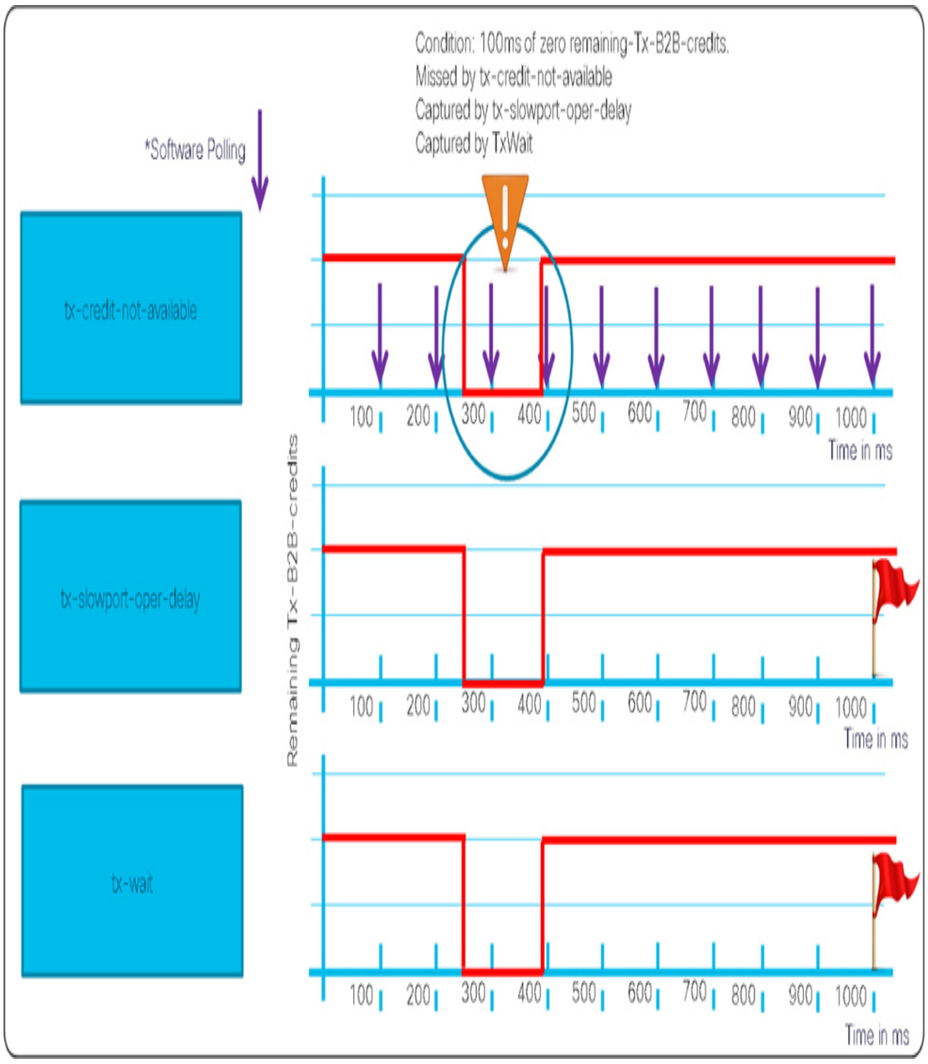


Figure 3-4 *TxWait, slowport-monitor, and tx-credit-not-available at 140ms of continuous credit unavailability*

- Tx-credit-not-available: None
- Slowport-monitor oper delay: 140ms — continuous
- TxWait: 140ms — not necessarily continuous.

If automatic alerting is enabled at a threshold of 100 ms, tx-credit-not-available does not detect, whereas slowport-monitor and TxWait detect this condition.

Further, consider Figure 3-5. Remaining-Tx-B2B-credits fall to zero multiple times in a poll-interval of 1 second. None of these durations is longer than 100ms on its own but the sum of these durations is longer than 100 ms.

If automated alerting is enabled at a threshold of 100 ms, TxWait is the only counter that detects this condition.

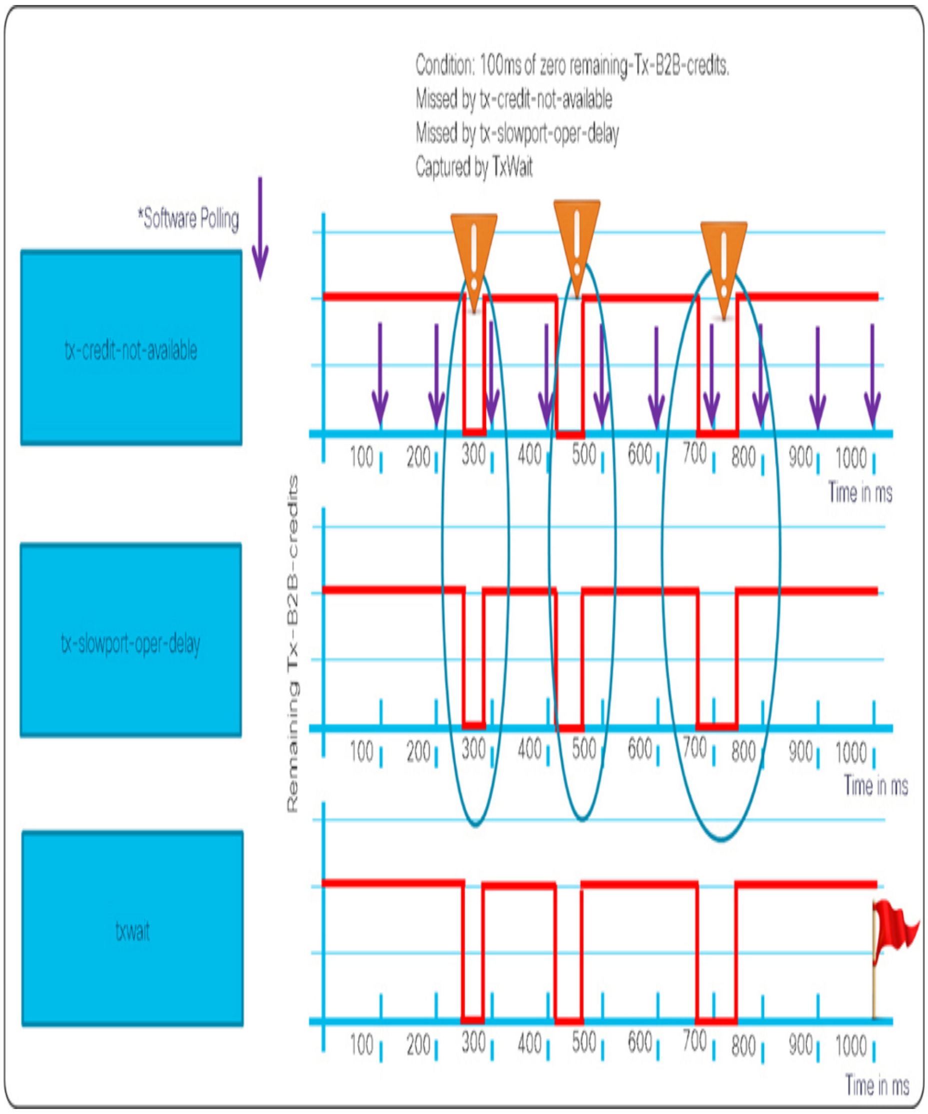


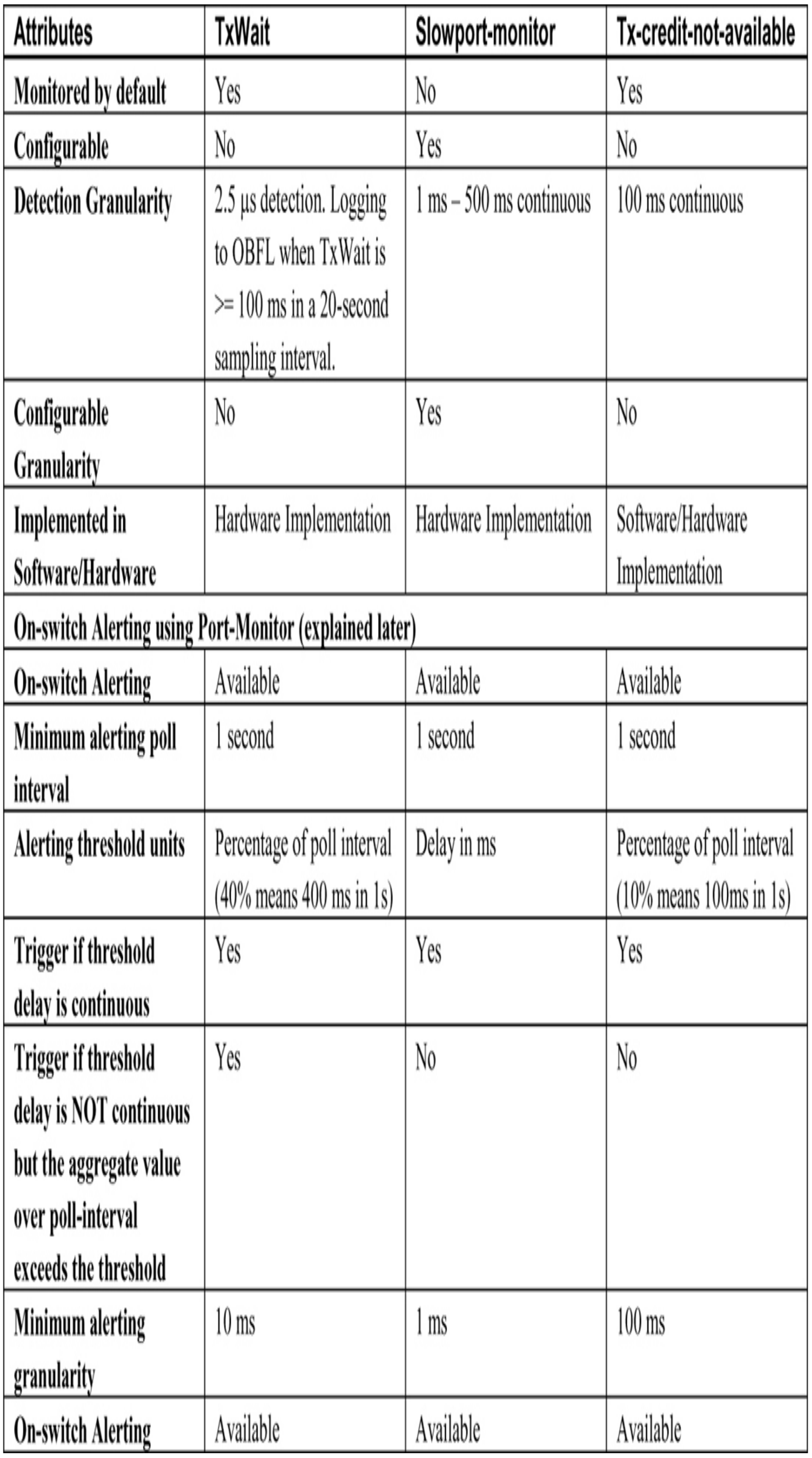
Figure 3-5 *TxWait, slowport-monitor, and tx-credit-not-available at multiple smaller durations of credit unavailability*

Note the following points:

- TxWait, Slowport-monitor, and Tx-credit-not-available complement each other.
- The low granularity of TxWait increases the accuracy and catches even micro-congestion events. Slowport-monitor and Tx-credit-not-available complement by detecting continuous duration of zero remaining-Tx-B2B-credits. For example, TxWait of 50 ms in 1 second is more severe if slowport-monitor also reports oper delay of 50 ms. Likewise, TxWait of 100 ms in 1 second is even more severe if Tx-credit-not-available also shows this event.
- Essentially, a port unable to transmit continuously for a longer duration is a more severe event than a port unable to transmit many times for small durations, although the collective duration of non-transmission may be the same.
- If TxWait increments by 300ms in a 20-second interval and if tx-credit-not-available increments by three in the same 20-second interval, it indicates that there were three instances, not necessarily contiguous, of 100ms credit unavailability that made up the 300 ms.
- The minimum granularity of slowport-monitor is 1 ms. It does not capture any shorter durations of zero remaining-Tx-B2B-credits, which TxWait can capture.

Table 3-2 compares these three counters side-by-side.

Table 3-2 *Comparison of TxWait, slowport-monitor, and tx-credit-not-available on Cisco MDS switches*



Attributes	TxWait	Slowport-monitor	Tx-credit-not-available
Monitored by default	Yes	No	Yes
Configurable	No	Yes	No
Detection Granularity	2.5 µs detection. Logging to OBFL when TxWait is >= 100 ms in a 20-second sampling interval.	1 ms – 500 ms continuous	100 ms continuous
Configurable Granularity	No	Yes	No
Implemented in Software/Hardware	Hardware Implementation	Hardware Implementation	Software/Hardware Implementation
On-switch Alerting using Port-Monitor (explained later)			
On-switch Alerting	Available	Available	Available
Minimum alerting poll interval	1 second	1 second	1 second
Alerting threshold units	Percentage of poll interval (40% means 400 ms in 1s)	Delay in ms	Percentage of poll interval (10% means 100ms in 1s)
Trigger if threshold delay is continuous	Yes	Yes	Yes
Trigger if threshold delay is NOT continuous but the aggregate value over poll-interval exceeds the threshold	Yes	No	No
Minimum alerting granularity	10 ms	1 ms	100 ms
On-switch Alerting	Available	Available	Available

When to Enable Slowport-monitor?

Slowport-monitor is not enabled by default. This section explains when to enable it, and at what admin delay.

First, let's analyze the other two metrics that detect the duration of Tx B2B credit unavailability and are enabled by default—TxWait and Tx-Credit-Not-available.

By default, the OBFL on Cisco MDS switches logs if TxWait is equal to or longer than 100 ms in a 20-second sampling interval. As mentioned previously, this is the cumulative duration of zero remaining-Tx-B2B-credits in 20-second duration.

The next granularity of capturing zero remaining-Tx-B2B-credits is 100 ms, which is captured by Tx-credit-not-available.

These two default granularities—100 ms of non-continuous TxWait in OBFL and 100 ms of continuous Tx-credit-not-available—do not capture the continuous events that fall in between. For example, by default, MDS switches do not log in OBFL if a switchport is at zero remaining-Tx-B2B-credits continuously for 10 ms in 20-seconds. Note that the raw TxWait and percentage TxWait counters increment, but they are not logged in the on-switch OBFL because the cumulative value is less than 100 ms. An external monitoring platform, which polls the metrics continuously, can still maintain this history but the MDS switches will not record any history. The 100 ms minimum threshold was chosen so minor amounts of TxWait do not fill up the OBFL log and cause more significant (longer duration) TxWait entries to be overwritten.

In such cases, slowport-monitor can capture the continuous durations of zero remaining-Tx-B2B-credits. This means that in most cases enable slowport-monitor only when TxWait and Tx-credit-not-available are unavailable in switch OBFL but you still suspect zero remaining-Tx-B2B-credits durations.

Overall, using TxWait together with 100 ms Tx-Credit-Not-Available is sufficient for most general congestion troubleshooting. TxWait entries are recorded in OBFL when there is just 100 ms of aggregate TxWait in a 20 second interval. That works out to be 0.5% congestion. Lower level of congestion is generally non-impactful and not even noticed. If the need arises to troubleshoot very low latency problems, then enable slowport-monitor with a low admin delay value such as 1ms or 10ms.

That answers if and when you should enable slowport-monitor.

The next consideration is to find an appropriate slowport-monitor admin delay value.

While enabling slowport-monitor, be aware that it captures all durations of zero remaining-Tx-B2B-credits (oper delay) that are longer than the configured admin delay. If admin delay is 1 ms, it may result in too many oper delay events resulting in flooding the OBFL and overwriting other important logs. Some users may argue that they need not log all the events when switchports are at zero remaining-Tx-B2B-credits occasionally for a few milliseconds. Because of these reasons, consider using a longer admin delay, such as 10 ms, when enabling slowport-monitor the first time. If it captures too many events, then increase the admin delay and solve the root cause. On the other hand, if it does not capture many events, lower the admin delay to capture more granular oper delay values.

The difference between TxWait, Tx-credit-not-available, and slowport-monitor explained in this section applies only to logging in the on-switch OBFL on MDS switches. A remote monitoring platform (such as MTM), that continuously monitors TxWait and Tx-credit-not-available can maintain the history of zero remaining-Tx-B2B-credits. TxWait can capture zero remaining-Tx-B2B-credits durations shorter than 100ms. The only added value of enabling slowport-monitor with a remote monitoring platform is to know if those durations are continuous or not.

Continuous Rx Credit Unavailability for 100 ms - Rx-credit-not-available

Direction — Ingress Congestion

Severity — Mild Congestion (Sub-Category: Major)

This counter increments when a switchport has zero remaining-Rx-B2B-credits for a continuous 100ms, which means that the interface is not returning credits to the adjacent device because of ingress congestion. This can be useful for determining what interfaces are affected by congestion, but it does not indicate which interfaces are causing congestion. In other words, it helps in finding the victims, not the culprits.

As the name suggests, Rx-credit-not-available is similar to Tx-credit-not-available in the reverse direction.

Cisco MDS switches do not alert for Rx-credit-not-available, but it is logged in OBFL error-stats as shown in Example 3-13. Interpret the output of this command similar to the Tx-credit-not-available, as explained earlier in Example 3-12.

Example 3-13 *Rx-credit-not-available for 100ms in show logging onboard error-stats*

Timeout-discards or Timeout-drops

Direction — Egress Congestion

Severity — Moderate

Under congestion, the time spent by a frame within a switch is much longer than the typical port-to-port switching latency. Under extreme congestion, a culprit device may not be ready to receive frames for an extended duration. Instead of allowing the frames to remain within a switch forever, the frames are dropped after a timeout duration and these frames are counted as timeout drops. Dropping the frames free up the buffers, allowing recovery from congestion. More details on this are explained in Chapter 6, the section on *Congestion Recovery by Dropping Frames*.

On Cisco MDS switches, dropping the fames based on their age in the switch is enabled by default at 500ms. This feature is called congestion-drop timeout. Fibre Channel standard calls it Frame Discard Timeout Value (F_D_TOV). Each frame received by the switch is time stamped. Congestion on the egress port to which the received frame is to be forwarded delays the received frame from being transmitted. When the frame eventually reaches the egress port in preparation for transmission the time stamp is compared against the current time. If the difference is greater than the congestion-drop timeout, then the frame is dropped as a timeout drop (or timeout discard).

Another feature on Cisco MDS switches, called no-credit-drop timeout drops the frames based on the continuous duration of zero remaining-Tx-B2B-credits on an edge port. But no-credit-drop timeout is not enabled by default. The design and benefits of this feature are explained in Chapter 6, the section on *Dropping the Frames based on Slow-Drain on an Edge Port*. Frames that are dropped because of no-credit-drop timeout, if enabled, are also counted as timeout drops (or timeout discards).

Note that the terms drop and discard are synonyms and consequently mean the same thing. Various places display 'timeout-drop' and other places show 'timeout-discard'. They are identical in their meaning.

Timeout-drops/discards are recorded via several commands on Cisco MDS switches:

1. **Show interface, show interface counters**, and **show interface counters detailed** (Example 3-14): Shows accumulated value since the counters were cleared last.
2. **Show logging onboard error-stats** (Example 3-15): Shows historic values with time and date stamps.

Example 3-14 *FC Timeout-discards in show interface counters detail*

Example 3-15 *show logging onboard error-stats showing timeout-drops*

Starting with 32GFC MDS switches, timeout-drops are also captured with flow-level details such as VSAN, Source FCID, Destination FCID, and the delta drops in that period. This is available in the NX-OS command **show logging onboard flow-congestion-drops** (Example 3-16), which is a useful command because it gives additional information as to the destination FCIDs affected when there are multiple FLOGIs on an interface. These are not additional drops than what is included in the NX-OS commands **show interface counters** and **show logging onboard error-stats** but just a bit more information as to which specific FCIDs were affected.

When an interface has multiple destination FCIDs on it like an E_Port or an F_Port to an NPV device just because a destination FCID shows up in this command output doesn't necessarily indicate it is a culprit device. When congestion occurs, other devices may be the victim devices and their frames may also be dropped.

Example 3-16 *show logging onboard flow-congestion-drops*

For remote monitoring platforms, Timeout-drops/discards is available via SNMP OID fcIfTimeOutDiscards.

Tx Credit Loss Recovery

Direction — Egress Congestion

Severity — Severe

Credit Loss Recovery is a mechanism to re-establish the agreed to B2B credits on a port after it is at zero remaining-Tx-B2B-credits for an extended duration, typically 1 second on F_Port and NP_Port and 1.5 seconds on the E_Port on Cisco MDS switches. As Chapter 2, the section on *Credit Loss Recovery Using Link Reset Protocol* explains, an FC port may reach a state of credit loss (zero remaining-Tx-B2B-credit for an extended duration) because of the bit errors or severe congestion.

Regardless of the actual cause, once the interface is at zero remaining-Tx-B2B-credit for a continuous 1 or 1.5 seconds, it invokes the Link Reset Protocol by sending a Link Reset (LR) primitive.

Each time an FC port attempts credit loss recovery, it is counted and logged and may be alerted via Port-Monitor on Cisco MDS switches.

Tx Credit Loss Recovery is recorded in several places on Cisco MDS switches:

1. NX-OS command **show interface counters detailed** (Example 3-17) command shows cumulative count.
2. NX-OS command **show process creditmon credit-loss-events** (Example 3-18) command shows the last 10 events

3. NX-OS command **show logging onboard error-stats** (Example 3-19) command shows the event history.

Example 3-17 *show interface counters detailed showing Tx credit loss*

Example 3-18 *show process creditmon credit-loss-events showing Tx credit loss*

Example 3-19 *show logging onboard error-stats showing Tx credit loss*

For remote monitoring platform, this counter is available via SNMP OID fcIfCreditLoss.

Link Failure — Link Reset Failed Nonempty Recv Queue (LR Rcvd B2B)

Direction — Ingress Congestion

Severity — Severe

LR Rcvd B2B stands for Link Reset (LR) primitive received but ingress frames are still queued.

As Figure 3-6 shows, a frame-sender invokes the Link Reset Protocol by sending a Link Reset (LR) primitive if it has zero remaining-Tx-B2B-credits for an extended duration. The frame-receiver, after receiving the LR primitive, checks if its ingress buffers are completely empty, and if no frames are queued in them, it sends a Link Reset Response (LRR) primitive. However, if at least one frame is still queued, the frame-receiver starts a 90 ms LR Rcvd B2B timer. If the frames are transmitted to the egress port and all the ingress buffers are free, then the LR Rcvd B2B timer is canceled and an LRR primitive is sent back to the frame-sender. Both of these situations indicate a successful link credit reset and do not generate an LR Rcvd B2B condition.

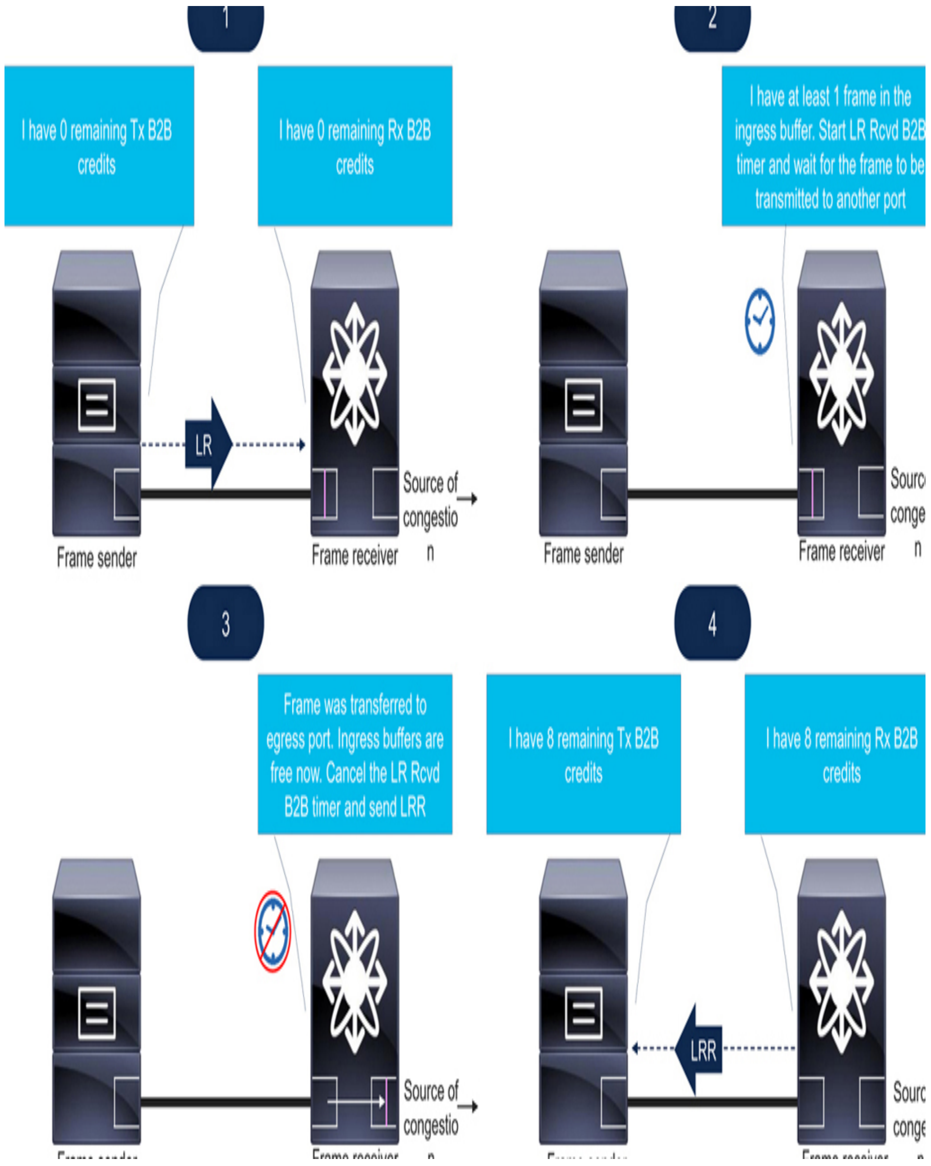

1
I have 0 remaining Tx B2B credits
I have 0 remaining Rx B2B credits
LR
Frame sender
Frame receiver
Source of congestion
2
I have at least 1 frame in the ingress buffer. Start LR Rcvd B2B timer and wait for the frame to be transmitted to another port
Frame sender
Frame receiver
3
Frame was transferred to egress port. Ingress buffers are free now. Cancel the LR Rcvd B2B timer and send LRR
Source of congestion
4
I have 8 remaining Tx B2B credits
I have 8 remaining Rx B2B credits
LRR

Figure 3-6 *Link Reset Protocol for credit loss recovery*

However, if the egress port remains congested and at least one frame is still queued at the ingress port while the LR Rcvd B2B timer expires, this results in the dropping of the queued frame(s) followed by link failure (Figure 3-7). Instead of sending an LRR, the frame-receiver sends a Not Operational Sequence (NOS) (a type of primitive sequence in Fibre Channel) to reinitialize the link.

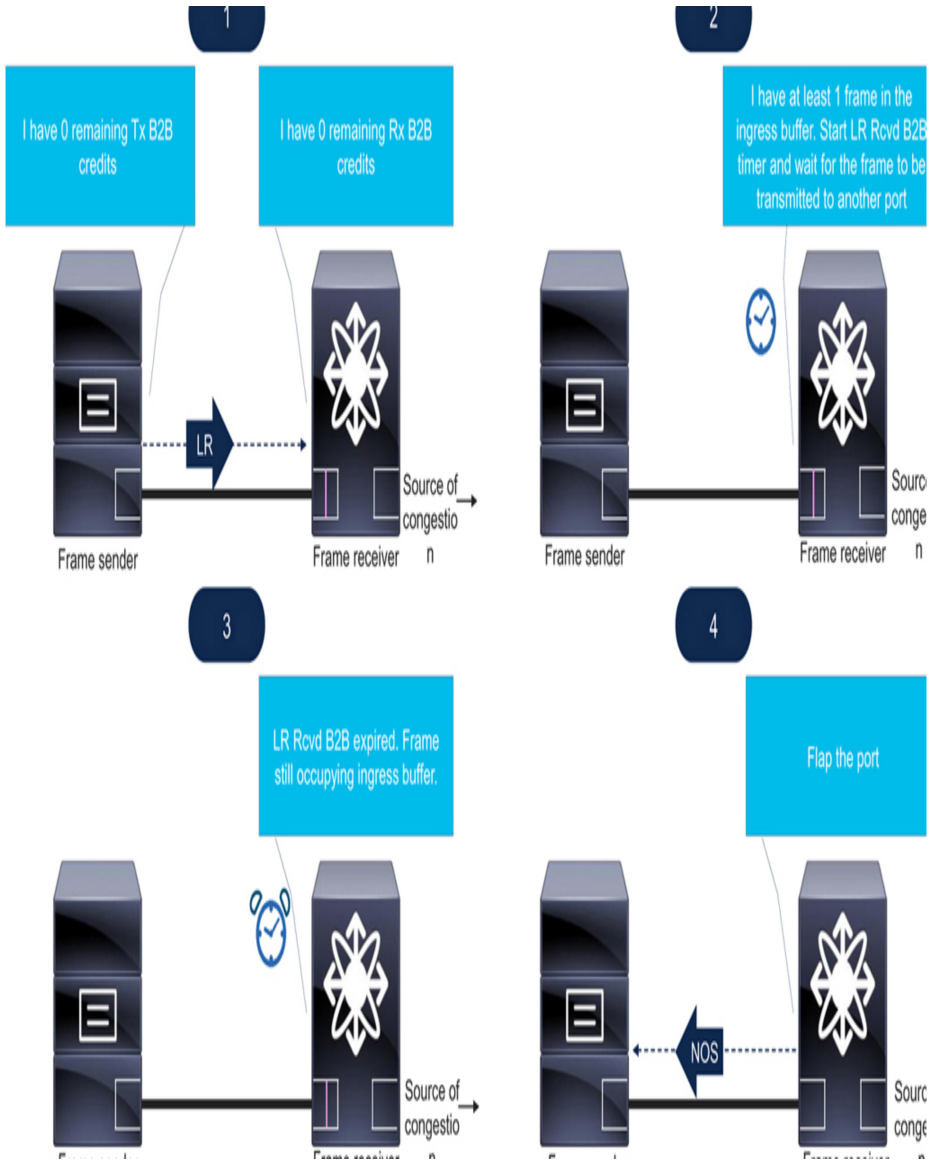

1
I have 0 remaining Tx B2B credits
I have 0 remaining Rx B2B credits
LR
Frame sender
Frame receiver
Source of congestion
2
I have at least 1 frame in the ingress buffer. Start LR Rcvd B2B timer and wait for the frame to be transmitted to another port
Frame sender
Frame receiver
3
LR Rcvd B2B expired. Frame still occupying ingress buffer.
4
Flap the port
NOS

Figure 3-7 *LR Rcvd B2B during credit loss recovery*

As Example 3-20 shows, when this occurs, a syslog message is logged stating 'Link failure Link Reset failed nonempty recv queue.'. This is an indication of severe ingress congestion at that port. This means the source of the congestion is not on this interface itself or the adjacent device but somewhere in the path to where this interface is forwarding frames that are received on the interface.

Example 3-20 *show logging logfile showing 'Link failure Link Reset failed nonempty recv queue'*

Credits and Remaining Credits

Direction — Both

Severity — Non-specific

As Chapter 2, the section on *B2B Credit Counters* explains, an FC port can have the following credit counters:

- **Rx B2B credits**: The number of receive buffers of an FC port. This does not change while the link is active.
- **Tx B2B credits**: The number of receive buffers of the neighbor FC port as informed during link initialization. This is established at FLOGI or ELP phase and does not change while the link is active.
- **Remaining-Rx-B2B-credits**: The number of free receive buffers on an FC port at the time its value is read, such as when **show interface** command was issued. Its minimum value can be zero and the maximum value can be the Rx B2B credits. Typically, this value keeps changing.
- **Remaining-Tx-B2B-credits**: The number of free receive buffers on the neighbor FC port at the time its value is read, such as when **show interface** command was issued. Its minimum value can be zero and the maximum value can be the Tx B2B credits. Typically, this value keeps changing.

Use the NX-OS command **show interface** for displaying the credit counters (Example 3-21).

Example 3-21 *show interface showing credits and remaining credits:*

As previously mentioned, the Rx and Tx credits do not change as long as the link is active. However, the remaining credits keep changing as frames and R_RDY are sent and received.

As a reminder, each time a frame-sender desires to send a frame it first checks the remaining-Tx-B2B-credits, and if this value is non-zero, the frame-sender decrements the remaining-Tx-B2B-credits by 1 and sends the frame. If the remaining-Tx-B2B-credits are zero, then the frame cannot be transmitted and must wait for a R_RDY (B2B credit) to arrive. When a credit arrives, the frame-sender increments the remaining-Tx-B2B-credits by 1. Using this non-zero count of remaining-Tx-B2B-credits, the frame-sender can now send a frame.

On a frame-receiver, each time a frame is received, the remaining-Rx-B2B-credits decrements by 1 indicating that it ‘owes’ a B2B credit to the frame-sender. When a received frame is processed and an ingress buffer is available for re-use, a R_RDY (B2B credit) is sent, and remaining-Rx-B2B-credits increment by 1.

Because the remaining-Tx-B2B-credits and remaining-Rx-B2B-credits are instantaneous values, they have limited use unless they are monitored over time to determine a pattern.

Credit Transition to Zero

Direction — Both

Severity — Non-specific

Tx Credit Transition to Zero counter increments when the remaining-Tx-B2B-credits fall to zero. This counter indicates that the port has depleted its remaining-Tx-B2B-credits and subsequent frames **may** not be immediately transmitted. If this port receives a Tx credit (by receiving a R_RDY) before it has the next frame for transmission, then there will be no delay in transmitting the frame. On the other hand, if the port does not receive a Tx credit, then any subsequent frames will have to wait.

A similar counter in the reverse direction, called Rx Credit Transition to Zero, increments when the remaining-Rx-B2B-credits fall to zero. This counter indicates that the port cannot accept any further frames from the directly connected device, and it is not returning credits by sending a R_RDY.

Tx Credit Transition to Zero SNMP variable is also available as fcIfTxBBCreditTransistionToZero.

Rx Credit Transition to Zero SNMP variable is also available as fcIfRxBBCreditTransistionToZero.

Use the NX-OS command **show interface counters detailed** on Cisco MDS switches for displaying the credit transitions to zero counters (Example 3-22).

Example 3-22 *show interface counters detailed showing Tx and Rx B2B transitions to zero*

The following are important points to remember about the credit transitions to zero counter:

1. When Credit Transition to Zero counter increments, there is no indication of how long the port was at zero remaining credits. It could be just an instant of time such as a microsecond (or less) or it could be much longer such as 100s of milliseconds up to a second. Because of this reason, it is not a strong measure of the amount of congestion.

2. Because TxWait (and RxWait) give an accurate indication of the actual amount of congestion on an interface and

they are time and date-stamped on Cisco MDS switches, they are superior indications of congestion.

3. Cisco MDS switches do not alert for the credit transition to zero counters.
4. Cisco MDS switches do not log the credit transition to zero counters with time and date stamps in OBFL.
5. It is expected that the credit-transition-to-zero counters increment minimally under normal conditions, which means that the B2B flow control mechanism is working well to achieve its function.
6. It is possible that the Tx credit transitions to zero increments without any effect on the frame transmission. For example, when an FC port has one remaining-Tx-B2B-credit, it decrements this counter to zero and sends a frame, which results in incrementing the Tx credit transitions to zero counter. The frame transmission takes some time depending on its size and the speed of the link. For instance, on a 32GFC port (bit rate of 28.025 Gbps), a full-size frame (2,148 bytes) takes approximately 600 nanoseconds for transmission. If an R_RDY is received during this time, the port is ready to send the next frame without any delay. In this condition, Tx credit transitions to zero increments with no TxWait and no real congestion.

Link Utilization

Cisco MDS switches capture the cumulative number of frames and the size of frames in the ingress and egress direction. Example 3-23 shows

- The 5-minute average rate in bits/second, bytes/second, and frames/second
- Accumulated input and output frames and bytes since the counters were cleared.

Example 3-23 *Link utilization in show interface*

To know if the throughput on a switchport can cause congestion due to over-utilization, it must be converted to percentage utilization., which can be calculated by dividing the 5-minute average throughput by the bit rate of the port. In Example 3-23, the 5-minute average ingress percent utilization is (1658604608 bps / 28.05 Gbps = 6%). A key factor in calculating the percentage utilization of a FC port is to use bit rate instead of advertised speed. Here, 28.05 Gbps is used instead of 32 Gbps. To know why, refer to Chapter 2, the section on *Difference Between Fibre Channel Speed and Bit Rate*.

To calculate percentage utilization at a smaller duration, poll the accumulated input and output bytes, take the delta, and divide by the bit rate. Of course, doing this calculation manually is not feasible in most production environments.

There are two better approaches and they both serve different purposes and have their trade-offs.

The first approach is to continuously export the metrics from all the ports to a remote monitoring platform (such as Cisco NDFC and MTM) and detect the percentage utilization there. These platforms can help in finding the trends and seasonality using intuitive charts, but they often lack the fine granularity that is needed to detect traffic bursts typically sent by All-Flash and NVMe storage arrays.

The second approach is to enable automatic alerting natively on the ports. This helps in detecting percentage utilization at a fine granularity in real-time. But this approach does not show trends, seasonality, and a historic traffic pattern, as available on remote monitoring platforms. On Cisco MDS switches, the Port-Monitor feature can automatically detect and alert the following counters that are relevant for link utilization for all the ports on a switch:

- Percentage egress utilization (Tx-datarate).
- Egress traffic burst (Tx-datarate-burst).
- Percentage ingress utilization (Rx-datarate).
- Ingress traffic burst (Rx-datarate-burst).

Collectively, these counters are called as datarate counters.

Before proceeding, we recommend a refresher of Chapter 1, the sections on *Defining Full Utilization vs High Utilization vs Over-Utilization* and *Monitoring Full Utilization vs High Utilization vs Over-Utilization*.

Tx-datarate

Tx-datarate is a Port-Monitor counter that automatically detects when the average egress utilization over an interval is greater than or equal to the configured threshold. For example, it can detect when a port's egress utilization is greater than or equal to 80% during a 10-second interval.

Tx-datarate is not a strict indication of congestion due to over-utilization since it is theoretically possible for an interface to run near 100% egress utilization and not cause congestion back to the source of the data frames. However, in our experience, there is a pretty good correlation between high Tx-datarate and congestion. Since tx-datarate is calculated as an average over the polling interval (which is a minimum of 10 seconds) a good value to use for the rising-threshold is 80%. Configuring a value much higher than that, say, 90%, may fail to detect high tx-datarate unless it is completely continuous over the 10 seconds.

Port Monitor allows taking automated actions when tx-datarate exceeds or falls below the configured threshold values.

When tx-datarate reaches the rising-threshold or falling-threshold, there are up to three things that can occur depending on the configuration:

1. Syslog message, which is written to the switch logs and optionally sent to a remote Syslog server.
2. SNMP trap sent to a remote receiver.
3. An entry is written to OBFL.

For detecting link utilization natively on the MDS switches the OBFL entry provides time and date stamped per interface information when the high Tx utilization started and when the high Tx utilization ended.

The rising-threshold indicates the time when the link utilization began to be high (actually the precise start time is the time recorded minus the polling interval). The falling-threshold indicates when the link utilization fell (again, the precise time is the time recorded minus the polling interval). The datarate percentage, Tx or Rx, was at or above the configured rising-threshold and remained above the configured falling-threshold until the falling-threshold entry time.

Tx-datarate-burst

Tx-datarate-burst detects multiple occurrences of tx-datarate at a lower granularity of one second, essentially helping in detecting traffic bursts. The OBFL on Cisco MDS switches logs the rising and falling threshold events with time and date stamps.

Refer to Example 3-24, which is based on Port-Monitor polling interval of 10 seconds, a rising-threshold of 5, a falling threshold of 1, and a datarate percentage of 90%. There are two time periods where interface fc1/13 was running at a high Tx utilization. The first interval of high utilization was Mar 3 11:18:12 through 11:18:35. These times are 10 seconds earlier than the displayed timestamps because the event started 10 seconds earlier but detected only after the polling interval. This is a 23-second duration. The second was on Mar 3 from 11:26:07 to 11:27:28. This is an 81-second duration. Since this was the tx-datarate-burst counter, both rising thresholds had five 1-second bursts of over 90% in the 10-second polling interval. When the falling-threshold hits, there were zero 1-second intervals where the tx-datarate was 90% or more.

The reason for using 90% threshold for tx-datarate-burst counter versus 80% for tx-datarate is that the sampling interval for tx-datarate-burst is 1 second versus a minimum of 10 seconds for tx-datarate.

Example 3-24 *show logging onboard datarate*

Rx-datarate

Rx-datarate is similar to Tx-datarate in the ingress direction. It detects ingress utilization on a switchport, which can be used as an alternative if the neighbor doesn't support Tx-datarate detection, alerting, and logging.

Rx-datarate-burst

Rx-datarate-burst is similar to Tx-datarate-burst in the ingress direction. It detects multiple occurrences of rx-datarate at one-second granularity. It can be used as an alternative if the neighbor does not support Tx-datarate-burst detection, alerting, and logging.

Bit Errors

Counters for detecting bit errors on Fibre Channel ports are already explained in Chapter 2, the sections on *Counters on Fibre Channel Ports* and *Fibre Channel Counters — Summary*. These counters are displayed by NX-OS commands **show interface counters detailed** and **show logging onboard error-stats** on Cisco MDS switches, as shown in Example 2-2. Those details are not repeated here for the sake of brevity.

Automatic Alerting

There are two high-level approaches for automatic alerting of congestion events.

The first approach is to detect and generate alerts natively on the network device ports. This approach allows for generating alerts at a fine granularity because the ports are the first to detect congestion. For example, Cisco MDS switches can alert for congestion (TxWait) at a granularity of 1 second.

The second approach is to export the metrics from the ports to a remote monitoring platform (such as Cisco NDFC and MTM) and then generate alerts from there. The benefit of alerting from a remote monitoring platform is that it has a long-term history of metrics, which can be used via machine learning for improved detection or in some cases predicting the congestion. On the downside, a remote monitoring platform has a longer detection granularity because of the delay in exporting the metrics from the switch.

We recommend using both these approaches because they complement each other.

The next sub-section explains automatic alerting of congestion events natively on the Cisco MDS switches. The same principles can also be used for enabling alerts on remote monitoring platforms.

Port-Monitor on Cisco MDS Switches

The Port-Monitor feature on Cisco MDS switches monitors various counters on each switchport at a low granularity (as low as 1 second). If the value of these counters exceeds the configured thresholds over a duration, it takes automated actions like sending a notification (Syslog and SNMP traps) and enabling preventive actions like Dynamic Ingress Rate Limiting (DIRL). Chapter 6 explains the actions taken by Port-Monitor in more detail.

Port Monitor has the following attributes:

Port-Monitor Policy Types

Port-Monitor can have three types of policies:

- **Logical-type All ports**: This type of policy applies to F_Ports, E_Ports, and NP_Ports. When a policy type of All is activated, there can only be a single policy active.
- **Logical-type Edge ports:** This type of policy applies to F_Ports except for trunking F_Ports which are connections to NPV switches. There can be a single Edge port policy active and optionally a Core port policy active.
- **Logical-type Core ports:** This type of policy applies to E_Ports and F_Ports to NPV switches. There can be a single Core port policy active and optionally an Edge port policy active.

Port Monitor Policy Parameters

Each policy contains one or more counters to monitor. Counters have the following parameters:

- **Counter name**: The name of the counter which describes its function.
- **Poll-interval**: The amount of time, in seconds, to achieve the counter's rising-threshold or falling-threshold.
- **Delta or absolute**: The way the counter's value is interpreted. A delta counter takes the difference between the counter's current value and the counter's previous value. Most counters are delta-type counters.
- **Rising-threshold**: The value the counter must reach to trigger a rising-threshold event.
- **Event**: The event-id number is an indication of the severity. The value that can be configured is 1 to 65535 but only values 1 through 5 are defined. These correspond to the RMON event values. The counters are categorized with the event severity values:
  - Event 1 — Fatal: None of the port monitor counters describe fatal type errors, so this event level should not be used.

    - Event 2 — Critical: Use this level for counters that indicate severe congestion such as link resets. The only congestion counter in Port-Monitor that could indicate a failure of a link is credit-loss-reco. Other Port-Monitor counters, such as link-loss and sync-loss, detect link failure but they are not always related to congestion and thus they are outside the scope of this writing.
    - Event 3 — Error: Use this level for counters that indicate moderate congestion such as frame drops/discards. The only congestion counter in Port-Monitor that indicates frame drops/discards is timeout-discards.
    - Event 4 — Warning: Use this level for counters that indicate mild congestion because of increased latency. The counters that indicate increased latency because of congestion due to slow-drain are TxWait, tx-credit-not-available, Slowport-monitor. The counters that indicate increased latency because of congestion due to over-utilization are tx-datarate, tx-datarate-burst, rx-datarate, and rx-datarate-burst.

    We strongly recommend using these event IDs because the differentiation of event levels by severity gives a visual distinction to the alerts in a GUI platforms such as Cisco DCNM/NDFC.
- **Falling-threshold**: The value the counter must fall to for triggering a falling-threshold event.
- **Alerts**: The various alerting actions that can be taken when the rising or falling threshold is reached. These include Syslog, RMON (SNMP alert), and OBFL (write to logging onboard).
- **Portguard**: Action to take when the rising threshold is reached. These include:
    - Flap (shutdown followed by no shutdown of port)
    - Errordisable (shutdown of port)
    - DIRL (Dynamic Ingress Rate Limiting of port)

- Cong-isolate (Congestion Isolate the port with or without automatic recovery)
- FPIN (Send Fabric Performance Impact Notification)

Refer to Chapter 6 for more details on these portguard actions.

If one set of counters and parameter values fits all the ports on an MDS switch, then a single All-ports-policy is enough. However, if different values are necessary for Edge and Core ports then two separate policies should be configured and activated.

Remember that in most cases congestion spreading on ISLs (E_Ports) is a consequence of congestion on F_Ports that are connected to the end devices. Because of this reason, for congestion counters, the portguard actions of flap or errordisable should only be on a logical-type edge policy. Even when an ISL is the source of congestion due to over-utilization, flapping or errordisabling it is not appropriate because doing so will only make congestion worse on other operational links. Portguard actions of flap and errordisable are acceptable on logical-type core ports for other non-congestion related counters, such as CRC-corrupted frames, just not on the counters related to congestion.

Port-Monitor Counters

Port Monitor has the following counters related to congestion:

- Credit-loss-reco
- Timeout-discards
- Tx-credit-not-available
- Txwait
- Slowport-monitor
- Tx-datarate
- Tx-datarate-burst
- Rx-datarate
- Rx-datarate-burst

These are the same counters that are previously explained in this chapter. So instead of explaining them again, this section focuses only on the logic for alerting on them.

Port-Monitor has many other counters, such as CRC and ITW, that are extremely useful. Alerting on them follows the same concept as explained in this section, but these counters are not explained here.

The following is a detailed explanation of each of these counters related to congestion:

Credit-loss-reco

The Credit-loss-reco Port-Monitor counter detects the instances of Credit Loss Recovery on a switchport. As the *Tx Credit Loss Recovery* section explains, when a switchport has zero remaining-Tx-B2B-credits for 1 second (F_Port) or 1.5 seconds (E_Port) continuously, it sends a Link Reset (LR) to reset the B2B credits on the port. This counter increments regardless of if the Link Reset Protocol is successful or not. This is the most severe indication of congestion. It is configured as a count within a poll-interval. We suggest rising threshold to be 1 or 2 in a 10-second or 60-second poll-interval. The falling threshold should always be 0. Since credit-loss recovery can severely impact a network, it is a good candidate for enabling portguard action of errdisable on edge ports.

Suggested values

- Poll-interval: 60 seconds
- Type: delta
- Rising-threshold: 1-2
- Falling-threshold: 0
- Event — 2 (Critical), indicates severe congestion.
- Optional: Portguard errordisable (logical-type edge policy only)

Timeout-discards

The timeout-discards Port-Monitor counter counts the frames that are discarded at an egress port due to the age of the frames in the switch exceeding the congestion-drop timeout (enabled by default at 500ms) and no-credit-drop timeout (not enabled by default). It is configured as a count within a poll-interval.

Suggested values

- Poll-interval: 60 seconds
- Type: delta
- Rising-threshold: 1-50
- Falling-threshold: 0
- Event — 3 (Error), indicates moderate congestion.
- Optional: Portguard errordisable (logical-type edge policy only)

Tx-credit-not-available

The tx-credit-not-available Port-Monitor counter detects the instances of continuous 100 ms duration at zero remaining-Tx-B2B-credits. It is configured as a percentage of the poll-interval. For example, a poll-interval of 1 second and a rising-threshold of 10 indicates 10% or 100 ms. There would have to be a single instance of tx-credit-not-available for this counter to trigger.

Suggested values

- Poll-interval: 1 second
- Type: delta
- Rising-threshold: 10%-40% - Detects 1 to 4 instances of 100ms intervals at zero remaining-Tx-B2B-credits.
- Falling-threshold: 0
- Event — 4 (Warning), indicates mild congestion.

Txwait

The txwait Port-Monitor counter detects the percentage TxWait in the configured poll-interval. For example, if 30 is configured as the rising threshold with a poll-interval of 1 second then Port-Monitor alerts when a switchport has a cumulative duration of 300ms at zero remaining-Tx-B2B-credits in the 1-second poll-interval. We suggest a poll-interval of 1 second, a rising threshold of 30% or 40%, and a falling-threshold of 10%.

Suggested values

- Poll-interval: 1 second
- Type: delta
- Rising-threshold: 30%-40%
- Falling-threshold: 0%-10%
- Event — 4 (Warning), indicates mild congestion.

Tx-slowport-oper-delay

The tx-slowport-oper-delay Port-Monitor counter detects the slowport-monitor events. Specifically, it triggers the “oper delay’ that the slowport-monitor records. The operational delay is the continuous duration for which a port has zero remaining-Tx-B2B-credits and only the durations longer than the configured admin delay are captured. Hence, slowport-monitor must be configured for Port-Monitor to detect tx-slowport-oper-delay. Slowport-monitor is useful for diagnosing relatively minor congestion problems that might affect latency-sensitive applications. It is configured in milliseconds.

Suggested values

- Poll-interval: 1 second
- Type: Absolute
- Rising-threshold: 30 ms
- Falling-threshold: 0 ms
- Event — 4 (Warning), indicates mild congestion.

Tx-datarate

The tx-datarate Port-Monitor counter detects when the egress utilization on a switchport reaches a configured percentage. It is a delta-type counter that works by calculating the number of bytes transmitted over the poll-interval, converting them to bits/second, and determining if the average rate exceeds the configured rising-threshold value. This counter is used to detect congestion due to over-utilization. The minimum (and recommended) poll-interval is 10 seconds which means that the rate percentage is an average rate over the 10 seconds. That is why a rising threshold of no more than 80%-85% should be used since congestion due to over-utilization can occur when the interface hits periods of high egress utilization for short durations.

One important feature of all the ‘datarate’ counters including Tx-datarate, is that MDS not only generates alerts and syslog messages when rising and falling thresholds are reached but also logs these events to OBFL datarate. This allows the recording of these types of events natively on the switch.

Suggested values

- Poll-interval: 10 seconds
- Type: Delta
- Rising-threshold: 80%-85%
- Falling-threshold: 70%-79%
  - If portguard DIRL is specified, then 70% falling-threshold is recommended to give a 10% ‘stasis’ range, assuming rising-threshold is 80%. Refer to Chapter 6 for more details on DIRL.
  - If portguard DIRL is not specified, then 79% falling-threshold is recommended to give better diagnostic information as to when the port was running at the 80% rising-threshold percentage or higher.
- Event — 4 (Warning), indicates mild congestion.

Tx-datarate-burst

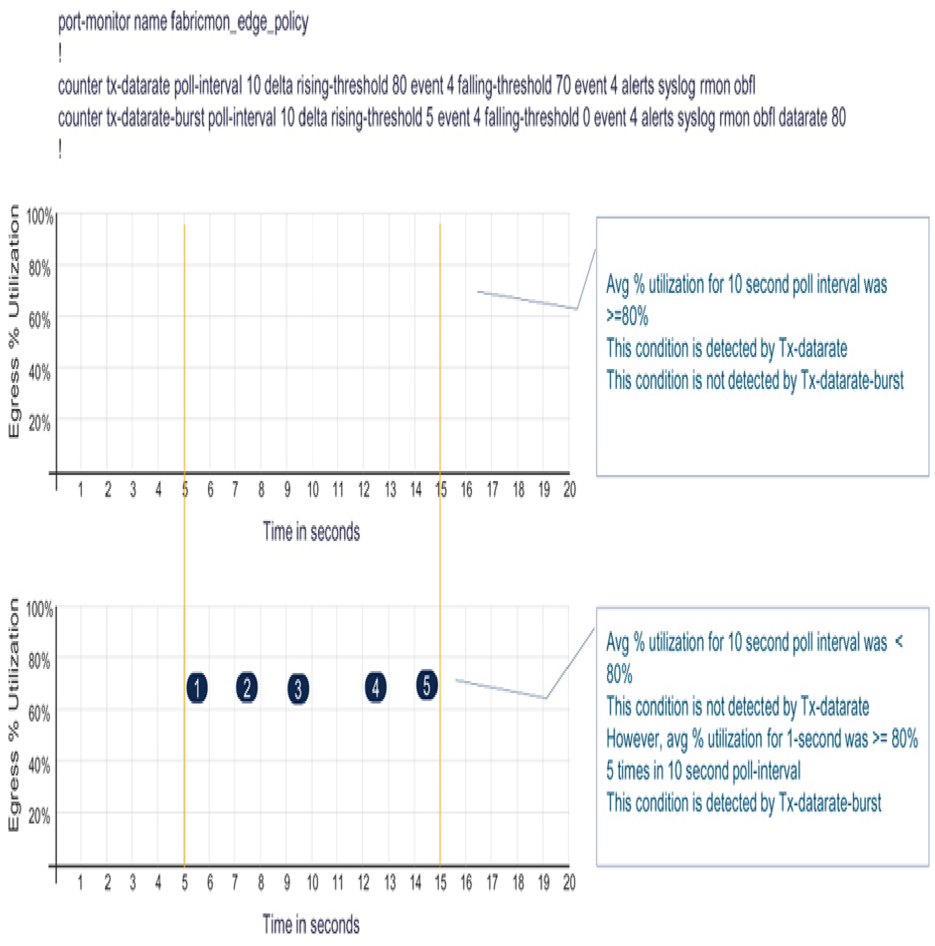
The tx-datarate-burst Port-Monitor counter is similar to the tx-datarate counter, but it is designed to be more granular and therefore it detects when there are shorter bursts of high egress utilization. This counter can detect over-utilization in smaller one-second intervals and does not rely on the average Tx-datarate to be high over the entire poll-interval.

Tx-datarate-burst has a parameter, called datarate. Every one second, the egress link utilization is calculated. If the one-second Tx-datarate percentage equals or exceeds the configured datarate percentage threshold, then an internal counter increments by 1. When the count of that internal counter equals or exceeds the rising-threshold value, Port-Monitor triggers a rising-threshold alert.

Like tx-datarate, tx-datarate-burst is a delta-type counter.

Besides generating alerts and syslog messages, Tx-datarate-burst also logs the events to OBFL. This allows recording of these types of events natively on the MDS switches.

As Figure 3-8 shows, tx-datarate and tx-datarate-burst can both be configured since they trigger slightly differently. However, if too many redundant alerts are generated, consider enabling just the tx-datarate-burst counter because it is designed to be more granular.



```
port-monitor name fabricmon_edge_policy
!
counter tx-datarate poll-interval 10 delta rising-threshold 80 event 4 falling-threshold 70 event 4 alerts syslog rmon obfl
counter tx-datarate-burst poll-interval 10 delta rising-threshold 5 event 4 falling-threshold 0 event 4 alerts syslog rmon obfl datarate 80
!
```

Avg % utilization for 10 second poll interval was >=80%
This condition is detected by Tx-datarate
This condition is not detected by Tx-datarate-burst

Avg % utilization for 10 second poll interval was < 80%
This condition is not detected by Tx-datarate
However, avg % utilization for 1-second was >= 80% 5 times in 10 second poll-interval
This condition is detected by Tx-datarate-burst

Figure 3-8 *Tx-datarate and Tx-datarate-burst counters in Port-Monitor*

Suggested values

- Poll-interval: 10 seconds
- Type: Delta
- Rising-threshold: 5-7. This is a count of one-second intervals where the Tx datarate is greater than or equal to the configured datarate parameter.
- Falling-threshold: 1-2. This is a count of one-second intervals where the Tx datarate is less than the configured datarate parameter.
- Datarate: 90%
- Event — 4 (Warning), indicates mild congestion.

Rx-datarate

The Rx-datarate Port-Monitor counter detects when the ingress utilization on a switchport reaches a configured percentage. This counter is the same as the Tx-datarate counter in the reverse direction.

Rx-datarate counter does not indicate congestion on the switch on which it is enabled. The real use-case for enabling Rx-datarate is to automatically alert on behalf of the upstream neighboring port if that device doesn’t support automatic alerting. Rx-datarate alert indicates congestion on the upstream switch in the same way as the Tx-datarate alert on the local switch.

Like other datarate counters, Rx-datarate generates alerts and syslog messages, and logs these events to OBFL.

Suggested values

- Poll-interval: 10 seconds
- Type: Delta
- Rising-threshold: 80%-85%
- Falling-threshold: 70%-79%
- Event – 4 (Warning), indicates mild congestion on the upstream switch.

Rx-datarate-burst

The rx-datarate-burst Port-Monitor counter is similar to the tx-datarate-burst counter, but in the ingress direction. It is designed to be more granular and therefore it detects when there are shorter bursts of high ingress utilization. This counter can detect potential over-utilization on the upstream neighbor port in smaller one-second intervals and doesn’t rely on the average Rx-datarate to be high over the entire poll-interval.

Every one second, the ingress link utilization is calculated. If the one-second Rx-datarate equals or exceeds the configured datarate threshold, then an internal counter increments by 1. When the count of that internal counter equals or exceeds the rising-threshold value, the Port-Monitor triggers a rising-threshold alert.

Like rx-datarate, rx-datarate-burst is a delta-type counter.

Like other datarate counters, Rx-datarate-burst generates alerts and syslog messages and logs these events to OBFL datarate.

Note that Rx-datarate and Rx-datarate-burst can both be configured since they trigger slightly differently. However, if too many redundant alerts are generated, consider enabling just the Rx-datarate-burst counter because it is designed to be more granular.

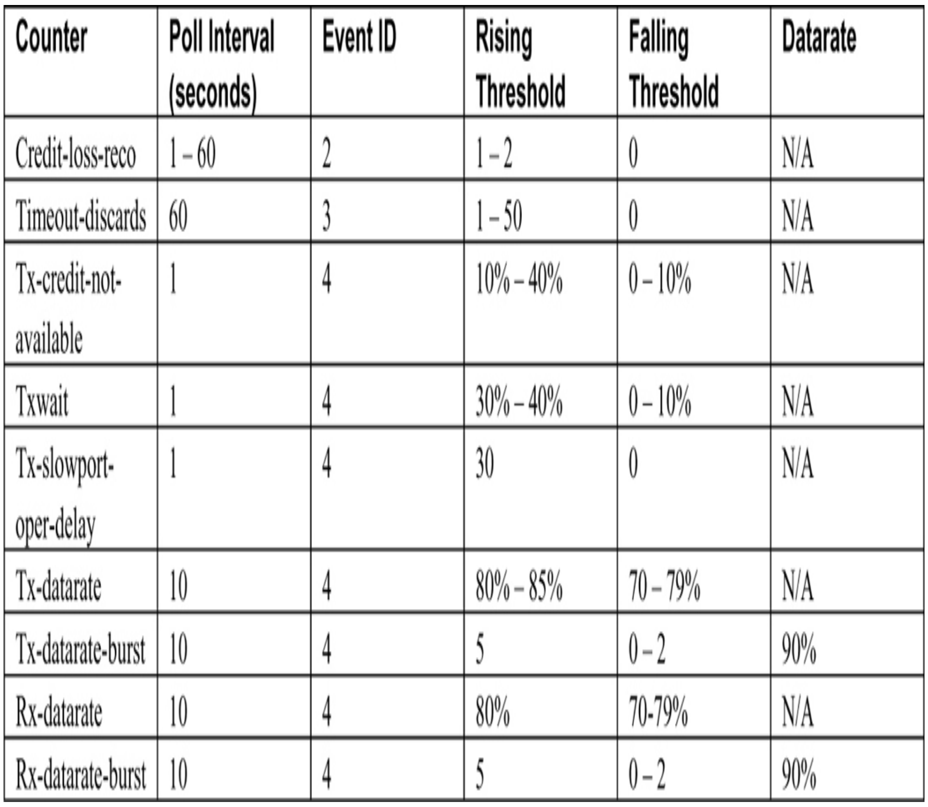
Remember that the configuration of Rx-datarate and/or Rx-datarate-burst is not necessary if the directly connected port is already enabled for alerting on Tx-datarate and/or Tx-datarate-burst.

Suggested values

- Poll-interval: 10 seconds
- Type: Delta
- Rising-threshold: 5-7. This is a count of one-second intervals where the Rx datarate is greater than or equal to the configured datarate parameter.
- Falling-threshold: 1-2. This is a count of one-second intervals where the Rx datarate is less than the configured datarate parameter.
- Datarate: 90%
- Event — 4 (Warning), indicates mild congestion.

Table 3-3 summarizes the Port-Monitor counters that are explained in this section.

Table 3-3 *Summary of Port-Monitor counters that are related to congestion*



Counter	Poll Interval (seconds)	Event ID	Rising Threshold	Falling Threshold	Datarate
Credit-loss-reco	1 – 60	2	1 – 2	0	N/A
Timeout-discards	60	3	1 – 50	0	N/A
Tx-credit-not-available	1	4	10% – 40%	0 – 10%	N/A
Txwait	1	4	30% – 40%	0 – 10%	N/A
Tx-slowport-oper-delay	1	4	30	0	N/A
Tx-datarate	10	4	80% – 85%	70 – 79%	N/A
Tx-datarate-burst	10	4	5	0 – 2	90%
Rx-datarate	10	4	80%	70-79%	N/A
Rx-datarate-burst	10	4	5	0 – 2	90%

Finally, regarding the use of Port-Monitor for logging the metrics in OBFL, make note of the following:

1. On Cisco MDS switches, almost all the congestion logging and metrics are recorded in OBFL by default even without configuring Port-Monitor.
2. However, without configuring Port-Monitor, automatic alerts will not be generated, and users will not be notified.
3. OBFL logging is not turned on by default for two features.
   a. The first is slowport-monitor. Refer to the earlier section on *When to enable slowport-monitor* for more details.
   b. The second is the ability of the MDS to detect over-utilization. To detect over-utilization one of the tx-datarate/tx-datarate-burst counters must be configured in Port Monitor. Once this is done entries will start to be recorded in OBFL.

Detecting Congestion on a Remote Monitoring Platform

Remote monitoring platforms can monitor all the ports across many networks simultaneously to provide single-pane-of-glass visibility.

The following are some more benefits of using remote monitoring platforms:

1. Always-on monitoring.
2. Scope across multiple networks to get a full view of traffic between hosts and storage arrays via all the paths, such as SAN-A and SAN-B.
3. Monitoring can be expanded to end devices as well. For example, when a host-connected switchport reports congestion, the health of the host can be monitored as well.
4. The on-switch alerts, such as those generated by Port-Monitor on Cisco MDS switches, can be sent to the same monitoring platforms.
5. Reporting and auditing.
6. Long-term trends, seasonality, and patterns.
7. Possibility of using machine learning for predicting congestion events based on the history data.

The remote monitoring platforms can be of the following types.

- A platform or application developed by the device manufacturer/vendor, such as Cisco Nexus Dashboard Fabric Controller (NDFC), formerly Data Center Network Manager (DCNM).
- A 3rd party or a custom-developed platforms/application, such as the MDS Traffic Monitoring (MTM) App.

NDFC/DCNM Congestion/Slow-drain Analysis

NDFC/DCNM congestion (slow-drain) analysis continuously polls the Fibre Channel fabrics in an always-on fashion. It produces a color-coded topology diagram showing where the congestion was found. It does not pull historic information from the on-switch OBFL. Rather it polls various congestion metrics in real-time from the switches and maintains its dedicated history.

The congestion (slow-drain) analysis job can be run 24 hours a day and the polled metrics are saved so that history can be investigated similar to the on-switch OBFL on Cisco MDS switches.

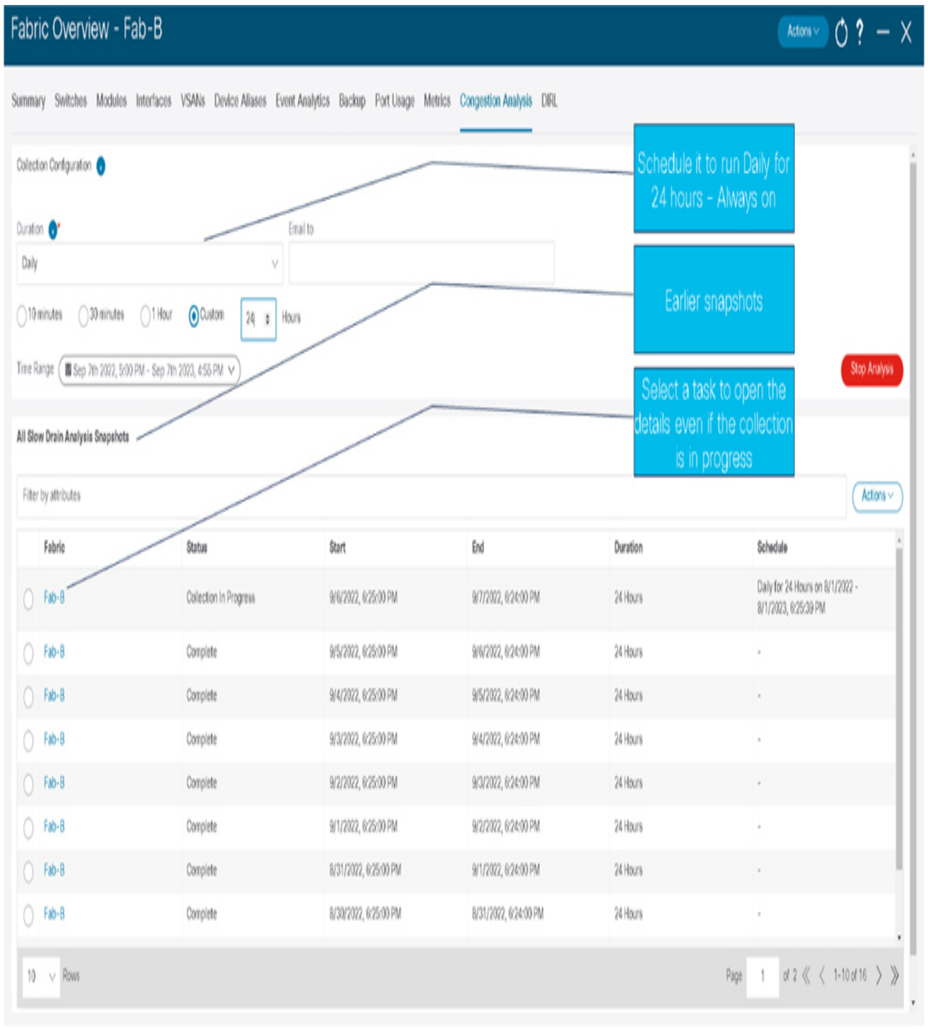
NDFC/DCNM congestion (slow-drain) analysis collects the following metrics from the switch via SNMP.

- Switch name, port, speed, and where it is connected to.
- Tx Credit Loss
- Link Reset (LR) received.
- Link Reset (LR) sent.
- Timeout discards
- Overall discards
- Tx-credit-not-available for 100ms
- Rx B2B credits transitions to zero
- Tx B2B credits transitions to zero
- TxWait in seconds
- TxWait in percentage

The output is arranged in ‘Level 3’, ‘Level 2’, ‘Level 1.5’, ‘Level 1’, and High Utilization and can be sorted so that the interfaces with the highest values are shown at the top. As previously explained, Level 3 is Severe congestion that involves Link Resets. Level 2 is moderate congestion that involves frame drops. Level 1 is mild congestion that involves increased latency.

There are time sliders so a specific period can be investigated.

Figure 3-9 shows the NDFC congestion analysis.



Collection Configuration

Duration: Daily

Email to:

10 minutes [✓ Custom] 30 minutes 1 Hour — 24 Hours

Time Range: Sep 7th 2022, 5:00 PM - Sep 7th 2023, 4:55 PM

All Slow Drain Analysis Snapshots

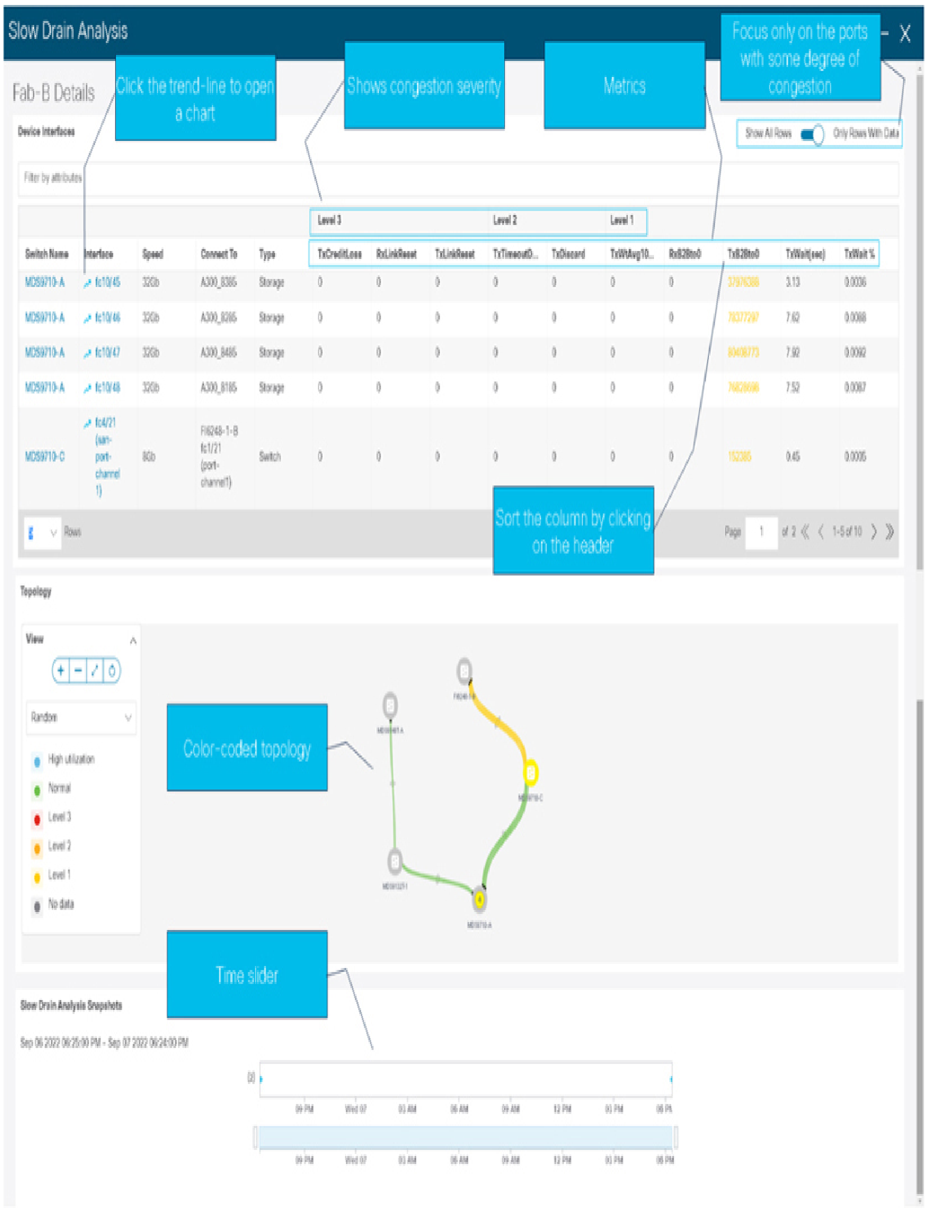
Fabric	Status	Start	End	Duration	Schedule
Fab-B	Collection In Progress	9/6/2022, 6:25:00 PM	9/7/2022, 6:24:00 PM	24 Hours	Daily for 24 Hours on 8/1/2022 - 8/1/2023, 6:25:39 PM
Fab-B	Complete	9/5/2022, 6:25:00 PM	9/6/2022, 6:24:00 PM	24 Hours	-
Fab-B	Complete	9/4/2022, 6:25:00 PM	9/5/2022, 6:24:00 PM	24 Hours	-
Fab-B	Complete	9/3/2022, 6:25:00 PM	9/4/2022, 6:24:00 PM	24 Hours	-
Fab-B	Complete	9/2/2022, 6:25:00 PM	9/3/2022, 6:24:00 PM	24 Hours	-
Fab-B	Complete	9/1/2022, 6:25:00 PM	9/2/2022, 6:24:00 PM	24 Hours	-
Fab-B	Complete	8/31/2022, 6:25:00 PM	9/1/2022, 6:24:00 PM	24 Hours	-
Fab-B	Complete	8/30/2022, 6:25:00 PM	8/31/2022, 6:24:00 PM	24 Hours	-

10 Rows — Page 1 of 2 — 1-10 of 16

Figure 3-9 *Opening and Scheduling NDFC Congestion Analysis*

We recommend scheduling congestion analysis daily for 24 hours. This helps in maintaining the historic metrics even before congestion occurs.

Select a job while the collection is in progress or an earlier finished job to see the results. Figure 3-10 shows congestion metrics arranged and sorted as per severities, color-coded topology, and time slider.



					Level 3					Level 2		Level 1		
Switch Name	Interface	Speed	Connect To	Type	TxCreditLoss	RxLinkReset	TxLinkReset	TxTimeoutD...	TxDiscard	TxWtAvg10...	RxB2Bto0	TxB2Bto0	TxWait(sec)	TxWait %
MDS9710-A	fc10/45	32Gb	A300_8385	Storage	0	0	0	0	0	0	0	37976388	3.13	0.0036
MDS9710-A	fc10/46	32Gb	A300_8285	Storage	0	0	0	0	0	0	0	70377297	7.62	0.0088
MDS9710-A	fc10/47	32Gb	A300_8485	Storage	0	0	0	0	0	0	0	80408773	7.92	0.0092
MDS9710-A	fc10/48	32Gb	A300_8185	Storage	0	0	0	0	0	0	0	76828696	7.52	0.0087
MDS9710-C	fc4/21 (san-port-channel 1)	8Gb	FI6248-1-B fc1/21 (port-channel1)	Switch	0	0	0	0	0	0	0	152385	0.45	0.0005

5 Rows Page 1 of 2 1-5 of 10

Sep 06 2022 06:25:00 PM - Sep 07 2022 06:24:00 PM

Figure 3-10 *Using NDFC Congestion Analysis*

The topology is interactive. Hover over a network link or a switch in the topology to see more details about it. Double-clicking on an entity limits the scope of the table to that entity.

Click on the trend line on the left of the interface to see the metric charts. As Figure 3-11 shows, a pop-up chart of congestion metrics for that interface. Hover over the chart to see the value at that time. Click on the legends below the graph to hide or show the chart for that metric.

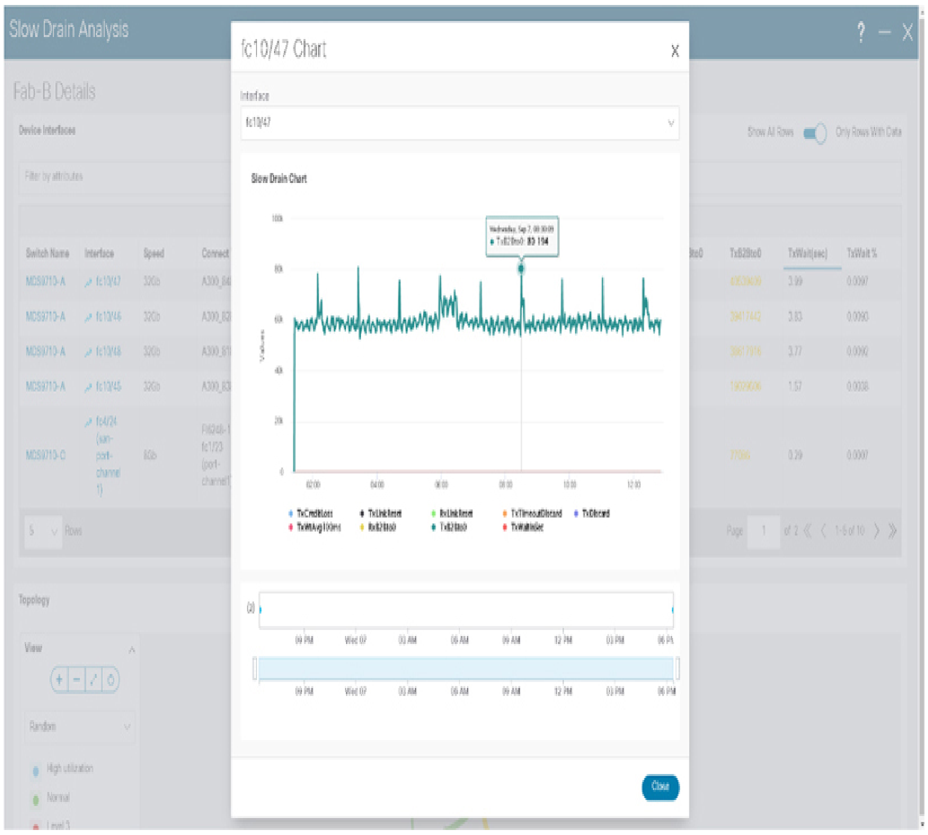


Figure 3-11 *Trends and seasonality of congestion in NDFC Congestion Analysis*

Congestion analysis in NDFC is under continuous evolution. New features are being added to it in every release. It is possible that it looks different in the latest release of NDFC. Please refer to the release notes of Cisco NDFC SAN for up-to-date information.

The MDS Traffic Monitoring (MTM) App

MDS Traffic Monitoring (MTM) is a simple application for monitoring a large number of MDS switches. It is developed by Paresh Gupta, author of this book, and it is available on GitHub (https://github.com/paregupt/mds_traffic_monitor) for free under the MIT license.

MTM Architecture

MTM polls the MDS switches every 30 seconds (by default) for metrics like inventory, health metrics like CPU and memory, port states, port speeds, port utilization, error counters, and transceiver health, such as temperature and Rx power.

MTM has a collector (written in Python), which pulls metrics from Cisco MDS switches using NX-API. The metrics are normalized and correlated before writing to InfluxDB (a time-series database). Finally, Grafana provides the visualization and use cases.

The MTM collector needs a read-only account on the MDS switches and NX-API must be enabled on them using the NX-OS command **feature nxapi**.

By default, all the metrics are stored in the database forever.

MTM Use-cases

MTM is a lightweight App. It is not designed to take place of a fully functional monitoring platforms like Cisco NDFC/DCNM. Instead, it is designed for users who like customizing the monitoring platforms based on the problems they are trying to solve. Many ready-made use cases are already available in MTM. Users can customize it further or integrate it with other similar apps like UCS Traffic Monitoring (UTM), which monitors traffic in the Cisco UCS servers. Refer to Chapter 9 for more details on the UTM app.

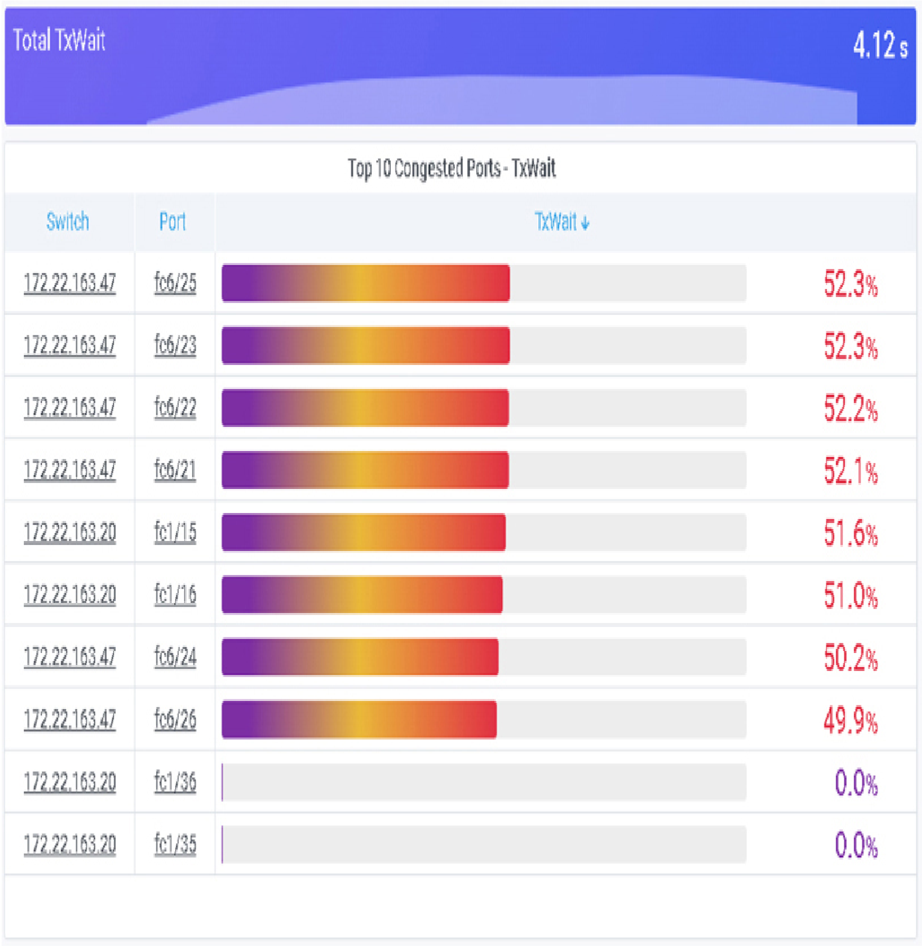
MTM groups the switches logically. For example, you can monitor all the switches at a site or all the switches in a fabric on a single dashboard. Then, you can drill down at a switch level or investigate a switchport.

At the time of this writing, MTM is used for monitoring many production environments. Chapter 2 *Case Study of An Online Retailer* uses MTM. Chapter 6 section *DIRL in Action* demonstrates DIRL using MTM. The illustrations in the later section on *Average and Peak Utilization* are taken from MTM running in another production environment.

The following sections explain a handful of MTM use-cases.

Finding Slow-drain Devices

As Figure 3-12 shows, MTM shows the top-10 ports with the most TxWait across all the monitored MDS switches. If these ports are of type edge, their connected devices are the source of slow-drain. Typically, you should start investigating the port at the top of this list.


Total TxWait 4.12 s

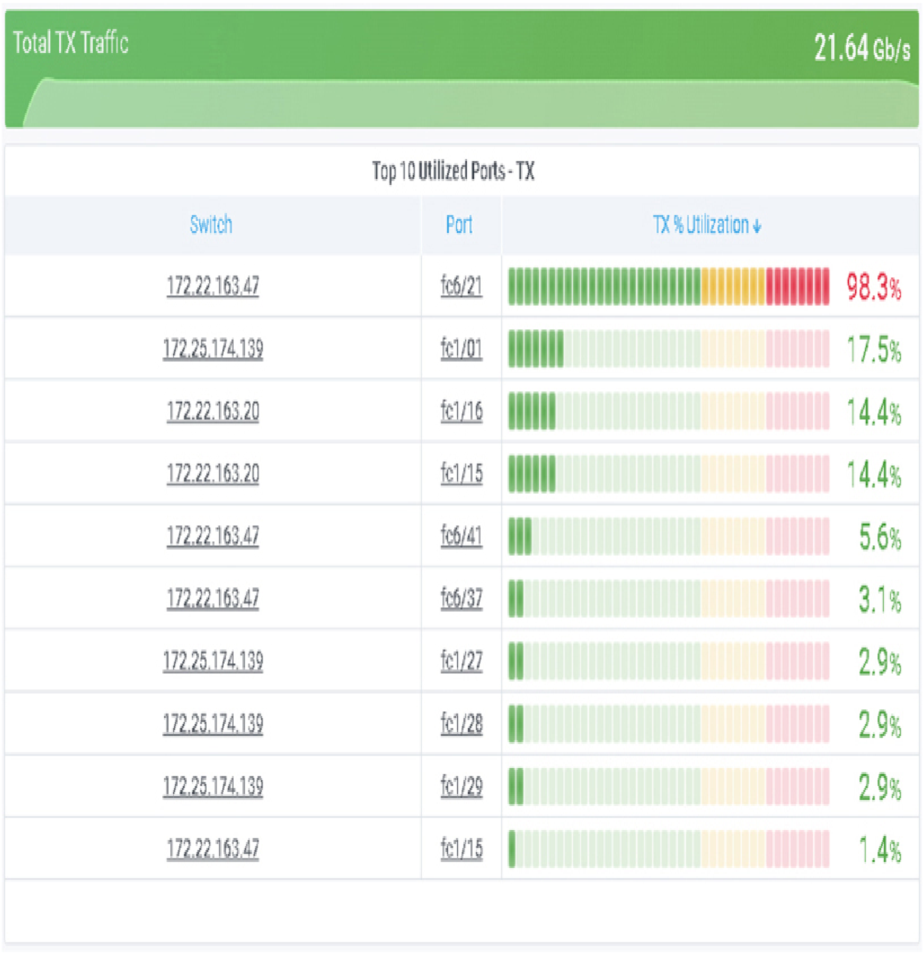
Top 10 Congested Ports - TxWait

Switch	Port	TxWait ↓
172.22.163.47	fc6/25	52.3%
172.22.163.47	fc6/23	52.3%
172.22.163.47	fc6/22	52.2%
172.22.163.47	fc6/21	52.1%
172.22.163.20	fc1/15	51.6%
172.22.163.20	fc1/16	51.0%
172.22.163.47	fc6/24	50.2%
172.22.163.47	fc6/26	49.9%
172.22.163.20	fc1/36	0.0%
172.22.163.20	fc1/35	0.0%

Figure 3-12 *Top-10 Congested Ports due to TxWait in the MTM app*

Finding Highly Utilized Links

As Figure 3-13 shows, MTM shows top-10 most utilized ports across all the monitored MDS switches. As previously explained, ports that show high utilization are good candidates that cause congestion due to over-utilization. In Figure 3-13, the port at the top of the list is running at 98.3% capacity and it might very well be causing congestion.


Total TX Traffic 21.64 Gb/s

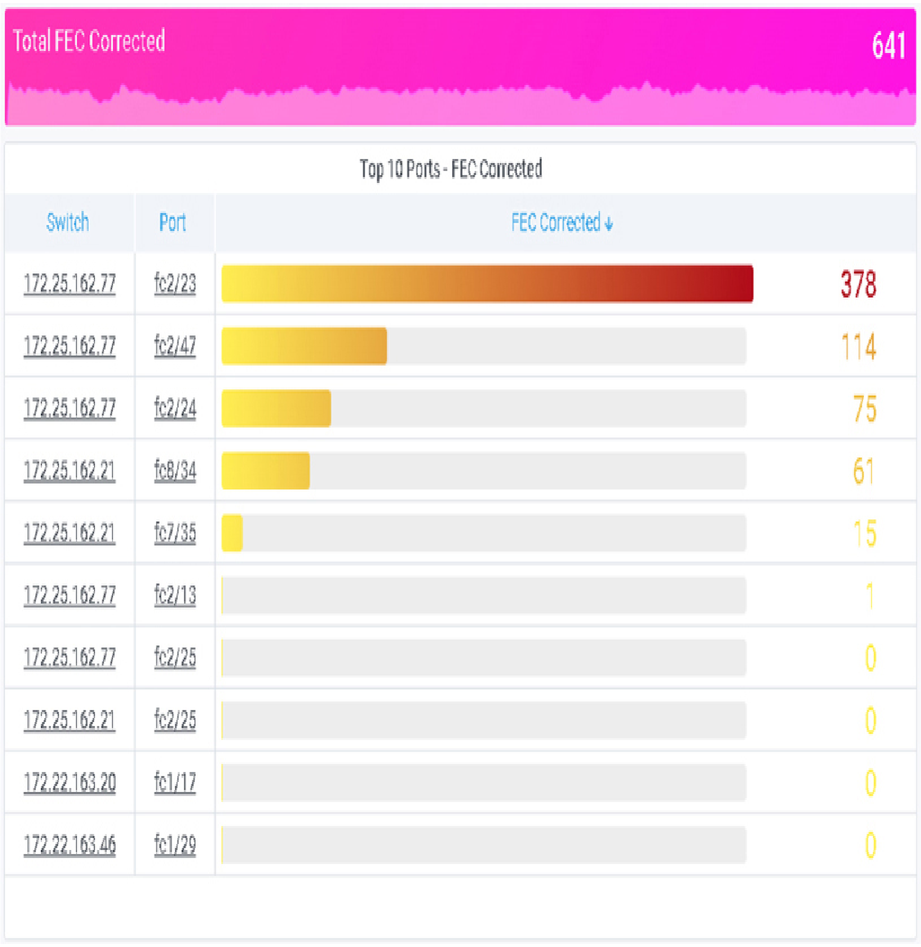
Top 10 Utilized Ports - TX

Switch	Port	TX % Utilization
172.22.163.47	fc6/21	98.3%
172.25.174.139	fc1/01	17.5%
172.22.163.20	fc1/16	14.4%
172.22.163.20	fc1/15	14.4%
172.22.163.47	fc6/41	5.6%
172.22.163.47	fc6/37	3.1%
172.25.174.139	fc1/27	2.9%
172.25.174.139	fc1/28	2.9%
172.25.174.139	fc1/29	2.9%
172.22.163.47	fc1/15	1.4%

Figure 3-13 *Top-10 Utilized Ports in the MTM app*

Finding Ports with the Most Errors

MTM shows the top-10 ports with the most errors across all the monitored MDS switches. Refer to Figure 3-14, which shows the top-10 ports showing FEC corrected blocks. As Chapter 2 explains, FEC corrected blocks indicate bit errors, which may lead to congestion or worsen existing congestion.


Total FEC Corrected 641

Top 10 Ports - FEC Corrected

Switch	Port	FEC Corrected ↓
172.25.162.77	fc2/23	378
172.25.162.77	fc2/47	114
172.25.162.77	fc2/24	75
172.25.162.21	fc8/34	61
172.25.162.21	fc7/35	15
172.25.162.77	fc2/13	1
172.25.162.77	fc2/25	0
172.25.162.21	fc2/25	0
172.22.163.20	fc1/17	0
172.22.163.46	fc1/29	0

Figure 3-14 *Top-10 Ports with the most number of FEC corrected blocks*

MTM monitors the following metrics from the MDS switches and shows the top-10 ports similar to Figure 3-14.

- CRC corrupted frames.
- FEC corrected blocks.
- FEC uncorrected blocks.
- Tx and Rx B2B credits transition to zero.
- Timeout discards.
- Invalid Transmission Words (ITW).
- Link Failures.
- Signal Loss.
- Sync Loss.
- Link Reset (LR) received and transmitted.

- Link Reset Response (LRR) received and transmitted.

Trending and Seasonality

MTM stores all the metrics forever to provide long-term trending and seasonality. Using the port utilization trends, you can predict when the port may reach more than 90% utilization and plan an upgrade accordingly. As Figure 3-15 shows, you can correlate multiple metrics on a switchport side-by-side.

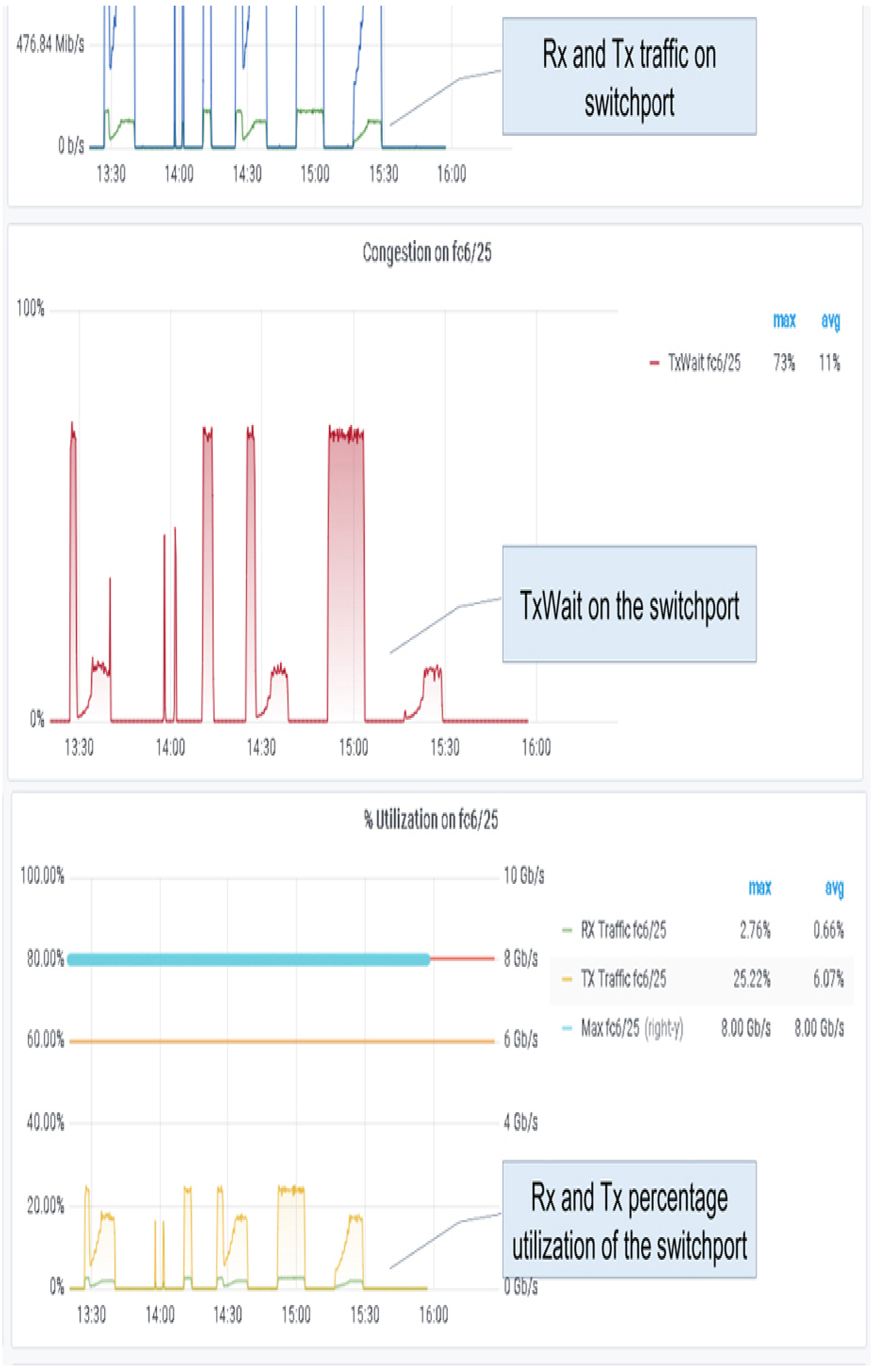


Figure 3-15 *Side-by-side trends and seasonality of multiple metrics*

End-to-end Correlation

For troubleshooting congestion end-to-end in a fabric, you can monitor the ports on multiple switches in the datapath. For example, in Chapter 6 the section on *DIRL in Action* uses the MTM app for monitoring the edge switchport side-by-side with the ISL port on the upstream switch. In the same demonstrations, metrics from the application layer on the connected hosts are also shown side-by-side, although this feature is not available in the MTM app by default.

Metric Export Mechanisms

This section explains various mechanisms and recommendations for exporting metrics from network devices to remote monitoring platforms.

Parsing the Command Line Output Over SSH

Remote monitoring platforms, applications, or scripts can open a SSH session to a device, run the commands, collect their outputs, and parse them for capturing the relevant metrics. Although this approach is widely used in production environments, its users often complain that the parsing logic breaks when the command output changes, and trying to catch up with the ongoing changes takes significant effort. Another drawback is the longer polling interval because opening SSH sessions and collecting the command output involves delay.

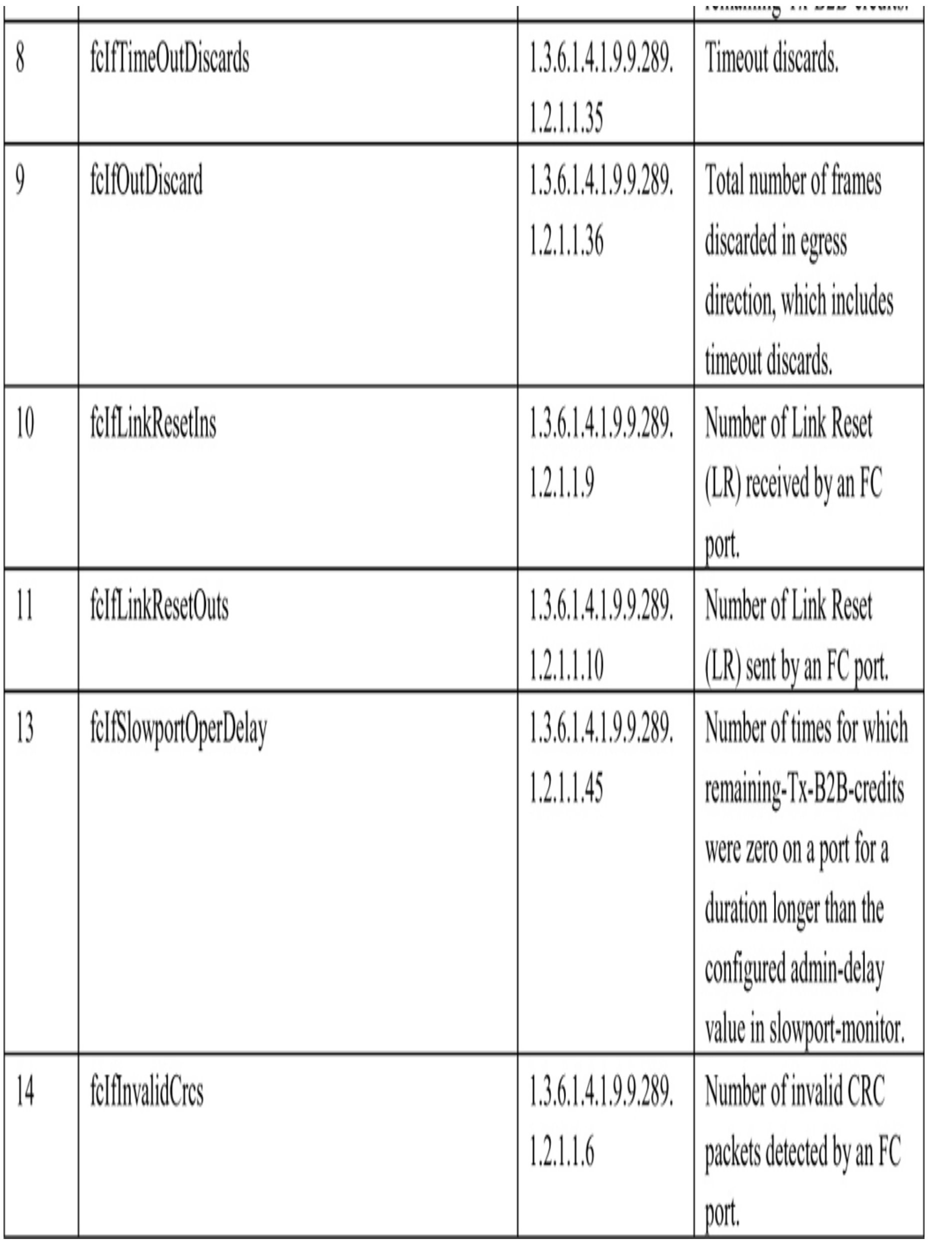
Simple Network Management Protocol

SNMP is the most common mechanism for exporting metrics from network devices. Typically, the remote monitoring platforms poll the SNMP OID at every polling interval. Some OIDs, such as ifHCInOctets and ifHCOutOctets, are standardized and thus they work on most devices. For polling device-specific metrics, refer to the documentation for finding their OIDs.

Table 3-4 shows commonly used SNMP MIBs and their OIDs for polling metrics from the ports on Cisco MDS switches.

Table 3-4 *Commonly used MIBs for monitoring the ports on Cisco MDS switches*

S.No	MIB Name	OID	Description
1	ifHCInOctets	1.3.6.1.2.1.31.1.1.1.6	Number of bytes received by the interface. This can be used to calculate Rx datarate in bits per second and percentage.
2	ifHCOutOctets	1.3.6.1.2.1.31.1.1.1.10	Number of bytes transmitted by the interface. This can be used to calculate Tx datarate in bits per second and percentage.
3	fcIfTxWaitCount	1.3.6.1.4.1.9.9.289.1.2.1.1.15	TxWait, the number of 2.5µs when a switch port could not transmit due to zero remaining-Tx-B2B-credits.
4	fcHCIfBBCreditTransistionFromZero	1.3.6.1.4.1.9.9.289.1.2.1.1.40	Tx B2B credit transition to zero.
5	fcIfBBCreditTransistionToZero	1.3.6.1.4.1.9.9.289.1.2.1.1.39	Rx B2B credit transition to zero.
6	fcIfTxWtAvgBBCreditTransitionToZero	1.3.6.1.4.1.9.9.289.1.2.1.1.38	Number of 100ms when a switchport could not transmit due to zeroremaining-Tx-B2B-credits (Tx-credit-not-available).
7	fcIfCreditLoss	1.3.6.1.4.1.9.9.289.1.2.1.1.37	Credit Loss (recovery) - Represents extended period (1 or 1.5 seconds) of zero remaining-Tx-B2B-credits.



8	fcIfTimeOutDiscards	1.3.6.1.4.1.9.9.289.1.2.1.1.35	Timeout discards.
9	fcIfOutDiscard	1.3.6.1.4.1.9.9.289.1.2.1.1.36	Total number of frames discarded in egress direction, which includes timeout discards.
10	fcIfLinkResetIns	1.3.6.1.4.1.9.9.289.1.2.1.1.9	Number of Link Reset (LR) received by an FC port.
11	fcIfLinkResetOuts	1.3.6.1.4.1.9.9.289.1.2.1.1.10	Number of Link Reset (LR) sent by an FC port.
13	fcIfSlowportOperDelay	1.3.6.1.4.1.9.9.289.1.2.1.1.45	Number of times for which remaining-Tx-B2B-credits were zero on a port for a duration longer than the configured admin-delay value in slowport-monitor.
14	fcIfInvalidCrcs	1.3.6.1.4.1.9.9.289.1.2.1.1.6	Number of invalid CRC packets detected by an FC port.

There are many benefits of using SNMP. It is an IETF standard protocol, and it is widely supported on most devices. It has been under use for many decades. Because of these reasons, SNMP can be used for monitoring a multi-vendor environment as well. In addition to polling, SNMP can be used for receiving alerts (via traps or informs) and configuring the network devices.

On the other hand, SNMP is not efficient. Many production devices have been affected by high-CPU issues caused by SNMP. Although modern devices have powerful CPUs, SNMP is still taxing and involves longer delays in exporting a large number of metrics.

Application Programming Interface (API)

APIs over HTTP(s) transport for programmable access to the devices are widely available and have become the recent norm. In addition to configuring the devices, the APIs can be used for monitoring as well.

On Cisco MDS switches, NX-API can be used for polling the metrics over HTTP(s) transport. The output can be in JSON or XML formats, which is easier to parse as compared to the raw command-line output, although NX-API support exporting the raw output also. Unlike exporting the command-line output via a SSH session, NX-API reduces the polling delay and thus enables monitoring at a fine granularity.

Refer to the Cisco MDS 9000 Series Programmability Guide for more details. The MTM app explained earlier uses NX-API for exporting metrics from the MDS switches. Refer to the MTM repository on GitHub for examples.

Streaming Telemetry

Streaming telemetry is a relatively newer mechanism for exporting a large amount of data efficiently, with minimal load on the control-plane CPU and minimal bandwidth consumption on the management port/network. This section explains its use on Cisco MDS switches.

While other export mechanisms (SSH sessions, SNMP MIB polling, and API) pull the metrics from the switches, streaming telemetry pushes the metrics from the switches to a configured destination at a configured interval. Cisco MDS uses gRPC transport and encodes the data using Google Protocol Buffers (GPB). The GPB encoding can be based on key-value (KV), or it can be much more efficient by using compact-GPB, which enables transporting data in binary format between the switches and the telemetry receiver. Instead of sending the full names (KV), the switches and the receiver agree on a naming scheme using a proto file for transporting minimal information between them. Because of these reasons, streaming telemetry is an extremely efficient export mechanism.

On Cisco MDS switches, streaming telemetry was initially added for exporting millions of I/O flow metrics that are collected by SAN Analytics. Refer to Chapter 5 for more details. Later, streaming telemetry was enhanced to export the switchport and transceiver (SFP) metrics as well.

Although technically streaming telemetry is superior to the other export mechanisms explained earlier, it has the following operational challenges.

1. It requires a custom-developed receiver to receive metrics from the switches, which most users may not be able to develop.
2. Most monitoring platforms do not natively support streaming telemetry.
3. It requires a relatively bigger configuration change on the switches.
4. Because being relatively newer, it cannot be used with devices running older software (NX-OS).

Recommendations

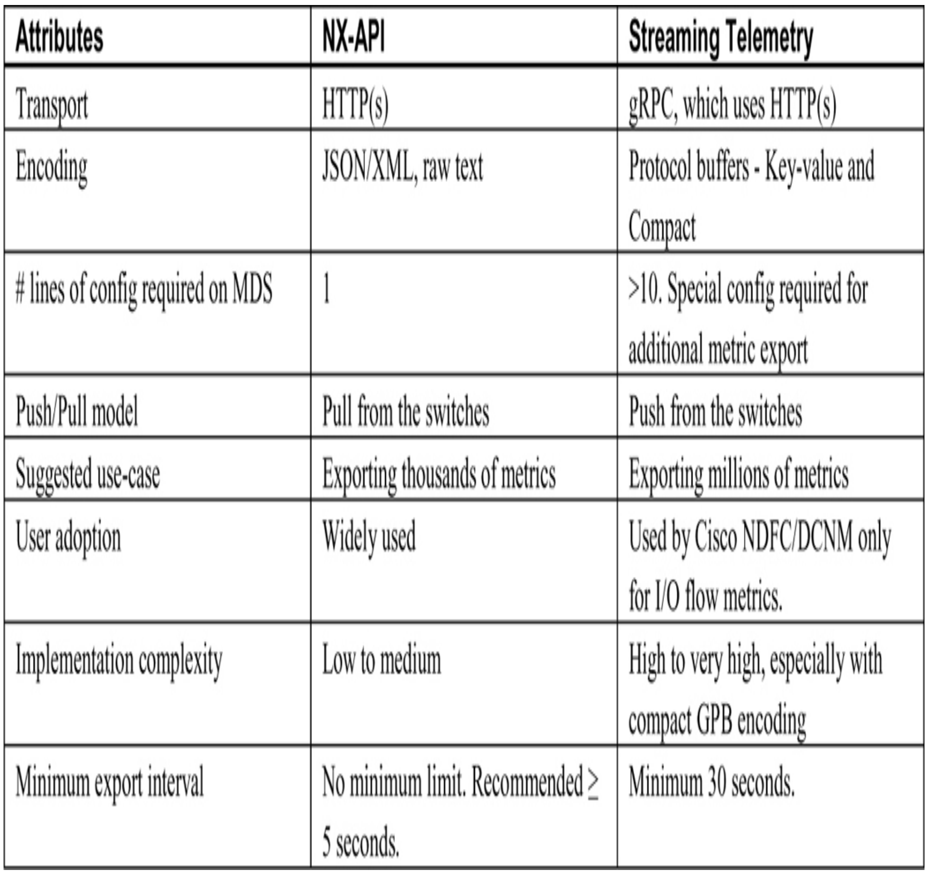
For newer developments, use NX-API on Cisco MDS switches (or similar APIs on other devices) because they are fast, easy to use, widely adopted, and require only a single-line configuration on the switches.

Legacy environments may continue using SNMP polling or parsing the command-line output via SSH sessions. But these mechanisms are slow and taxing on the switches. Although re-writing existing platforms/applications may not be worth the effort, prefer using NX-APIs for future enhancements.

Streaming telemetry is best used only when exporting millions of metrics, such as for exporting I/O flow metrics collected by SAN analytics on Cisco MDS switches. Although it is the most efficient export mechanism, rewriting existing platforms/applications may not be worth the effort. Also, for exporting only thousands of metrics, performance gain of streaming telemetry over NX-API is not visible.

Table 3-5 compares NX-API and streaming telemetry on Cisco MDS switches based on the facts at the time of this writing.

Table 3-5 *NX-API and streaming telemetry on Cisco MDS switches*



Attributes	NX-API	Streaming Telemetry
Transport	HTTP(s)	gRPC, which uses HTTP(s)
Encoding	JSON/XML, raw text	Protocol buffers - Key-value and Compact
# lines of config required on MDS	1	>10. Special config required for additional metric export
Push/Pull model	Pull from the switches	Push from the switches
Suggested use-case	Exporting thousands of metrics	Exporting millions of metrics
User adoption	Widely used	Used by Cisco NDFC/DCNM only for I/O flow metrics.
Implementation complexity	Low to medium	High to very high, especially with compact GPB encoding
Minimum export interval	No minimum limit. Recommended ≥ 5 seconds.	Minimum 30 seconds.

Besides other attributes, pay attention to the minimum export interval, which cannot be configured lower than 30 seconds for streaming telemetry at the time of this writing. In contrast, NX-API can poll the switches at a lower granularity. Some production environments monitor a fully loaded Cisco MDS 9718 with 768 ports every 10 seconds using the MTM app, which uses NX-API. While monitoring fewer ports, the polling interval can be reduced even further as long as the previous poll completes before the next one starts. A lower polling interval is especially useful when investigating only a few congested ports. Near real-time streaming graphs that update every few seconds make congestion monitoring much more visually powerful.

The Pitfalls of Monitoring Network Traffic

A key consideration for monitoring traffic in lossless networks is that for all practical purposes, treat occurrences of full or high utilization the same as over-utilization. This is because in our experience, most occurrences of full utilization and high utilization have also shown some duration of over-utilization. To understand this in details, refer to Chapter 1, the sections on *Defining Full Utilization vs High Utilization vs Over-Utilization* and *Monitoring Full Utilization vs High Utilization vs Over-Utilization*.

Next, this section explains pitfalls for monitoring network traffic in production networks using GUI platforms, applications, or scripts. The first point is about calculating the percentage utilization, which is specific to Fibre Channel ports. The second point about the average and peak utilization applies to all types of network ports (Fibre Channel or Ethernet).

Percentage Utilization of Fibre Channel Ports

For calculating the percentage utilization of a Fibre Channel port, use the bit rate instead of its advertised speed. As Chapter 2, section *Fibre Channel Data Rate* explains, the advertised speed of a FC port is higher than its bit rate. Consequently, using the advertised speed instead of the bit rate will never result in more than 90% utilization, which is inaccurate and falsely reduces the severity of the problem. For example, the bit rate of a 32GFC port is 28.025 Gbps. If 32 instead of 28.025 is used in the calculation, the percentage utilization will never exceed 89%. Because high utilization is a key indication of congestion, failing to use the bit rate of an FC port for calculating its percentage utilization is a common reason for many users to miss this indication.

This problem is more deep-rooted in custom-developed remote monitoring platforms, applications, or scripts because their developers may not be aware of these intricacies of Fibre Channel. On the other hand, the users of these platforms/applications (typically network, storage, or compute teams) may either be unaware themselves or may be under the false impression that the platforms/application would have accounted for the difference in Fibre Channel advertised speed and bit rate. If such monitoring platforms/applications are configured to alert when a port exceeds 90% utilization, Fibre Channel ports will never generate this alert. In production environments, this issue (bug) becomes difficult to detect.

A simple solution to this problem is to multiply by 0.85 (or 0.9) the calculated percentage utilization of Fibre Channel links based on the advertised speed (such as 32GFC). To know why, refer to Chapter 2, section on *Fibre Channel Bit Rate*. This value can be obtained after dividing the bit rate by the advertised speed (28.025 / 32 = 0.87). Although the value may vary slightly for different speeds, a consistent multiplication factor of 0.85 is a simpler solution.

Average and Peak Utilization

The reported utilization of a network port is typically an average value over a poll interval. When calculated natively on network devices, the interval is short, whereas the interval is longer on remote monitoring platforms. This first level of averaging cannot be avoided because the network ports typically capture ingress and egress bytes, and link utilization is indirectly calculated by taking a delta over an interval. In general, the aim should be to limit the averaging over long durations because excessive averaging causes occurrences of high utilization to be missed.

Some devices may support advanced telemetry for microburst detection, but their details are outside the scope of this writing.

This first level of averaging is generally known to the users, but sometimes a second level of averaging may go unnoticed leading to false conclusions.

When monitoring link utilization on GUI platforms/applications, be aware that the values may be aggregated before being plotted in a graph. This is because a graph can only show a limited number of data points. When a utilization graph is plotted for a long duration, the number of data points exceeds what the graph can show. So multiple data points are aggregated using a function and the result of this function is plotted instead. Generally, average (mean) or peak (max) functions are used to create the aggregates. When using the average function, the problem severity may be reduced. For example, refer to Figure 3-16, which shows the top-10 utilized ports in a production environment. For 1 hour, the peak utilization of the topmost utilized port was 93.1%, whereas the average utilization was only 24%.

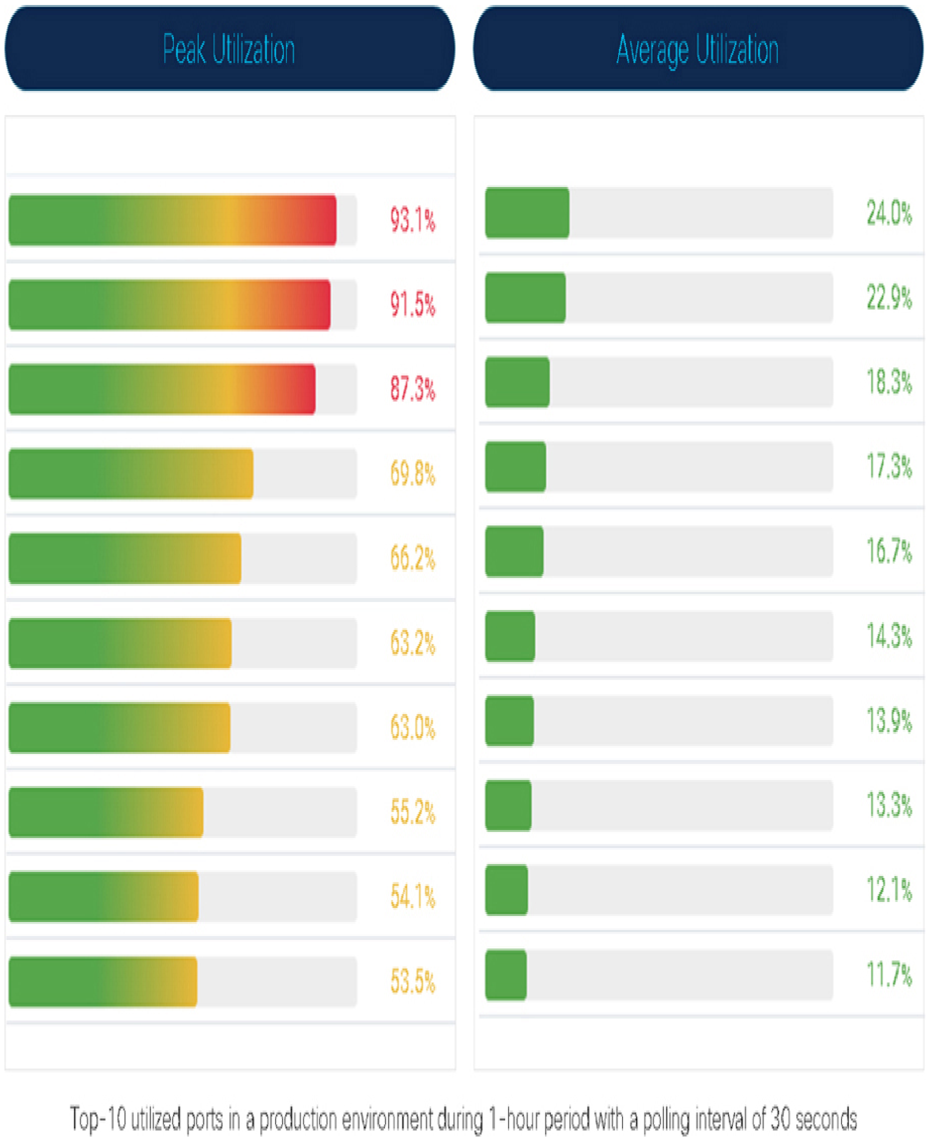


Figure 3-16 *Difference in peak and average utilization of the same ports*

Figure 3-17 shows the utilization of the same port during a week. The peak utilization values generate two alerts for greater than 80% utilization. On the other hand, the utilization of the same port does not generate any alert with average values. Because of using a longer duration (1 week), the data points are aggregated to fit in the graph. While the data points are the same, the outcome of aggregates created using the peak function is different from the aggregates created using an average function.

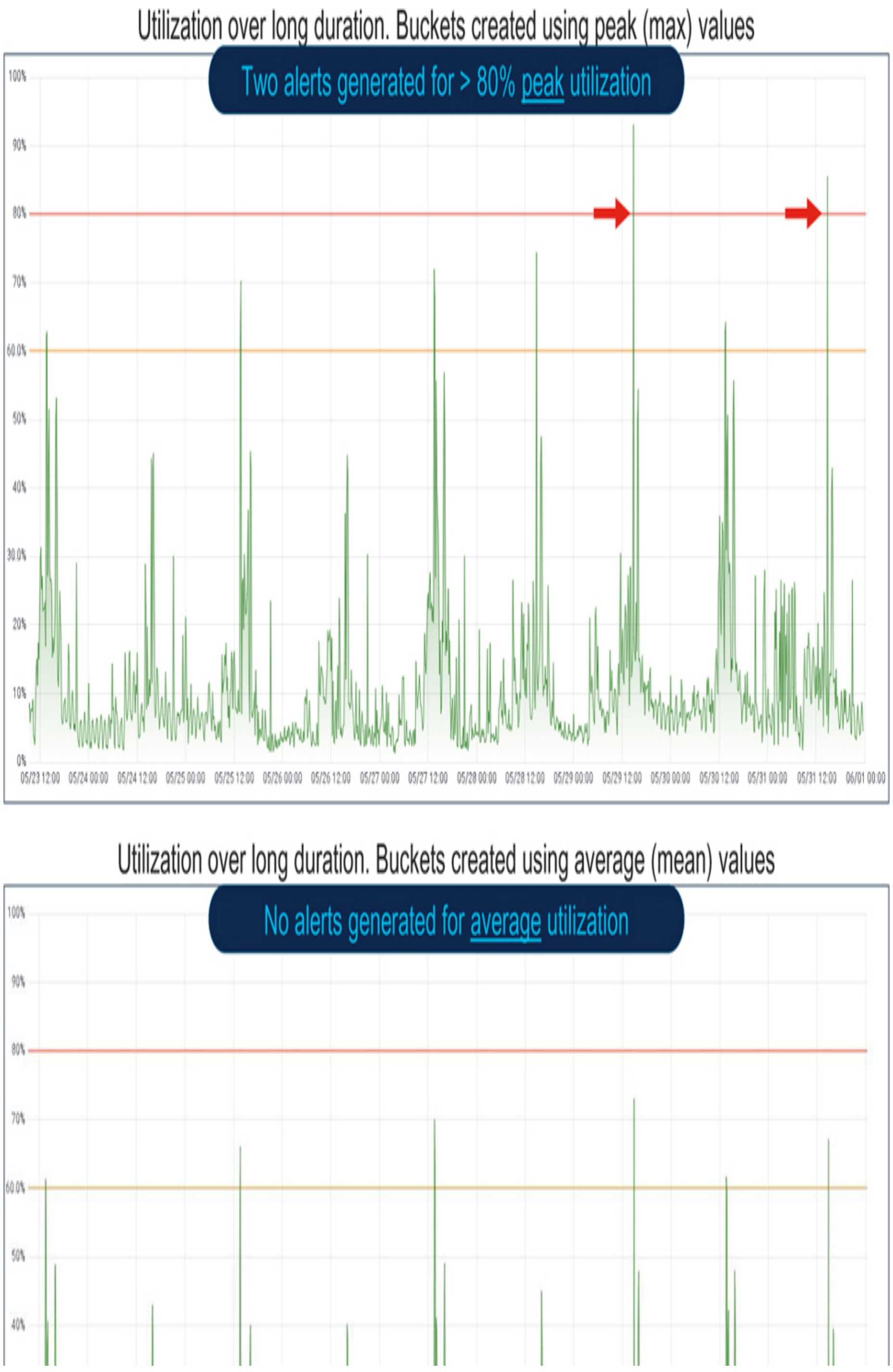


Figure 3-17 *Different peak and average utilization over long duration*

Figure 3-18 shows the utilization of the same port for two days when Figure 3-17 shows the first alert. Because of fewer data points, no aggregates are created this time. As a result, the peak and the average utilization graphs are identical and both result in generating the same alert.

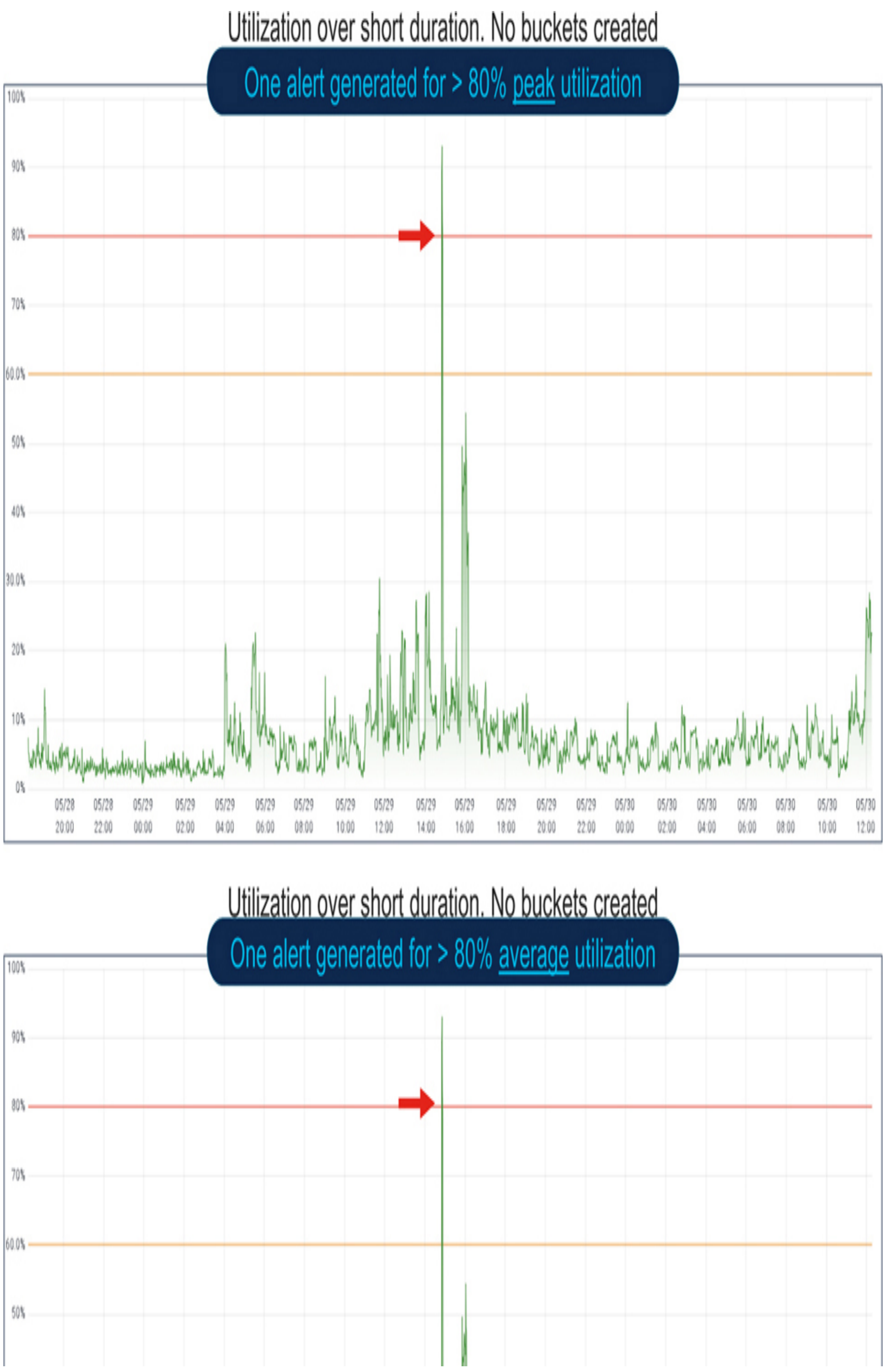


Figure 3-18 *Same peak and average utilization over a short duration.*

As evident from Figure 3-17 and 3-18, the different outcomes of peak and average functions may be unnoticed at a short duration (2 days), whereas the results are misleading at a longer duration (1 week).

Between the peak and the average utilization, one may not be necessarily better than the other. Both have their use cases. While peak values capture all the events, some users do not want to be alarmed about the peak values, especially when these events are already being investigated. Other users prefer to keep the peak-utilization events for themselves to investigate, whereas for reporting to higher management, they prefer using average utilization.

To sum up, GUI network monitoring platforms/applications may show average and/or peak utilization. Both have their use cases. You should know which ones you see. Do not be misled by average values.

Detecting Congestion due to Slow-drain and Over-utilization

The following are the best practices and our recommendations for detecting congestion on Cisco MDS switch.

- We recommend enabling Port-Monitor on Cisco MDS switches for detecting congestion automatically. Use the thresholds as explained earlier in the Port-Monitor sections.
- We recommend enabling congestion (slow-drain) analysis on NDFC/DCNM. Schedule it to run daily for 24 hours, essentially monitoring the fabrics in an always-on fashion.
- We recommend sending the alerts from the MDS switches to NDFC/DCNM or a similar platform that's integrated with your workflow. Sending the alerts from the switches is of little value if nobody is watching them.
- The **show tech-support slowdrain** command on Cisco MDS switches gather all congestion-related commands. We recommend collecting its output from all the switches

in the fabric that are suspected of in the path of congestion.

In addition to the always-on monitoring of the switches using congestion (slow-drain) analysis, NDFC/DCNM can also receive the Syslog and SNMP traps from the switches. When Port-Monitor counters on the Cisco MDS switches exceed thresholds, the alerts can be seen in the Event Analytics feature in NDFC. Refer to the documentation for setting it up correctly. We recommend enabling all these features together and using them in a typical congestion detection and troubleshooting workflow. Use the alerts generated by Port-Monitor on the switches as a starting point. Then, use the NDFC congestion analysis for fabric-wide correlation and trends.

Example 3-25 shows Port-Monitor syslog alert indicating that the device that connects to fc6/21 on this switch caused slow-drain on May 11, 2021, at 21:07:42.

Example 3-25 *A Port-Monitor alert showing the source of congestion due to slow-drain*

```
2021 May 11 21:07:42 MDS9706-C %PMON-SLOT6-4-
RISING_THRESHOLD_REACHED_WARNING:
TXWait has reached the rising threshold (port=fc6/21
[0x1294000], value=61%) .
```

Example 3-26 shows that the MDS switch captured this slow-drain event in the OBFL with time and date stamps.

Example 3-26 *show logging onboard txwait showing the source of congestion due to slow-drain*.

```
MDS9710# show logging onboard txwait
<snip>
---------------------------------
Notes:
     - Sampling period is 20 seconds
     - Only txwait delta >= 100 ms are logged


-----------------------------------------------------
-------------------------
```

```
| Interface | Delta TxWait Time      | Congestion |
Timestamp                  |
|           | 2.5us ticks | seconds |            |
|
----------------------------------------------------
-------------------------
|   fc6/21   |    4880000 |    12    |      61%   |
Tue May 11 21:07:42 2021 |
```

Example 3-27 shows a Port-Monitor syslog alert indicating that the device that connects to fc6/21 on this switch might have caused congestion due to over-utilization on September 8, 2021, at 08:58:00.

Example 3-27 *A Port-Monitor alert showing the source of congestion due to over-utilization.*

```
2021 Sep  8 08:58:00 MDS9706-C %PMON-SLOT6-4-
RISING_THRESHOLD_REACHED_WARNING: TX
Datarate has reached the rising threshold
(port=fc6/21 [0x1294000],
value=9423991900) .
```

Example 3-28 shows that the MDS switch captured this high-utilization (and a possible congestion) event in the OBFL with time and date stamp.

Example 3-28 *show logging onboard datarate showing high utilization*

```
MDS9706-C# show logging onboard datarate module 6
<snip>
----------------------------------------------------
--------------------------
| Interface | Speed |     Alarm-types      |   Rate
|       Timestamp        |
----------------------------------------------------
---------------------------
|  fc6/21   |  8G |  TX_DATARATE_RISING  |   98%   |
Wed Sep  8 08:58:00 2021 |
```

As Chapter 2 explains, finding the source of congestion is possible only with the edge switch that directly connects to the culprit device. On the ISL ports (core ports), both slow-drain and over-utilization events result in Tx B2B credit unavailability which may trigger alerts on the core ports. Such alerts should be time matched with the alerts generated by the edge ports on the downstream switches.

Sometimes, alerts may not be generated because the actual value may not exceed the configured thresholds. Use the historic values from the OBFL on the MDS switches or a remote monitoring platform, such as NDFC/DCNM or MTM, for finding the real source of congestion.

To sum up, a slow-drain device triggers alerts on metrics that detect Tx B2B credit unavailability such as TxWait, Tx-credit-not-available, credit-loss-reco, and Tx-slowport-oper-delay (if enabled), but does not trigger alerts on Tx-datarate and Tx-datarate-burst. In contrast, a device that causes congestion due to over-utilization of its link triggers alerts on Tx-datarate and/or Tx-datarate-burst, but does not trigger alerts on the Tx B2B credit unavailability metrics.

Slow-drain and Over-utilization at the Same Time

An end device cannot cause slow-drain and over-utilization at the same time. This is because slow-drain is caused when an end device tries to reduce the rate of ingress traffic. In contrast, over-utilization happens when its link is highly utilized in the ingress direction and the switchport still has more traffic to be sent. Therefore, these two conditions cannot occur simultaneously.

On Cisco MDS switches, slow-drain is primarily detected by TxWait, and over-utilization is detected by Tx-datarate. TxWait can be measured as a number of seconds in a given monitoring interval and can also be converted into a percentage. For example, 3 seconds of cumulative TxWait in a 10-second monitoring interval means that a switchport was unable to transmit frames for 3 seconds in that 10-second monitoring interval due to zero remaining-Tx-B2B-credits. This makes the TxWait percentage 30%. During the same 10-second interval, the average egress utilization (measured by Tx-datarate) cannot be more than 70%.

In rare cases, TxWait and Tx-datarate may seem to occur or even alert at the same time. This is mostly due to the monitoring or alerting granularity. For example, the Port-Monitor feature on Cisco MDS switches can alert on TxWait at a minimum granularity of 1 second, whereas the minimum granularity for Tx-datarate is 10 seconds. It is possible that, in a 10-second monitoring interval, a port observes high egress utilization for initial 9 seconds resulting in the Tx-datarate alert. However, the following 1-second monitoring interval observes slow-drain resulting in TxWait alert, which would match the timestamp of the Tx-datarate closely. This is just an example to explain the concept. As mentioned, such cases should be rare and when observed are mostly due to the monitoring granularity. Chapter 4, Case Study 1 explains one such anomaly, which was initially suspected to have TxWait and Tx-datarate at the same time, but careful investigation of the logs proved that wasn't happening.

Detecting Congestion on Long-Distance Links

Typically, replication and backup traffic flow through long-distance links between primary and disaster-recovery sites. Native Fibre Channel links may be used for a few hundred kilometers. Fibre Channel over IP (FCIP) links can go even longer distances, for example between continents. These long-distance links may be owned by 3rd parties and may not be as healthy as the intra-data center links. When using FCIP, the WAN service provider may not guarantee the assigned capacity, or the devices may try to send more traffic than the capacity of the WAN links. In all such cases, congestion on long-distance links results in backpressure towards the source of the traffic.

Detecting congestion on long-distance FC links is the same as explained throughout this chapter.

The only difference for long-distance FC links is that they may not have enough credits for the distance, speed, and average frame size of the traffic that flows on the link. The distance of the link should be already known (although in certain DWDM environments, the actual distance may be longer than mentioned), and the speed does not change while the link is operational. However, the frame size may be variable. Cisco MDS switches report bytes/second and frames/sec. Dividing the first counter by the second gives the average frame size on that link. To account for practical variants, increasing the number of credits by an extra amount is recommended.

Refer to Chapter 2 section on *Lack of B2B Credits for the Distance, Speed, and Frame Size of ISL* for more details.

Chapter 4 explains a case study when a lower than needed number of B2B credits was the cause of congestion.

Detecting and troubleshooting congestion on FCIP links is explained in Chapter 8, the section on *Fibre Channel over TCP/IP (FCIP)*.

Summary

This chapter explains a congestion detecting workflow that includes finding the culprits, victims, cause, time, location, and direction of congestion.

Next, this chapter provides an overview of the congestion detecting metrics on Fibre Channel switches with deep-dive examples from Cisco MDS switches. The most important metrics are percentage TxWait, timeout drops, and the number of credit loss recovery attempts. For detecting congestion natively on the MDS switches, use the OBFL command that displays events with date and time stamps. For detecting congestion proactively, configure the alerts on the congestion metrics, such as those generated by the Port-Monitor feature on MDS switches.

The metrics can be exported from the switches to remote monitoring platforms, such as Cisco NDFC/DCNM, for intuitive GUI, topology views, trending, seasonality, etc.

Further, this chapter explains the use cases of the MDS Traffic Monitoring (MTM) app in detecting congestion. While using such custom-developed platforms, applications, or scripts, be aware of the pitfalls such as using the Fibre Channel advertised speeds instead of the bit rate and the use of average values instead of the peak values.

For exporting metrics from network devices, SNMP is a standard-based and widely adopted approach, whereas the use of APIs has become the recent norm. Streaming telemetry is a relatively newer mechanism for exporting millions of metrics.

The focus of this chapter is on Fibre Channel transport. But the same congestion detection mechanisms apply to lossless Ethernet storage networks as well. The metrics are similar, although the implementation may be different (R_RDY versus pause frames). Chapter 7 explains lossless Ethernet networks, but instead of repeating everything, we will refer to this chapter for the same basic concepts.

References In This Chapter

- INCITS 488-2016, FC-FS-4, Fibre Channel Framing, and Signaling – 4
- INCITS 545-2018, FC-FS-5, Fibre Channel Framing, and Signaling – 5
- INCITS 512-2015, FC-PI-6, Fibre Channel Physical Interfaces – 6
- INCITS 533-2016, FC-PI-6P, Fibre Channel Physical Interfaces - 6P
- INCITS 479-2011, FC-PI-5, Fibre Channel Physical Interfaces – 5
- IEEE 803.2-2018, IEEE Standard for Ethernet
- Cisco Slow-Drain Device Detection, Troubleshooting, and Automatic Recovery White Paper

- Cisco MDS 9000 NX-OS Software Configuration Guides
- Getting Started with Monitoring and Alerting in a Cisco SAN Fabric White Paper
- Cisco NDFC-SAN Controller Configuration Guides
- The Cisco MDS Traffic Monitoring (MTM) app – https://github.com/paregupt/mds_traffic_monitor
- SAN Congestion! Understanding, Troubleshooting, Mitigating in a Cisco Fabric - BRKSAN-3446, Cisco Live 2017, Las Vegas
- DCNM SAN Insights - Next Generation Network Visibility - BRKDCN-2271. Cisco Live 2019, San Diego.
- DCNM SAN Insights - Next Generation Network Visibility - BRKDCN-3645. Cisco Live 2022, Las Vegas.
- Detecting, Alerting, Identifying and Preventing, SAN Congestion - BRKDCN-3241, Cisco Live 2022, Las Vegas.
- Cisco SNMP Object Navigator: https://snmp.cloudapps.cisco.com/Support/SNMP/do/BrowseOID.do?local=en
- Cisco MDS 9000 Series Programmability Guide, Release.
- gRPC - A high performance, open source universal RPC framework https://grpc.io/
- Cisco MDS 9000 Series SAN Analytics and SAN Telemetry Streaming Configuration Guide

Chapter 4. Troubleshooting Congestion in Fibre Channel Fabrics

This chapter covers the following topics:

- Congestion troubleshooting methodology and workflow.
- Hints and tips for troubleshooting congestion.
- Cisco MDS NX-OS commands for troubleshooting congestion.
- Case studies demonstrating troubleshooting in production environments.

Troubleshooting Methodology

Chapter 2, "Understanding Congestion in Fibre Channel Fabrics," and Chapter 3, "Detecting Congestion in Fibre Channel Fabrics," explain congestion and its detection metrics. The next questions are 'Where to begin?' or even 'How to start the investigation?'

Remember this quote:

"A problem well stated is a problem half solved" (Charles Kettering).

If you can define the problem, you are well on the way to solving it. Having an incorrect or inadequate problem description wastes time and effort.

The next sections help in defining the problem, associated levels of performance degradation, and relate them to congestion severities.

Congestion Severities and Levels

As Chapter 3 explains, congestion in Fibre Channel fabrics can have the following severity levels (Figure 4-1)

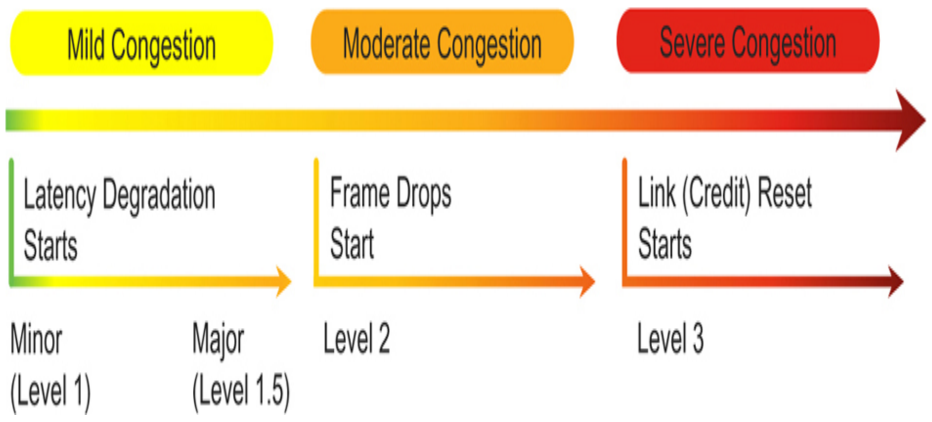


Figure 4-1 *Congestion severities and levels in Fibre Channel fabrics*

1. **Mild Congestion (Level 1 and Level 1.5)**: Increased latency but no frame drops and no Link Reset Protocol.
2. **Moderate Congestion (Level 2)**: Increased latency and frame drops but no Link Reset Protocol
3. **Severe Congestion (Level 3)**: Increased latency, frame drops, and Link Reset Protocol.

Mild Congestion (Level 1 and Level 1.5)

During mild congestion, frames are buffered in a switch longer than the typical port-to-port switching latency, which increases the I/O or Exchange Completion Time (Refer to Chapter 5, "Solving Congestion by Storage I/O Performance Monitoring," for more details about ECT). It can be caused by a lack of remaining-Tx-B2B-credits (measured by TxWait on Cisco MDS switches) or Over-Utilization (measured by Tx-datarate on Cisco MDS switches) at egress ports. Both conditions lead to congestion spreading resulting in ingress congestion on the ingress port that receives traffic from the upstream devices to be sent out of the congested port on this switch and egress congestion on the transmitting port on the upstream device.

Since even well-performing fabrics have some TxWait and maybe even some high Tx-datarate and fabrics can have thousands of ports, a good rule of thumb is to categorize percentage TxWait instances below 30% as minor and 30% or higher as major. Because of this reason, mild congestion is sub-categorized as:

- **Level 1**: Relatively minor latency problems. (Measured by TxWait < 30% on Cisco MDS switches)
- **Level 1.5**: More severe latency problems but still no I/O errors due to frame drop. (Measured by TxWait >= 30% on Cisco MDS switches)

The TxWait threshold of 30% separating minor from major is not a scientifically derived value. It is just a value that we, the authors of this book, have used over many years to separate the most serious indications of congestion from the less serious. This sub-categorization has proven to be effective in most production environments. If the applications have a smaller tolerance for congestion then a lower value, such as 20%, could be used.

Different ports in a fabric may have different amounts of transmit and receive delays. When investigating latency degradation problems, look for the Level 1.5 problems (TxWait >= 30%) and focus on them first followed by Level 1 problems. Having the Level 1.5 sub-categorization allows the ability to focus on the more serious delays.

Moderate Congestion (Level 2)

During moderate congestion, frames remain within a switch for an extended duration. But instead of waiting forever, the switch drops the frames after an amount of time. This time is called Frame Discard Timeout Value (F_D_TOV) in the Fibre Channel standards. Cisco MDS switches call it congestion-drop timeout. By default, it is enabled at 500ms. Refer to Chapter 6, “Preventing Congestion in Fibre Channel Fabrics,“

the section on *Congestion Recovery by Dropping Frames* for more details.

In storage networks, whenever frames are discarded for any reason, the exchange (I/O operation) does not complete. Typically, when the I/O operation does not complete, the initiator has its own separate timeout value of 30 - 60 seconds depending on the storage protocol stack that initiated the I/O. When this timeout expires, the I/O operation is incomplete and needs to be aborted. Only then can it be re-issued as a separate exchange. Note that there are many reasons for frame drops and only some of them are related to congestion. The application, normally, has no idea why an exchange (I/O Operation) didn't complete. It just knows that the exchange did not complete. The host records I/O errors of some kind and then re-issues the I/O operation as a separate exchange. So, when there are I/O errors, keep an open mind when investigating them. Look for frame drops due to congestion as well as all other reasons, such as CRC corrupted frames.

When packets are dropped due to congestion, they are usually preceded by increased latency in the form of high TxWait. This means that Level 2 problems (frame drops) almost certainly have lots of concurrent Level1/1.5 indications.

Unlike the sub-categorizations of Mild congestion to Level 1 and Level 1.5, we don't sub-categorize moderate congestion because even one dropped frame is a problem in storage networks.

Severe Congestion (Level 3)

During severe congestion, an FC port triggers credit loss recovery by initiating Link Reset Protocol because it is unable to send frames for an extended duration (1 or 1.5 continuous seconds) due to credit unavailability. Link Reset Protocol involves the Link Reset (LR) and Link Reset Response (LRR) primitives. This is the most serious level of congestion.

Note that the Link Reset primitive does not actually reset the link, it just resets the B2B credits on the link. When successful, the link itself does not bounce/flap. If the Link

Reset Protocol fails, it results in a link failure, which is recovered via bounce/flap.

Regardless of the success of the Link Reset Protocol, the extended delay normally causes large numbers of timeout drops. Consequently, hosts report many I/O errors and aborted exchanges, and the end devices could even logout from the fabric.

Level 3 problems contain frame drops (Level 2 indications) and latency degradation (Level 1/1.5 indications like TxWait).

Unlike the sub-categorization of Mild congestion to Level 1 and Level 1.5, we don't sub-categorize severe congestion because even one instance of credit loss recovery (Link Reset primitive) is a severe problem. However, if there are multiple instances of credit loss recovery within a certain period, that should be considered more serious. When that occurs automatically shutting down the link should be considered. This can be accomplished using the Port-Monitor feature in Cisco MDS switches as explained in Chapter 3.

Goals of Troubleshooting

The goals of troubleshooting congestion are twofold: Identify the culprits, and identify the victims.

Identify the Source (Culprits) and Cause of Congestion

The primary goal of the investigation and troubleshooting is to determine if there is congestion, the source of congestion, and the reason for congestion. This could be any of the four reasons discussed in Chapter 1, "Introduction to Congestion in Storage Networks," but usually will be either slow-drain (detected by high TxWait) or over-utilization (detected by high Tx-datarate). Once the source and cause of congestion are known, the culprit end device(s) can be investigated in detail. If the congestion is caused by long-distance ISLs then the number of credits on the ISL for the speed, distance, and average frame size must be considered.

Switchports that are directly connected to the culprit end device typically show symptoms of egress congestion.

In Figure 4-2, Host-1 (connected to switchport 1) is the source of congestion (culprit). Figure 4-2 shows that Host-1 causes congestion due to slow-drain, but the cause can also be over-utilization of its link from switchport 1.

Identify the Affected Devices (Victims)

The secondary goal is to identify the set of devices that are adversely affected by the culprit device(s). These are the victims, and they may report the same kind of performance degradation as the culprit device(s). But until the investigation is completed, you may not know if a device is a culprit or a victim. Also, the problem can only be solved by investigating the culprit(s), not the victims. Later in this chapter, the case studies illustrate this detail.

Victims can be of three types — Direct, Same-Path, and Indirect.

Direct Victims

These are the devices that are in direct communication with the culprit device. Normally, this is determined by looking at the zoning database to understand the devices communicating with the culprit devices. End-to-end flow visibility as provided by Cisco SAN Analytics can also provide this information.

Switchports that are directly connected to the direct victim end devices typically show symptoms of ingress congestion.

In Figure 4-2, Target-1 (connected to switchport 7) is a direct victim because it is sending traffic to Host-1 (connected to switchport 1), which is a culprit.

Same-Path Victims

These are the devices that are transmitting over the same congested network path as the culprit and its direct victims.

Switchports that are directly connected to the same-path victim end devices typically show symptoms of ingress congestion.

In Figure 4-2, Target-2 (connected to switchport 8) is a same-path victim because it is sending traffic to Host-2 (connected to switchport 2) via the ISL between Switch-1 (switchport a) and Switch-3 (switchport b). This ISL is congested because of Host-1, and hence congestion spread to Target-2 as well.

Indirect Victims

These are the devices that communicate with the direct victims or same-path victims. For example, if a host is a culprit, then all the targets communicating with this host are the direct victims. End devices that transmit over that congested network path are same-path victims. The devices in communication with these end devices (direct or same-path victims) are likely indirect victims. Normally, this is determined by looking at the zoning database to understand the devices communicating with the direct victims and same-path victims, or flow visibility as provided by SAN Analytics can also be used.

One key fact is that the switchports that are directly connected to the indirect victims do not have congestion indications, even though these end devices report degraded I/O performance. This is because there is no congestion specifically at this device. Its I/O performance is degraded because the other end devices (direct and same-path victims) in communication with this indirect victim are themselves being adversely affected.

In Figure 4-2, Host-2 (connected to switchport 2) and Host-5 (connected to switchport 5) are indirect victims because they are receiving traffic from Target-1 (connected to switchport 7), which is a direct victim. Host-3 and Host-6 are indirect victims because they are receiving traffic from Target-2 (connected to switchport 8), which is a same-path victim.

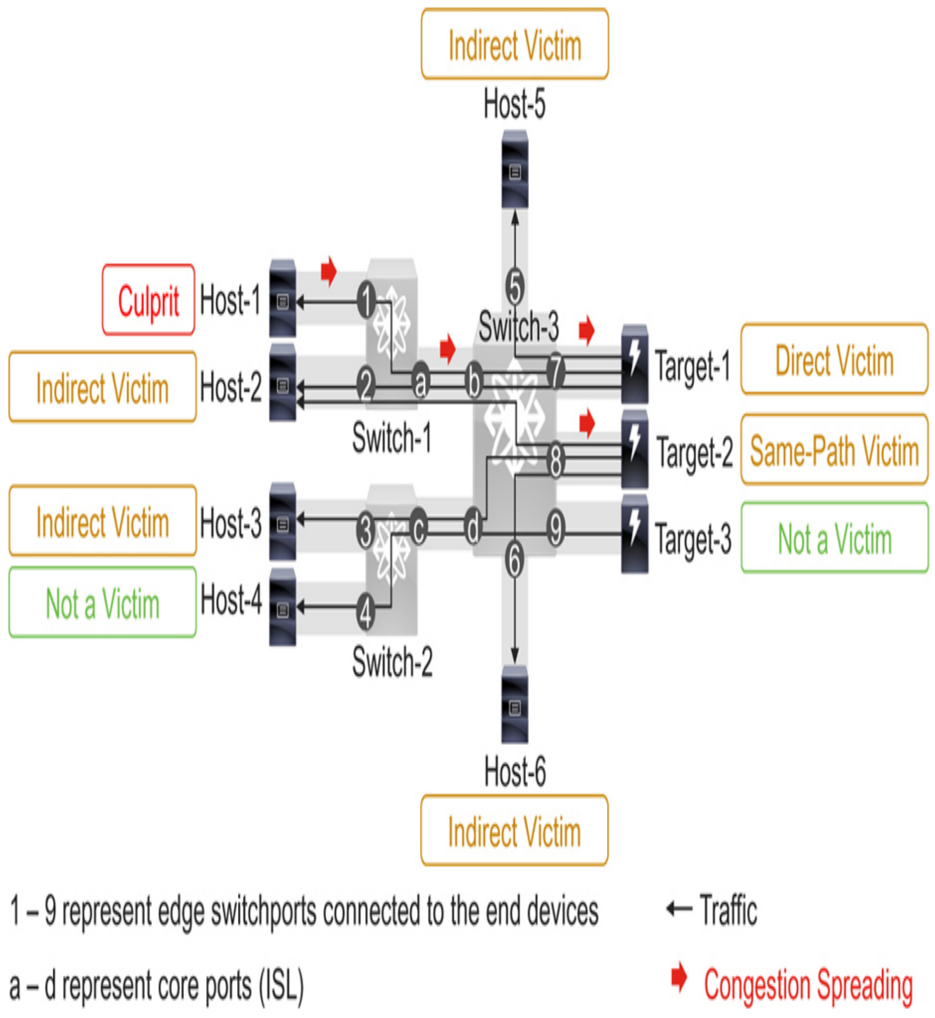


Figure 4-2 *Types of Victims — Direct, Same-Path, and Indirect*

Host-4 and Target-3 are not victims because they don't send traffic to Host-1, don't receive traffic from Target-1 and Target-2, and don't share network paths between Host-1 and Target-1.

To summarize:

- Edge switchport connected to a culprit shows symptoms of egress congestion.
- Edge switchport connected to a direct victim shows symptoms of ingress congestion. This direct victim sends traffic to the culprit.
- Edge switchport connected to a same-path victim also shows symptoms of ingress congestion. But, unlike a direct victim, a same-path victim does not send traffic to the culprit.

- Edge switchport connected to an indirect victim does not show symptoms of egress or ingress congestion.

These differences in the symptoms on the edge switchports help in identifying the connected end devices as culprits, victims, and types of victims.

Methodology

Congestion troubleshooting methodology involves two high-level steps. The first step is to detect ingress or egress congestion on a device in the decreasing severity levels. The second step is the workflow of chasing the source of congestion to the culprit.

Troubleshooting Methodology—Decreasing Severity Levels

We recommend troubleshooting congestion in decreasing levels of severity. That is, start at the highest level of severity first and if no indications of that level of severity are found, then continue to the next lower level of severity. Refer to Figure 4-3.

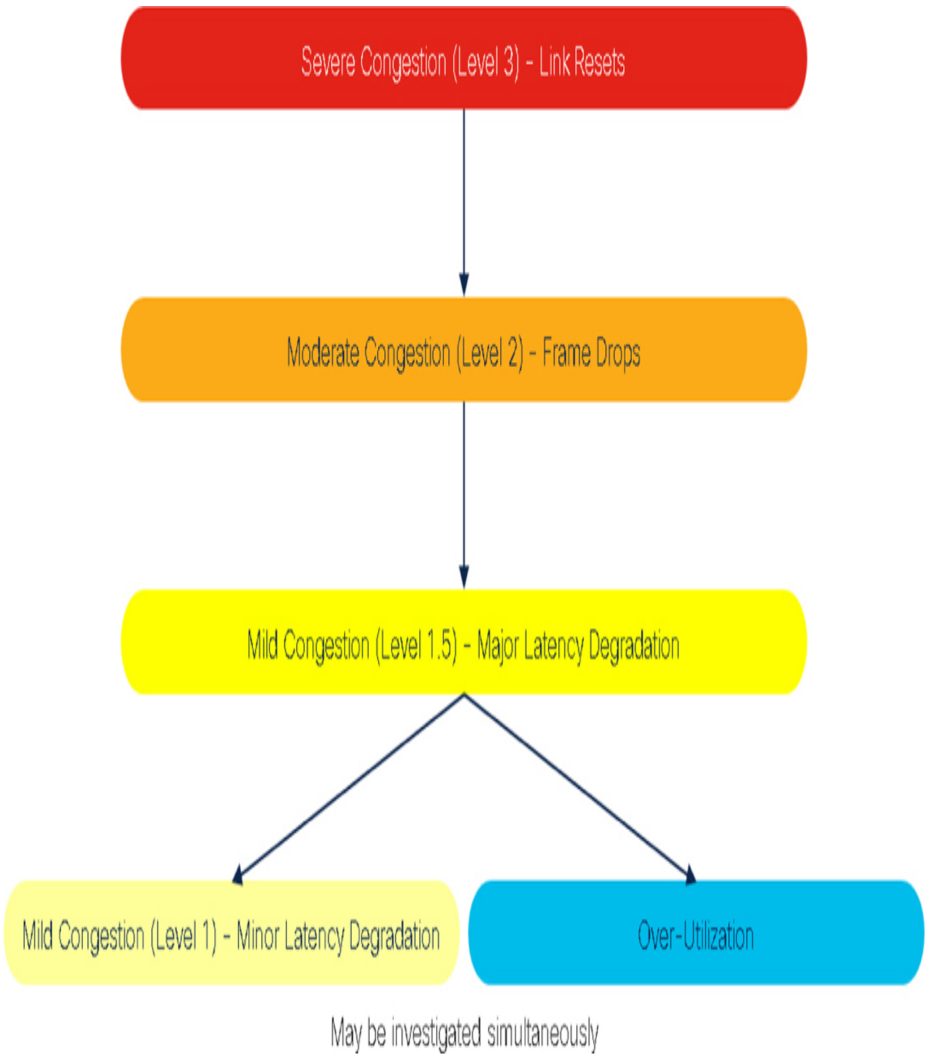

Severe Congestion (Level 3) - Link Resets
Moderate Congestion (Level 2) - Frame Drops
Mild Congestion (Level 1.5) - Major Latency Degradation
Mild Congestion (Level 1) - Minor Latency Degradation
Over-Utilization
May be investigated simultaneously

Figure 4-3 *Recommended Troubleshooting Methodology - Decreasing Levels*

There are at least two possible motivations for investigating congestion in a Storage Network:

1. An ongoing or a recent problem.
2. General evaluation of the fabric.

Both of these reasons for investigating congestion follow a similar methodology. If there is an ongoing or past problem then classify the problem (Level 1/1.5, Level 2, or Level 3) and start with the most severe indications.

If there is no ongoing or past problem being investigated and a general evaluation of the fabric is being done, then we recommend starting the investigation as if there is a severe (Level 3) problem. If there are no indications of a Level 3 problem, then look for a moderate (Level 2) problem. If no

Level 2 problem, look for indications of a mild (major Level 1.5) problem. Finally, if there are no indications of a Level 1.5 problem look for indications of a mild (minor Level 1) problem. The reason for looking for more severe (Level 3) types of problems followed by Level 2 and Level 1 problems is that more severe problems will impact the network more and can even cause widespread failures. Additionally, more severe congestion events are fewer and because of that easier to spot. In short, the more severe the problem the easier it is to detect. Finally, resolving a severe problem can address many moderate and mild problems.

Workflow for Chasing the Source of Congestion (Culprit)

This section explains the workflow of chasing congestion to its source (culprit) based on the symptoms seen (Figure 4-4). As mentioned, these symptoms may surface anywhere in a fabric, or even from a general evaluation.

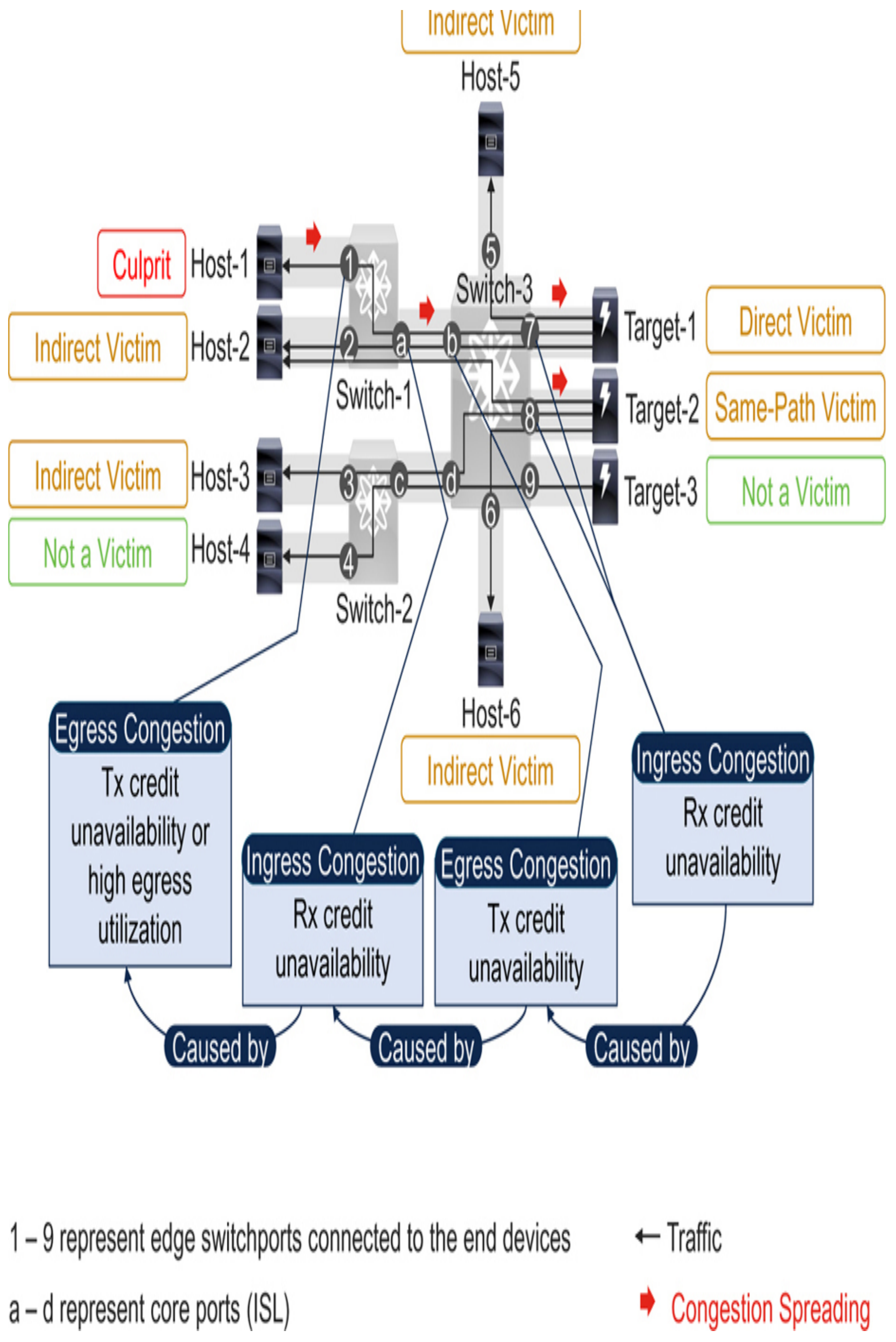


Figure 4-4 *Chasing congestion to its source to find the culprit*

The Need for a Workflow for Chasing Congestion to its Source

Before understanding this workflow, it is important to know why such a workflow is needed. A common question is, why not proactively monitor egress congestion on all the ports in a fabric, which would directly allow finding the source of congestion (culprit) instead of trying to chase it. Although we recommend proactive monitoring of congestion symptoms on all the ports, the workflow of chasing the culprit is needed for the following reasons:

1. The first point to understand is that this section explains a superset workflow. It is not mandatory to enter this workflow from the beginning. Enter this workflow where symptoms are found and follow it to find the culprit(s). Sometimes it means entering this workflow on the culprit-connected switchport, therefore directly finding the source of congestion. Other times, the workflow may start with no symptoms of congestion on the switchports (typically connected to indirect victims) and then chase the culprit.

2. Even if a culprit is found in the beginning, a workflow is still needed to correctly correlate the problem reported by a victim (such as slow I/O response) with a congestion alert by a culprit (such as TxWait alert on its connected switchport). Many victims and culprits may exist simultaneously. A workflow, as explained in this section, ensures a correct correlation between the culprit and the victims. Imagine if two host connected-switchports generate alerts almost at the same time for high egress utilization and five other hosts report slow I/O performance. How to find which victim is affected by which culprit? The workflow in this section helps in finding the correct culprit-victim relationships and, therefore helps in finding the location where a corrective action should be applied.

3. The third key point is that when congestion spreads in lossless networks, one culprit may victimize many end devices, and users of these victim devices may report performance degradation as well. Because of this reason, it is natural to start troubleshooting where a problem was reported first, which can be victim devices or their connected switchports.

4. Finally, understand that sometimes congestion problems go unnoticed despite having a robust monitoring and

alerting infrastructure. This is because the congestion severity may remain below the alerting thresholds. For example, 29% TxWait, while the alerting threshold is 30%, does not generate an alert, although it has the potential of creating victims at the same severity.

Overall, this workflow makes the congestion troubleshooting procedure repeatable, therefore increasing its effectiveness. It does not eliminate the need for proactive monitoring of switchports. As Chapter 3, the section on *How to Detect Congestion* explains, continuous monitoring enables proactive and predictive approaches for detecting congestion. But these approaches first need a foundation of reactive approaches, which is achieved by a troubleshooting workflow as explained in this section and demonstrated later using the case studies.

Where to Start the Troubleshooting Workflow?

As mentioned, it is impossible to predict on which switch or end device the investigation will start. It can be a switch connected to a culprit, somewhere in between, or even a victim. If a victim, it can be a direct victim, same-path victim, or indirect victim. Typically, an investigation starts where a problem is reported. For example, an alert generated by a switchport, an error reported by a host, or even slow I/O performance by a storage array. In most cases, it is natural to start an investigation from the switchport that generates an alert or the switchport connected to the end devices that reported a problem.

These switchports are covered under the following scenarios:

1. **Symptom of egress congestion is found on a switchport**: This switchport is in the congestion path (Figure 4-5 and Figure 4-6). Follow congestion to its source to find the culprit.

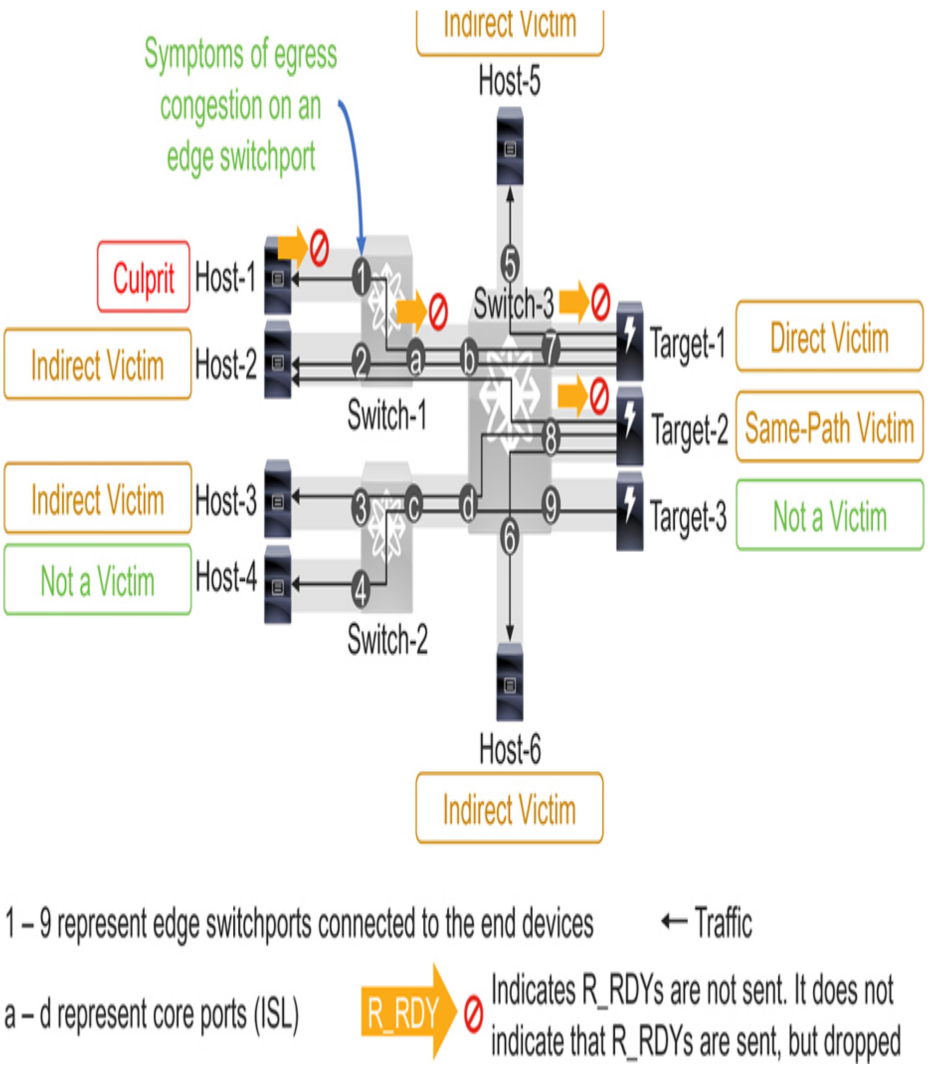


Figure 4-5 *Congestion troubleshooting workflow when egress congestion symptoms are found on an edge switchport*

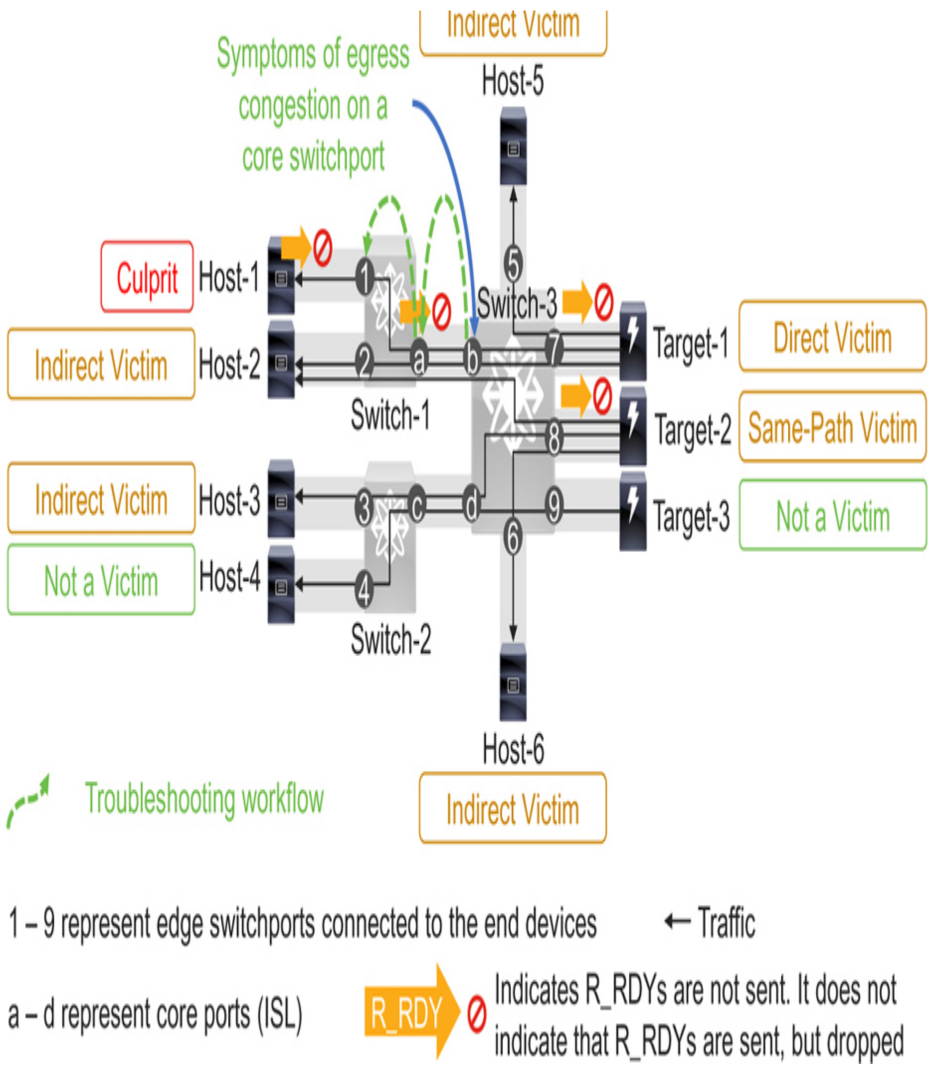


Figure 4-6 *Congestion troubleshooting workflow when egress congestion symptoms are found on a core switchport*

2. **Symptom of ingress congestion is found on a switchport**: This switchport is also in the congestion path (Figure 4-7). But this time determine the ports that are receiving traffic from this port on the same switch and look for symptoms of egress congestion on those ports. This takes to (1) above.

3. **No symptom of egress or ingress congestion is found on a switchport**: Find the devices in communication with this switchport and continue investigation there to find symptoms of egress or ingress congestion (Figure 4-4). This takes to (1) or (2) above.

The following subsections explain these scenarios in detail.

Egress Congestion Symptoms on a Switchport

Symptoms of egress congestion on a switchport indicate that this port is in the congestion path. These symptoms can be credit-loss, timeout-drops, TxWait, or high egress utilization (Tx-datarate).

In Figure 4-5, egress congestion symptoms are found on edge switchport 1. Being an edge port, switchport 1 is directly connected to the culprit end device.

Egress congestion symptoms can also be found on core switchports, such as switchport b in Figure 4-6. In these cases, follow congestion to its source by using the *Detailed Workflow* explained shortly.

Ingress Congestion Symptoms on a Switchport

The next possibility is that the switchport where the investigation starts shows symptoms of ingress congestion. In Figure 4-7, these would be switchport 7 (connected to the direct victim) and switchport 8 (connected to same-path victims).

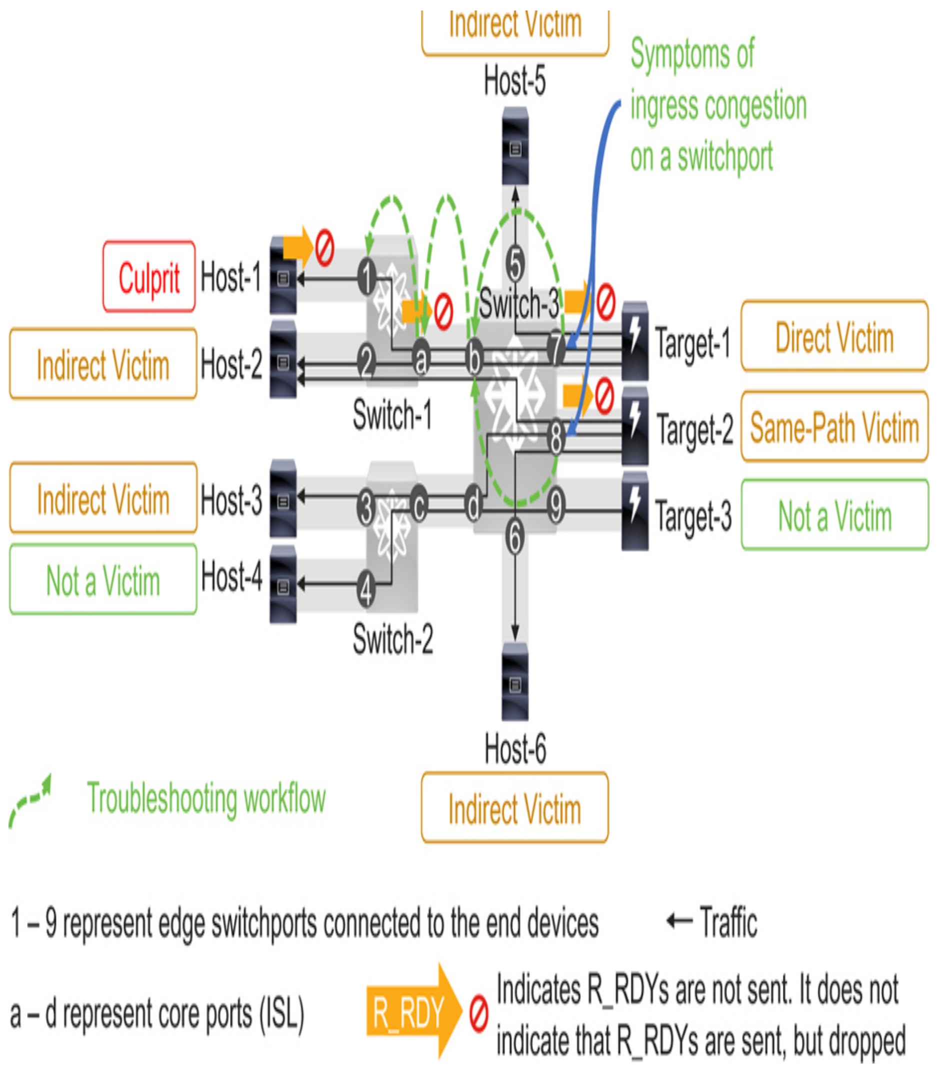


Figure 4-7 *Congestion troubleshooting workflow when ingress congestion symptoms are found on a switchport*

The symptoms of ingress congestion on Cisco MDS switches are not as prevalent as egress congestion. So ingress congestion may exist, but may not be captured or shown. Such cases can be found by egress congestion symptoms on other ports of the same switch (such as switchport b) that are receiving traffic from the switchport under investigation (such as switchport 7 and 8).

Once the ports with egress congestion symptoms are found, as mentioned, chase the culprit by using the *Detailed Workflow* explained shortly.

No Congestion Symptoms on a Switchport

The last possibility is that the switchport does not show any symptoms of egress or ingress congestion. These would be the switchports connected to the indirect victims, such as switchport 2, 3, 5, and 6 in Figure 4-8, and the core ports, such as switchport c and d.

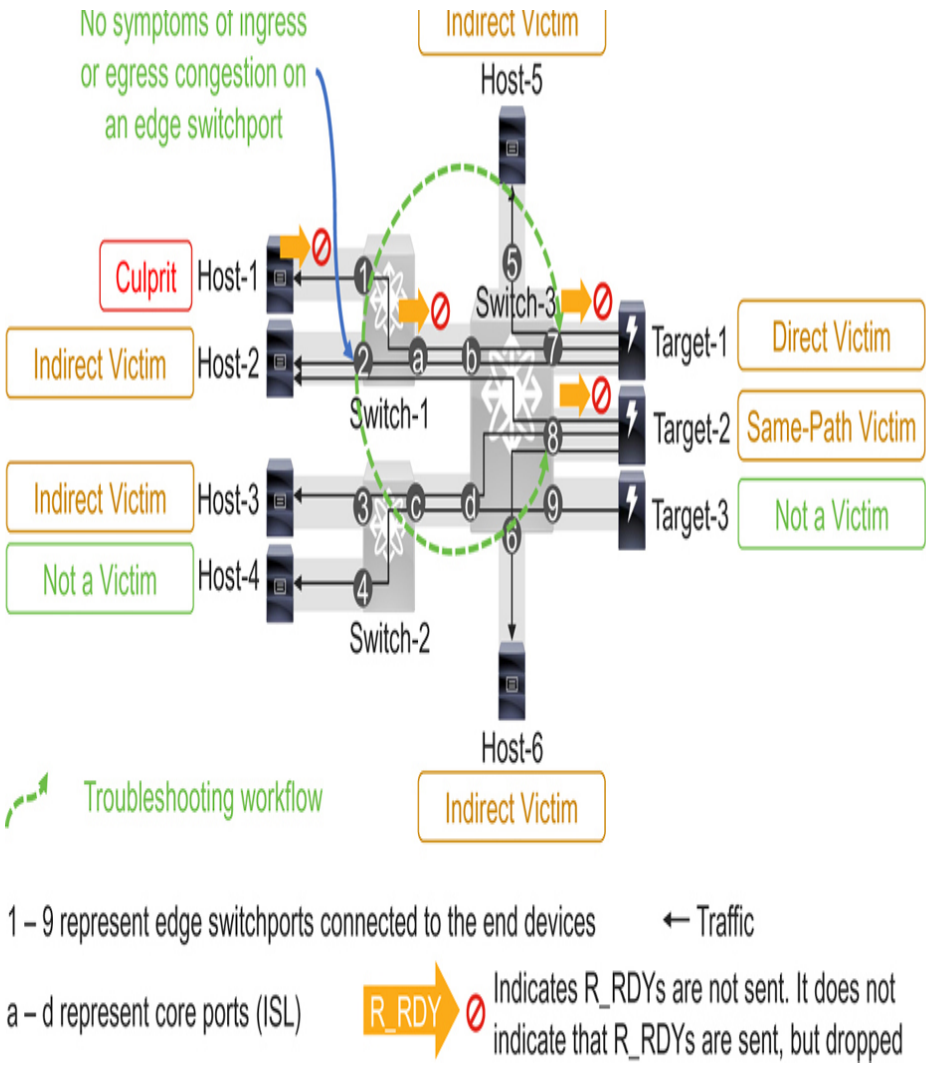

No symptoms of ingress or egress congestion on an edge switchport
Indirect Victim
Host-5
Culprit
Host-1
Indirect Victim
Host-2
Switch-1
Switch-3
Target-1
Direct Victim
Target-2
Same-Path Victim
Indirect Victim
Host-3
Not a Victim
Host-4
Switch-2
Target-3
Not a Victim
Host-6
Indirect Victim
Troubleshooting workflow
1 – 9 represent edge switchports connected to the end devices
← Traffic
a – d represent core ports (ISL)
R_RDY
Indicates R_RDYs are not sent. It does not indicate that R_RDYs are sent, but dropped

Figure 4-8 *Congestion troubleshooting workflow when no congestion symptoms are found on a switchport*

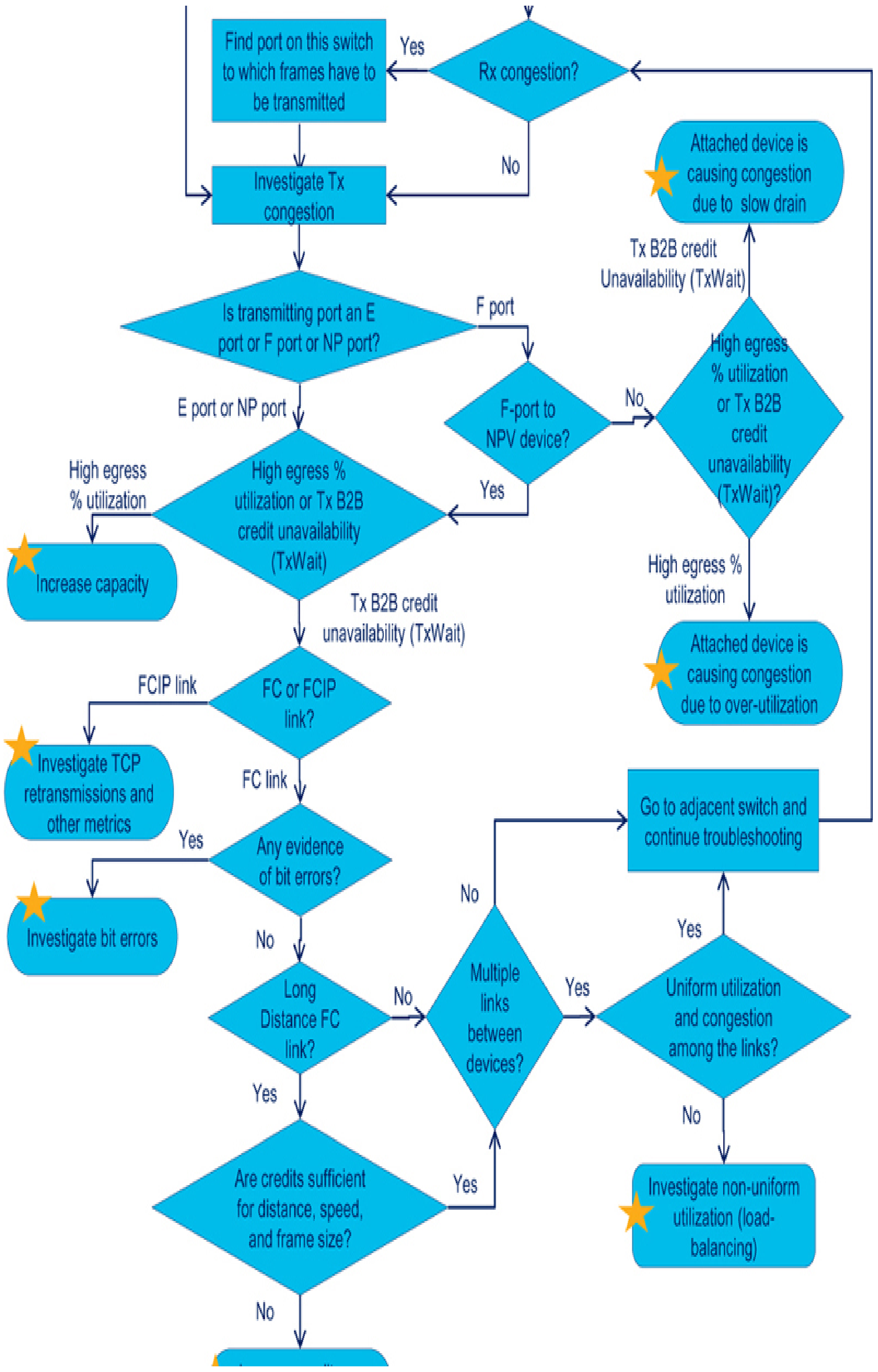
In such cases, find the end devices or switchports that are in communication via the switchports under investigation. This is typically done by using the zoning database. Flow visibility as provided by Cisco SAN Analytics can also be used on Cisco MDS switches.

Then repeat these steps until a port is found that shows symptoms of ingress or egress congestion.

Overall, the aim is to narrow the scope to the congestion path. Then, chase the culprit by using the *Detailed Workflow* explained next.

The Detailed Workflow

Figure 4-9 explains a detailed workflow for chasing congestion to its source.



Rx congestion?
Yes
Find port on this switch to which frames have to be transmitted
No
Investigate Tx congestion
Attached device is causing congestion due to slow drain
Tx B2B credit Unavailability (TxWait)
Is transmitting port an E port or F port or NP port?
F port
E port or NP port
F-port to NPV device?
No
High egress % utilization or Tx B2B credit unavailability (TxWait)?
Yes
High egress % utilization
High egress % utilization or Tx B2B credit unavailability (TxWait)
Increase capacity
Tx B2B credit unavailability (TxWait)
High egress % utilization
Attached device is causing congestion due to over-utilization
FCIP link
FC or FCIP link?
Investigate TCP retransmissions and other metrics
FC link
Go to adjacent switch and continue troubleshooting
Yes
Any evidence of bit errors?
Investigate bit errors
No
No
Yes
No
Long Distance FC link?
Multiple links between devices?
Yes
Uniform utilization and congestion among the links?
Yes
No
Are credits sufficient for distance, speed, and frame size?
Yes
Investigate non-uniform utilization (load-balancing)
No

Figure 4-9 *Congestion troubleshooting workflow*

If a switchport shows indications of ingress congestion (e.g. RxWait), such as switchport 7 in Figure 4-8, find the other ports on the same switch that communicate with it via zoning, topology, etc., and show indications of egress congestion (e.g. TxWait or high Tx-datarate), such as switchport b. Finding switchports with egress congestion is important because as explained in Chapter 3, egress congestion causes ingress congestion on switches. The reverse is not true, which means ingress congestion does not cause egress congestion.

Once a port with egress congestion is found, determine the port type.

If the port type is an E_Port (switchport 7 in Figure 4-8) then the port is an ISL to another switch. Except for the cases of insufficient B2B credits for long-distance ISLs, over-utilization of the ISLs, and perhaps credit loss due to bit errors, the ISL itself is not the source of the congestion. The source of the congestion needs to be ‘chased’ to the adjacent switch towards the destination of the traffic.

If the ISLs are aggregated into a multi-link port-channel, then most or all the port-channel members should show a similar amount of congestion indication. If the problem is seen on just a single member of the port-channel, then perhaps something else, like physical errors and not congestion, may be the cause of the problem. Check for various indications of bit errors to confirm this.

Normally, the load balancing algorithms produce fairly evenly balanced traffic when there are multiple ISLs between switches. In some cases, however, multiple links between two neighboring devices may not be uniformly utilized. When traffic spikes, one link may be over-utilized, whereas other links may be under-utilized. In such cases, the over-utilized link is the source of congestion. Since the Tx-datarate indications in the Port-Monitor feature on MDS switches are at a switchport level, this condition can be easily determined.

Once the congestion is chased to the adjacent switch (Switch-1 in Figure 4-8), then continue looking for congestion on that switch and repeat these steps. If the congestion indication on the ISL port on the local switch (Switch-3 in Figure 4-8) is TxWait yet the adjacent switch (Switch-1 in Figure 4-8) does not indicate egress or ingress congestion on any ports, this means congestion could be because of insufficient B2B credits for the ISL's speed, distance, and average frame size. This scenario is demonstrated in Case Study 4.

If the congestion indications show high percentage utilization on an ISL then the remedy is simple: increase the capacity between the switches. Do this by adding additional individual ISLs, adding additional links to the existing port-channel, or even adding an additional equal bandwidth port-channel. Any of these increases the capacity between switches eliminating the high percentage utilization.

If the egress port type is F_Port then this is an edge port—a port to an actual end device or a link to an NPV switch. If it is a F_Port to an end device (switchport 1 in Figure 4-8), the culprit is the end device itself, which should be investigated.

If it is a F_Port to an NPV switch, these links function logically as E_Ports because they are interconnections to switches and not end devices. Many times, these links are aggregated into port-channels as well. These should be treated the same as E_Ports and the investigation should continue in that adjacent NPV device. The same methodology applies, namely, all the port-channel members should be showing a similar amount of congestion indication. Chase congestion to the adjacent NPV switch until you find the F_Port with egress congestion indications, such as TxWait.

Note that when looking at credit loss, timeout-drops, and even high TxWait, once arriving at the edge switch (Switch-1 in Figure 4-8) where the culprit device(s) presumably reside, it may be that the cause is Over-Utilization (detected by high Tx-datarate) and not Slow-Drain (detected by TxWait). That is, if high TxWait is being chased from switch to switch, once arriving at the edge switch the TxWait may "disappear" in the sense that there will be no F_Ports that record TxWait. In this case, look for one or more ports with high egress utilization (detected by Tx-datarate), which indicates congestion due to over-utilization.

Since TxWait is a measure of the amount of time a port cannot transmit because it lacks Tx B2B credits, it should be clear that the higher the TxWait percentage the more severe the congestion. Therefore, TxWait of 30% and greater is more severe than TxWait less than 30%. But for high Tx-datarate, it is not so clear what makes for more severe Over-Utilization. In practice, longer periods of Tx-datarate where the average egress link utilization is 80% or greater are known to cause more problems than shorter periods of high utilization.

Hints and Tips for Troubleshooting Congestion

This section explains helpful tips for troubleshooting congestion on Cisco MDS switches.

Investigate Higher Congestion Levels First

As explained in the earlier section on *Troubleshooting Methodology— Decreasing Severity Levels*, when there are higher levels of congestion like credit loss (Level 3), it almost certainly includes lower levels of congestion indications like timeout-drops (Level 2) and TxWait (Level 1/1.5). Similarly, if there are timeout-drops (Level 2), then there are almost certainly TxWait (Level 1/1.5).

A word of warning, if the problem type indicates a Level 2 or Level 3 problem but the fabric is being investigated like it is a Level 1 or 1.5 problem, then most likely there will be lots of ports found with TxWait and it will be more difficult to track down the problem. That is because, as mentioned, Level 3 problems will almost certainly include timeout drops and a large amount of TxWait. Likewise, Level 2 problems will almost certainly include a large amount of TxWait. So, getting the problem classification correct makes the investigation more efficient.

Therefore, when initially investigating a problem first look for these higher, more severe levels of congestion before looking for less severe indications at the lower levels. Practically that means using the **show logging onboard error-stats** command to look for errors like credit loss, timeout-drops, etc. followed by the less severe indications like TxWait.

The **show logging onboard** command allows investigating specific time range using the **starttime mm/dd/yy-hh:mm:ss** and **endtime mm/dd/yy-hh:mm:ss** parameters.

The following sub-sections provide only a quick reference of the relevant counters in OBFL on Cisco MDS switches. These counters and commands are explained in detail in Chapter 3. Information from this section can be used for reactive troubleshooting. For proactive and predictive monitoring, refer to Chapter 3 for details on how these and many other counters can be exported to a remote monitoring platform, such as Cisco NDFC/DCNM.

Finding Level 3 Congestion — Credit Loss

The quickest place to look for indications of credit loss is in OBFL error-stats on Cisco MDS switches (Example 4-1). If investigating real-time, use the NX-OS command **show logging onboard starttime mm/dd/yy-hh:mm:ss error-stats | include CREDIT_LOSS**.

Example 4-1 *Credit Loss — Level 3 congestion symptoms in OBFL on Cisco MDS switches*

```
MDS9706# show logging onboard error-stats | include
CREDIT_LOSS
----------------------------------------------------
----------------------
 Interface  |                                 |
|    Time Stamp
   Range    |       Error Stat Counter Name    |
Count   | MM/DD/YY HH:MM:SS
            |                                 |
|
----------------------------------------------------
----------------------
```

```
fc6/3        |  F32_MAC_KLM_CNTR_CREDIT_LOSS   |
5057 | 11/07/22 22:32:39
fc6/3        |  F32_MAC_KLM_CNTR_CREDIT_LOSS   |
5057 | 08/03/22 18:24:11
fc6/3        |  F32_MAC_KLM_CNTR_CREDIT_LOSS   |
5056 | 08/03/22 18:23:51
```

Also, look in the logfile for ‘Link Reset failed’ via the NX-OS command **show logging logfile | include “Link Reset failed”** (Example 4-2).

Example 4-2 *Link Reset failed — Level 3 congestion symptoms in OBFL on Cisco MDS switches*

```
MDS9706# show logging logfile | i “Link Reset
failed”
2020 Dec  4 21:48:19 MDS9706 %PORT-5-
IF_DOWN_LINK_FAILURE: %$VSAN 13%$ Interface
fc1/12 is down (Link failure Link reset failed due
to timeout)
```

Another place to look for is if Port-Monitor is active and the ‘credit-loss-reco’ counter is being monitored, there would normally be SNMP alerts and Syslog messages.

Finding Level 2 Congestion — Frame Drops

Like the credit loss events, the best place to look for frame drops for a variety of reasons is also in the OBFL error-stats using the NX-OS command **show logging onboard error-stats** (Example 4-3). This has the advantage of reflecting many ‘normal’ types of frame-drop reasons like timeout-drops, offline drops (frames that hit a port that just went offline), and invalid CRCs. As with other **show logging onboard** commands the output is date and time stamped and is per-module (on director-class switches). For finding targeted timeout-drops only, use the NX-OS command **show logging onboard starttime mm/dd/yy-hh:mm:ss flow-control timeout-drops**.

Example 4-3 *Timeout drops — Level 2 congestion symptoms in OBFL on Cisco MDS switches*

```
MDS9706# show logging onboard starttime 01/02/22-
00:00:00 flow-control timeout-
drops
----------------------------------------------------
------------------------
 Interface  |                              |
|    Time Stamp
   Range    |    Error Stat Counter Name    |  Count
| MM/DD/YY HH:MM:SS
            |                              |
|
----------------------------------------------------
------------------------
 fc6/3      | F32_TMM_PORT_TIMEOUT_DROP     |
10001684 | 08/03/22 18:24:11
 fc6/3      | F32_TMM_PORT_TIMEOUT_DROP     |
9999792 | 08/03/22 18:23:51
 fc6/3      | F32_TMM_PORT_TIMEOUT_DROP     |
9926197 | 08/03/22 18:23:31
```

Another place to look is if Port-Monitor is active and the 'timeout-discards' counter is being monitored, there would normally be SNMP alerts and Syslog messages. Remember, the Cisco MDS switches refer to drops and discards interchangeably.

Finding Level 1/1.5 Congestion—TxWait and Over-Utilization

These can be investigated by OBFL using the NX-OS command **show logging onboard txwait** (Example 4-4) and **show logging onboard datarate** (Example 4-5).

Example 4-4 *TxWait — Level 1/1.5 congestion symptoms in OBFL on Cisco MDS switches*

```
MDS9706# show logging onboard starttime 01/02/22-
00:00:00 txwait
<snip>
----------------------------------------------------
-----------------------
```

```
| Intf | Virtual | Delta TxWait Time     |
Congestion | Timestamp
|      |   Link  | 2.5us ticks | seconds |
|
----------------------------------------------------
-----------------------
| fc6/3 | None   |  2393955    |    5    |      29%
| Aug 3 18:24:11 2022
| fc6/3 | None   |  2139918    |    5    |      26%
| Aug 3 18:23:51 2022
| fc6/3 | None   |  3752005    |    9    |      46%
| Aug 3 18:23:31 2022
```

Example 4-5 *Datarate — Level 1/1.5 congestion symptoms in OBFL on Cisco MDS switches*

```
MDS9706# show logging onboard starttime 01/01/23-
00:00:00 datarate
<snip>
 ---------------------------------------------------
-------------------------
| Interface |  Speed  |     Alarm-types     | Rate |
Timestamp
 ---------------------------------------------------
-------------------------
|  fc6/17   |    4G   | TX_DATARATE_FALLING | 68%  |
Fri Feb 24 20:43:18 2023
|  fc6/17   |    4G   | TX_DATARATE_RISING  | 98%  |
Fri Feb 24 20:43:07 2023
```

The greater the percentage TxWait, the more severe the impact. Similarly, the greater the Tx-datarate percentage and the longer the time of the event (time between rising and falling threshold) the more severe the impact. Again, another place to look is if Port-Monitor is active and the 'txwait' counter is being monitored, there would normally be SNMP alerts and Syslog messages.

Utilizing the show tech-support slowdrain Command

The **show tech-support slowdrain** is a single command on Cisco MDS switches that aggregates all the other commands normally necessary for troubleshooting congestion into a single output. From its output, you can find the individual commands needed for troubleshooting.

The following are best practices for collecting the **show tech-support slowdrain** command output:

1. Collect the output of this command from all the switches. This is because before investigating a congestion problem, it is impossible to determine in advance, the source and cause of congestion or even all the victims. Having the information from all the switches allows the investigation to go where the congestion indications lead.
2. Collect the outputs from all the switches in the fabric at the same time. This allows time correlation of the same events across multiple switches.
3. Periodically (such as every week) saving the output of **show tech-support slowdrain** command helps in preserving past congestion events. The OBFL cyclical buffers are overwritten as new entries are logged, if there is a high rate of new logs, OBFL may not go back that far in time. Later MDS NX-OS software releases have increased the size of the OBFL txwait buffer significantly. As a result of this change, all the congestion-related OBFL log files should normally be able to contain weeks and even months of data. But we still recommend periodically saving this output to preserve the historic events because frequent events may still result in overwriting of the older entries.

Synchronize Clocks and Consider Timings

Because congestion is a fabric-wide phenomenon, it results in various indications on multiple devices at the same time. Having time synchronization between the switches and the end devices allows various indications on one device to be correlated to the indications on another device. Time synchronization effectively ties those events together. For example, high TxWait, timeout-drops, or credit loss on a switch port at 10 PM may be caused by another switch port that reports high Tx-datarate at 10 PM and not by a different switch port that reports high Tx-datarate at 11 PM.

This correlation is necessary when determining the victims (direct, same-path, or indirect). A culprit device should only cause congestion on ports in the data path to all the devices it is zoned with (direct and same-path victims). Additionally, the indirect victims should be then determined by considering the zoning of the direct and same-path victims. If there is congestion in switches or ISLs outside the scope of the direct, indirect, and same-path victims, then it is likely the congestion is due to some other culprit device.

Timeout-drop Anomaly

An observation, which is based on our experience, is that in a multi-switch environment, timeout-drops may not be recorded on the entire path to the culprit device, but mostly on the switches that are closer to the traffic source (victims) and farther from the culprits.

Consider the topology in Figure 4-10. Host-1 is the slow-drain device (traffic destination).

The instances and amounts of timeout-drops reduce on the switchports that are closer to the culprit device in the data path. This is because when a burst of frames is transmitted by the source, they are dropped in large numbers due to a large amount of congestion on the ISL to the next switch. In Figure 4-10, the ISL port on Switch-C to Switch-B typically shows the most timeout-drops. Because of that, Switch-C only transmits the number of frames to Switch-B that are allowed through the congested ISLs. Then once those frames are received on switch-B, they may already be in sufficiently reduced numbers to allow them all to be transmitted to Switch-A and finally to the end device.

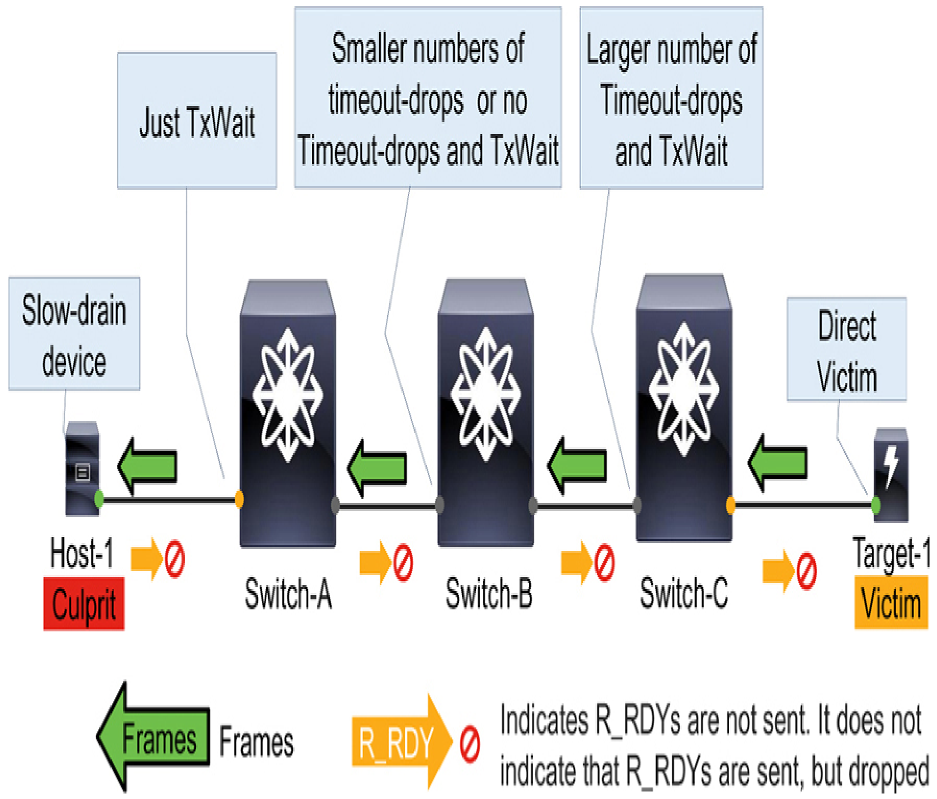

Just TxWait
Smaller numbers of timeout-drops or no Timeout-drops and TxWait
Larger number of Timeout-drops and TxWait
Slow-drain device
Direct Victim
Host-1
Culprit
Switch-A
Switch-B
Switch-C
Target-1
Victim
Frames
Frames
R_RDY
Indicates R_RDYs are not sent. It does not indicate that R_RDYs are sent, but dropped

Figure 4-10 *Timeout-drop anomaly*

On Switch-A, it is almost certain that it has even fewer or no instances of timeout-drops. It may have TxWait or Over-Utilization, but Switch-A most likely does not drop any frames to Host-1.

This timeout drop anomaly is demonstrated in Case Study 3.

Enable and Use Automatic Alerting

Automatic alerting on various indications significantly helps in troubleshooting congestion. For example, a host generating alerts at the same time when a switch generates credit loss is a strong correlation between the cause and the effect of the problems.

Cisco MDS switches have the Port-Monitor feature for alerting on congestion symptoms, as explained in Chapter 3, the section on *Port-Monitor on Cisco MDS Switches*.

Remember to not just enable the alerts, it is important to use them and ensure that these alerts are part of your workflow. We often come across production environments where alerts are enabled but nobody knows where they go. If the alerts are not generating notifications, such as emails, they fail to get the due attention. Essentially, there is a gap between the devices detecting a problem and the users being aware of them.

Use a Remote Monitoring Platform (NDFC/DCNM)

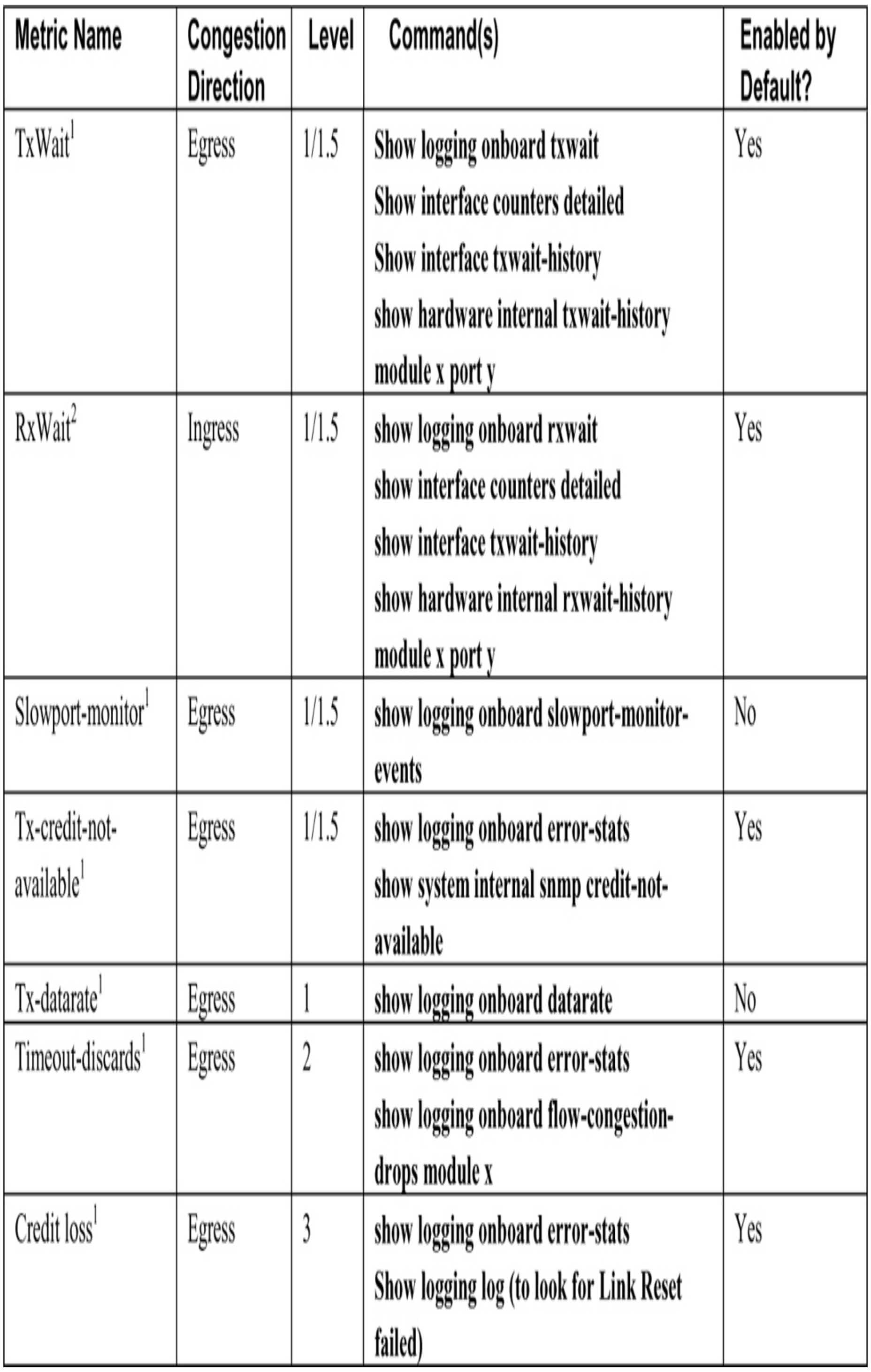
Remote monitoring platforms, such as Cisco NDFC/DCNM, can significantly help in troubleshooting congestion because they show an end-to-end traffic path using topology, centralized monitoring of congestion metrics, alerts generated by multiple switches, etc. For more details, refer to Chapter 3, the section on *Detecting Congestion on a Remote Monitoring Platform*.

The case studies in this chapter use only the NX-OS commands on the Cisco MDS switches for educational purpose. A remote monitoring platform (such as Cisco NDFC/DCNM) would have significantly simplified many basic steps, such as finding FCID, switchport, and VSAN of an end device, members of a port-channel, topology, traffic distribution, etc.

Cisco MDS NX-OS Commands for Troubleshooting Congestion

Cisco MDS switches have a wide variety of commands to detect and troubleshoot congestion. The most useful commands are those that time and date stamp the various indications. Table 4-1 provides a summary, and this section explains each of the commands and when they should be used.

Table 4-1 *Congestion troubleshooting commands on Cisco MDS switches*.



Metric Name	Congestion Direction	Level	Command(s)	Enabled by Default?
TxWait[1]	Egress	1/1.5	Show logging onboard txwait Show interface counters detailed Show interface txwait-history show hardware internal txwait-history module x port y	Yes
RxWait[2]	Ingress	1/1.5	show logging onboard rxwait show interface counters detailed show interface txwait-history show hardware internal rxwait-history module x port y	Yes
Slowport-monitor[1]	Egress	1/1.5	show logging onboard slowport-monitor-events	No
Tx-credit-not-available[1]	Egress	1/1.5	show logging onboard error-stats show system internal snmp credit-not-available	Yes
Tx-datarate[1]	Egress	1	show logging onboard datarate	No
Timeout-discards[1]	Egress	2	show logging onboard error-stats show logging onboard flow-congestion-drops module x	Yes
Credit loss[1]	Egress	3	show logging onboard error-stats Show logging log (to look for Link Reset failed)	Yes

The congestion detection commands on Cisco MDS switches are in four general categories:

1. **Show interface**
2. **Show interface counters** [detailed]
3. **Show interface txwait-history | rxwait-history**
4. **OBFL** commands.

The show interface Command

The **show interface** command (Example 4-6) displays the following information:

- Rx and Tx B2B credits. The Tx credits are the Rx credits of the neighbor.
- The number of free receive buffers on an FC port at the time its value is read, such as when **show interface** command was issued
- Input and output discards (drops)
- Bit errors such as CRC corrupted frames
- Operational state of the B2B State Change mechanism.
- FC link metrics such as OLS, NOS, and LRR.

Use this command for investigating an ongoing issue in real-time

Example 4-6 *NX-OS command **show interface fc** on Cisco MDS switches*

```
fc1/20 is up
    Hardware is Fibre Channel, SFP is short wave
laser w/o OFC (SN)
    Port WWN is 20:14:00:de:fb:d9:49:d0
    Peer port WWN is 20:04:00:3a:9c:f1:ec:40
    Admin port mode is auto, trunk mode is off
    snmp link state traps are enabled
    Port mode is E
    Port vsan is 102
    Admin Speed is 8 Gbps
    Operating Speed is 8 Gbps
    Rate mode is dedicated
    Port flow-control is R_RDY

    Transmit B2B Credit is 64
    Receive B2B Credit is 64
    B2B State Change: Admin(on), Oper(up),
Negotiated Value(14)
    Receive data field Size is 2112
    Beacon is turned off
    Logical type is core
    5 minutes input rate 229120 bits/sec,28640
bytes/sec, 520 frames/sec
    5 minutes output rate 572399872
bits/sec,71549984 bytes/sec, 34529 frames/sec
      10370546596 frames input,16891526339644 bytes
        0 discards,1053 errors
        1 invalid CRC/FCS,0 unknown class
        0 too long,67 too short
      98916424003 frames output,201863442086204
bytes
        4 discards,0 errors
      15 input OLS,13  LRR,172 NOS,20 loop inits
      51 output OLS,64 LRR, 135 NOS, 33 loop inits
    64 receive B2B credit remaining
    8 transmit B2B credit remaining
    0 low priority transmit B2B credit remaining
    Interface last changed at Fri Jan 21 16:24:37
2022

    Last clearing of "show interface" counters:
never
```

Example 4-6 shows the following:

- Agreed to transmit B2B credits are 64
- Agreed to receive B2B credits are 64
- There are just 8 transmit credits remaining at the time the command was issued.
- There are 64 receive credits remaining at the time the command was issued. That indicates this switch does not "owe" any credits to the adjacent device.
- The B2B state change mechanism was negotiated successfully and it is operational (up). The BB_SC_N (number) is 14 which means the count is 2^{14} = 16,384. Consequently, each side of the link sends a BB_SC_S after 16,384 frames have been sent and sends a BB_SC_R after 16,384 R_RDYs have been sent. Refer to

Chapter 2, the section on *Credit Loss Recovery Using B2B State Change Mechanism* for more details.

- There are 1053 frames dropped due to input errors. Input errors are an aggregation of multiple different types of frame errors such as frames that are too short, missing an EOF or contain CRC errors.
- There is 1 frame dropped due to invalid CRC
- There are 4 output discards. This is an aggregation of all output discard types.

The show interface counters [detailed] Command

The **show interface counters** and **show interface counters detailed** are similar but the number of counters is more in the **show interface counters detailed** (Example 4-7) command. This command outputs a range of important counters grouped in the following categories:

- Rate stats
- Total Stats
- Link Stats
- Loop Stats
- Congestion Stats

For troubleshooting congestion, the most important counters are grouped in the Congestion Stats section but the BB_SCs, BB_SCr, FEC and LR counters listed in the Link Stats group are also important. These counters give a cumulative view of the interface but do not indicate exactly when they incremented except for the "last Cleared" date/time.

Use this command for investigating an ongoing issue in real-time. It is concise, and it is module-type independent, unlike OBFL. It gives a good quick view of any problems that have occurred on a link. For a port channel (link aggregation) interface, this command shows the sum of the counters from all the member interfaces.

Example 4-7 *NX-OS command* ***show interface fc counters detailed*** *on Cisco MDS switches*

```
MDS# show interface fc10/48 counters detailed
fc10/48
  Rx 5 min rate bit/sec:
0
  Tx 5 min rate bit/sec:
320
  Rx 5 min rate bytes/sec:
0
  Tx 5 min rate bytes/sec:
40
  Rx 5 min rate frames/sec:
0
  Tx 5 min rate frames/sec:
0

 Total Stats:
  Rx total frames:
22592
  Tx total frames:
141972
  Rx total bytes:
35001588
  Tx total bytes:
35381224
  Rx total multicast:
0
  Tx total multicast:
0
  Rx total broadcast:
0
  Tx total broadcast:
12
  Rx total unicast:
22592
  Tx total unicast:
141972
  Rx total discards:
0
  Tx total discards:
```

```
1224
  Rx total errors:
0
  Tx total errors:
0
  Rx class-2 frames:
0
  Tx class-2 frames:
0
  Rx class-2 bytes:
0
  Tx class-2 bytes:
0
  Rx class-2 frames discards:
0
  Rx class-2 port reject frames:
0
  Rx class-3 frames:
16126
  Tx class-3 frames:
10210
  Rx class-3 bytes:
34630360
  Tx class-3 bytes:
21898144
  Rx class-3 frames discards:
0
  Rx class-f frames:
6466
  Tx class-f frames:
131762
  Rx class-f bytes:
371228
  Tx class-f bytes:
13483080
  Rx class-f frames discards:
0

 Link Stats:
  Rx Link failures:
1
  Rx Sync losses:
```

```
0
  Rx Signal losses:
0
  Rx Primitive sequence protocol errors:
0
  Rx Invalid transmission words:
0
  Rx Invalid CRCs:
0
  Rx Delimiter errors:
0
  Rx fragmented frames:
0
  Rx frames with EOF aborts:
0
  Rx unknown class frames:
0
  Rx Runt frames:
0
  Rx Jabber frames:
0
  Rx too long:
0
  Rx too short:
0
  Rx FEC corrected blocks:
0
  Rx FEC uncorrected blocks:
0
  Rx Link Reset(LR) while link is active:
1
  Tx Link Reset(LR) while link is active:
1
  Rx Link Reset Responses(LRR):
1
  Tx Link Reset Responses(LRR):
3
  Rx Offline Sequences(OLS):
1
  Tx Offline Sequences(OLS):
2
  Rx Non-Operational Sequences(NOS):
```

```
1
  Tx Non-Operational Sequences(NOS):
1
  BB_SCs credit resend actions:
0
  BB_SCr Tx credit increment actions:
0

 Loop Stats:
  Rx F8 type LIP sequence errors:
0
  Tx F8 type LIP sequence errors:
0
  Rx Non F8 type LIP sequence errors:
0
  Tx Non F8 type LIP sequence errors:
0

 Congestion Stats:
  Tx Timeout discards:
1224
  Tx Credit loss:
0
  TxWait 2.5us due to lack of transmit credits:
3446713
  Percentage TxWait for last 1s/1m/1h/72h:
0%/0%/0%/0%
  Rx B2B credit remaining:
1000
  Tx B2B credit remaining:
500
  Tx Low Priority B2B credit remaining:
500
  Rx B2B credit transitions to zero:
37
  Tx B2B credit transitions to zero:
200

 Last clearing of "show interface" counters :
never
```

The show interface txwait-history and rxwait-history Commands

The **show interface txwait-history** and **show interface txwait-history** graphs show three ASCII graphs for TxWait and RxWait.

- Last one minute — Each column shows the cumulative TxWait or RxWait in a second
- Last one hour — Each column shows the cumulative TxWait or RxWait in a minute
- Last 72 hours — Each column shows the cumulative TxWait or RxWait in an hour

Use this command for investigating an ongoing congestion issue in real time.

Example 4-8 shows TxWait history for the last minute on interface fc1/13. Each column shows TxWait in that second. The most current second (the time in which the command was issued) is on the left. So the X-axis shows seconds before issuing the command. At the top of the graph in vertical format is the actual number of milliseconds of TxWait in that one second. For example, 15 seconds earlier, there were 957ms of cumulative TxWait in that one second.

Example 4-8 *NX-OS command **show interface txwait-history** for last one minute on Cisco MDS switches*

```
MDS9710-1# show interface fc1/13 txwait-history

TxWait history for port fc1/13:
==============================
       6999999999999999999999999999999996
      39666655555545454555554554455554545   1

0540000441811176718262177686618726690005000000000000
00000000
1000    ########## # # ##### ##  #### #
 900    ################################
 800    ################################
 700   ##################################
 600   ##################################
 500   ##################################
 400   ##################################
 300   ##################################
 200   ##################################
 100   ##################################

0....5....1....1....2....2....3....3....4....4....5.
...5....6
              0    5    0    5    0    5    0    5
0    5    0

          Tx Credit Not Available per second (last
60 seconds)
                 # = TxWait (ms)
```

The output for one hour and 72 hours is similar. Refer to Chapter 3, the section on *TxWait History Graphs* for an example and explanation.

The OBFL Commands—show logging onboard

The On Board Failure Logging (OBFL) is a feature on Cisco MDS switches that stores important metrics and logs with time and date stamps. The following are more details about OBFL.

1. Each module on modular platforms (MDS 9700 Directors) has a dedicated OBFL, whereas the fixed fabric switches maintain a per-switch OBFL.
2. MDS switches maintain OBFL in Non-Volatile Random Access Memory (NVRAM), which is persistent memory. As a result, OBFL logs are not lost even if the module or switch is restarted, reloaded, or power cycled.
3. There are several event types in OBFL and each has its own fixed-size cyclical file. Once it fills up, the oldest entry, chronologically, is overwritten with the newest entry. This is called cyclical file wrapping.

4. There is no need to clear the OBFL entries. We don't even recommend doing this because clearing the entries may lead to loss of information, as seen in Case Study 1. Clearing OBFL counters is not even needed because OBFL is a date and time-stamped cyclical logfile. The oldest entries are automatically overwritten when the newest entries are added. Note that the **clear logging onboard** command clears the file and data in it, whereas the **clear statistics** command clears only the data in the file. These commands should be avoided unless there are special reasons for using them.
5. These individual OBFL event types are all accessed by the **show logging onboard** command on a per-module basis. To detect congestion there are four main event types to be covered. Note that for fabric switches the module number is always 1 and does not need to be specified.
6. When investigating in real time a helpful feature is the "starttime" parameter. It allows finding only events recorded after a certain date and time. For example, if the day the investigation begins is May 2nd, 2022 and it is known that the problem started the day before then the "starttime 05/01/22-00:00:00" can be included in the command to only display the events that occurred on May 1st and 2nd. To look for timeout-drops the command would be **show logging onboard starttime 05/01/22-00:00:00 flow-control timeout-drops**.

Because all OBFL information is date- and time-stamped, it allows the ability to 'look back' and see what indications occurred during the time of the congestion natively on Cisco MDS switches. Congestion problems come and go. Before (or without) OBFL, someone had to actively investigate the various indications and counters to be able to see if they were actively incrementing. Many people have had very difficult times trying to predict and catch the congestion 'in the act'. OBFL ends all this since all the important congestion indications have date and time stamps. Various OBFL logs can be gathered at one time and a unified, time-correlated picture of the congestion in the entire SAN can be determined. This facilitates a completely different and powerful way of troubleshooting.

TxWait

The **show logging onboard txwait module <x>** command records any interfaces that have 100ms or more of cumulative TxWait in a 20-second polling interval. This means that even if the TxWait increments in small amounts as long as it adds up to 100ms or more in the 20-second interval, it is recorded. If the interface has less than 100ms of cumulative TxWait time in that 20-second interval, then no entry is recorded in OBFL, although the raw TxWait counter still increments.

Refer to Chapter 3, the section on *TxWait History in OBFL* for an example of **show logging onboard txwait** command.

RxWait

The **show logging onboard rxwait module <x>** command records any interfaces that have 100ms or more of cumulative RxWait in a 20-second polling interval. This means that even if the RxWait increments in small amounts as long as it adds up to 100ms or more in the 20-second interval, it is recorded. If the interface has less than 100ms of cumulative RxWait time in that 20-second interval, then no entry is recorded.

Note: RxWait for Fibre Channel ports is available only in 64G MDS modules and switches.

Refer to Chapter 3, the section on *Rx Credit Unavailability in Microseconds — RxWait* for an example of **show logging onboard rxwait** command.

Error-statistics

The **show logging onboard module <x> error-stats** command displays specific error-statistics, like TxWait, that are recorded every 20 seconds. Each module checks each of its interfaces' error counters every 20 seconds to see if any of the applicable error counters have incremented (even by 1). It's important to note that when an error counter increments the current value of the error counter is recorded not just the delta value (the amount it incremented in that 20 seconds). Consequently, to determine the delta value for a specific interface and error counter entry, an earlier (in time) entry must be located, and the two counts must be subtracted. If no earlier entry for the same interface and error counter is found, then it could be because this is the first entry for that specific interface and error counter combination, or it could mean that the prior entry was overwritten due to the OBFL cyclical file wrapping.

Most MDS switches record their counter names with some hardware-specific information. Consequently, the same error or indication may look different on different types of switches or modules.

The following are meaningful error-stats counters for troubleshooting congestion.

Tx-credit-not-available (100ms continuous)

Refer to Chapter 3, the section on *Continuous Tx Credit Unavailability for 100 ms — Tx-credit-not-available* for more details on this counter. The following are the counters that detect the continuous duration of zero remaining-Tx-B2B-credits on MDS switches.

- FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO: MDS 9700 16G module, MDS 9148S, 9250i, 9396S
- F32_MAC_KLM_CNTR_TX_WT_AVG_B2B_ZERO: MDS 9700 32G module, MDS 9132T, 9148T, 9220i, 9396T
- F64_CMON_TX_WT_100MS_CH0_TMR1_HIT: R_RDY Mode on MDS 9700 64G module, MDS 9124V, 9148V, 9396V
- F64_CMON_TX_WT_100MS_CH3_TMR1_HIT: ER_RDY Mode on MDS 9700 64G module, MDS 9124V, 9148V, 9396V

CHx in the name indicates the channel or Virtual Link (VL) number.

F64_CMON_TX_WT_100MS_CH0_TMR1_HIT is used when the interface is in R_RDY mode since there are no separate VLs and F64_CMON_TX_WT_100MS_CH3_TMR1_HIT is used for when the interface is in ER_RDY mode since VL3 is for the normal data traffic. As Chapter 6 explains, Fibre Channel links can be logically separated into multiple virtual links or VLs. When this occurs, the physical link is in Enhanced R_RDY mode, or ER_RDY mode.

If these counters increment by, say 5, then that indicates five 100ms intervals of continuous zero remaining-Tx-B2B-credits. That is 500ms of time but not necessarily 500ms of continuous time.

Rx-credit-not-available (100ms continuous)

Refer to Chapter 3, the section on *Continuous Rx Credit Unavailability for 100 ms - Rx-credit-not-available* for more details on this counter. The following are the counters that detect the continuous duration of zero remaining-Rx-B2B-credits on MDS switches:

- FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO: MDS 9700 16G module, MDS 9148S, 9250i, 9396S
- F32_MAC_KLM_CNTR_RX_WT_AVG_B2B_ZERO: MDS 9700 32G module, MDS 9132T, 9148T, 9220i, 9396T
- F64_CMON_RX_WT_100MS_CH0_CNT1: R_RDY Mode on MDS 9700 64G module, MDS 9124V, 9148V, 9396V
- F64_CMON_RX_WT_100MS_CH3_CNT1: ER_RDY Mode on MDS 9700 64G module, MDS 9124V, 9148V, 9396V

CHx in the name indicates the channel or Virtual Link number. F64_CMON_RX_WT_100MS_CH0_CNT1 is used when the interface is in R_RDY mode since there are no separate VLs and F64_CMON_RX_WT_100MS_CH3_CNT1 is used for when the interfaces is in ER_RDY mode since VL3 is the normal data traffic VL.

If these counter increments by, say 3, then that indicates three 100ms intervals of continuous zero remaining-Rx-B2B-credits. That is 300ms of time but not necessarily 300ms of continuous time that the switch has withheld B2B credits from the adjacent device.

Timeout-drops

The following counters indicate that a frame was dropped due to it exceeding the congestion-drop timeout or the no-credit-drop timeout for that interface. Refer to Chapter 6 for more details on congestion-drop timeout and no-credit-drop timeout. The following counters detect timeout-drops on Cisco MDS switches:

- VIP_TMM_TO_DROP_CNT: MDS 9148S, 9250i
- F16_TMM_TOLB_TIMEOUT_DROP_CNT: MDS 9700 16G module, MDS 9396S
- F32_TMM_PORT_TIMEOUT_DROP: MDS 9700 32G module, MDS 9132T, 9148T, 9220i, 9396T
- F64_TMM_PORT_TIMEOUT_DROP: MDS 9700 64G module, MDS 9124V, 9148V, 9396V

Credit Loss

The following counters indicate that an interface was at zero remaining-Tx-B2B-credits for 1 or 1.5 continuous seconds and a Link Reset (LR) was transmitted to recover the B2B credits. It does not indicate if the Link Reset was successful. The following counters detect credit-loss events on Cisco MDS switches:

- FCP_SW_CNTR_CREDIT_LOSS: MDS 9700 16G module, MDS 9148S, 9250i, 9396S
- F32_MAC_KLM_CNTR_CREDIT_LOSS: MDS 9700 32G module, MDS 9132T, 9148T, 9220i, 9396T
- F64_CMON_CREDIT_LOSS_CH0_TMR2_HIT: R_RDY Mode on MDS 9700 64G module, MDS 9124V, 9148V, 9396V

- F64_CMON_CREDIT_LOSS_CH3_TMR2_HIT: ER_RDY Mode on MDS 9700 64G module, MDS 9124V, 9148V, 9396V

Note CHx in the name indicates the channel or Virtual Link number. F64_CMON_CREDIT_LOSS_CH0_TMR2_HIT is used when the interface is in R_RDY mode since there are no separate VLs and F64_CMON_CREDIT_LOSS_CH3_TMR2_CNT1_HIT is used for when the interfaces are in ER_RDY mode since VL3 is the normal data traffic VL.

Refer to Example 4-9 for the output of **show logging onboard error-stats** command. It shows:

- fc1/50's indication of 100ms continuous ingress congestion incremented to 12 in the 20-second interval ending in 13:23:21. A prior entry for the same counter on 06/05/2021 01:36:12 shows the value was 4. This indicates that it incremented by 8 which is 800ms in 20 seconds ending at 13:23:21. It cannot be determined how the 8 individual 100ms intervals are distributed over the 20 seconds. They could be in one continuous block, in 8 separate blocks of time, or any combination. Regardless, it indicates a significant amount of ingress congestion on fc1/50.
- fc1/31 had both the timeout drop and credit loss events.

Example 4-9 ***show logging onboard error-stats*** *command on Cisco MDS switches*

```
show logging onboard module 1 error-stats


----------------------------

    Module:  1

----------------------------



----------------------------

 Show Clock

----------------------------

2021-06-22 20:57:00
```

```
---------------------------------
 Module: 1 error-stats
---------------------------------


----------------------------------------------------
-------------------------
 ERROR STATISTICS INFORMATION FOR DEVICE DEVICE:
FCMAC
----------------------------------------------------
-------------------------
Interface |                                    |
|    Time Stamp
  Range   |    Error Stat Counter Name         |
Count  | MM/DD/YY HH:MM:SS
          |                                    |
|
----------------------------------------------------
-------------------------
fc1/50    |F32_MAC_KLM_CNTR_RX_WT_AVG_B2B_ZERO | 12
| 06/22/21 13:23:21
fc1/31    |F32_TMM_PORT_TIMEOUT_DROP           |
46139   | 06/22/21 13:23:21
fc1/31    |F32_MAC_KLM_CNTR_CREDIT_LOSS        | 12
| 06/22/21 13:23:21
fc1/31    |F32_TMM_PORT_TIMEOUT_DROP           |
42175   | 06/08/21 06:05:42
fc1/31    |F32_MAC_KLM_CNTR_CREDIT_LOSS        | 11
| 06/08/21 06:05:42
fc1/31    |F32_TMM_PORT_TIMEOUT_DROP           |
40004   | 06/08/21 05:36:59
fc1/31    |F32_MAC_KLM_CNTR_CREDIT_LOSS        | 10
| 06/08/21 05:36:59
fc1/96    |F32_MAC_KLM_CNTR_RX_WT_AVG_B2B_ZERO | 13
| 06/08/21 05:14:37
fc1/31    |F32_TMM_PORT_TIMEOUT_DROP           |
37913   | 06/08/21 05:14:37
fc1/31    |F32_MAC_KLM_CNTR_CREDIT_LOSS        | 9
| 06/08/21 05:14:37
fc1/96    |F32_MAC_KLM_CNTR_RX_WT_AVG_B2B_ZERO | 10
| 06/08/21 05:07:36
...
```

```
fc1/50    |F32_MAC_KLM_CNTR_RX_WT_AVG_B2B_ZERO | 4
| 06/05/21 01:36:12
```

Invalid Transmission Words (ITWs)

The following counters display ITWs on the OBFL on Cisco MDS switches.

- IP_FCMAC_CNT_STATS_ERRORS_RX_BAD_WORDS_FROM_DECODER: MDS 9700 16G module, MDS 9148S, 9250i, 9396S
- F32_MAC_KLM_CNTR_RX_BAD_WORDS_FROM_DECODER: MDS 9700 32G module, MDS 9132T, 9148T, 9220i, 9396T
- F64_MAC_KLM_CNTR_RX_BAD_WORDS_FROM_DECODER: on MDS 9700 64G module, MDS 9124V, 9148V, 9396V

Refer to Chapter 2, the section on *Fibre Channel Counters — Summary* for more details on ITW.

Input CRC Errors

The following counters display CRC errors on the OBFL on Cisco MDS switches.

- ’IP_FCMAC_CNT_STATS_ERRORS_RX_CRC: MDS 9700 16G module, MDS 9148S, 9250i, 9396S
- IP_FCMAC_ERRORS_RX_CRC: MDS 9700 32G and 64G modules, MDS 9132T, 9148T, 9220i, 9396T, 9124V, 9148V, 9396V

Refer to Chapter 2, the section on *Fibre Channel Counters — Summary* for more details on CRC errors.

Counter-statistics

The **show logging onboard module <x> counter-stats** command displays specific statistical (not necessarily errors) events that are recorded every 10 minutes. Each module checks each of its interfaces every 10 minutes to see if any of the applicable statistical counters have incremented (even by 1). It's important to note that when a statistical counter increment, its current value is recorded not just the delta value (the amount it incremented in that 10 minutes). Consequently, to determine the delta value for a specific interface and statistical counter entry, an early (in time) entry must be located, and the two counts must be subtracted. If no earlier entry for the same interface and statistical counter is found, then it could be because this is the first entry for that specific interface and statistical counter combination, or it could mean that the earlier entry was overwritten due to the OBFL cyclical file wrapping.

Slowport-monitor Events

Slowport-monitor measures the continuous duration when a switchport has zero remaining-Tx-B2B-credits for a duration longer than the configured admin delay. Slowport-monitor history is captured in OBFL and can be displayed using the **show logging onboard slowport-monitor-events** command on Cisco MDS switches. Refer to Chapter 3, the section on *Slowport-monitor History in OBFL* for an example of this command output.

High-datarate

The **show logging onboard datarate** command on Cisco MDS switches records instances of Port-Monitor tx-datarate, rx-datarate, tx-datarate-burst, and rx-datarate-burst rising-thresholds and falling-thresholds. This can pin-point occurrences of high link utilization which may lead to congestion due to over-utilization.

The datarate counters must be enabled using the Port-Monitor feature on Cisco MDS switches before these are recorded in OBFL. Once Port-Monitor is configured and activated with at least one of the datarate counters monitored, whenever the counter's rising or falling threshold is hit, in addition to the

SNMP alert and Syslog, an entry is recorded in OBFL datarate.

Refer to Chapter 3, the section on *Link utilization* for more details, and the output of this command.

Request-timeouts

The **show logging onboard flow-control request-timeout** command indicates that an ingress interface wants to send frames to an egress interface but it is not receiving a grant from the central arbiter to allow it to transmit to the egress interface. Refer to Chapter 2, the section on *Frame Switching Within a Cisco MDS switch* for more details about the central arbiter and how the grant works.

When an ingress interface receives a frame, it determines the egress interface and sends a request to the Central Arbiter. If the egress interface has sufficient buffer space to hold the frame, the Central Arbiter returns a grant to the ingress interface and the ingress interface transmits the frame to the egress interface through the internal switching infrastructure (crossbar). If the egress interface does not have sufficient buffer space to hold the frame, the Central Arbiter does not return a grant. The ingress interface waits a small amount of time and resends the request. Again, if the egress interface still does not have sufficient buffer space to hold the frame, the Central Arbiter continues to not return a grant. This process repeats until the egress interface can accept the frame and the arbiter returns a grant.

Refer to Example 4-10. The ‘Dest Intf’ fc1/47 is the congested port and fc1/1-fc1/16 are the ports that are attempting to send traffic to fc1/47. It would be expected to find either high Tx-datarate or high TxWait on fc1/47 at the time stamps shown.

If an ingress frame is sufficiently delayed, approximately 200ms, then an entry is recorded in the module’s OBFL flow-control request-timeouts. Note that request-timeouts are not frame timeout-drops. When an entry is recorded here it simply means a delay and not a loss of frame. When the frame reaches the egress port, the timestamp of the frame is checked and if it exceeds the Frame Discard Timeout Value (called congestion-drop timeout on Cisco MDS switches), it is dropped as a timeout-drop.

Example 4-10 ***show logging onboard flow-control request-timeout*** *on Cisco MDS switches*

```
---------------------------------
 Module: 1 flow-control request-timeout
---------------------------------


----------------------------
Module: 1 show clock
----------------------------
2022-02-17 10:22:08

---------------------------------
-----------------------------------------------------
----------------------------
|  Dest          |     Source    | Events |
Timestamp     |     Timestamp
|  Intf          |      Intf     |  Count |
Earliest     |       Latest
-----------------------------------------------------
----------------------------
|fc1/47(DI:94)  |fc1/1,fc1/2, |       1|02/14/2022-
23:32:15|02/14/2022 23:32:15|
|               |fc1/3,fc1/4,  |        |
|                   |
|               |fc1/5,fc1/6,  |        |
|                   |
|               |fc1/7,fc1/8,  |        |
|                   |
|               |fc1/9,fc1/10, |        |
|                   |
|               |fc1/11,fc1/12,|        |
|                   |
|               |fc1/13,fc1/14,|        |
|                   |
|               |fc1/15,fc1/16,|        |
|                   |
-----------------------------------------------------
----------------------------
```

```
|fc1/47(DI:94)  |fc1/17,fc1/18,|       1|01/15/2022-
23:05:45|01/15/2022-23:05:45|
|               |fc1/19,fc1/20,|        |
|                   |
|               |fc1/21,fc1/22,|        |
|                   |
|               |fc1/23,fc1/24,|        |
|                   |
|               |fc1/25,fc1/26,|        |
|                   |
|               |fc1/27,fc1/28,|        |
|                   |
|               |fc1/29,fc1/30,|        |
|                   |
|               |fc1/31,fc1/32,|        |
|                   |
----------------------------------------------------
----------------------------------
```

Flow Congestion Drops

The MDS 32G and 64G modules and switches can give more information for frames that are dropped due to timeout. In addition to recording the fact that there are timeout drops in OBFL error-stats, the VSAN, source FCID, destination FCID, and the delta count of drops can be displayed using the **show logging onboard flow-congestion-drops** command on Cisco MDS switches. This can help identify the affected devices (victims) when the interface is an E_Port (ISL) or the F_Port has multiple logins (FLOGIs) on it.

Refer to Chapter 3, the section on *Timeout-discards or Timeout-drops* for an example output of this command.

Generic Troubleshooting Commands

The following NX-OS commands on Cisco MDS switches can be used to determine the topology of the fabric, the optimum routing path through the fabric, and the end-device communication relationships. This allows chasing the culprit in the congestion direction, to find the victims, and to find additional information about them.

- **show topology**: Shows the directly connected switches, their domain ID, and local and remote ports.
- **show flogi database**: Shows the FCID and WWPN of a connected/logged-in device and the port/interface on the local switch where this device is connected.
- **show fcns database**: Shows the name server database, which carries the FCID and WWPN of an end device, and optionally if the device is an initiator or a target. The first octet of the FCID of a device is the domain ID in the VSAN of the switch to where this device is connected. For example, a device with FCID 0x050160 is connected to a switch with domain ID 0x05.
- **show zone member**: Shows the name of the zone to which a device belongs to. This command can be used to find the victims of a culprit device or vice versa.
- **show zone active**: Shows the members of a zone. An asterisk (*) before a device indicates that it is currently logged in to the Fibre Channel fabric.
- **show zoneset active**: Shows the active zoneset and all its members. This command can be used to find the victims of a culprit device.
- **show fcs ie**: Shows all the switches in a fabric including their proximity (Local, Adjacent, or Remote), mgmt-id including domain-id, IP address and switch names.
- **show fcdomain domain-list**: Shows the list of the switches in the fabric. As mentioned, the domain ID of a switch makes the first octet of the FCID of a device.
- **show fspf database vsan <vsan_id> domain <domain_id>**: Shows the details of each switch in the fabric with its domain ID and all links in the fabric including their costs. This can be used to determine the route from culprit to victim (and vice versa).
- **show rdp**: Read Diagnostic Parameter (RDP) can be used to display the information of an HBA port, such as its

B2B credits, SFP information, and bit errors like CRC, ITW, and FEC as seen by the HBA port. This bit-error information is important because if the bits are corrupted after a switchport sends data but before it is received by the HBA, the local switchport can't detect these errors, but RDP can pull these metrics in-band from the HBA port and display them on the switch. The **show interface** command displays the information about the local switchport, whereas **show rdp** command displays the information of the peer HBA on the end device that is connected to the local switchport. RDP works in-band on the Fibre Channel links.

- **show fdmi database**: Fabric Device Management Interface (FDMI) shows additional management information of an end device, such as its model, firmware, and host name. FDMI works in-band on the Fibre Channel links.

Following sections show the output of these commands.

Show Topology

Example 4-11 shows the NX-OS command **show topology** on Cisco MDS switches.

Example 4-11 *NX-OS command **show topology** on Cisco MDS switches*

```
MDS9706-C# show topology

FC Topology for VSAN 1 :
----------------------------------------------------
----------------------------
       Interface  Peer Domain Peer Interface
Peer IP Address(Switch Name)
----------------------------------------------------
----------------------------
    port-channel2   0x05(5)    port-channel2
A.B.C.D (MDS9718-A)
```

Show FLOGI Database

Example 4-12 shows the NX-OS command **show flogi database** on Cisco MDS switches.

Example 4-12 *NX-OS command **show flogi database** on Cisco MDS switches*

```
MDS9706-C# show flogi database
---------------------------------------------------------------------
----------------------------
INTERFACE         VSAN     FCID              PORT NAME
NODE NAME
---------------------------------------------------------------------
----------------------------
port-channel1     20     0x320072
20:a1:00:25:b5:00:00:6d 20:ff:00:25:b5:00:00:6e
                                  [B230-7-A6D]
```

Show FCNS Database

Example 4-13 shows the NX-OS command **show fcns database** on Cisco MDS switches.

Example 4-13 *NX-OS command **show fcns database** on Cisco MDS switches*

```
MDS9706-C# show fcns database

VSAN 20:
---------------------------------------------------------------------
----------------------
FCID         TYPE   PWWN                      (VENDOR)
FC4-TYPE:FEATURE
---------------------------------------------------------------------
----------------------
0x050160     N      21:00:00:24:ff:3d:6d:0f (Qlogic)
scsi-fcp:target
                  [Tiktok_0F]
```

Show Zone Member

Example 4-14 shows the NX-OS command **show zone member** on Cisco MDS switches.

Example 4-14 *NX-OS command* ***show zone member*** *on Cisco MDS switches*

```
MDS9706-C# show zone member fcid 0x320072
fcid 0x320072 vsan 20
  zone Z_B230-7
```

Show Zone Name

Example 4-15 shows the NX-OS command **show zone name** on Cisco MDS switches.

Example 4-15 *NX-OS command* ***show zone name*** *on Cisco MDS switches*

```
MDS9706-C# show zone name Z_B230-7 active vsan 20
zone name Z_B230-7 vsan 20
  device-alias B230-7-B6D  init
  device-alias B230-7-B7D  init
* fcid 0x320072 [device-alias B230-7-A6D]  init
* fcid 0x320073 [device-alias B230-7-A7D]  init
* fcid 0x050140 [device-alias Tiktok_0E]  target
* fcid 0x050160 [device-alias Tiktok_0F]  target
  device-alias Tiktok_10  target
  device-alias Tiktok_11  target
```

Show Zoneset Active

Example 4-16 shows the NX-OS command **show zoneset active** on Cisco MDS switches.

Example 4-16 *NX-OS command* ***show zoneset active*** *on Cisco MDS switches*

```
MDS9706-C# show zoneset active | begin Z_B230-7
<snip>
  zone name Z_B230-7 vsan 20
    device-alias B230-7-B6D  init
    device-alias B230-7-B7D  init
  * fcid 0x320072 [device-alias B230-7-A6D]  init
  * fcid 0x320073 [device-alias B230-7-A7D]  init
  * fcid 0x050140 [device-alias Tiktok_0E]  target
  * fcid 0x050160 [device-alias Tiktok_0F]  target
    device-alias Tiktok_10  target
    device-alias Tiktok_11  target
<snip>
```

Show FCS Ie

Example 4-17 shows the NX-OS command **show fcs ie** on Cisco MDS switches.

Example 4-17 *NX-OS command **show fcs ie** on Cisco MDS switches*

```
MDS9706-C# show fcs ie

IE List for VSAN: 20
----------------------------------------------------
---------------------------
IE-WWN                   IE      Mgmt-Id  Mgmt-Addr
(Switch-name)
----------------------------------------------------
---------------------------
20:14:00:de:fb:b1:84:21  S(Rem)  0xfffc33 AA.BB.CC.DD
(MDS9396T-A)
20:14:8c:60:4f:54:51:01  S(Loc)  0xfffc32 A1.B1.C1.D1
(MDS9706-C)
20:14:8c:60:4f:9e:2b:01  S(Adj)  0xfffc05 A.B.C.D
(MDS9718-A)
[Total 3 IEs in Fabric]
```

Show Fcdomain

Example 4-18 shows the NX-OS command **show fcdomain domain-list** on Cisco MDS switches.

Example 4-18 *NX-OS command **show fcdomain domain-list** on Cisco MDS switches*

```
MDS9706-C# show fcdomain domain-list vsan 20

Number of domains: 3
Domain ID              WWN
```

```
---------    -----------------------
0x05(5)     20:14:8c:60:4f:9e:2b:01 [Principal]
0x32(50)     20:14:8c:60:4f:54:51:01 [Local]
0x33(51)     20:14:00:de:fb:b1:84:21
MDS9706-C#
```

Show FSPF Database

Example 4-19 shows the NX-OS command **show fspf database** on Cisco MDS switches.

Example 4-19 *NX-OS command* ***show fspf database*** *on Cisco MDS switches*

```
MDS9706-C# show fspf database vsan 20 domain 5

FSPF Link State Database for VSAN 20 Domain 0x05(5)
LSR Type                = 1
Advertising domain ID   = 0x05(5)
LSR Age                 = 1641
LSR Incarnation number  = 0x80002496
LSR Checksum            = 0x8655
Number of links         = 2
    NbrDomainId         IfIndex      NbrIfIndex
Link Type            Cost
----------------------------------------------------
------------------------
    0x32(50)       0x00040001(port-channel2)
0x00040001       1             15
    0x33(51)       0x00040004(port-channel5)
0x00040004       1              15
```

Show RDP

Example 4-20 shows the NX-OS command **show rdp** on Cisco MDS switches.

Example 4-20 *NX-OS command* ***show rdp*** *on Cisco MDS switches*

```
MDS9396T-A# show rdp fcid 0x330020 vsan 20
----------------------------------------------------
```

```
---------
                   RDP frame details
----------------------------------------------------
---------
Link Service Request Info:
------------------------------


Port Speed Descriptor Info:
------------------------------
Port speed capabilities : 16G 8G 4G
Port Oper speed         : 16000 Mbps

Link Error Status:
------------------------------
VN PHY port type               : FC
Link failure count             : 3
Loss of sync count             : 0
Loss of signal count           : 0
Primitive sequence proto error : 0
Invalid Transmission word      : 0
Invalid CRC count              : 0

Port Name Descriptor:
------------------------------
Node WWN            : 20:00:00:24:ff:74:e6:c4
Port WWN            : 21:00:00:24:ff:74:e6:c4
Attached Node WWN   : 20:14:8c:60:4f:9e:2b:01
Attached Port WWN   : 20:27:00:de:fb:b1:84:20

SFP Diag params:
------------------------------
SFP flags       :
SFP Tx Type     : Short Wave

FEC Status:
------------------------------
Corrected blocks   : 0
Uncorrected blocks : 0

Buffer Credit Descriptor:
------------------------------
Rx B2B credit   : 500
```

```
Tx B2B credit    : 16
Port RTT         : 0 ns

Optical Product Data:
------------------------------
Vendor Name     :
Model No.       :
Serial No.      :
Revision        :
Date            :

  --------------------------------------------------
--------------------------
                Current             Alarms
Warnings
                Measurement     High        Low
High          Low
  --------------------------------------------------
--------------------------
  Temperature   53.04 C          0.00 C      0.00 C
0.00 C        0.00 C
  Voltage        3.43 V          0.00 V      0.00 V
0.00 V        0.00 V
  Current        6.58 mA         0.00 mA     0.00 mA
0.00 mA       0.00 mA
  Tx Power      -2.22 dBm        0.00 dBm    0.00
dBm    0.00 dBm      0.00 dBm
  Rx Power      -3.54 dBm        0.00 dBm    0.00
dBm    0.00 dBm      0.00 dBm
  --------------------------------------------------
--------------------------
  Note: ++  high-alarm; +  high-warning; --  low-
alarm; -  low-warning
```

Show FDMI Database

Example 4-21 shows the NX-OS command **show fdmi database** on Cisco MDS switches.

Example 4-21 *NX-OS command* ***show fdmi database*** *on Cisco MDS switches*

```
MDS9396T-A# show fdmi database detail hba-id
21:00:00:24:ff:74:e6:c4 vsan 20
Node Name         :20:00:00:24:ff:74:e6:c4
Manufacturer      :QLogic Corporation
Serial Num        :RFD1604J61092
Model             :QLE2742
Model Description:QLogic 32Gb 2-port FC to PCIe Gen3
x8 Adapter
Hardware Ver      :BK3210407-01  A
Driver Ver        :10.01.00.15.08.1-k1
ROM Ver           :3.36
Firmware Ver      :8.03.04 (d0d5)
  Port-id: 21:00:00:24:ff:74:e6:c4
    Supported FC4 types:scsi-fcp
    Supported Speed    :8G 16G 32G
    Current Speed      :16G
    Maximum Frame Size :2048
    OS Device Name     :qla2xxx:host8
    Host Name          :stg-tme-esxi-c240-
2.cisco.com
```

System Messages—show logging log

There are several system messages that the Cisco MDS switches log for congestion troubleshooting. These messages can be optionally sent to a remote Syslog receiver, such as Cisco NDFC/DCNM.

Link failure Link Reset failed nonempty recv queue

Example 4-22 shows this message. It indicates that the neighbor sent a Link Reset (LR) primitive to the MDS interface/swichport while the link was active, but the MDS interface/switchport did not respond with a Link Reset Response (LRR) due to ingress congestion on it.

Example 4-22 *System message - Link failure Link Reset failed nonempty recv queue*

```
%PORT-5-IF_DOWN_LINK_FAILURE: %$VSAN 1%$ Interface
fc2/5 is down (Link failure
Link Reset failed nonempty recv queue)
```

The neighbor likely transmitted the LR because it was at zero remaining-Tx-B2B-credits for an extended duration. When an interface is at zero remaining-Tx-B2B-credits for an extended period (1 or 1.5 seconds) it initiates Credit Loss Recovery by transmitting an LR. The LR is an attempt to reinitialize the bidirectional B2B credits to their initial, agreed-upon values.

If the MDS interface/switchport, which received the LR, still has frames queued that are received from the neighbor, and it was not able to deliver them to the appropriate egress port after 90ms, then the MDS does not send back an LRR, and the link fails. Refer to Chapter 3, the section on *Link Failure — Link Reset Failed Nonempty Recv Queue (LR Rcvd B2B)* for more details.

Essentially, this message indicates severe ingress congestion on a port. Most likely another interface in the switch has severe egress congestion. The next troubleshooting step should be to find the source of severe egress congestion. Use the **show logging onboard flow-control request-timeout** command (Example 4-10) and match the interface in the 'Source Intf' column for the same date and time. Once found the interface listed in the 'Destf- Intf' is the interface with the severe egress congestion.

Link failure Link reset failed due to timeout

Example 4-23 shows this message. It indicates that the interface transmitted a LR primitive but did not receive a LRR back within 100ms. This is likely due to the interface being at zero remaining-Tx-B2B-credits for an extended time (1 or 1.5 seconds). Hence, it initiates Credit Loss Recovery by transmitting an LR. Since LR is transmitted during normal link initialization it could also be a non-congestion problem during link activation. Under normal situations, this should exactly correlate to instances of Credit Loss Recovery by LR. If the neighbor is also an MDS it may not have returned the LRR due to severe ingress congestion. If so, see the earlier Syslog message indicating ‘Link failure Link Reset failed nonempty recv queue’.

Example 4-23 *System message - Link failure Link reset failed due to timeout*

```
%PORT-5-IF_DOWN_LINK_FAILURE: %$VSAN 200%$ Interface
fc1/30 is down (Link failure
Link reset failed due to timeout)
```

Refer to Chapter 3, the section on *Link Failure — Link Reset Failed Nonempty Recv Queue (LR Rcvd B2B)* for more details.

TCP conn. closed - retransmit failure

Example 4-24 shows this message. It indicates that a Fibre Channel over IP (FCIP) link has failed due to exceeding the number of allowed retransmits. FCIP uses TCP as its reliable transport and when there are packet drops and a lack of TCP acknowledgments, the FCIP interface will stop transmitting new FC frames until the packet(s) being retransmitted receives an acknowledgment. The reason this is an indication of congestion is because when the FCIP interface starts retransmitting, it stops accepting ingress frames on other FC ports on the switch, and thus exerts backpressure on those interface(s). FCIP TCP retransmits can take a significant amount of time which looks very similar to egress congestion caused by slow-drain.

Example 4-24 *System message - Link failure Link reset failed due to timeout*

```
%PORT-5-IF_DOWN_TCP_MAX_RETRANSMIT: %$VSAN 1%$
Interface fcip13 is down(TCP conn.
closed - retransmit failure)
```

Refer to Chapter 8, “Congestion Management in TCP Storage Networks,” for more details on TCP transport and FCIP.

Case Study 1—Finding Congestion Culprits and Victims in a Single-Switch Fabric

This case study explains finding the culprit of congestion and the victims in a single-switch fabric. It also explains the correlation of congestion symptoms using graphs, anomalies, and best practices. Being the first case study, it is based on a simple environment of a switch-single fabric. The same workflow can be expanded further to larger fabrics as explained in the following case studies.

A large entertainment company reported that their storage arrays were alerting for congestion in the fabric. Not much detail was provided about how exactly these alerts were generated. This environment used Dell XtremeIO storage arrays and Cisco MDS switches. They suspected slow-drain and requested a detailed investigation.

The first step was to gather **show tech-support slowdrain** commands for all the MDS switches. As mentioned earlier, this command aggregates many other commands important for troubleshooting congestion on Cisco MDS switches. Two such outputs were provided indicating two switches in the environment. As would be found during the investigation, these two switches belonged to the same environment but separate fabrics, similar to as shown in Figure 4-11.

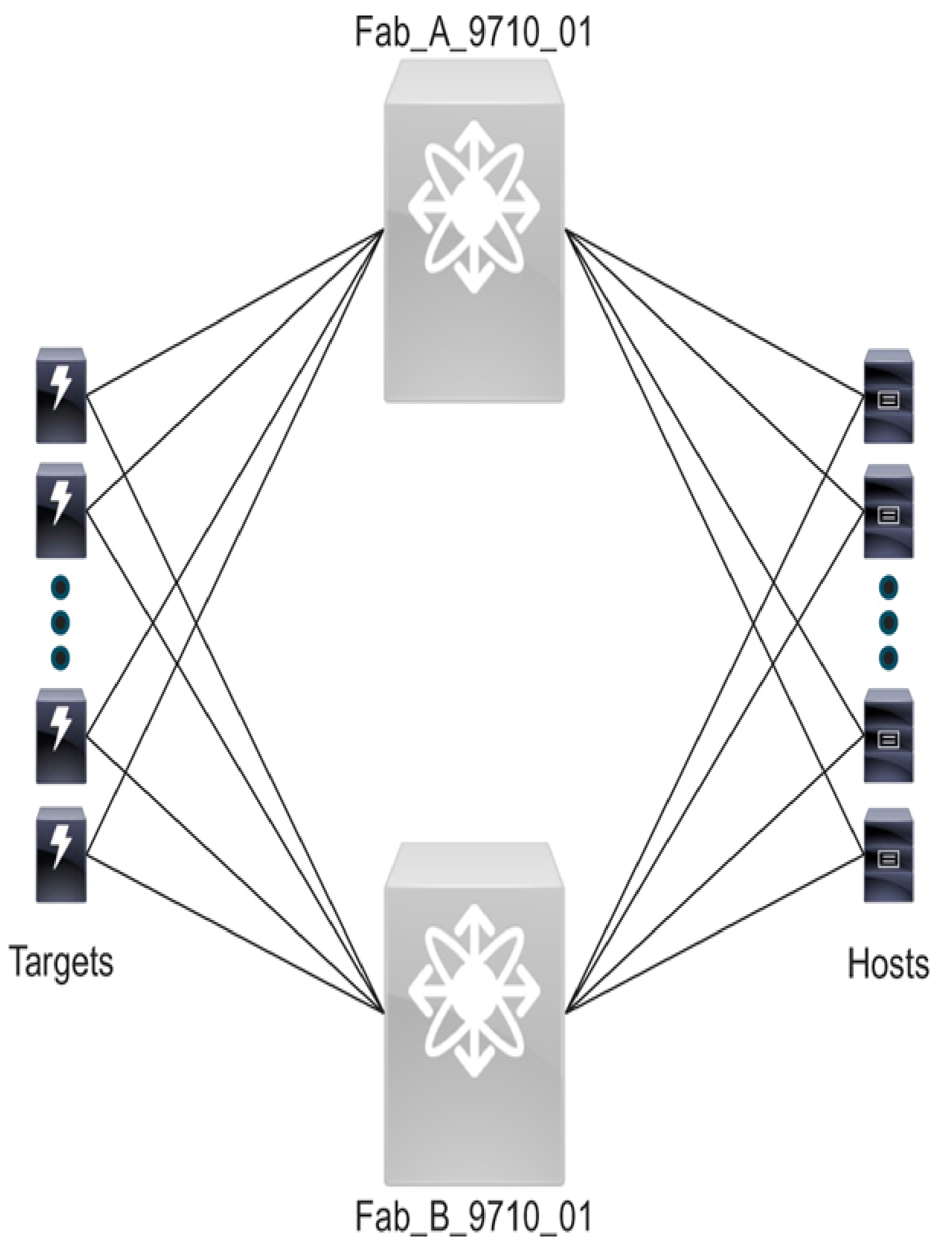


Figure 4-11 *Environment under investigation for congestion*

As Example 4-25 shows, when the command output was analyzed for the first switch, it was found that the model of this switch was MDS 9710 with switchname Fab_A_MDS_9710_01, and it was running NX-OS version 8.2(1).

Example 4-25 *Finding switch model, NX-OS version, and switchname*

```
`show switchname`
Fab_A_MDS_9710_01
```

```
`show version`
...
Software
  BIOS:      version 3.1.0
  kickstart: version 8.2(1)
  system:    version 8.2(1)
...
Hardware
  cisco MDS 9710 (10 Slot) Chassis ("Supervisor
Module-3")
```

Even though the Cisco MDS 9710 switch can hold up to 8 port modules, as Example 4-26 shows, the switch under investigation had just four 16G modules in slots 1 — 4.

Example 4-26 *Finding modules in a Cisco MDS switch*

```
`show module`
Mod  Ports  Module-Type
Model                 Status
---  -----  ----------------------------------- ----
-------------- ----------
1    48     2/4/8/10/16 Gbps Advanced FC Module DS-
X9448-768K9     ok
2    48     2/4/8/10/16 Gbps Advanced FC Module DS-
X9448-768K9     ok
3    48     2/4/8/10/16 Gbps Advanced FC Module DS-
X9448-768K9     ok
4    48     2/4/8/10/16 Gbps Advanced FC Module DS-
X9448-768K9     ok
5    0      Supervisor Module-3                 DS-
X97-SF1-K9      active *
6    0      Supervisor Module-3                 DS-
X97-SF1-K9      ha-standby
```

The other switch's hardware and software configuration was the same except that its switchname was Fab_B_MDS_9710_01. The output from this switch was similar to Example 4-25 and Example 4-26, but not shown for the sake of brevity.

Fabric A Analysis

Continuing the investigation on Fab_A_MDS_9710_01 switch, it was found to be a single-switch fabric. As Example 4-27 shows, no output of the **show topology** command indicated no E_Ports/ISLs on this switch. Therefore, for congestion troubleshooting, there was no need to chase the culprit to another switch. This type of basic information about the fabric design is extremely helpful before investigating congestion symptoms.

Example 4-27 *No output of* ***show topology*** *command indicates single-switch fabric*

```
`show topology`
```

Next, the troubleshooting workflow followed congestion symptoms from highest severity to lowest severity as explained earlier in the section on *Troubleshooting Methodology—Decreasing Severity Levels*. The first thing to check was whether there were any interfaces with credit loss events, which can be found using the **show logging onboard error-stats** command. This command shows instances of credit loss, timeout-drops, and 100ms Tx/Rx credit not available as well as other error counters.

As Example 4-28 shows, fc2/45 on Fab_A_MDS_9710_01 switch had many congestion symptoms.

Example 4-28 *Locating problems on Fab_A_MDS_9710_01 ports using* ***show logging onboard error-stats*** *command*

```
`show logging onboard error-stats`
...

----------------------------
 Show Clock
----------------------------
2020-09-08 14:56:51

---------------------------------
 Module: 2 error-stats
```

```
---------------------------------


-----------------------------------------------------
--------------------------
 ERROR STATISTICS INFORMATION FOR DEVICE DEVICE:
FCMAC
-----------------------------------------------------
--------------------------
 Interface  |                                    |
|    Time Stamp
   Range    |       Error Stat Counter Name      |
Count   | MM/DD/YY HH:MM:SS
            |                                    |
|
-----------------------------------------------------
---------------------------
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO     |
99      | 09/08/20 13:51:03
fc2/45      | F16_TMM_TOLB_TIMEOUT_DROP_CNT      |
5303    | 09/08/20 13:31:03
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO     |
94      | 09/08/20 13:31:03
fc2/45      | F16_TMM_TOLB_TIMEOUT_DROP_CNT      |
4705    | 09/08/20 12:31:41
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO     |
83      | 09/08/20 12:31:41
fc2/45      | FCP_SW_CNTR_CREDIT_LOSS            | 4
| 09/08/20 12:31:41
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO     |
73      | 09/08/20 12:11:21
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO     |
72      | 09/08/20 09:30:58
fc2/45      | F16_TMM_TOLB_TIMEOUT_DROP_CNT      |
4593    | 09/07/20 22:11:08
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO     |
71      | 09/07/20 22:11:08
fc2/45      | FCP_SW_CNTR_CREDIT_LOSS            | 3
| 09/07/20 22:11:08
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO     |
60      | 09/07/20 14:31:00
fc2/45      | F16_TMM_TOLB_TIMEOUT_DROP_CNT      |
3934    | 09/07/20 12:29:18
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO      |
58      | 09/07/20 12:29:18
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO      |
49      | 09/07/20 12:09:38
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO      |
46      | 09/07/20 11:09:37
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO      |
45      | 09/07/20 10:50:16
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO      |
44      | 09/07/20 10:49:36
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO      |
43      | 09/07/20 09:31:16
fc2/45      | F16_TMM_TOLB_TIMEOUT_DROP_CNT       |
3404    | 09/06/20 17:31:00
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO      |
41      | 09/06/20 17:31:00
fc2/45      | F16_TMM_TOLB_TIMEOUT_DROP_CNT       |
2666    | 09/05/20 11:19:31
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO      |
36      | 09/05/20 11:19:31
fc2/45      | FCP_SW_CNTR_CREDIT_LOSS             | 2
| 09/05/20 11:19:31
fc2/45      | F16_TMM_TOLB_TIMEOUT_DROP_CNT       |
1851    | 09/05/20 08:39:29
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO      |
26      | 09/05/20 08:39:29
fc2/45      | F16_TMM_TOLB_TIMEOUT_DROP_CNT       |
1565    | 09/05/20 08:37:29
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO      |
22      | 09/05/20 08:37:29
fc2/45      | F16_TMM_TOLB_TIMEOUT_DROP_CNT       |
693     | 09/05/20 00:51:03
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO      |
13      | 09/05/20 00:51:03
fc2/45      | FCP_SW_CNTR_CREDIT_LOSS             | 1
| 09/05/20 00:51:03
...
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO      | 3
| 09/04/20 23:11:01
...
```

```
fc2/45       | F16_TMM_TOLB_TIMEOUT_DROP_CNT       |
6619    | 09/04/20 11:51:10
```

These egress congestion symptoms on fc2/45 of various severities indicated that the device connected to fc2/45 was a culprit. As explained in the section on *Goals of Troubleshooting*, fc2/45 was an edge port, and egress congestion symptoms on it made the connected device a culprit.

If this environment had used automatic alerting using Port-Monitor feature on the MDS switches, these symptoms would have served as a proactive mechanism to detect congestion. After receiving those alerts, the next steps should be the same as explained next.

Example 4-28 leads to the following observations:

1. Starting at the bottom at 23:11:01, the TX_WT_AVG_B2B_ZERO counter was 3. Then at 00:51:03 it incremented to 13 for a delta of 10. This indicated there were ten 100ms intervals in the 20-second interval ending at 00:51:03 where fc2/45 was at zero remaining-Tx-B2B-credits. This is a total of 1 second which also reflects the increment of CREDIT_LOSS counter at the same time. Typically, these ten 100ms intervals of TX_WT_AVG_B2B_ZERO need not be contiguous but, in this case, those ten 100ms intervals were contiguous because the CREDIT_LOSS counter also incremented.

2. In the same 20-second interval, the F16_TMM_TOLB_TIMEOUT_DROP_CNT was recorded as 693. Looking back in the logs, the previous entry showed a value of 6619. This indicated the hardware counters were cleared. So it was assumed that there were 693 frames dropped in the 20-second interval ending at 00:51:03. This was a loss of information due to clearing the OBFL counters.

3. There were increments of both TX_WT_AVG_B2B_ZERO (Level 1, mild congestion) and F16_TMM_TOLB_TIMEOUT_DROP_CNT (Level

2, moderate congestion) that occured without the CREDIT_LOSS (Level 3, severe congestion) counter incrementing. For example, look at 09/05/20 08:37:29. The TX_WT_AVG_B2B_ZERO shows 22. The previous entry of TX_WT_AVG_B2B_ZERO shows 13 at 09/05/20 00:51:03. This means it incremented by 9 (22 - 13 = 9) indicating nine 100ms intervals of zero remaining-Tx-B2B-credits. Even if these were contiguous representing a single 900ms interval of zero remaining-Tx-B2B-credits, that is not sufficient to trigger the 1-second credit loss recovery. Note the F16_TMM_TOLB_TIMEOUT_DROP_CNT entries at the same time. In the 20-second interval ending at 09/05/20 08:37:29, there were 872 (1565 - 693 = 872) frames dropped due to timeout.

4. As explained in Chapter 2, the section on *B2B Credit Loss and Recover*, credit loss can occur due to severe congestion or physical bit errors on the link which causes a loss of B2B credits. But when there are 100ms intervals of zero remaining-Tx-B2B-credits without credit loss occurring, this indicates that it is due to congestion and not due to bit errors. This is because the credits are returning without going through the credit loss recovery (Link Reset Protocol). So, in this case, it seems clear that the end device connected on fc2/45 experienced severe congestion and not physical link bit errors.

Loss of Information Due to Clearing the OBFL Counters

Example 4-28 indicates a loss of information due to the clearing of the OBFL counters. To make it more evident, Example 4-29 shows only the CREDIT_LOSS entries, which is a subset of the error-stats shown in Example 4-28.

Example 4-29 *Credit loss recovery events on Fab_A_MDS_9710_01 by* ***show logging onboard credit-loss***

```
`show logging onboard credit-loss`


...
```

```
----------------------------------------------------
----------------------------
 ERROR STATISTICS INFORMATION FOR DEVICE DEVICE:
FCMAC
----------------------------------------------------
----------------------------
    Interface   |                             |
|     Time Stamp
      Range     |   Error Stat Counter Name   |
Count   | MM/DD/YY HH:MM:SS
                |                             |
|
----------------------------------------------------
----------------------------
```
fc2/45 | FCP_SW_CNTR_CREDIT_LOSS | 4
| 09/08/20 12:31:41
```
fc2/45          | FCP_SW_CNTR_CREDIT_LOSS     | 3
| 09/07/20 22:11:08
fc2/45          | FCP_SW_CNTR_CREDIT_LOSS     | 2
| 09/05/20 11:19:31
```
fc2/45 | FCP_SW_CNTR_CREDIT_LOSS | 1
| 09/05/20 00:51:03
fc2/45 | FCP_SW_CNTR_CREDIT_LOSS | 9
| 09/04/20 11:31:10
```
fc2/45          | FCP_SW_CNTR_CREDIT_LOSS     | 8
| 09/04/20 10:31:09
fc2/45          | FCP_SW_CNTR_CREDIT_LOSS     | 7
| 09/04/20 08:03:07
fc2/45          | FCP_SW_CNTR_CREDIT_LOSS     | 6
| 09/03/20 17:50:54
fc2/45          | FCP_SW_CNTR_CREDIT_LOSS     | 5
| 09/03/20 16:30:54
fc2/45          | FCP_SW_CNTR_CREDIT_LOSS     | 4
| 09/03/20 16:11:14
fc2/45          | FCP_SW_CNTR_CREDIT_LOSS     | 3
| 09/02/20 11:11:27
fc2/45          | FCP_SW_CNTR_CREDIT_LOSS     | 2
| 09/02/20 10:31:07
fc2/45          | FCP_SW_CNTR_CREDIT_LOSS     | 1
| 09/02/20 09:31:05
fc2/45          | FCP_SW_CNTR_CREDIT_LOSS     | 13
```

```
| 09/01/20 12:31:06
fc2/45          | FCP_SW_CNTR_CREDIT_LOSS     | 12
| 09/01/20 09:51:04
```

Note that the credit loss count increments and apparently resets and starts over again a couple of times. This was because of clearing the hardware counters on a sporadic basis using the **clear statistics** command on Cisco MDS switches. Recall that the count is the total count. To determine how many it incremented, the previous value must be located and then the two subtracted. On 09/04/20 11:31:10 the count was 9 but then later at 09/05/20 00:51:03 the count incremented and showed a value of just 1. That means somewhere between those two times the hardware counters were cleared and the counter just incremented by 1 in the 20-second interval ending at 09/05/20 00:51:03. In this case, all the counters in OBFL error-stats showed this behavior because they were all cleared at the same time.

As explained in the section on *The OBFL Commands—show logging onboard* and is evident from this case, clearing the OBFL entries (using **clear statistics** or **clear logging onboard** command) is not necessary or recommended. Doing this leads to a loss of information, which can hinder analysis and even cause confusion. The OBFL is a date and time-stamped cyclical logfile in which the oldest entries are automatically overwritten when the newest entries are added.

TxWait Analysis

Looking at the OBFL TxWait logs (Example 4-30) for any additional clues, there were many low-level congestion symptoms on fc2/45 in addition to the spikes of higher amounts. The first entry that matched a CREDIT_LOSS entry was on September 08 12:31:41. Match this time in Example 4-28. At that time the TxWait incremented by 644277 ticks. This is 1.61 seconds (644277 * 2.5 / 1000000 = 1.61) or 8% congestion. This is a sum of the time at zero remaining-Tx-B2B-credits and doesn't indicate 1.61 seconds of contiguous time by itself. Only when we look at CREDIT_LOSS (Example 4-28) and TX_WT_AVG_B2B_ZERO (Example 4-

28), we can see that there were ten 100ms intervals, which was 1 full continuous second of zero remaining-Tx-B2B-credits.

Example 4-30 *TxWait events on fc2/45 on Fab_A_MDS_9710_01 by* ***show logging onboard txwait***

```
`show logging onboard txwait`
...
---------------------------------
 Module: 2 txwait
---------------------------------
 Notes:
     - Sampling period is 20 seconds
     - Only txwait delta >= 100 ms are logged
----------------------------------------------------
-------------------------
| Interface | Delta TxWait Time     | Congestion |
Timestamp                 |
|           | 2.5us ticks | seconds |            |
|
----------------------------------------------------
-------------------------
|   fc2/45  |   309054    |    0    |       3%   |
Tue Sep  8 14:51:24 2020 |
|   fc2/45  |   405323    |    1    |       5%   |
Tue Sep  8 14:51:04 2020 |
|   fc2/45  |    56231    |    0    |       0%   |
Tue Sep  8 14:31:23 2020 |
|   fc2/45  |   316771    |    0    |       3%   |
Tue Sep  8 14:31:03 2020 |
|   fc2/45  |    70688    |    0    |       0%   |
Tue Sep  8 14:11:23 2020 |
|   fc2/45  |   436370    |    1    |       5%   |
Tue Sep  8 14:11:03 2020 |
|   fc2/45  |   501390    |    1    |       6%   |
Tue Sep  8 13:51:03 2020 |
|   fc2/45  |   871572    |    2    |      10%   |
Tue Sep  8 13:31:03 2020 |
|   fc2/45  |   191331    |    0    |       2%   |
Tue Sep  8 13:11:03 2020 |
|   fc2/45  |   227862    |    0    |       2%   |
Tue Sep  8 12:51:01 2020 |
|   fc2/45  |   644277    |    1    |       8%   |
Tue Sep  8 12:31:41 2020 |
```

Traffic Utilization (Tx-datarate) Analysis

To complete the analysis of fc2/45, Tx utilization was also investigated. As Example 4-31 shows, these are represented by the tx-datarate counter and are found in OBFL datarate. There were lots of entries for this port which was somewhat unusual. This indicates at times the device was a severe slow draining port but at other times it was running at near 100% Tx utilization. It's important to note that an end device cannot cause slow-drain and over-utilization at the same time. These two indications are mutually exclusive. This means that a switchport can not show high TxWait and high Tx-datarate at the same time. Refer to Chapter 3, the section on *Slow-drain and Over-utilization at the Same Time* for details. This observation is revisted shortly under the section on *Anomaly — Simultaneous TxWait and Tx-datarate*.

To understand these Tx-datarate entries, start with a TX_DATARATE_RISING and then look for a corresponding TX_DATARATE_FALLING entry. For example the TX_DATARATE_RISING at Sep 8 01:54:47 was followed by a TX_DATARATE_FALLING entry at Sep 8 01:55:19. In this case port-monitor had the Tx-datarate polling-interval set to the recommended 10 seconds so the high Tx utilization started 10 seconds before Sep 8 01:54:47 (at Sep 8 01:54:37) and the high Tx utilization ceased at 10 seconds before at Sep 8 01:55:19 (at Sep 8 01:55:09). This made the duration of the high Tx utilization to 32 seconds.

Example 4-31 *Utilization analysis on fc2/45 on Fab_A_MDS_9710_01 by* ***show logging onboard datarate***

```
`show logging onboard datarate`
---------------------------------
 Module: 2 datarate
---------------------------------


----------------------------
```

```
 Show Clock
----------------------------
2020-09-08 14:57:01

---------------------------------
 Module: 2 datarate
---------------------------------
      - DATARATE INFORMATION FROM FCMAC

 ---------------------------------------------------
---------------------------
| Interface | Speed |     Alarm-types      | Rate  |
Timestamp          |
 ---------------------------------------------------
---------------------------
```
| fc2/45 | 8G | TX_DATARATE_FALLING | 71% |
Tue Sep 8 01:55:19 2020 |
| fc2/45 | 8G | TX_DATARATE_RISING | 86% |
Tue Sep 8 01:54:47 2020 |
```
|  fc2/45   |   8G  |  TX_DATARATE_FALLING |  61%  |
Tue Sep  8 01:54:36 2020 |
|  fc2/45   |   8G  |  TX_DATARATE_RISING  |  85%  |
Tue Sep  8 01:54:25 2020 |
|  fc2/45   |   8G  |  TX_DATARATE_FALLING |  35%  |
Tue Sep  8 01:54:15 2020 |
|  fc2/45   |   8G  |  TX_DATARATE_RISING  |  89%  |
Tue Sep  8 01:53:53 2020 |
|  fc2/45   |   8G  |  TX_DATARATE_FALLING |  75%  |
Tue Sep  8 01:53:32 2020 |
|  fc2/45   |   8G  |  TX_DATARATE_RISING  |  89%  |
Tue Sep  8 01:53:00 2020 |
|  fc2/45   |   8G  |  TX_DATARATE_FALLING |  68%  |
Tue Sep  8 01:52:49 2020 |
|  fc2/45   |   8G  |  TX_DATARATE_RISING  |  89%  |
Tue Sep  8 01:52:38 2020 |
|  fc2/45   |   8G  |  TX_DATARATE_FALLING |  77%  |
Tue Sep  8 01:52:27 2020 |
|  fc2/45   |   8G  |  TX_DATARATE_RISING  |  89%  |
Tue Sep  8 01:52:17 2020 |
|  fc2/45   |   8G  |  TX_DATARATE_FALLING |  73%  |
Tue Sep  8 01:52:06 2020 |
|  fc2/45   |   8G  |  TX_DATARATE_RISING  |  89%  |
Tue Sep  8 01:51:55 2020 |
```

Whether these high Tx-datarate events were really instances of over-utilization could be found by investigating if the congestion spread to the end devices that were sending traffic to the culprit end device connected to fc2/45.

Graphical Correlation of Congestion Symptoms

Plotting the congestion symptoms in a graphical format simplifies the understanding and correlation of these events. As Chapter 3, the section on *Detecting Congestion on a Remote Monitoring Platform* explains, these graphs can be created by a vendor-developed platform, such as Cisco Nexus Dashboard Fabric Controller (NDFC), or a 3rd party or custom-developed platform such as the MDS Traffic Monitoring (MTM) app.

The case studies in this Chapter plot the OBFL commands using MatPlotLib Python library. This is done to simplify the interpretation of the commands for educational purposes. We encourage users to invest in developing such utilities to save troubleshooting time, increase accuracy, and help solve problems that can't be solved just by the command outputs. These benefits are demonstrated throughout this and the following case studies.

Refer to Figure 4-12. Y-axis shows different congestion events and the X-axis shows the time with the major tick marks every 12 hours (12 AM and 12 PM) and minor tick marks every hour. The same format is used in the rest of this chapter.

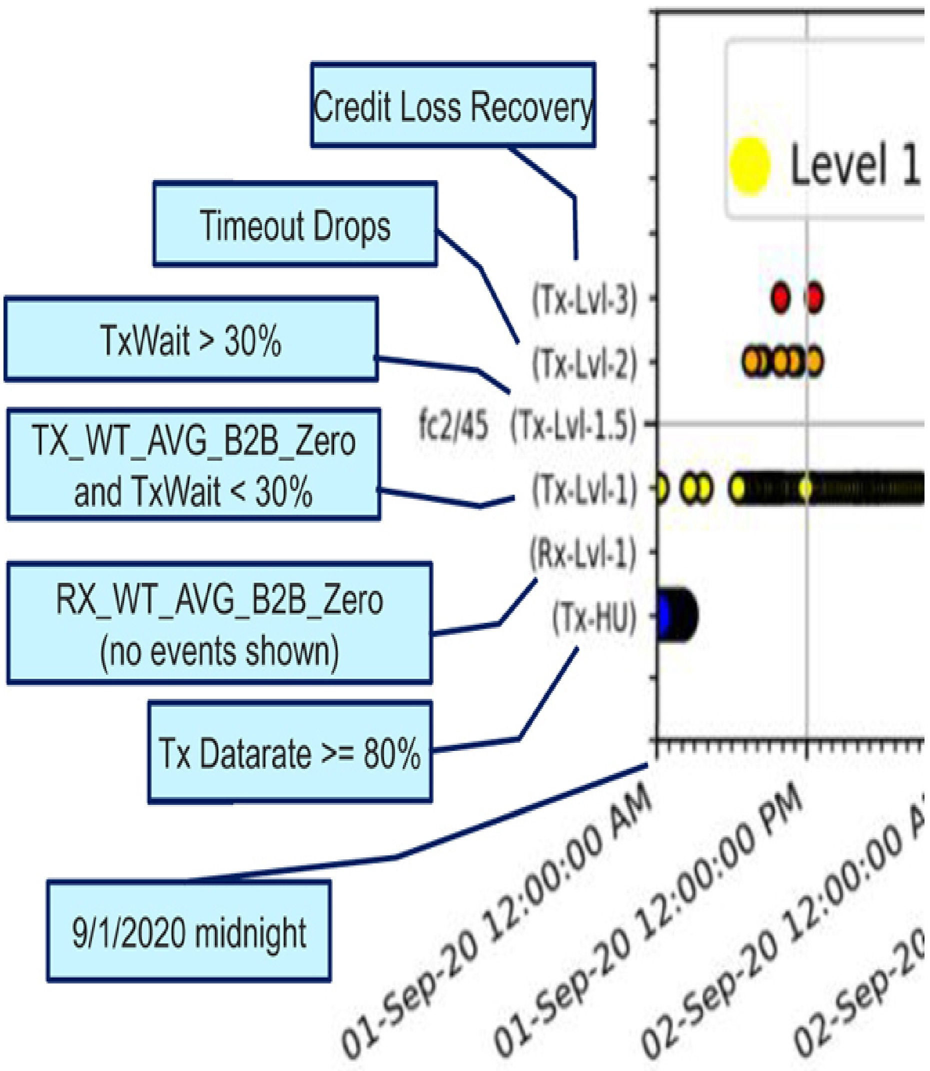


Figure 4-12 *Graphical representation of congestion events*

In Figure 4-12,

- Tx-Lvl-3 — Credit Loss Recovery — Red dots.
- Tx-Lvl-2 — Timeout Drops — Orange dots.
- Tx-Lvl-1.5 — TxWait >= 30% - Khaki dot (none shown in Figure 4-12).
- Tx-Lvl-1 — TxWait < 30% and Tx-Credit-Not-Available(100ms) — Yellow dots.
- Rx-Lvl-1 — Rx-Credit-Not-Available(100ms) — Yellow dots (none shown in Figure 4-12).
- Tx-HU — High Utilization — Tx-datarate — Blue dots.

Figure 4-13 shows all the congestion relayed events for switchname Fab_A_9710_01, interface fc2/45 for the previous 8 days.

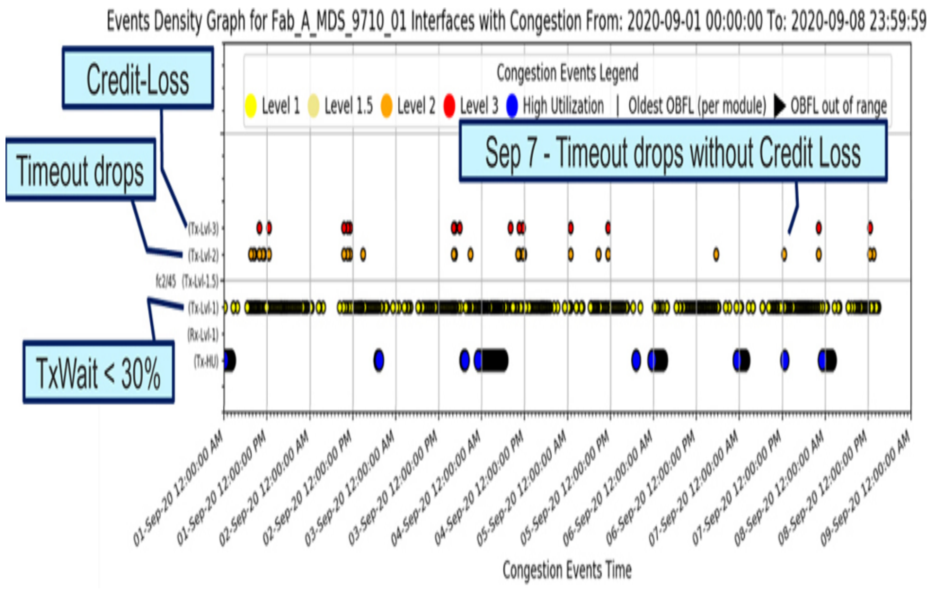


Figure 4-13 *Graph of congestion events on fc2/45 on Fab_A_9710_01 from last 8 days*

Figure 4-13 indicates the following:

1. The credit-loss events (red dots) and most timeout drop events (orange dots) were grouped around 12PM noon on most days.

2. Even on September 07 there was a timeout drop event (orange dot) without a credit loss event (red dot). The OBFL error-stats entry shows it at exactly 12:29:18 (Example 4-28). This seems to indicate that the device was close to credit loss but didn't quite meet the threshold. This was mentioned previously when discussing the observations of Example 4-28.

3. The grouping of the most severe congestion indications at certain times each day was another indication that the credit loss events were not related to physical link bit errors but were just simple severe congestion.

That concludes the congestion analysis of Fabric A.

Fabric B Analysis

Fabric B analysis repeats the same steps as Fabric A.

Interface fc2/45 on switch Fab_B_9710_01 showed severe congestion and was not experiencing physical bit errors. These were similar indications as fc2/45 on Fab_A_9710_01. OBFL error-stats and TxWait on Fab_B_9710_01 showed the same indications at approximately the same times as Fab_A_9710_01. For the sake of brevity, this section skips over the OBFL error-stats outputs. Instead, Figure 4-14 directly shows the graph of congestion symptoms for fc2/45 on switch Fab_B_9710_01.

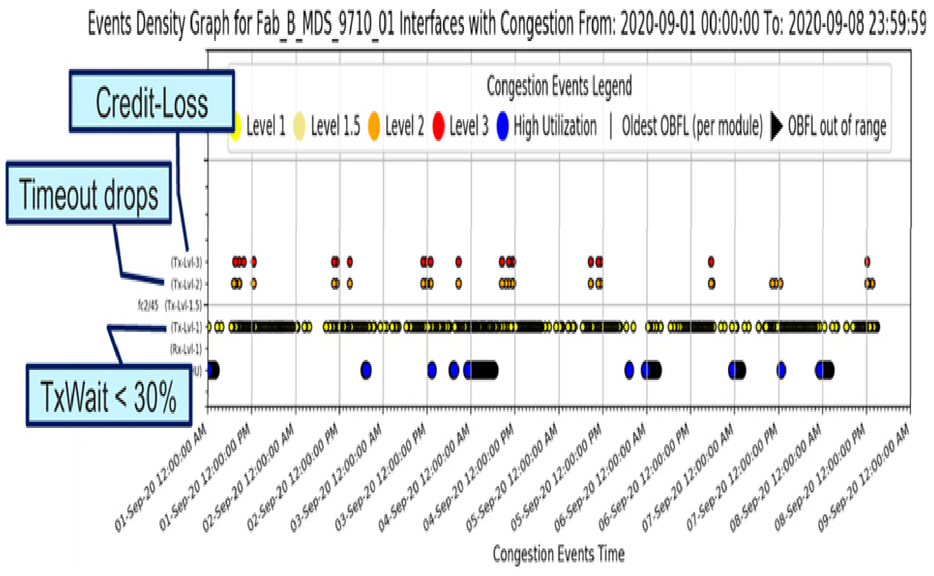


Figure 4-14 *Graph of congestion symptoms from last 8 days on fc2/45 on Fab_B_9710_01*

Figure 4-14 shows that the congestion events in Fabric B, especially the credit loss and timeout drop events, look very similar and are grouped at very similar times as Fabric A (Figure 4-13). This was yet another indication that the credit loss, timeout drops, TxWait, etc. were all a result of a problem in the end device itself and not a result of physical link bit errors. This is because physical bit errors would not be occurring on two separate links with different transceivers, cables, patch panel ports, etc.

To summarize, three reasons indicated that the credit loss events were caused by severe congestion in the end device itself and not physical link bit errors:

1. TX_WT_AVG_B2B_ZERO and TxWait were incrementing without credit loss occurring. This indicated that the end device was withholding B2B credits at times but then restoring the credits later without the Link Reset Protocol occurring.
2. Second, the credit loss events (and timeout drops and other indications) were occurring periodically at similar times each day. Physical link bit errors would occur more randomly and sporadically.
3. Third, credit loss events were occurring on both fabrics at approximately the same time. This again, indicated that the problem was in the end device itself and not physical link bit errors.

Culprit Analysis

The end device connected to fc2/45 on both the switches was identified as the culprit because this switchport showed many egress congestion symptoms of all severities.

The MDS **show tech-support slowdrain** command showed some good information on the end device. Example 4-32 shows the entry of the device from the Fibre Channel Name Server (FCNS) database in Fabirc A.

Example 4-32 *Details of the culprit end device from the FCNS database on Fab_A_MDS_9710_01*

```
`show fcns database detail`

------------------------
VSAN:1501  FCID:0x3d0340
------------------------
port-wwn (vendor)            :10:00:00:00:c9:83:1a:ee
(Emulex)
                              [xxxxx71b_A]
node-wwn                     :20:00:00:00:c9:83:1a:ee
class                        :2,3
node-ip-addr                 :0.0.0.0
ipa                          :ff ff ff ff ff ff ff ff
fc4-types:fc4_features       :scsi-fcp:init
```

```
symbolic-port-name           :0
symbolic-node-name           :Emulex LPe12002-E
FV2.02A1 DV12.0.0.13. HN:xxx71b.
                              OS:Linux
port-type                    :N
port-ip-addr                 :0.0.0.0
fabric-port-wwn              :20:6d:00:de:fb:78:40:00
hard-addr                    :0x000000
permanent-port-wwn (vendor) :10:00:00:00:c9:83:1a:ee
(Emulex)
connected interface          :fc2/45
switch name (IP address)     :Fab_A_MDS_9710_01
(w.x.y.z)
```

Example 4-32 shows the following:

- The culprit end device was assigned to VSAN 1501 on Fab_A_MDS_9710_01.
- Its FCID was 0x3d0340.
- It had registered the FC4 feature of scsi-fcp and was type 'init' (initiator).
- Device-alias name of xxxxx71b_A.
- Hostname was xxx71b.
- The HBA was an Emulex LPe12002-E.
- The OS type was Linux.

Additionally, the **show interface** command (Example 4-33) displays more information such as the operating speed of 8 Gbps.

Example 4-33 *Details of the switchport connected to the culprit end device on Fab_A_MDS_9710_01*

```
`show interface fc2/45`

fc2/45 is up
    Port description is xxx71b_A
    Hardware is Fibre Channel, SFP is short wave
laser w/o OFC (SN)
    Port WWN is 20:6d:00:de:fb:78:40:00
    Admin port mode is auto, trunk mode is on
    snmp link state traps are enabled
    Port mode is F, FCID is 0x3d0340
    Port vsan is 1501
    Admin Speed is auto
    Operating Speed is 8 Gbps
```

The same information from switch Fab_B_MDS_9710_01 was also verified but not shown here for the sake of brevity.

Victim Analysis

Victim devices should be evaluated in the following order:

1. Direct victims.
2. Same-path victims.
3. Indirect victims.

Direct Victims

These victims are in direct communication with the culprit device. To determine the direct victims, we need to evaluate the zoning in VSAN 1501 on Fab_A_MDS_9710_01, specifically the active zoneset (Example 4-34). We looked for all the zoning entries that have FCID of the culprit end device.

Example 4-34 *Active zoneset to find the end devices zoned with the culprit device on Fab_A_MDS_9710_01*

```
` show zoneset active `

zoneset name Fab_A_MDS_9710_01 _08272018 vsan 1501
...
  zone name xxx71b_xio_A vsan 1501
  * fcid 0x3d0340 [device-alias xxx71b_A]
  * fcid 0x3d0060 [device-alias xio-sc1-fc2]
  * fcid 0x3d0040 [device-alias xio-sc2-fc2]
```

Example 4-34 shows a single zone with the culprit end device (fcid 0x3d0340) and two targets (fcid 0x3d0060 and fcid 0x3d0040).

To be sure that the other two end devices were targets, FCNS database can be used as shown in Example 4-35.

Example 4-35 *Details from the FCNS database about the targets that were zoned with the culprit end device*

```
` show fcns database detail `

------------------------
VSAN:1501  FCID:0x3d0060
------------------------
port-wwn (vendor)           :51:4f:0c:50:07:17:0f:11
(Xtremio)
                             [xio-sc1-fc2]
node-wwn                    :51:4f:0c:50:07:16:03:60
class                       :3
node-ip-addr                :0.0.0.0
ipa                         :ff ff ff ff ff ff ff ff
fc4-types:fc4_features      :scsi-fcp:target
symbolic-port-name          :
symbolic-node-name          :QLE2562 FW:v5.08.05
DVR:v8.02.01-k4-tgt
port-type                   :N
port-ip-addr                :0.0.0.0
fabric-port-wwn             :20:87:00:de:fb:78:40:00
hard-addr                   :0x000000
permanent-port-wwn (vendor) :51:4f:0c:50:07:17:0f:11
(Xtremio)
connected interface         :fc3/7
switch name (IP address)    : Fab_A_MDS_9710_01
(w.x.y.z)



------------------------
VSAN:1501  FCID:0x3d0040
------------------------
port-wwn (vendor)           :51:4f:0c:50:07:17:0f:15
(Xtremio)
                             [xio-sc2-fc2]
node-wwn                    :51:4f:0c:50:07:16:03:60
class                       :3
node-ip-addr                :0.0.0.0
```

```
ipa                           :ff ff ff ff ff ff ff ff
fc4-types:fc4_features        :scsi-fcp:target
symbolic-port-name            :
symbolic-node-name            :QLE2562 FW:v5.08.05
DVR:v8.02.01-k4-tgt
port-type                     :N
port-ip-addr                  :0.0.0.0
fabric-port-wwn               :20:c7:00:de:fb:78:40:00
hard-addr                     :0x000000
permanent-port-wwn (vendor)   :51:4f:0c:50:07:17:0f:15
(Xtremio)
connected interface           :fc4/7
switch name (IP address)      : Fab_A_MDS_9710_01
(w.x.y.z)
```

Example 4-35 shows that these two end devices were DellEMC Xtremio storage arrays on interfaces fc3/7 and fc4/7 on switch Fab_A_MDS_9710_01. They registered as scsi-fcp targets and were both using QLogic QLE2562 HBAs.

The operating speed of these two ports was verified to 8 Gbps using the **show interface** command on switch Fab_A_MDS_9710_01. There was no speed-mismatch with the initiator (culprit), which was connected to fc2/45.

Figure 4-15 shows the findings so far.

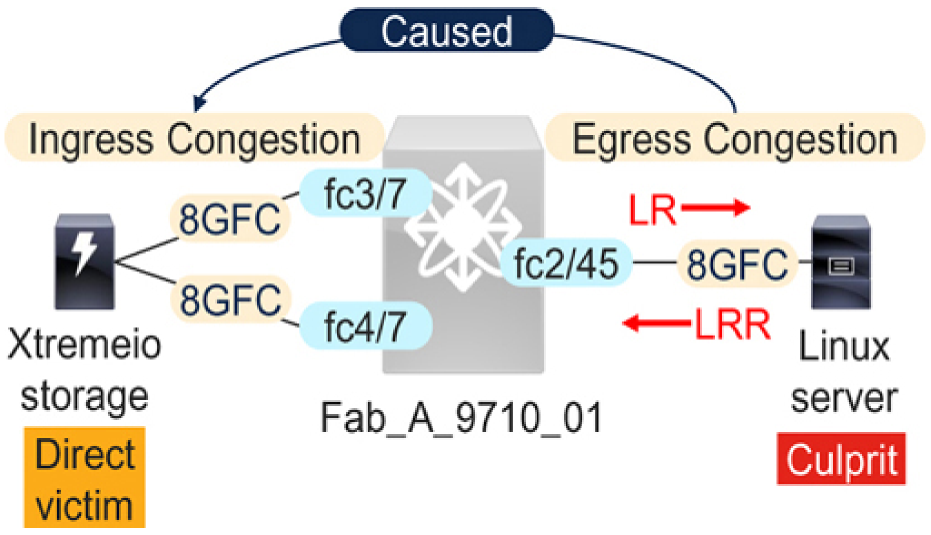


Figure 4-15 *Culprit and direct victim relationship in Fabric A*

The target ports connected to fc3/7 and fc4/7 on Fab_A_MDS_9710_01 were ‘theoretical’ direct victims because, at this point, we didn’t know which target port(s) the culprit server was actually communicating with at the time.

To know if these targets were really direct victims, we looked at symptoms of ingress congestion (RX_WT_AVG_B2B_ZERO) on fc3/7 and fc4/7 and compared their time with egress congestion symptoms (TX_WT_AVG_B2B_ZERO) on fc2/45.

Example 4-36 shows TX_WT_AVG_B2B_ZERO for fc2/45 over 3 days. This information is the same as shown earlier in Example 4-28. It is shown here again to make it easy to compare with ingress congestion symptoms on fc3/7 and fc4/7.

Example 4-36 *TX_WT_AVG_B2B_ZERO for fc2/45 (connected to culprit)*

```
`show logging onboard error-stats`

----------------------------
 Show Clock
----------------------------
2020-09-08 14:56:51
---------------------------------
 Module: 2 error-stats
---------------------------------
----------------------------------------------------
----------------------
 ERROR STATISTICS INFORMATION FOR DEVICE DEVICE:
FCMAC
----------------------------------------------------
----------------------
 Interface |                                  |
|    Time Stamp
   Range    |      Error Stat Counter Name      |
Count | MM/DD/YY HH:MM:SS
           |                                  |
|
----------------------------------------------------
----------------------
```

```
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO    | 99
| 09/08/20 13:51:03
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO    | 94
| 09/08/20 13:31:03
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO    | 83
| 09/08/20 12:31:41
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO    | 73
| 09/08/20 12:11:21
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO    | 72
| 09/08/20 09:30:58
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO    | 71
| 09/07/20 22:11:08
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO    | 60
| 09/07/20 14:31:00
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO    | 58
| 09/07/20 12:29:18
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO    | 49
| 09/07/20 12:09:38
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO    | 46
| 09/07/20 11:09:37
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO    | 45
| 09/07/20 10:50:16
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO    | 44
| 09/07/20 10:49:36
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO    | 43
| 09/07/20 09:31:16
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO    | 41
| 09/06/20 17:31:00
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO    | 36
| 09/05/20 11:19:31
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO    | 26
| 09/05/20 08:39:29
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO    | 22
| 09/05/20 08:37:29
fc2/45      | FCP_SW_CNTR_TX_WT_AVG_B2B_ZERO    | 13
| 09/05/20 00:51:03
```

Example 4-37 shows RX_WT_AVG_B2B_ZERO for fc3/7 over 3 days. In Example 4-37, we added asterisks next to the RX_WT_AVG_B2B_ZERO to show the entries that match the times of the TX_WT_AVG_B2B_ZERO for fc2/45 shown in

Example 4-36. These astricks are not added automatically by the switch.

Example 4-37 *RX_WT_AVG_B2B_ZERO for fc3/7 (potential direct victim)*

```
`show logging onboard error-stats`
----------------------------
 Show Clock
----------------------------
2020-09-08 14:56:51


---------------------------------
 Module: 3 error-stats
---------------------------------
----------------------------------------------------
-------------------------
 ERROR STATISTICS INFORMATION FOR DEVICE DEVICE:
FCMAC
----------------------------------------------------
-------------------------
 Interface  |                                  |
|    Time Stamp
   Range    |     Error Stat Counter Name      |
Count   | MM/DD/YY HH:MM:SS
            |                                  |
|
----------------------------------------------------
-------------------------
fc3/7       | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO   | 86
| 09/08/20 13:50:53
fc3/7       | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO   | 81
| 09/08/20 13:31:13  *
fc3/7       | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO   | 73
| 09/08/20 12:31:31  *
fc3/7       | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO   | 67
| 09/08/20 06:33:47
fc3/7       | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO   | 66
| 09/08/20 05:28:26
fc3/7       | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO   | 65
| 09/07/20 23:04:01
fc3/7       | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO   | 64
```

```
| 09/07/20 22:11:00  *
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 58
| 09/07/20 21:59:00
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 57
| 09/07/20 21:31:19
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 56
| 09/07/20 15:13:54
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 55
| 09/07/20 14:30:53
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 54
| 09/07/20 14:28:33
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 53
| 09/07/20 12:29:32
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 47
| 09/07/20 12:09:52  *
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 45
| 09/07/20 11:09:50  *
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 44
| 09/07/20 10:49:30  *
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 43
| 09/07/20 09:31:10  *
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 42
| 09/07/20 05:42:46
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 41
| 09/07/20 05:30:46
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 40
| 09/07/20 02:34:23
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 39
| 09/06/20 17:31:16  *
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 35
| 09/06/20 11:51:11
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 34
| 09/06/20 08:59:29
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 33
| 09/06/20 08:06:48
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 32
| 09/06/20 06:27:27
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 31
| 09/06/20 03:42:25
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 30
| 09/05/20 20:07:38
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 29
```

```
| 09/05/20 14:01:13
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 28
| 09/05/20 11:19:31  *
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 20
| 09/05/20 10:59:30
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 19
| 09/05/20 10:19:30
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 18
| 09/05/20 08:39:28  *
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 16
| 09/05/20 08:37:28
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 11
| 09/05/20 07:59:28
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 10
| 09/05/20 06:14:46
fc3/7        | FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO  | 9
| 09/05/20 00:51:03  *
```

This clearly showed a positive correlation making the target port connected to interface fc3/7 an actual victim and not just theoretical.

Repeating the same steps for fc4/7 led to the same conclusion that even fc4/7 was the direct victim of the culprit connected to fc2/45. Those steps are not repeated for the sake of brevity.

Instead, the congestion events on the culprit-connected switchport (fc2/45) are plotted on the same graphs as with fc3/7 and fc4/7 (connected to the direct victims). Figure 4-16 shows this graph. There was a strong correlation between the Tx congestion on fc2/45 and the Rx congestion on fc3/7 and fc4/7. Those correlations are circled.

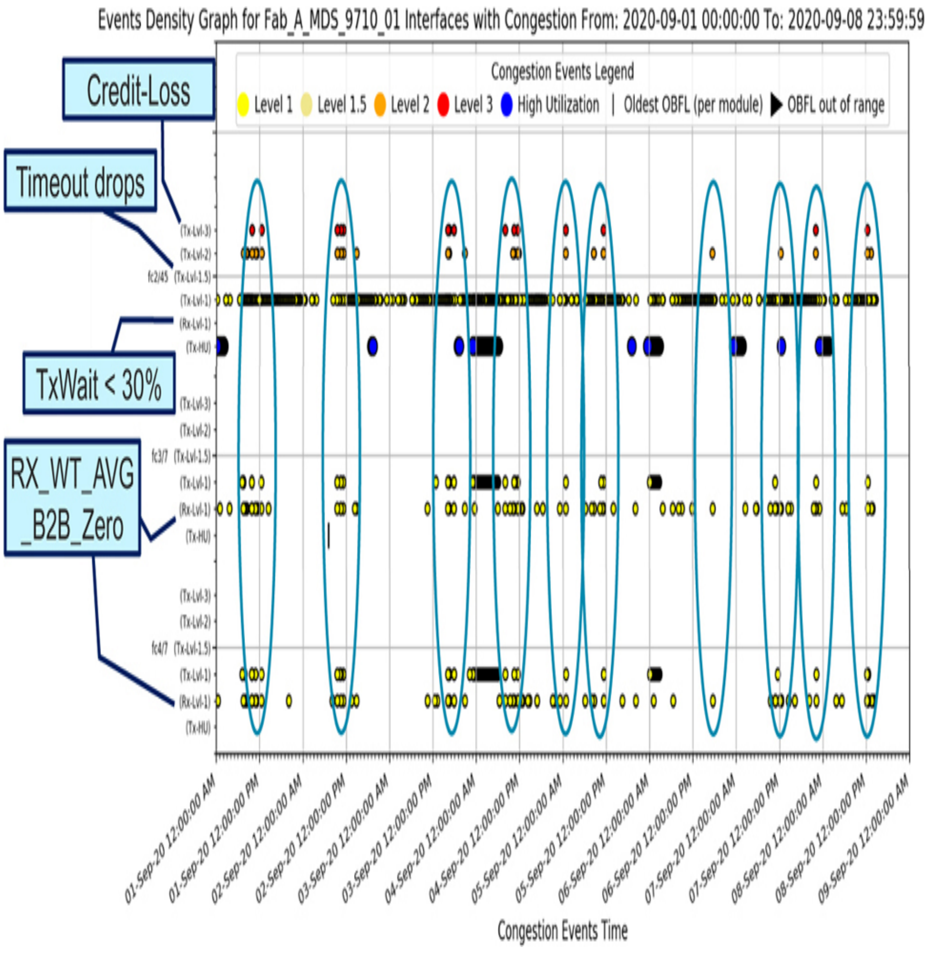


Figure 4-16 *Graph of congestion events on culprit-connected and direct-victim-connected switchports on Fabric A*

Going through the same exercise for Fab_B_MDS_9710_01 we saw similar results on Fabric B. When the active zoneset was checked, the initiator on fc2/45 was once again in just a single zone with two DellEMC Xtremio target ports. However, the two targets were on ports fc1/7 and fc2/7 in Fabric B instead of fc3/7 and fc4/7 in Fabric A.

Skipping over the OBFL error-stats and OBFL TxWait output for the sake of brevity and plotting those events for the eight days prior to when the **show tech-support slowdrain** was created, we saw a very similar pattern shown in Figure 4-17.

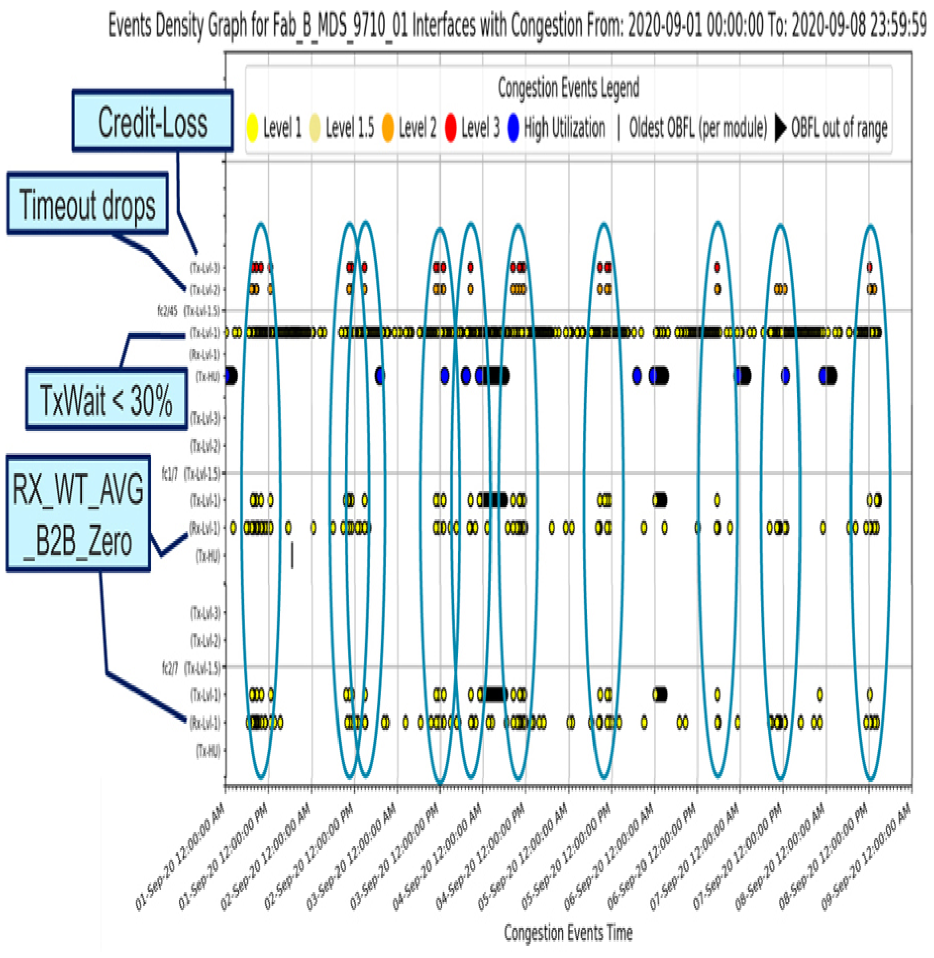


Figure 4-17 *Graph of congestion events on culprit-connected and direct-victim-connected switchports on Fabric B*

Figure 4-16 and Figure 4-17 indicate that the two targets in both fabrics were truly direct victims since they all experience ingress congestion as fc2/45 experiences egress congestion.

Investigating Tx-datarate on Culprit-Connected Switchport

As reported earlier in the section on *Traffic Utilization (Tx-datarate) Analysis*, the culprit-connected switchport reported some periods of high egress utilization. Figure 4-18 shows the graph of congestion symptoms on interface fc2/45, fc3/7, and fc4/7 again in Fabric A. This is the same graph as Figure 4-16, but this time the focus is on the high utilization events.

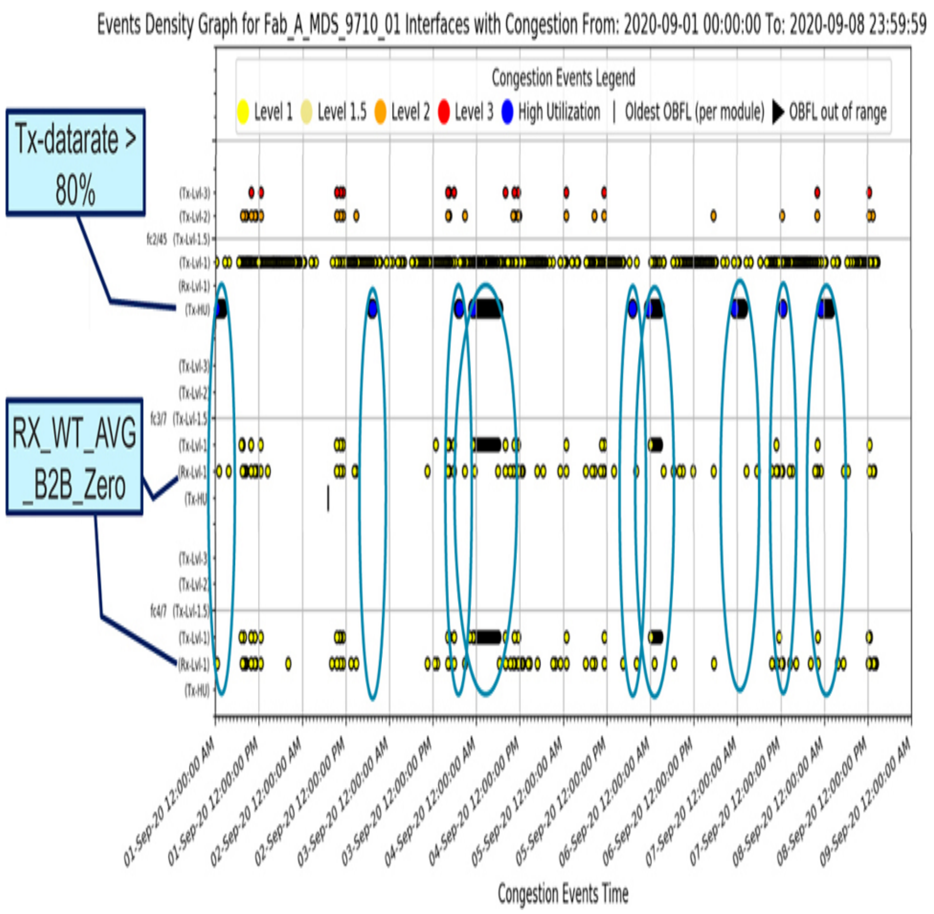


Figure 4-18 *Graph for culprit-connected and direct-victim-connected switchports on Fabric A focusing on high utilization*

If the high Tx utilization on culprit-connected switchport was causing backpressure (ingress congestion on the two target ports), then that could be an indication that the culprit device (initiator) connected to fc2/45 was requesting much more data than can be delivered (over-utilization) to it by the switch. To determine this, those Tx-datarate events on fc2/45 were time correlated with the RX_WT_AVG_B2B_ZERO entries on fc3/7 and fc4/7. But there was only a weak time correlation between them. That may indicate that the high utilization events, at least at this time, were not causing severe backpressure in Fabric A.

However, remember that the RX_WT_AVG_B2B_ZERO counter is a relatively coarse measurement of time since there must be 100ms of continuous zero remaining-Rx-B2B-credits. If there is 99ms of continuous zero remaining-Rx-B2B-credits, then no indication would be recorded. So, in this case, there may be more ingress congestion on the target-connected switchports, but those symptoms might not have been captured due to the granularity of the counter. The RX_WT_AVG_B2B_ZERO counter was the best ingress congestion symptom available on 16G modules in this case. The 64G MDS switches and modules support RxWait for measuring ingress congestion at a granularity of 2.5 microseconds.

Figure 4-19 shows a similar graph for Fabric B. There was a similar weak time correlation between the Tx-datarate entries on fc2/45 (culprit-connected switchport) and the RX_WT_AVG_B2B_ZERO entries on fc1/7 and fc2/7 (victim-connected switchports).

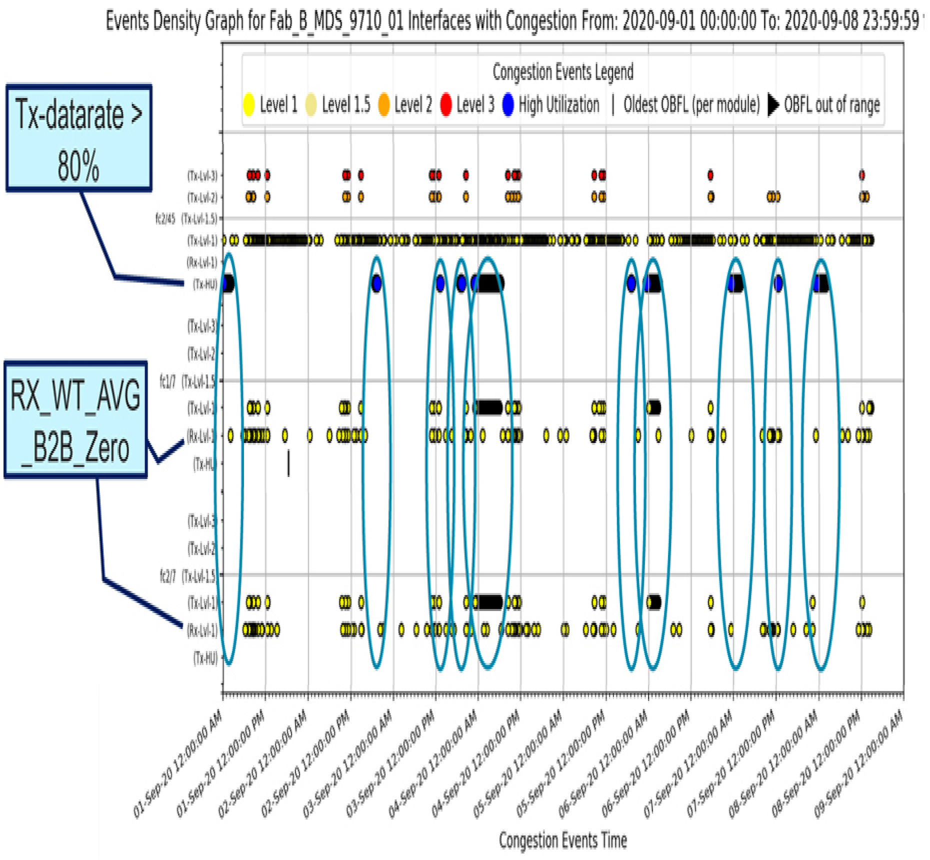


Figure 4-19 *Graph for culprit-connected and direct-victim-connected switchports on Fabric B focusing on high utilization*

The conclusion is that there was a weak time correlation between the Tx-datarate entries on the culprit-connected switchport and the RX_WT_AVG_B2B_ZERO entries on the two target-connected switchports. Also, recall that the initiator port and target ports have the same speed of 8 GFC. Therefore, additional bandwidth on the culprit-connected switchport is not needed at this time (but it should be monitored).

Typically, the analysis to find the direct victim would end here. However, in this case, some anomalies are seen, which are worth investigating further.

Anomaly — Simultaneous TxWait and Tx-datarate

There are a few observations in Figure 4-18 that at first glance look “anomalous”.

First, if we focus on the culprit-connected switchport, fc2/45, and the indications of high Tx-datarate (Tx-HU — blue dots) and look above in the same graph to the Tx-Lvl-1 (TxWait <30%) row, it appears that there is simultaneous high Tx-datarate at the same time as TxWait. As Chapter 3, the section on *Slow-drain and Over-utilization at the Same Time* explains, high Tx utilization cannot occur at the same time as a significant TxWait. The higher the TxWait the lower the Tx utilization on that interface.

If we zoom in and only display the 30 minutes prior to midnight on September 4, we get a better look (Figure 4-20). In this 30-minute graph, most indications of high egress utilization occur at slightly different times than the TxWait < 30% indications. So most are not happening together.

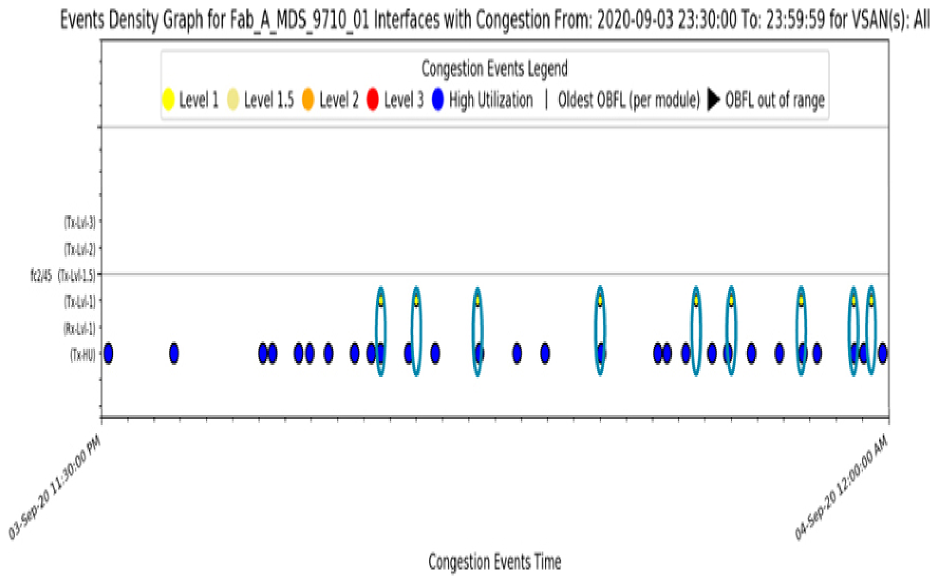


Figure 4-20 *Fabric A — Investigating anomaly for TxWait and Tx-datarate at the same time*

For the few TxWait and Tx-datarate entries that seem to be occurring at similar times, if we look even closer (Example 4-38), we see that all these TxWait entries at that time were very small percentages. So, it's possible that there was a small amount of TxWait and still achieved a high Tx-datarate on that port.

Example 4-38 *TxWait analysis on Fab_A_MDS_9710_01*

```
` show logging onboard txwait `

---------------------------------
 Module: 2 txwait
---------------------------------
 Notes:
     - Sampling period is 20 seconds
     - Only txwait delta >= 100 ms are logged

----------------------------------------------------
-------------------------
| Interface | Delta TxWait Time     | Congestion |
Timestamp                |
|           | 2.5us ticks | seconds |            |
|
----------------------------------------------------
-------------------------
|   fc2/45  |    67640    |    0    |       0%   |
Thu Sep  3 23:59:20 2020 |
|   fc2/45  |   102128    |    0    |       1%   |
Thu Sep  3 23:58:40 2020 |
|   fc2/45  |    75209    |    0    |       0%   |
Thu Sep  3 23:56:40 2020 |
|   fc2/45  |    50419    |    0    |       0%   |
Thu Sep  3 23:54:00 2020 |
|   fc2/45  |    53714    |    0    |       0%   |
Thu Sep  3 23:52:40 2020 |
|   fc2/45  |    51174    |    0    |       0%   |
Thu Sep  3 23:49:00 2020 |
|   fc2/45  |    77379    |    0    |       0%   |
Thu Sep  3 23:44:20 2020 |
|    fc2/45   |    115811     |     0     |         1%    |
Thu Sep  3 23:42:00 2020 |
|    fc2/45   |     46593     |     0     |         0%    |
Thu Sep  3 23:40:39 2020 |
```

So, after investigation, it was concluded that what seemed like an anomaly in the beginning, is really not an anomaly. This is because most of the TxWait is not happening at exactly the same time as the high Tx-datarate and even in the instances where it is, the TxWait is very low.

Anomaly — End Device Reflecting Congestion in the Fabric in Opposite Direction

The second anomaly is even more interesting. In the graphs shown in Figure 4-18 and Figure 4-19, for both Fabrics, there seems to be a time correlation between the Tx-datarate > 80% (high utilization) events on host port (fc2/45) and TxWait < 30% (Tx-Lvl-1 on the Y-Axis) on both target ports (such as fc3/7 and fc4/7 in Fabric A). This is interesting because the Tx-datarate > 80% (high utilization) events indicate traffic going towards the host on fc2/45 and the TxWait < 30% entries to the two targets on fc3/7 and fc4/7 indicate congestion in traffic going toward the two arrays which is the opposite direction of the Tx-datarate entries. So, somehow, the large number of Read I/O operations, which is generating a large amount of traffic towards the host is causing some minor congestion even in the flow of command frames (which initiate I/O operations) going in the opposite direction. These details are covered in Chapter 5, the section on *I/O Operations and Network Traffic Patterns*.

Figure 4-21 is the same as Figure 4-18, except that the focus is different. Notice the two circled times of high utilization on fc2/45 and the corresponding TxWait entries on fc3/7 and fc4/7 (inside circles).

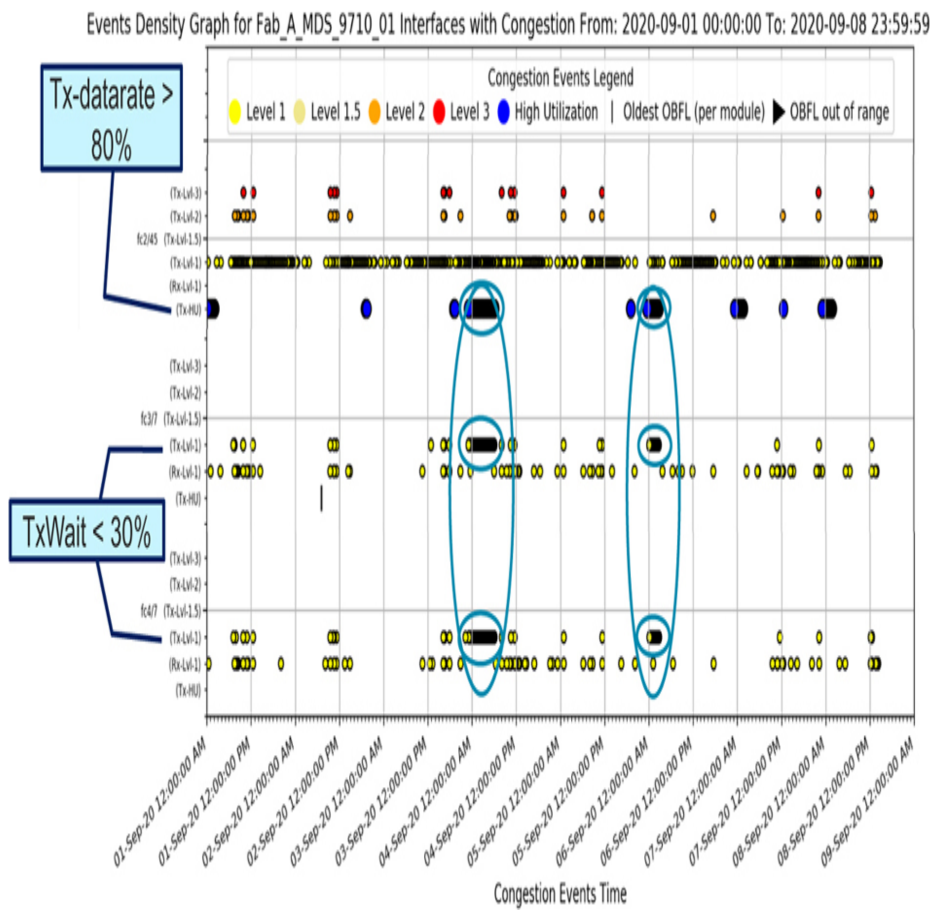


Figure 4-21 *Fabric A — Investigating anomaly*

A close analysis of the TxWait entries for fc3/7 shows that low-level TxWait starts around 11PM (23:03:02) on September 3rd and is fairly continuous till about 6AM the next morning (Example 4-39). This time matches with the high Tx-datarate entries on fc2/45 (Example 4-31).

Example 4-39 *TxWait on target-connected switchport fc3/7*

```
` show loggin onboard txwait `


---------------------------------
 Module: 3 txwait count
---------------------------------


----------------------------
 Show Clock
----------------------------
2020-09-08 14:57:00


---------------------------------
```

```
 Module: 3 txwait
---------------------------------
 Notes:
     - Sampling period is 20 seconds
     - Only txwait delta >= 100 ms are logged

----------------------------------------------------
-------------------------
| Interface | Delta TxWait Time     | Congestion |
Timestamp                |
|           | 2.5us ticks | seconds |            |
|
----------------------------------------------------
-------------------------
...
|   fc3/7   |   117571    |    0    |       1%   |
Sat Sep  5 00:51:03 2020 |
|   fc3/7   |   221867    |    0    |       2%   |
Fri Sep  4 11:31:11 2020 |
|   fc3/7   |    50575    |    0    |       0%   |
Fri Sep  4 10:51:11 2020 |
|   fc3/7   |   196307    |    0    |       2%   |
Fri Sep  4 10:31:11 2020 |
|   fc3/7   |   125900    |    0    |       1%   |
Fri Sep  4 08:03:09 2020 |
|   fc3/7   |   100716    |    0    |       1%   |
Fri Sep  4 06:05:27 2020 |
|   fc3/7   |   194388    |    0    |       2%   |
Fri Sep  4 06:04:27 2020 |
|   fc3/7   |   291829    |    0    |       3%   |
Fri Sep  4 06:03:27 2020 |
|   fc3/7   |   470741    |    1    |       5%   |
Fri Sep  4 06:02:27 2020 |
|   fc3/7   |   285401    |    0    |       3%   |
Fri Sep  4 06:01:27 2020 |
|   fc3/7   |    41430    |    0    |       0%   |
Fri Sep  4 05:55:07 2020 |
...274 lines deleted
|   fc3/7   |   198507    |    0    |       2%   |
Thu Sep  3 23:10:42 2020 |
|   fc3/7   |   338465    |    0    |       4%   |
Thu Sep  3 23:09:42 2020 |
|    fc3/7    |    218939     |     0     |         2%    |
Thu Sep  3 23:06:42 2020 |
|    fc3/7    |    168113     |     0     |         2%    |
Thu Sep  3 23:05:42 2020 |
|    fc3/7    |     71623     |     0     |         0%    |
Thu Sep  3 23:04:42 2020 |
|    fc3/7    |    223882     |     0     |         2%    |
Thu Sep  3 23:03:22 2020 |
|    fc3/7    |     59713     |     0     |         0%    |
Thu Sep  3 17:50:59 2020 |
|    fc3/7    |    207993     |     0     |         2%    |
Thu Sep  3 16:30:58 2020 |
|    fc3/7    |    171679     |     0     |         2%    |
Thu Sep  3 16:11:18 2020 |
|    fc3/7    |     71374     |     0     |         0%    |
Thu Sep  3 12:51:14 2020 |
```

TxWait entries for fc4/7 in Fabric A showed a similar pattern.

Looking back at the graph in Figure 4-19 Fabric B shows almost identical behavior between fc2/45 and the two targets on fc1/7 and fc2/7.

What was causing this behavior? We could not be sure because the exact reason for this anomaly can only be found by a detailed investigation of the array, which is beyond the scope of the book.

But we suspected that the high egress utilization (Tx-datarate) on the host-connected switchport (fc2/45) might have caused congestion due to over-utilization. This congestion spreading resulted in a significant amount of ingress congestion on the target-connected switchports, but it went undetected because of the RX_WT_AVG_B2B_ZERO counter. As mentioned, this counter has a minimum granularity of 100ms, therefore if the switch exerts 99ms of continuous zero remaining-Rx-B2B-credits to the target ports, the MDS does not record any RX_WT_AVG_B2B_ZERO. It needs to be a full 100ms of time. This significant backpressure to the arrays was because of the excessive traffic sent by the array itself, which further caused over-utilization of the initiator port. We suspected that due to some sort of internal resource contention, to reduce the egress traffic rate from the array, it attempted to reduce the ingress traffic rate. A reduced ingress traffic rate also means a reduced rate of new incoming I/O operations, and when the rate of I/O operations reduces, the throughput also reduces. To reduce the ingress traffic rate, the array used the B2B flow control that resulted in TxWait on the directly connected MDS switchports. This happened at the same time as high egress utilization (Tx-datarate) of the initiator-connected switchport.

Another way to look at this anomaly is that the target ports reflected congestion back to the fabric in the opposite direction. Typically, congestion is directional. But this case shows that when some end devices observe egress congestion (seen as ingress congestion on their directly connected switchports), they exert ingress congestion (seen as egress congestion on their directly connected switchports).

Same-Path Victims

There are no same-path victims because this is a single-switch fabric.

Indirect Victims

As was mentioned earlier, indirect victims are the devices that communicate with the direct victims and same-path victims. To find the indirect victims, the active zoneset showed the end devices that the two Xtremio storage array ports were zoned with.

As Example 4-40 shows, the active zoneset on Fab_A_MDS_9710_01 had just one additional zone containing both direct victim targets.

Example 4-40 *Fabric A — Locating the end devices that were in communication with the direct victim*

```
`show zoneset active`

zoneset name Fab_A_MDS_9710_01 _08272018 vsan 1501
...
  zone name xxx943a_xio_A vsan 1501
  * fcid 0x3d0180 [device-alias xxx943a_A]
```

```
   * fcid 0x3d0060 [device-alias xio-sc1-fc2]
   * fcid 0x3d0040 [device-alias xio-sc2-fc2]
```

The potential indirect victim had an FCID of 0x3d0180 in Fabric A. More details about it can be found using FCNS database. As Example 4-41 shows, it was a Linux initiator connected to fc3/40 on Fab_A_MDS_9710_01.

Example 4-41 *Fabric A — Details about the potential indirect victim from FCNS database*

```
` show fcns database detail `
...
VSAN:1501  FCID:0x3d0180
port-wwn (vendor)           :10:00:00:90:fa:af:77:48
(Emulex)
                             [xxxxxxxxx09943a_A]
node-wwn                    :20:00:00:90:fa:af:77:48
class                       :2,3
node-ip-addr                :0.0.0.0
ipa                         :ff ff ff ff ff ff ff ff
fc4-types:fc4_features      :scsi-fcp:init
symbolic-port-name          :Emulex PPN-
10:00:00:90:fa:af:77:48
symbolic-node-name          :Emulex AJ763B/AH403A
FV2.01A4 DV11.0.1.6 HN⊗none)
                             OS:Linux
port-type                   :N
port-ip-addr                :0.0.0.0
fabric-port-wwn             :20:a8:00:de:fb:78:40:00
hard-addr                   :0x000000
permanent-port-wwn (vendor) :10:00:00:90:fa:af:77:48
(Emulex)
connected interface         :fc3/40
switch name (IP address)    : Fab_A_MDS_9710_01
(w.x.y.z)
```

The sames steps were repeated on Fabric B. As Example 4-42 shows, the active zoneset on Fab_B_MDS_9710_01 had just one additional zone containing both direct victim targets.

Example 4-42 *Fabric B — Locating the end devices that are in communication with the direct victim*

```
`show zoneset active`

zoneset name Fab_B_MDS_9710_01 _08272018 vsan 1601
...
  zone name xxx643c_xio_B vsan 1601
  * fcid 0x3d0100 [device-alias xxx643c_B]
  * fcid 0x3d0320 [device-alias xio-sc1-fc1]
  * fcid 0x3d0120 [device-alias xio-sc2-fc1]
```

The potential indirect victim had an FCID of 0x3d0100 in Fabric B. More details about it can be found using FCNS database. As Example 4-43 shows, it was a Linux initiator connected to fc3/42 on Fab_B_MDS_9710_01.

Example 4-43 *Fabric A - Details about the potential indirect victim from FCNS database*

```
` show fcns database detail `
...
------------------------
VSAN:1601  FCID:0x3d0100
------------------------
port-wwn (vendor)           :10:00:00:90:fa:bb:ad:d9
(Emulex)
                             [xxx643c_B]
node-wwn                    :20:00:00:90:fa:bb:ad:d9
class                       :2,3
node-ip-addr                :0.0.0.0
ipa                         :ff ff ff ff ff ff ff ff
fc4-types:fc4_features      :scsi-fcp:init
symbolic-port-name          :Emulex PPN-
10:00:00:90:fa:bb:ad:d9
symbolic-node-name          :Emulex AJ763B/AH403A
FV2.01A4 DV11.0.1.6 HN:(none)
                             OS:Linux
port-type                   :N
port-ip-addr                :0.0.0.0
fabric-port-wwn             :20:aa:00:de:fb:78:4e:00
hard-addr                   :0x000000
```

```
permanent-port-wwn (vendor) :10:00:00:90:fa:bb:ad:d9
(Emulex)
connected interface         :fc3/42
switch name (IP address)    : Fab_B_MDS_9710_01
(w.x.y.z)
```

Example 4-41 and Example 4-43 show that Fabric A had an indirect victim on interface fc3/40, whereas Fabric B had it is on fc3/42. These two ports on two different fabrics connect to the same end device as verified by the operating system of that end device.

A related note—it is a best practice to use identical port numbers on both fabrics, although not a technical requirement and not even related to finding the indirect victim.

Figure 4-22 explains the findings.

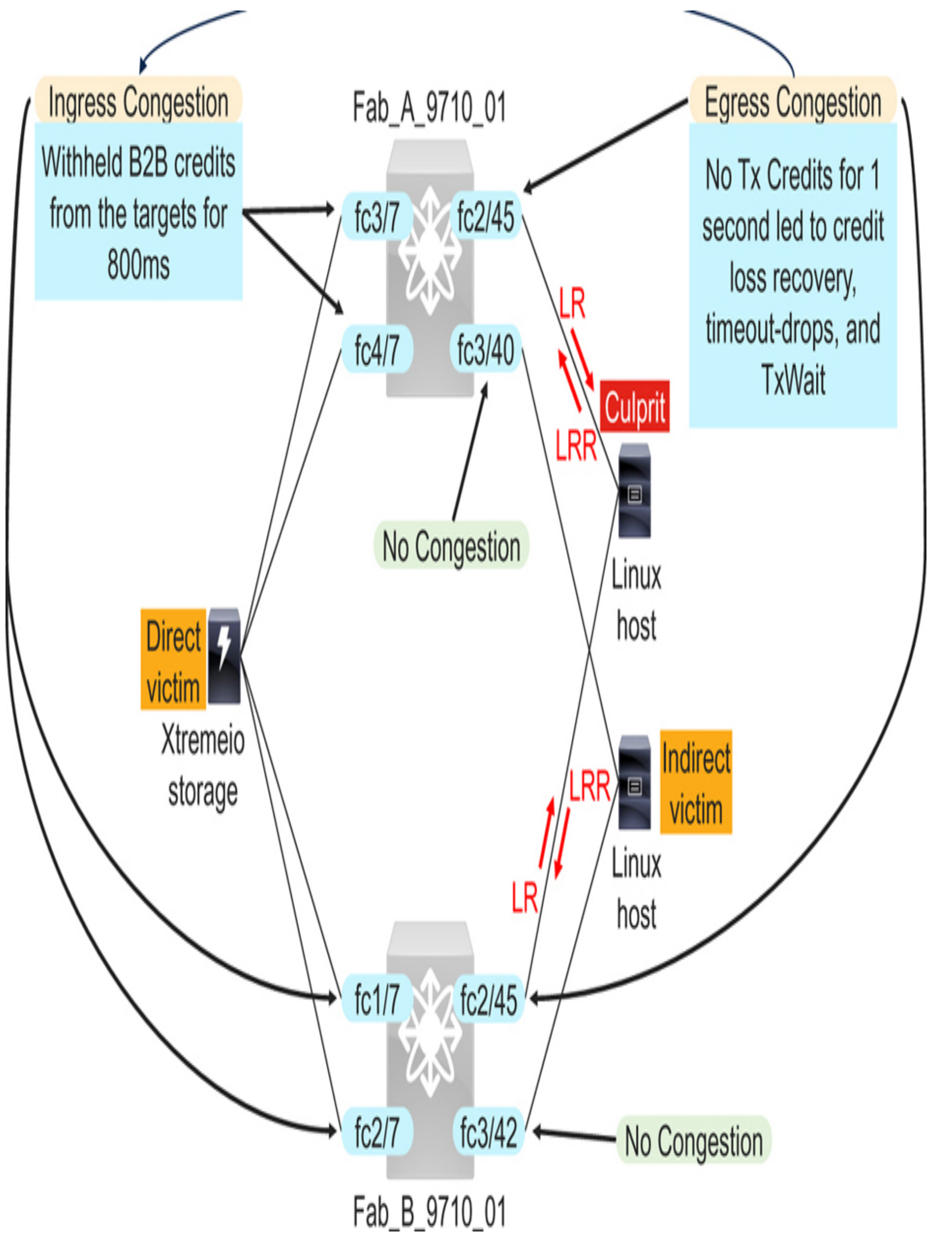


Figure 4-22 *Culprit and Victims*

Case Study 1—Summary

Even relatively simple, single-switch fabrics can have congestion problems. The congestion spreading will be less, resulting in fewer victims of all types. In this case, as Figure 4-22 shows, the culprit Linux host was zoned with just two storage ports and those two storage ports were zoned with just one other host. This makes a total of one culprit and three victims (two direct victims and one indirect victim) in each fabric. This is about as simple and straightforward as it gets.

In this case, the host connected on fc2/45 on both fabrics was experiencing severe congestion resulting in the MDS initiating Credit Loss Recovery at times. Other times the congestion wasn’t that severe and even though there were timeout drops the duration at zero remaining-Tx-B2B-credits wasn’t long enough for the MDS to initiate Credit Loss Recovery. Because of this fact and the periodic nature of the congestion as well as the congestion occurring on two separate physical connections, it was concluded that this congestion was not due to bit errors but just severe congestion in the server.

The reason for the Linux host (connected to fc2/45) to be a culprit could be because of many possible reasons, such as excessive load, bugs, or traffic patterns. These details were outside the scope of troubleshooting network congestion.

This case study demonstrates finding the cause and the source of congestion (culprit) and the end devices that are adversely affected by it (victims). For educational purposes, this case study provides a much more detailed workflow than is typically needed. It also explains some side details such as analyzing the anomalies of simultaneous TxWait and Tx-datarate, and when the end devices reflect congestion back to the fabric.

Case Study 2—Credit Loss Recovery Causing Frame Drops

A financial services company reported ‘Event 153’ errors on a specific Windows server on June 22, 2021. These errors indicate SCSI I/O errors. The server was connected to two fabrics, called Fabric A and Fabric B. The errors were not associated with a specific fabric.

As Figure 4-23 shows, these fabrics had an edge-core design with two core switches and 47 edge switches. All hosts (the initiators) were connected to the edge switches and the storage arrays (the targets) were connected to the two core MDS 9718 switches named Fab_A_MDS_9718_01 and Fab_A_MDS_9718_02 in Fabric A, and Fab_B_MDS_9718_01 and Fab_B_MDS_9718_02 in Fabric

B. Each edge switch was connected to each core switch by a port-channel containing multiple member interfaces. Figure 4-23 shows Fabric A. The design of Fabric B was similar except for different switch names.

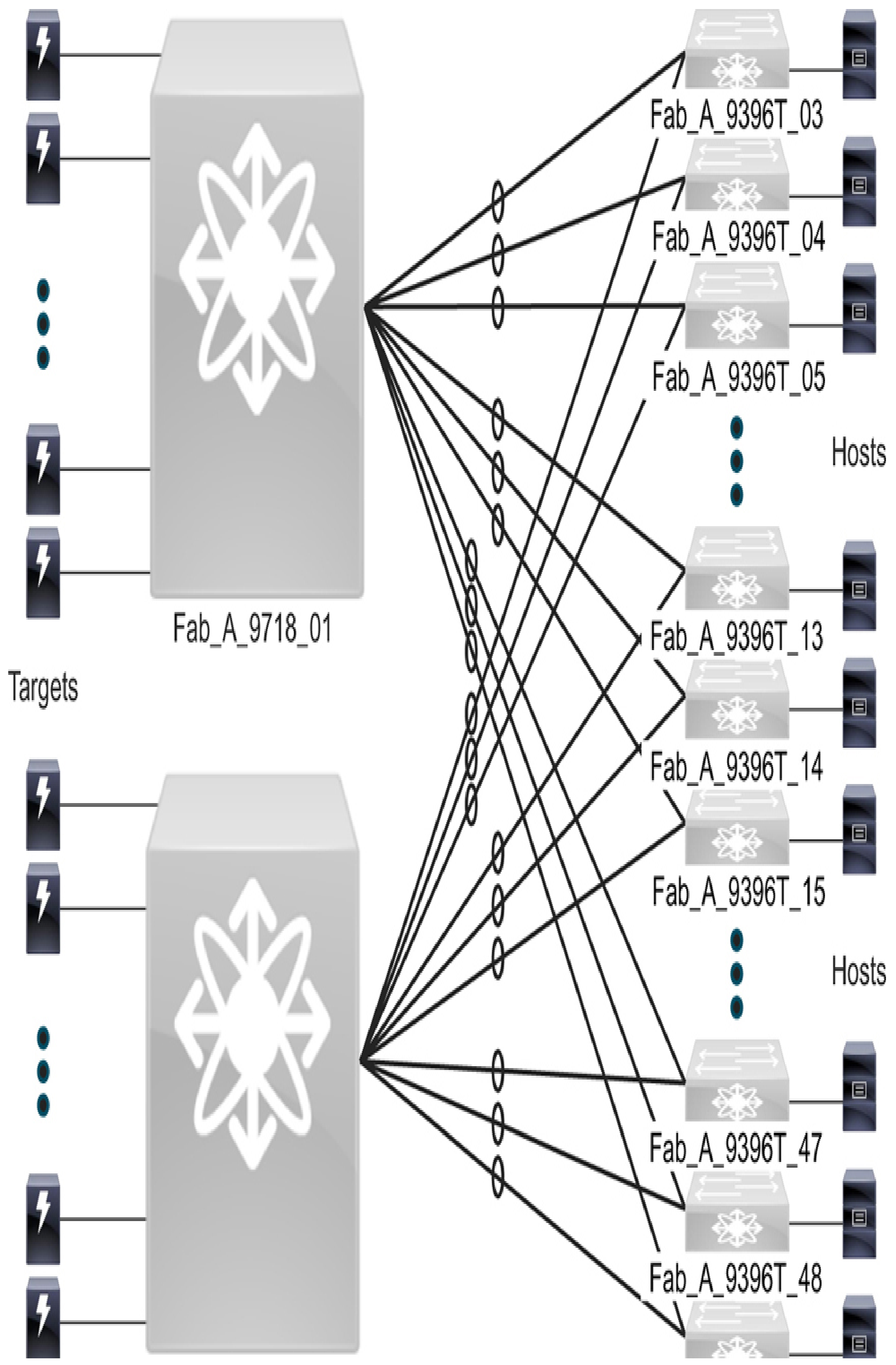


Figure 4-23 *Edge-Core topology of Fabric A. Fabric B had a similar design.*

This case study is purposefully based on a 49-switch fabric to demonstrate that the congestion troubleshooting methodology should remain the same regardless of the size of the fabric. As the section on *Workflow for Chasing the Source of Congestion* explains, focus on the end devices and switches with relevant symptoms instead of being overwhelmed by the size of the fabric.

Initial Investigation

In this case, although the Windows server reported the problem, it was not known if it was a culprit or a victim. Since we had no better place to start our investigation, it is natural to start the investigation where the problem was reported.

The initial investigation was to find where this Windows server was connected to the fabric. For this purpose, the following steps were taken:

1. Found the WWPNs of this server. This server was found to have two WWPNs. Finding the WWPN of HBA ports is dependent on the host operating system.
2. Then, on any MDS switch in the fabrics, the FCIDs associated with those WWPNs were found using the **show fcns database | include VSAN|<wwpn>** command. It was found to have FCID of 0x690140 in VSAN101 in Fabric A and FCID of 0x6a0160 in VSAN 100 in Fabric B.
3. Next, using the **show fcns database fcid <fcid> detail vsan <vsan_id>** command, it was found that this device was connected to switch Fab_A_MDS_9396T_14 in Fabric A and to switch Fab_B_MDS_9396T_14 in Fabric B.
4. Finally, using the **show flogi database fcid <fcid>** command on Fab_A_MDS_9396T_14, it was found that one HBA port of this Window server was connected to fc1/62. In Fabric B, the same command on Fab_B_MDS_9396T_14 (with fcid 0x6a0160) showed that even on this switch, the other HBA port of this Windows server was connected to fc1/62.

This initial investigation gave a starting location for continuing the congestion troubleshooting methodology. This was interface fc1/62 on Fab_A_MDS_9396T_14 and Fab_B_MDS_9396T_14.

As mentioned earlier, there is no need to be overwhelmed by the size of the fabric. We started the investigation by limiting the focus to the edge MDS9396T, where the server connects.

Fabric A Analysis

Example 4-44 shows a snippet of the **show flogi database detail** command from switch name Fab_A_MDS_9396T_14, which connected to the server in question via interface fc1/62. As mentioned, the PORT NAME and NODE NAME from this output should match with that of the HBA port on the Windows server that reported the problem. This command also confirms that fc1/62 was assigned to VSAN 101.

Example 4-44 *FLOGI database on Fab_A_MDS_9396T_14*

```
show flogi database detail

----------------------------------------------------
---------------------------
INTERFACE    VSAN    FCID           PORT NAME
NODE NAME           FLAGS
----------------------------------------------------
---------------------------

fc1/62       101   0x690140  10:00:00:10:9b:97:14:c0
20:00:00:10:9b:97:14:c0 P
```

The VSAN 101 topology in Fab_A_MDS_9396T_14 (Example 4-45) shows that it was connected to Fab_A_MDS_9718_01 via port-channel 142 and to Fab_A_MDS_9718_02 via port-channel 143.

Example 4-45 *Neighbors of Fab_A_MDS_9396T_14*

```
show topology

FC Topology for VSAN 101 :
-----------------------------------------------------
---------------------------
       Interface  Peer Domain Peer Interface
Peer IP Address(Switch Name)
-----------------------------------------------------
---------------------------
  port-channel142  0x1f(31)  port-channel142
x.x.x.x(Fab_A_MDS_9718_01)
  port-channel143  0x21(33)  port-channel143
x.x.x.x(Fab_A_MDS_9718_02)
```

Figure 4-24 depicts the topology of Fabric A including the two core MDS 9718 and the edge MDS 9396T in question.

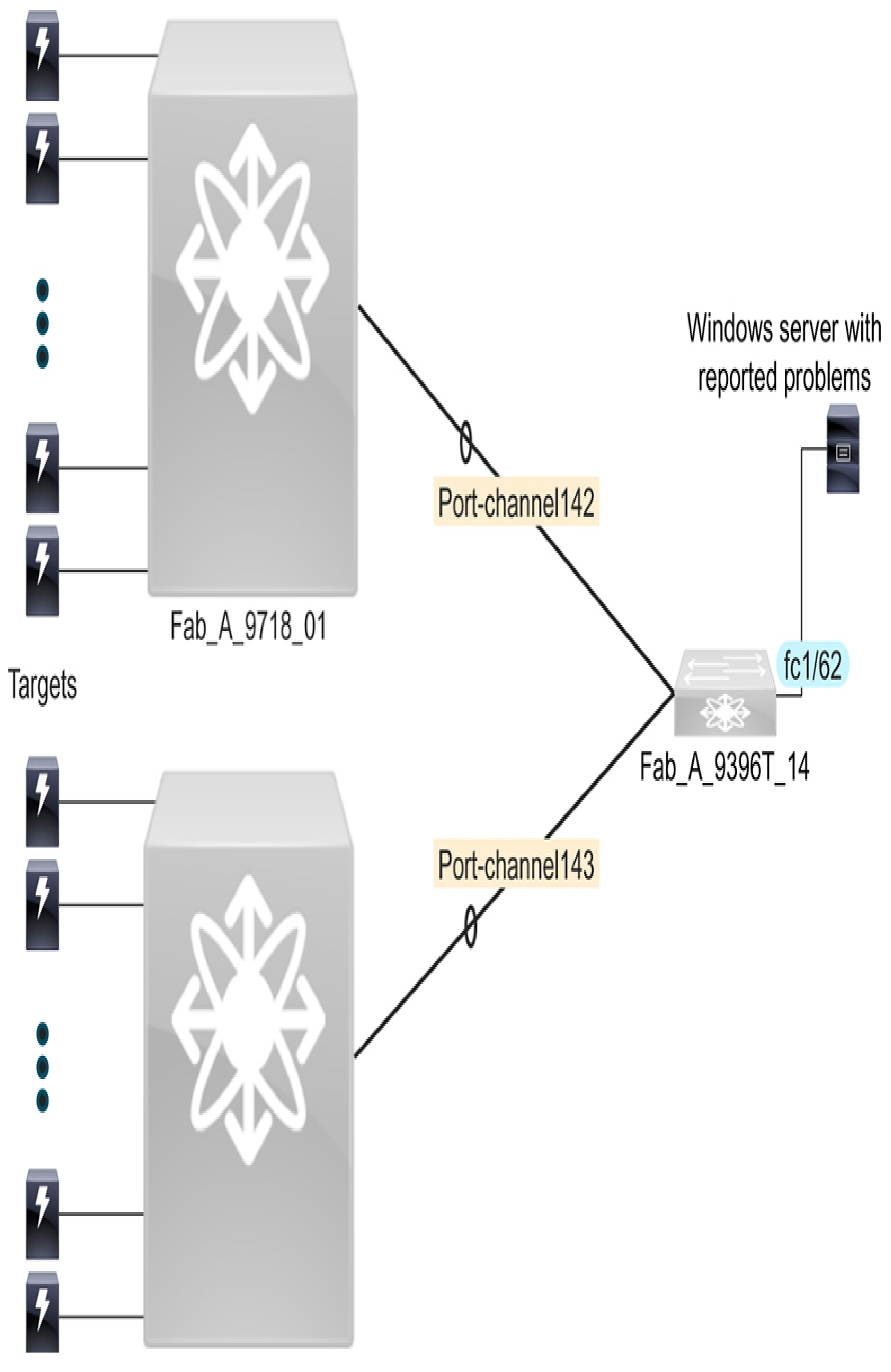


Figure 4-24 *Fabric A - Edge-Core topology for Fab_A_MDS_9396T_14 and its two core switches*

Edge Switch Fab_A_MDS_9396T_14

Because the reported error was at the SCSI layer, this probably involved dropped frames somewhere in the path between the server and the targets. A good place to start was with the **show**

logging onboard error-stats command (OBFL), which contains most interface-level errors.

Investigating Fab_A_MDS_9396T_14, there were no events for 6/22/2021 (MM/DD/YY) (Example 4-46):

Example 4-46 *Error stats history on Fab_A_MDS_9396T_14*

```
show logging onboard error-stats

----------------------------
    Module:  1
----------------------------


----------------------------
 Show Clock
----------------------------
2021-06-22 18:43:59

---------------------------------
 Module: 1 error-stats
---------------------------------


----------------------------------------------------
--------------------------
 ERROR STATISTICS INFORMATION FOR DEVICE DEVICE:
FCMAC
----------------------------------------------------
--------------------------
 Interface  |                                      |
|    Time Stamp
   Range    |       Error Stat Counter Name        |
Count   |MM/DD/YY HH:MM:SS
            |                                      |
|
----------------------------------------------------
--------------------------
<no entries for 6/22/2021>
```

If there were errors found, understanding which member interfaces were in port-channel142 and port-channel143 would be important. But in this case, that step was skipped because no errors were found.

Since SCSI errors would typically be reported if there are frame drops anywhere along the path from server to target in either direction, both of the core MDS 9718 in the path were investigated as well.

Core Switch Fab_A_MDS_9718_01

From Fab_A_MDS_9718_01, we can see that port-channel 142 had 6 members and they were all nicely using port 19 of modules 1 — 6 (Example 4-47). This kind of attention to detail makes understanding the fabric topology much easier.

Example 4-47 *Members of port-channel interface*

```
show interface port-channel142

port-channel142 is trunking
    Port description is ISL to
Fab_A_MDS_9396T_14_PC142
    Hardware is Fibre Channel
    Port WWN is 24:8e:00:3a:9c:30:8c:80
    Admin port mode is auto, trunk mode is on
    snmp link state traps are enabled
    Port mode is TE
    Port vsan is 1
    Speed is 96 Gbps
    Logical type is core
    Trunk vsans (admin allowed and active)
(1,51,101)
    Trunk vsans (up)                        (1,101)
    Trunk vsans (isolated)                  (51)
    Trunk vsans (initializing)              ()
    5 minutes input rate 341086720 bits/sec,42635840
bytes/sec, 22756 frames/sec
    5 minutes output rate 711011264
bits/sec,88876408 bytes/sec, 46539 frames/sec
      208358302240 frames input,396467381532436
```

```
bytes
        **0 discards,0 errors**
        0 invalid CRC/FCS,0 unknown class
        0 too long,0 too short
      895755562158 frames output,1819180641786024
bytes
        **0 discards,0 errors**
      0 input OLS,0  LRR,0 NOS,0 loop inits
      0 output OLS,0 LRR, 0 NOS, 0 loop inits
    **Member[1] : fc1/19    [up] ***
    **Member[2] : fc2/19    [up]**
    **Member[3] : fc3/19    [up]**
    **Member[4] : fc4/19    [up]**
    **Member[5] : fc5/19    [up]**
    **Member[6] : fc6/19    [up]**
    Interface last changed at Thu Sep 17 02:13:35
2020
```

Notice that there were ‘0 discards, 0 errors’ listed under both input and output. A port-channel interface aggregates all of the counters for the member interfaces. Consequently, having ‘0 discards, 0 errors’ indicated that all of the member interfaces also had ‘0 discards, 0 errors’. That would seem to indicate there weren’t any drops/discards under all the interfaces that made up the port-channel.

But, to look closer, modules 1 — 6 were investigated in OBFL error-stats. Example 4-48 shows the output for module 1. The most recent entry was from interface fc1/21 (which was not part of port-channel 142) and is from 09/26/20 (MM/DD/YY). Since the most current entry was at the top, this indicated there were not any entries for interface fc1/19 for the event on 06/22/21 (MM/DD/YY). Likewise, the rest of the modules in question had no entries for 06/22/21 (MM/DD/YY).

To complete the analysis on this switch, we inspected the zoning database and found that the host connected on switch Fab_A_MDS_9396T_14 interface fc1/62 was zoned with two targets on Fab_A_MDS_9718_01. One was connected to interface fc2/12 and the other was connected to interface fc7/12. We’ve already concluded that there were no entries in **show logging onboard error-stats** for modules 1 through 6 on Fab_A_MDS_9718_01 so no packet loss occurred there. When module 7 was checked for interface fc7/12, there were no entries for the event on 06/22/21 (MM/DD/YY). That was the complete data path pertaining to this switch.

Example 4-48 ***OBFL*** *Error stats history*

```
show logging onboard error-stats

----------------------------
 Show Clock
----------------------------
2021-06-22 18:44:19

---------------------------------
 Module: 1 error-stats
---------------------------------

-----------------------------------------------------
-------------------------
 ERROR STATISTICS INFORMATION FOR DEVICE DEVICE:
FCMAC
-----------------------------------------------------
-------------------------
 Interface  |                                      |
|    Time Stamp
   Range    |        Error Stat Counter Name       |
Count   |MM/DD/YY HH:MM:SS
            |                                      |
|
-----------------------------------------------------
-------------------------
fc1/21      |F32_TMM_PORT_TIMEOUT_DROP
|6898     |09/26/20 07:19:17
fc1/21      |F32_TMM_PORT_TIMEOUT_DROP
|6545     |09/25/20 03:03:40
```

Core Switch Fab_A_MDS_9718_02

From Fab_A_MDS_9718_02, we saw that port-channel 142 also had 6 members using port 19 on modules 1 — 6 (Example 4-49).

Example 4-49 *Members of port-channel interface*

```
show interface


port-channel143 is trunking
```

```
    Port description is ISL to
Fab_A_MDS_9396T_14_PC143
    Hardware is Fibre Channel
    Port WWN is 24:8f:00:3a:9c:30:72:80
    Admin port mode is auto, trunk mode is on
    snmp link state traps are enabled
    Port mode is TE
    Port vsan is 1
    Speed is 96 Gbps
    Logical type is core
    Trunk vsans (admin allowed and active)
(1,51,101)
    Trunk vsans (up)                       (1,101)
    Trunk vsans (isolated)                 (51)
    Trunk vsans (initializing)             ()
    5 minutes input rate 323695680 bits/sec,40461960
bytes/sec, 21752 frames/sec
    5 minutes output rate 670048896
bits/sec,83756112 bytes/sec, 43852 frames/sec
      304350075683 frames input,578146921564556
bytes
        0 discards,0 errors
        0 invalid CRC/FCS,0 unknown class
        0 too long,0 too short
      951418650549 frames output,1897004671422240
bytes
        0 discards,0 errors
      0 input OLS,0  LRR,0 NOS,0 loop inits
      0 output OLS,0 LRR, 0 NOS, 0 loop inits
    Member[1] : fc1/19    [up]
    Member[2] : fc2/19    [up] *
    Member[3] : fc3/19    [up]
    Member[4] : fc4/19    [up]
    Member[5] : fc5/19    [up]
    Member[6] : fc6/19    [up]
    Interface last changed at Thu Sep 17 02:19:26
2020
```

Again, there were no input or output discards, or errors. Looking closer at modules 1 — 6 using the **show logging**

onboard error-stats command, as before, there was nothing for 06/22/21 for any of the interfaces in the port-channel.

To complete the analysis on this switch, we inspected the zoning database and found that the host connected to switch Fab_A_MDS_9396T_14 interface fc1/62 was zoned with two targets on Fab_A_MDS_9718_02 as well. Identically, one was connected to interface fc2/12 and the other to interface fc7/12. We've already concluded that there were no entries in **show logging onboard error-stats** for modules 1 through 6 so no packet loss occurred there. When module 7 was checked for interface fc7/12, there were no entries for the event on 06/22/21 (MM/DD/YY). That was the complete data path pertaining to this switch.

Fabric A Conclusion

At this point, Fabric A was pretty much ruled out for problems. On to Fabric B!

Fabric B Analysis

Fabric B had a similar design as Fabric A, and it was investigated using the same commands as were used to investigate Fabric A.

Edge Switch Fab_B MDS_9396T_14

Example 4-50 shows a snippet of the **show flogi database detail** command from the switch Fab_B_MDS_9396T_14, which was connected to the server in question that reported the problems via interface fc1/62. In Fabric B, fc1/62 on Fab_B_MDS_9396T_14 was assigned to VSAN 100.

Example 4-50 *FLOGI database on Fab_B_MDS_9396T_14*

```
show flogi database detail
-----------------------------------------------------
-----------------------------
INTERFACE      VSAN    FCID            PORT NAME
NODE NAME         FLAGS
-----------------------------------------------------
```

```
-----------------------------

fc1/62        100   0x6a0160
10:00:00:10:9b:97:14:c1 20:00:00:10:9b:97:14:c1  P
```

The VSAN 100 topology in Fab_B_MDS_9396T_14 (Example 4-51) shows that it was connected to Fab_B_MDS_9718_01 via port-channel 142 and to Fab_B_MDS_9718_02 via port-channel 143.

Example 4-51 *Neighbors of Fab_B_MDS_9718_01*

```
show topology

FC Topology for VSAN 100 :
----------------------------------------------------
---------------------------
       Interface  Peer Domain Peer Interface
Peer IP Address(Switch Name)
----------------------------------------------------
---------------------------
  port-channel142  0x20(32)  port-channel142
x.x.x.x(Fab_B_MDS_9718_01)
  port-channel143  0x22(34)  port-channel143
x.x.x.x(Fab_B_MDS_9718_02)
```

Figure 4-25 depicts the topology of Fabric B just including the two core MDS9718 and the edge MDS9396T in question.

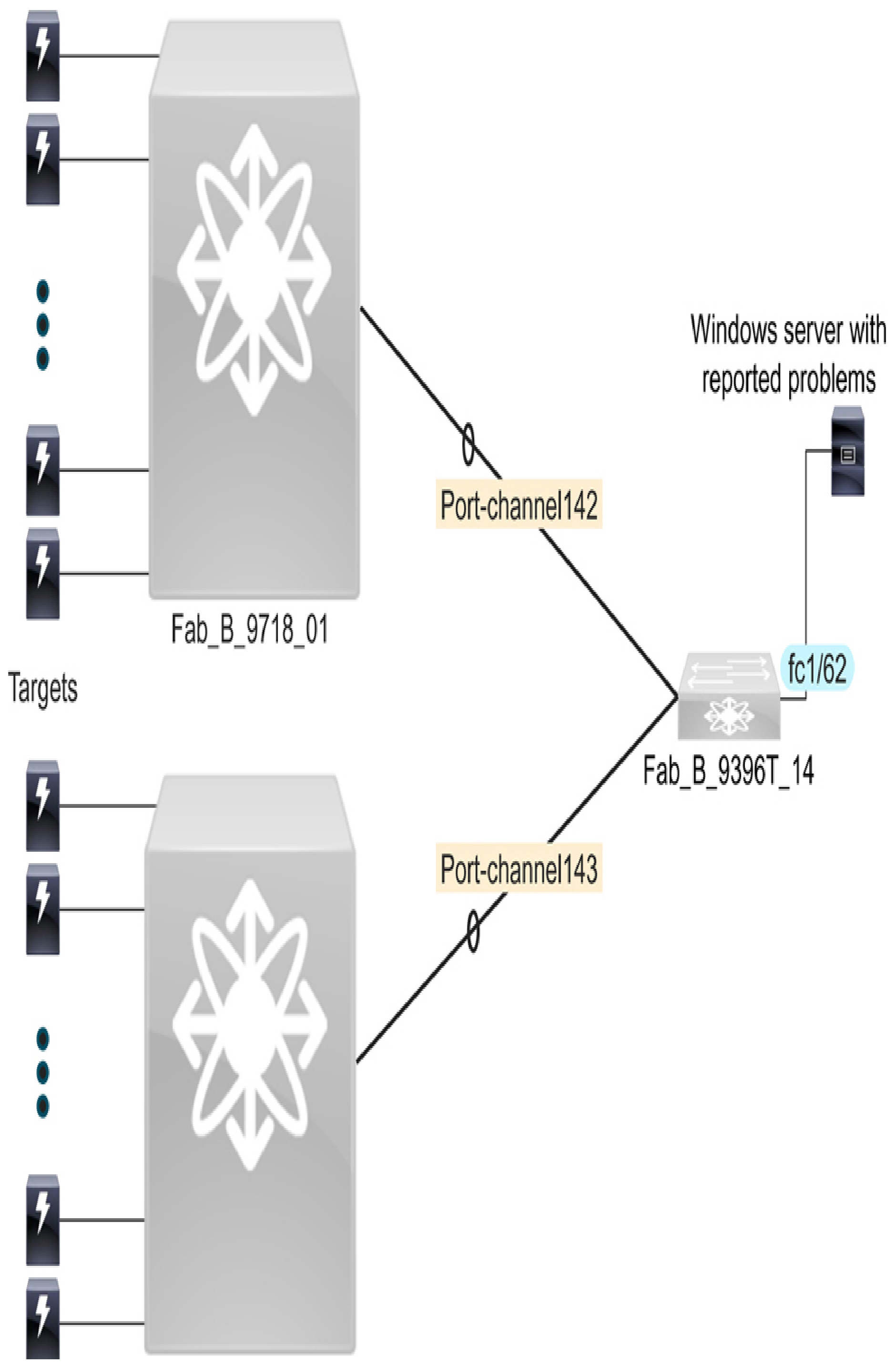


Figure 4-25 *Fabric B — Edge-Core topology for Fab_B_9396T_14 and its two core switches.*

Investigating Fab_B_MDS_9396T_14 OBFL error-stats showed the events recorded on 6/22/2021 (MM/DD/YY) (Example 4-52).

Example 4-52 *Error stats history on Fab_B_MDS_9396T_14*

```
show logging onboard error-stats

----------------------------
    Module:  1
----------------------------


----------------------------
 Show Clock
----------------------------
2021-06-22 20:57:00

---------------------------------
 Module: 1 error-stats
---------------------------------



----------------------------------------------------
--------------------------
 ERROR STATISTICS INFORMATION FOR DEVICE DEVICE:
FCMAC
----------------------------------------------------
--------------------------
 Interface  |                                      |
|    Time Stamp
   Range    |       Error Stat Counter Name        |
Count  |MM/DD/YY HH:MM:SS
            |                                      |
|
----------------------------------------------------
--------------------------
...
fc1/50      |F32_MAC_KLM_CNTR_RX_WT_AVG_B2B_ZERO
|12       |06/22/21 13:23:21
fc1/31      |F32_TMM_PORT_TIMEOUT_DROP
|46139    |06/22/21 13:23:21
fc1/31      |F32_MAC_KLM_CNTR_CREDIT_LOSS
|12       |06/22/21 13:23:21
...
fc1/31      |F32_TMM_PORT_TIMEOUT_DROP
|42175    |06/08/21 06:05:42
fc1/31      |F32_MAC_KLM_CNTR_CREDIT_LOSS
```

```
|11        |06/08/21 06:05:42
...
fc1/50       |F32_MAC_KLM_CNTR_RX_WT_AVG_B2B_ZERO  |4
|06/05/21 01:36:12
```

Interface fc1/31 had CREDIT_LOSS and the corresponding timeout drops on June 22, 2021. This time matched with the time of the error reported by the Windows server. These error-stats outputs are recorded every 20 seconds independently on each module when their values change. To determine how much each counter changed in the 20-second interval, the previous value must be subtracted from the current value. In Example 4-52, these counters were previously recorded on 06/08/21 (MM/DD/YY). Subtracting the two values it is seen that there was a single occurrence of CREDIT_LOSS (12 - 11 = 1) and there were 3964 (46139 - 42175) timeout drops in the 20-second interval ending at 13:23:21 on 06/22/21 (MM/DD/YY).

Before investigating these congestion symptoms in detail, we found more details about fc1/31. As Example 4-53 shows, the **show interface** command shows that fc1/31 was an F_Port.

Example 4-53 *show interface command on Fab_B_MDS_9396T_14*

```
show interface
...

fc1/31 is up
    Port description is dc1drwfnpdb0003_1
    Hardware is Fibre Channel, SFP is short wave
laser w/o OFC (SN)
    Port WWN is 20:1f:00:3a:9c:c4:06:40
    Admin port mode is auto, trunk mode is on
    snmp link state traps are enabled
    Port mode is F, FCID is 0x6a0440
    Port vsan is 100
    Admin Speed is auto max 32 Gbps
    Operating Speed is 32 Gbps
    Rate mode is dedicated
    Port flow-control is R_RDY
    Transmit B2B Credit is 80
    Receive B2B Credit is 32
    B2B State Change Number is 14
    Receive data field Size is 2112
    Beacon is turned off
    fec is enabled by default
    Logical type is edge
    5 minutes input rate 630560 bits/sec,78820
bytes/sec, 20 frames/sec
    5 minutes output rate 2317376 bits/sec,289672
bytes/sec, 112 frames/sec
      8431559473 frames input,16999304569824 bytes
        0 discards,0 errors
        0 invalid CRC/FCS,0 unknown class
        0 too long,0 too short
      5009622615 frames output,9613624736124 bytes
        46139 discards,0 errors
      9 input OLS,20  LRR,8 NOS,0 loop inits
      8 output OLS,33 LRR, 7 NOS, 0 loop inits
      32 receive B2B credit remaining
      80 transmit B2B credit remaining
      80 low priority transmit B2B credit remaining
    Interface last changed at Sat Mar 20 02:54:43
2021

    Last clearing of "show interface" counters :
never
...
```

Example 4-52 shows many Level 3, Level 2, and Level 1 symptoms on fc1/31. CREDIT_LOSS indicated there was 1 full second of continuous zero remaining-Tx-B2B-credits on the interface. This was severe (Level 3) egress congestion towards the adjacent device. Refer to Chapter 2 for more details on Credit Loss Recovery using Link Reset Protocol, Chapter 3 for more details on Credit Loss Recovery, and the beginning of this chapter for Congestion Severities and Levels.

Timeout Drops indicated the frames that were dropped because they could not be transmitted in the 500ms congestion-drop timeout. This made perfect sense since fc1/31 was ‘stuck’ for 1 full continuous second.

To get a more detailed look at how much time the interface was at zero remaining-Tx-B2B-credits we looked at the same switch’s OBFL TxWait. Example 4-54 shows there was 5% TxWait congestion on fc1/31 in exactly the same 20-second interval from 13:23:01 - 13:23:21. Here, 5% TxWait indicates 5% of 20 seconds which is 1 second. To get a more precise value, we used the number of ticks, which incremented by 423108 (Delta). This value is the actual number of TxWait ticks that occurred in the 20-second interval and does not need to be subtracted from a previous value. Each TxWait tick is 2.5µs so to convert that to seconds, it was (423108 * 2.5) / 1,000,000 = 1.05777 seconds. This didn’t add a lot of information but it was good confirmation of the CREDIT_LOSS indications in OBFL error-stats.

Example 4-54 *TxWait history on Fab_B_MDS_9396T_14*

```
show logging onboard txwait

----------------------------
    Module:  1
----------------------------


---------------------------------
 Module: 1 txwait count
---------------------------------


----------------------------
 Show Clock
----------------------------
2021-06-22 20:57:07


---------------------------------
 Module: 1 txwait
---------------------------------
 Notes:
     - Sampling period is 20 seconds
     - Only txwait delta >= 100 ms are logged
```

```
---------------------------------------------------------------------
-------------------------
| Interface | Delta TxWait Time      | Congestion |
Timestamp                          |
|           | 2.5us ticks | seconds |             |
|
---------------------------------------------------------------------
-------------------------
|   fc1/31   |   423108    |    1    |      5%     |
Tue Jun 22 13:23:21 2021 |
```

Another observation from the OBFL error-stats output in switch Fab_B_9396T_14 (Example 4-52) shows the F32_MAC_KLM_CNTR_RX_WT_AVG_B2B_ZERO counter incremented on fc1/50 within the same 20 second interval as the CREDIT_LOSS and timeout-drops to fc1/62. This indicates ingress congestion on that interface. To determine what kind of port this was, the **show interface** command was used. As shown in Example 4-55, fc1/50 is a member of port-channel143 going to Fab_B_MDS_9718_02.

To determine which port fc1/50 was connected to on Fab_B_MDS_9718_02, the 'Peer port WWN' was used in the **show interface** command (Example 4-55). It was matched to the 'Port WWN' on an interface in Fab_B_MDS_9718_02.

Example 4-55 *Peer WWN from Fab_B_MDS_9396T_14 in* ***show interface*** *command*

```
fc1/50 is trunking
    Port description is Fab_B_MDS_9718_02_PC143
    Hardware is Fibre Channel, SFP is short wave
laser w/o OFC (SN)
    Port WWN is 20:32:00:3a:9c:c4:06:40
    Peer port WWN is 20:53:00:3a:9c:30:f9:80
    Admin port mode is auto, trunk mode is on
    snmp link state traps are enabled
    Port mode is TE
..
    Belongs to port-channel143
```

Fab_B_MDS_9718_02 (Example 4-56) showed fc2/19 was connected to fc1/50 on Fab_B_MDS_9396T_14.

Example 4-56 *Port WWN in* ***show interface*** *command on Fab_B_MDS_9718_02*

```
fc2/19 is trunking
    Port description is Fab_B_MDS_9718_02_PC143
    Hardware is Fibre Channel, SFP is short wave
laser w/o OFC (SN)
    Port WWN is 20:53:00:3a:9c:30:f9:80
    Peer port WWN is 20:32:00:3a:9c:c4:06:40
```

The F32_MAC_KLM_CNTR_RX_WT_AVG_B2B_ZERO counter in Example 4-52 records the count of continuous 100ms intervals of time when Fab_B_MDS_9396T_14 withheld B2B credits from Fab_B_MDS_9718_02. This was ingress congestion on Fab_B_MDS_9396T_14 ISL port. By withholding B2B credits from Fab_B_MDS_9718_02, Fab_B_MDS_9396T_14 was preventing any inbound traffic from Fab_B_MDS_9718_02 on that interface.

The F32_MAC_KLM_CNTR_RX_WT_AVG_B2B_ZERO counter only increments if there is an entire 100ms of continuous zero remaining-Rx-B2B-credits on the port. If there was even 1 B2B credit sent back from fc1/50 on Fab_B_MDS_9396T_14 to fc2/29 on Fab_B_MDS_9718_02 during a 100ms interval, then this counter would not increment. Because of this, there still could be significant ingress congestion without this counter incrementing.

The count was 12 and since the previous instance of the counter for this port was 4 (from 06/05/21), it was determined that this counter incremented by 8 during this 20-second interval. This indicates there were 8 100ms intervals of time (not necessarily contiguous) that fc1/50 withheld B2B credits from fc2/19 on Fab_B-MDS_9718_02. This was a total of 8 * 100ms = 800ms.

This 800ms of egress congestion on Fab_B_MDS_9718_02 was close to 1 second of egress congestion on Fab_B_MDS_9396T_14 indicated by TxWait in Example 4-54 and CREDIT_LOSS in Example 4-52. This proves significant congestion spreading and is explained in Figure 4-26.

Next, we investigated the two core MDS 9718 switches to find symptoms of congestion spreading.

Core Switch Fab_B MDS_9718_01

To find symptoms of congestion spreading on switch Fab_B_MDS_9718_01, **show logging onboard error-stats** command was checked. But no events were available for 06/22/21 (MM/DD/YY). This indicated that as per the traffic pattern on 06/22/21 (MM/DD/YY), the level 2 congestion did not spread to Fab_B_MDS_9718_01.

For a complete analysis, individual member interfaces of port-channel 142 were also analyzed for any errors. These members can be found using the **show interface** command. As Example 4-57 shows, Fab_B_MDS_9718_01 had 6 member interfaces aggregated in port-channel 142.

Example 4-57 *Members of a port-channel interface on Fab_B_MDS_9718_01*

```
show interface

port-channel142 is trunking
    Port description is ISL to
Fab_B_MDS_9396T_14_PC142
    Hardware is Fibre Channel
    Port WWN is 24:8e:00:3a:9c:40:23:80
    Admin port mode is auto, trunk mode is on
    snmp link state traps are enabled
    Port mode is TE
    Port vsan is 1
    Speed is 96 Gbps
    Logical type is core
    Trunk vsans (admin allowed and active)
(1,50,100)
    Trunk vsans (up)                          (1,100)
    Trunk vsans (isolated)                    (50)
    Trunk vsans (initializing)                ()
    5 minutes input rate 305907744 bits/sec,38238468
```

```
bytes/sec, 20008 frames/sec
    5 minutes output rate 327390240
bits/sec,40923780 bytes/sec, 22390 frames/sec
      297399115568 frames input,561803823267632
bytes
        0 discards,0 errors
        0 invalid CRC/FCS,0 unknown class
        0 too long,0 too short
      883949066858 frames output,1752330935153904
bytes
        0 discards,0 errors
      0 input OLS,0  LRR,0 NOS,0 loop inits
      0 output OLS,0 LRR, 0 NOS, 0 loop inits
    Member[1] : fc1/19    [up]
    Member[2] : fc2/19    [up]
    Member[3] : fc3/19    [up]
    Member[4] : fc4/19    [up]
    Member[5] : fc5/19    [up] *
    Member[6] : fc6/19    [up]
    Interface last changed at Thu Sep 17 03:20:04
2020
```

None of these member interfaces showed any congestion symptoms as verified by **show interface counter detailed** command. The outputs are not shown here for the sake of brevity.

Core Switch Fab_B MDS_9718_02

Switch Fab_B_MDS_9396T_14 already showed ingress congestion on the switchport fc1/50 connected to fc2/19 on Fab_B MDS_9718_02. So, we expected to see egress congestion symptoms on fc2/19 on Fab_B MDS_9718_02.

From Fab_B_MDS_9718_02, we found that port-channel 143 had 6 members and they were on port 19 on modules 1 - 6 (Example 4-58).

Example 4-58 *Members of a port-channel interface on Fab_B_MDS_9718_02*

```
show interface

port-channel143 is trunking
    Port description is ISL to
Fab_B_MDS_9396T_14_PC143
    Hardware is Fibre Channel
    Port WWN is 24:8f:00:3a:9c:30:f9:80
    Admin port mode is auto, trunk mode is on
    snmp link state traps are enabled
    Port mode is TE
    Port vsan is 1
    Speed is 96 Gbps
    Logical type is core
    Trunk vsans (admin allowed and active)
(1,50,100)
    Trunk vsans (up)                       (1,100)
    Trunk vsans (isolated)                 (50)
    Trunk vsans (initializing)             ()
    5 minutes input rate 205973696 bits/sec,25746712
bytes/sec, 14033 frames/sec
    5 minutes output rate 253791392
bits/sec,31723924 bytes/sec, 17954 frames/sec
      296679693935 frames input,556372660069660
bytes
        0 discards,0 errors
        0 invalid CRC/FCS,0 unknown class
        0 too long,0 too short
      876485569623 frames output,1729856511686916
bytes
        670 discards,0 errors
      0 input OLS,0  LRR,0 NOS,0 loop inits
      0 output OLS,0 LRR, 0 NOS, 0 loop inits
    Member[1] : fc1/19    [up]
    Member[2] : fc2/19    [up]
    Member[3] : fc3/19    [up]
    Member[4] : fc4/19    [up] *
    Member[5] : fc5/19    [up]
    Member[6] : fc6/19    [up]
    Interface last changed at Thu Sep 17 03:09:13
2020
```

Example 4-58 shows that port-channel143 had a total of 670 output discards among all six members. This doesn't indicate which member(s) had those or when they incremented. This was just a sum across all member interfaces. Consequently, modules 1 - 6 were investigated in OBFL error-stats using **show logging onboard error-stats** command.

Example 4-59 shows that module 2 interface fc2/19 had a TIMEOUT_DROP entry on 06/22/21. As previously determined, interface fc2/19 was part of port-channel143 going to Fab_B_MDS_9396T_14. Fc2/19 was connected to Fab_B_MDS_9396T_14 fc1/50.

No other modules had any entries for that date. Note that the time listed exactly matched the TIMEOUT_DROP and CREDIT_LOSS events on Fab_B_MDS_9396T_14 (Example 4-52) although it did not have to exactly match to still be relevant. Remember that counters are recorded every 20 seconds only when they increment. When that 20 second interval occurs, it may be different across different switches or even different modules in the same switch. What is important is that there was an overlap in the 20-second interval between this module 2 and switch Fab_B_MDS_9396T_14. This again emphasizes the importance of clock synchronization across all the switches in the fabric.

Example 4-59 *Error stats history in OBFL on Fab_B_MDS_9718_02*

```
--------------------------------
 Module: 2 error-stats
--------------------------------



---------------------------------------------------
-------------------------
 ERROR STATISTICS INFORMATION FOR DEVICE DEVICE:
FCMAC
---------------------------------------------------
-------------------------
```

```
 Interface  |                                      |
|    Time Stamp
   Range    |       Error Stat Counter Name       |
Count   |MM/DD/YY HH:MM:SS
----------------------------------------------------
--------------------------

fc2/19      |F32_TMM_PORT_TIMEOUT_DROP
|535      |06/22/21 13:23:21
```

To determine the number of frames dropped in the 20-second interval from 13:23:01 to 13:23:21 a previous TIMEOUT_DROP entry was found for that same interface, and the two values subtracted. In this case, there was no previous entry, so we assume the counter incremented by 535 during the 20-second interval ending on 06/22/2021 13:23:21.

For a more detailed analysis, we also investigated TxWait. The TxWait entries in (Example 4-60) indicate fc2/19 was unable to transmit for approximately 1 second because of zero remaining-Tx-B2B-credits. This finding was very similar to how long fc1/31 on Fab_B_MDS_9396T_14 was unable to transmit. However, the difference is that there was credit loss recovery on Fab_B_MDS_9396T_14 fc1/31 but not on Fab_B_MDS_9718_02 fc2/19. This is because fc2/19 was an ISL (E_Port) and the time to invoke credit loss recovery is 1.5 seconds instead of 1 second on an F_Port. Also, note the TxWait entry doesn't guarantee that the 420746 2.5µs ticks occurred continuously.

Example 4-60 *TxWait history in OBFL on Fab_B_MDS_9718_02*

```

---------------------------------
 Module: 2 txwait
---------------------------------
 Notes:
     - Sampling period is 20 seconds
     - Only txwait delta >= 100 ms are logged


----------------------------------------------------
```

```
-------------------------
| Interface | Delta TxWait Time      | Congestion |
Timestamp                |
|           | 2.5us ticks | seconds |            |
|
-----------------------------------------------------
-------------------------


|   fc2/19  |   420746    |    1    |       5%   |
Tue Jun 22 13:23:21 2021 |
```

This again proved that congestion spread to Fab_B_MDS_9718_02.

At this point, as Figure 4-26 shows, it was established that Fab_B_MDS_9396T_14 fc1/31 was unable to transmit frames to the server due to zero remaining-Tx-B2B-credits for 1 second. This caused 3964 timeout drops at that time. Those dropped frames were only for the server on fc1/31.

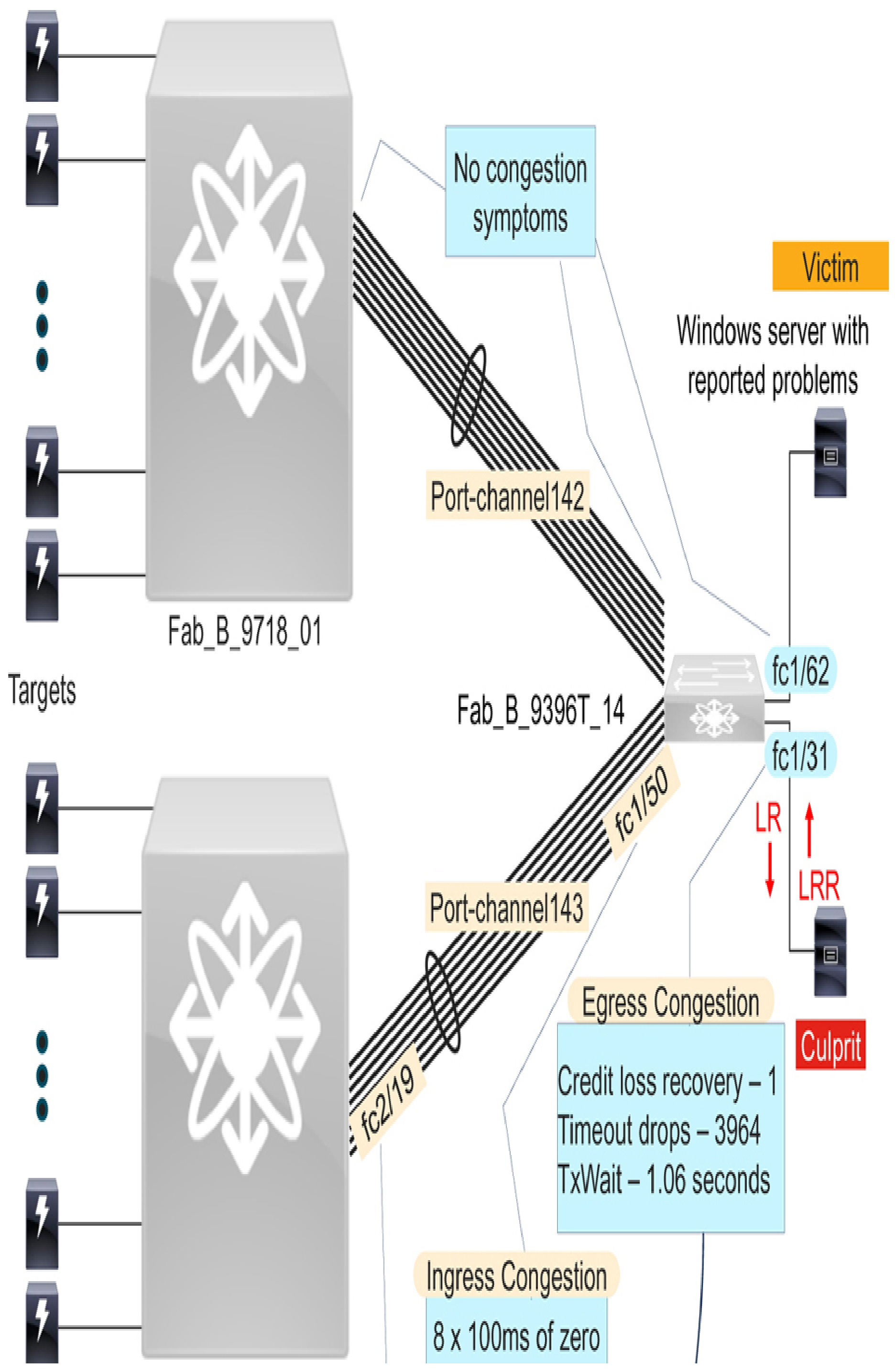


Figure 4-26 *Fabric B investigation*

Because of this congestion on fc1/31, traffic on the upstream switch, Fab_B_MDS_9718_02 interface fc2/19 (member of port-channel143) backed up causing it to be unable to transmit. It also dropped frames but only 535. Since this was an ISL it would be carrying traffic for multiple destination servers located on Fab_B_MDS_9396T_14. This would potentially affect every server on Fab_B_MDS_9396T_14 for that 1 second period.

Fabric B Conclusion

Fabric B clearly showed symptoms of congestion spreading.

Figure 4-27 shows the key interfaces and events in a graphical format plotted according to the type of congestion indication and the time in which they occurred. There are two graphs shown. The top one is from switch Fab_B_9396T_14 and the bottom one is from switch Fab_B_9718_02. On the Y-axis, there are the interfaces and for each interface, there are the six congestion indication types(Tx-Lvl-3, Tx-Lvl-2, Tx-Lvl-1.5, Tx-Lvl-1, Rx-Lvl-1 and Tx-HU for High Utilization). Notice how all the events occured at the same time as shown by their vertical time alignment.

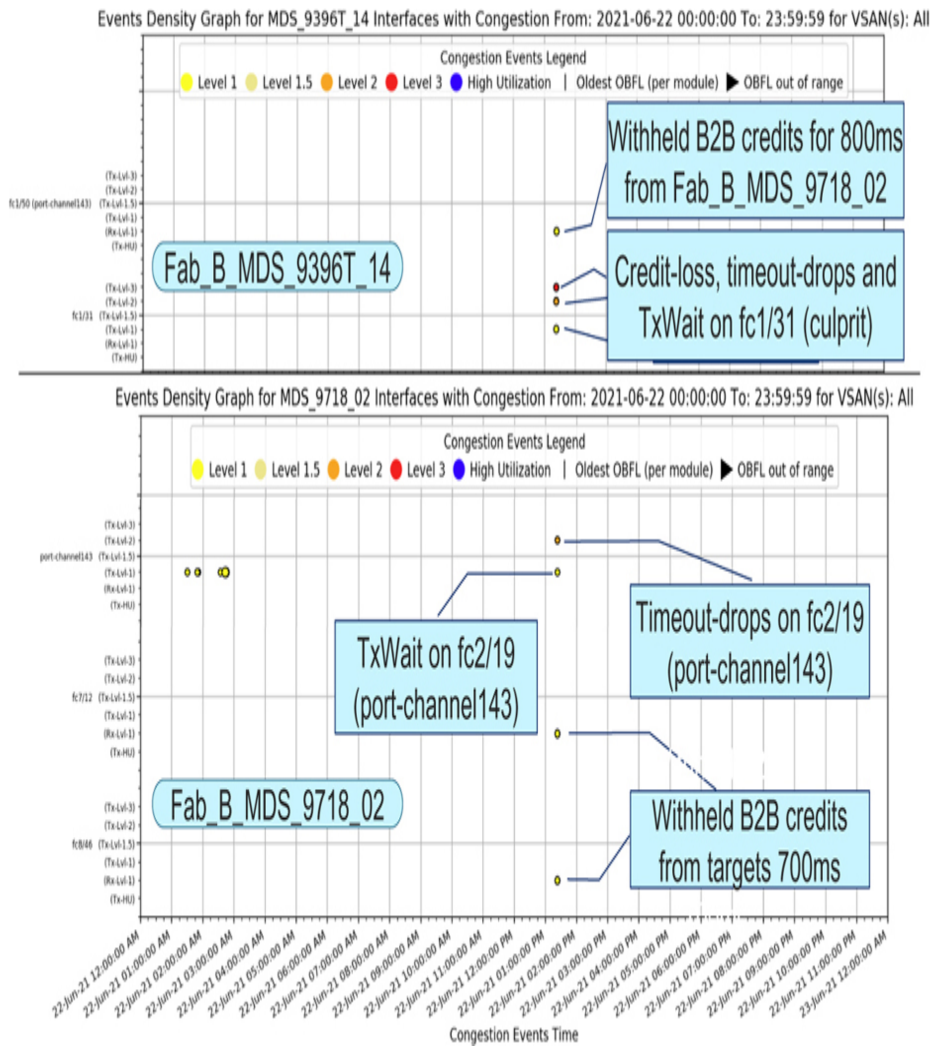


Figure 4-27 *Time correlation of key congestion events on Fab_B_MDS_9396T_14 and Fab_B_MDS_9718_02*

Culprit Analysis

As Figure 4-26 shows, it has been determined that fc1/31 on Fab_B_MDS_9396T_14 had zero remaining-Tx-B2B-credits for 1.06 seconds. During this time the switch dropped 3964 frames to that interface as timeouts. The server connected to fc1/31 was the culprit device and it would have experienced SCSI errors, but the problem was reported by the server that was connected to fc1/62, not fc1/31. The culprit end device was found to be a Linux server as reported by the **show fcns database detail** command on the MDS switch and verified by logging in to that device.

It was possible that the Windows server that reported the problem was an indirect victim of congestion because it's connected switchport fc1/62 on Fab_B_MDS_9396T_14 showed no egress or ingress congestion symptoms. It might have been in communication with other direct or same-path victims. This possibility is explored shortly under the section on *Victim Analysis*.

To understand the root cause for zero remaining-Tx-B2B-credits for 1.06 seconds (which resulted in credit loss recovery) on fc1/31 on Fab_B_MDS_9396T_14, a few more things were checked. As Chapter 2, the section on *B2B Credit Loss and Recovery* explains, credit loss can occur for two basic reasons:

1. Bit errors, causing B2B credits to be lost over time.
2. Severe congestion in the end device due to which it does not return the credits.

These were investigated one by one.

Bit errors can be detected by invalid CRCs, Invalid Transmission Words, FEC errors (especially uncorrectable), and Buffer-to-Buffer State Change Notification counters. Cisco MDS switches show these counters via **show interface counters detailed** command. As Example 4-61 shows, no indications of bit errors were found on fc1/31 on Fab_B_MDS_9396T_14.

Example 4-61 ***show interface counters detailed*** *command.*

```
show interface fc1/31 counters detailed
fc9/17
...

 Link Stats:
  Rx Link failures:
0
  Rx Sync losses:
0
  Rx Signal losses:
0
  Rx Primitive sequence protocol errors:
0
  Rx Invalid transmission words:
0
  Rx Invalid CRCs:
0
  Rx Delimiter errors:
0
  Rx fragmented frames:
0
  Rx frames with EOF aborts:
0
  Rx unknown class frames:
0
  Rx Runt frames:
0
  Rx Jabber frames:
0
  Rx too long:
0
  Rx too short:
0
  Rx FEC corrected blocks:
0
  Rx FEC uncorrected blocks:
```

```
0
  Rx Link Reset(LR) while link is active:
0
  Tx Link Reset(LR) while link is active:
0
  Rx Link Reset Responses(LRR):
0
  Tx Link Reset Responses(LRR):
1
  Rx Offline Sequences(OLS):
0
  Tx Offline Sequences(OLS):
1
  Rx Non-Operational Sequences(NOS):
0
  Tx Non-Operational Sequences(NOS):
0
  BB_SCs credit resend actions:
0
  BB_SCr Tx credit increment actions:
0
```

Bear in mind that these are the errors that the MDS switchport sees. These are not the errors that the adjacent device sees. Errors can occur in one direction only. Errors seen at the adjacent device would normally have to be retrieved from the adjacent device itself.

Alternatively, Cisco MDS switches allow retrieving counters from the connected HBA port using the Read Diagnostic Parameters (RDP) feature. As explained earlier in the section on *Generic Troubleshooting Commands*, RDP can pull the counters of the HBA ports in-band and display them on the switch. The end device HBA needs to support the RDP function for this to work.

Because of the lack of any symptoms, bit errors were ruled out as the reason for congestion.

The second possibility was severe congestion within the end device itself, which could be the HBA layer or its storage stack because of many possible reasons, such as excessive load, bugs, or traffic patterns. These details were outside the scope of this writing. This server should be monitored closely and the fabric can make use of congestion prevention mechanisms explained in Chapter 6.

Overall, it was concluded that the Linux server connected to fc1/31 on Fab_B_MDS_9396T_14 was a culprit due to severe congestion within that server itself. Even though the exact cause of the internal congestion within that server wasn't conclusive, the main goal of this case study is to explore all the possibilities and follow the workflow.

Victim Analysis

Victim devices should be evaluated in the following order:

1. Direct victims.
2. Same-path victims.
3. Indirect victims.

Direct Victims

To find the direct victims, the zoning database was analyzed to find the targets that were in direct communication with the culprit server connected to fc1/31 on Fab_B_MDS_9396T_14. This can be found by **show zone member** and **show zone name** commands on Cisco MDS switches. The next step was to find where these targets were connected. This can be found by **show fcsn database** command. Finally, the **show flogi database** command displayed the interface to which those devices are connected. These steps are already demonstrated in Case Study 1 and under the section on *Generic Troubleshooting Commands*, and are not repeated here for the sake of brevity.

It was found that the culprit server on Fab_B_MDS_9396T_14 interface fc1/31 was zoned with 8 targets connected to the two core 9718 switches. The FLOGI database revealed that these targets were connected to fc1/48, fc2/48, fc7/46, and fc8/46 on both core 9718s.

Then the **show logging error-stats** command for each core switch was investigated for congestion symptoms on any of these 8 interfaces. As Example 4-62 shows, only fc8/46 on Fab_B_MDS_9718_02 had an entry at that time. This entry was for the RX_WT_AVG_B2B_ZERO counter. This counter indicated ingress congestion on a switchport because the switchport was withholding B2B credits from the target.

Example 4-62 *RX_WT_AVG_B2B_ZERO on fc8/46 on Fab_B_MDS_9718_02*

```
---------------------------------
 Module: 8 error-stats
---------------------------------



----------------------------------------------------
--------------------------
 ERROR STATISTICS INFORMATION FOR DEVICE DEVICE:
FCMAC
----------------------------------------------------
--------------------------
 Interface  |                                      |
|    Time Stamp
   Range    |       Error Stat Counter Name       |
Count   |MM/DD/YY HH:MM:SS
            |                                      |
|
fc8/46      |F32_MAC_KLM_CNTR_RX_WT_AVG_B2B_ZERO
|11       |06/22/21 13:23:19
fc8/46      |F32_MAC_KLM_CNTR_RX_WT_AVG_B2B_ZERO  |4
|06/08/21 00:34:12
```

Example 4-62 shows that the target on Fab_B_MDS_9718_02 fc8/46 was a direct victim of the server on Fab_B_MDS_9396T_14 fc1/31.

To determine the amount of time the Fab_B_MDS_9718_02 switch withheld B2B credits from the target on fc8/46 the current value of 11 was subtracted from the previous value of 4, which results in (11 - 4 = 7) 700ms of zero remaining-Rx-B2B-credits. Note that the time stamp in the OBFL entry for module 8 doesn't exactly match the time stamp for the error-stats and TxWait entries for module 2 (Example 4-59 and Example 4-60). This is because each module initializes at a different time in a modular chassis so the time stamps can vary by a small amount. The important thing is that the 20-second interval for fc8/46 overlaps with the timeout-drop and TxWait entry for interface fc2/19 (ISL port).

Other targets communicating with fc1/31 on Fab_B_9396T_14 might also be direct victims but their switchports did not have any RX_WT_AVG_B2B_ZERO increments. This, again, might be due to the 100ms granularity of the RX_WT_AVG_B2B_ZERO counter itself, which misses smaller durations of credit unavailability.

Same-Path Victims

Congestion on ISL (port-channel 143) could have created many same-path victims.

In short, any of the targets on switchname Fab_B_MDS_9718_02 that might have been transmitting to switchname Fab_B_9396T_14 could have been affected. To determine actual same-path victims, we looked for ingress congestion indications on Fab_B_MDS_9718_02. This could potentially be a substantial number of interfaces but besides fc8/46 (already identified direct victim) there was only interface fc7/12. See Example 4-63 for the details. This shows that the switchport withheld B2B credits for a total of 700ms. This made fc7/12 a same-path victim. This appeared to be exactly the same as the backpressure exerted against fc8/46, which was found to be a direct victim.

Example 4-63 *RX_WT_AVG_B2B_ZERO on fc7/12 on Fab_B_MDS_9718_02*

```
 Module: 7 error-stats



 ERROR STATISTICS INFORMATION FOR DEVICE DEVICE:
FCMAC
 Interface  |                                            |
|    Time Stamp
```

```
   Range     |        Error Stat Counter Name       |
Count  |MM/DD/YY HH:MM:SS
             |                                      |
|
fc7/12       |F32_MAC_KLM_CNTR_RX_WT_AVG_B2B_ZERO
|11       |06/22/21 13:23:25
fc7/12       |F32_MAC_KLM_CNTR_RX_WT_AVG_B2B_ZERO  |4
|06/08/21 05:07:29
```

Indirect Victims

As mentioned earlier, the server connected to fc1/62 on Fab_B_MDS_9396T_14 was suspected to be an indirect victim. Recall that indirect victims communicate with direct victims or same-path victims. Let’s verify this.

Next, we found if this server communicated with direct victims. For this, we found if this server was zoned with the same set of targets as the culprit server on Fab_B_MDS_9396T_14 fc1/31. When that was checked, it was found that fc1/31 (FCID 0x6a0440) was zoned with 8 targets across both 9718 core switches. Interface fc1/62 (FCID 0x6a0160) was only zoned with 4 targets, also distributed across both 9718 core switches. These two sets of targets were distinct from each other. Consequently, the server connected to fc1/62 on Fab_B_MDS_9396T_14 was not in communication with the direct victims.

Then we found if this server communicated with the same-path victim. To determine which targets the same-path victim Fab_B_MDS_9718_02 fc7/12 was communicating with, the zoning database was checked. The server connected to Fab_B_MDS_9396T_14 fc1/62 was zoned with four total targets: fc2/12 and fc7/12 on both Fab_B_MDS_9718_01 and Fab_B_MDS_9718_02. Since there was backpressure exerted against fc7/12 (identified previously as a same-path victim) and fc7/12 was in communication with fc1/62, fc1/62 looked like an indirect victim.

However, to prove this, the actual dropped frames were investigated. When frames are dropped due to timeout, Cisco

MDS switches maintain the headers for the last few dropped frames.

Example 4-64 shows that for fc2/19 (slot 2 port 19), the last 4 frames dropped due to timeout were for destination FCIDs 0x6a0160 and 0x6a0440. FCID 0x6a0440 was expected because it was for fc1/31 (Example 4-53) on Fab_B_MDS_9396T_14 (the culprit interface with CREDIT_LOSS). FCID 0x6a0160 was for fc1/62 on Fab_B_MDS_9396T_14 (Example 4-40), which was the server that originally reported the errors. FCID to interface mapping is available in the **show fcns database detail** command. The source FCID was 0x220960 which was for fc7/12(same-path victim). With this information, we now had the tie-in between the credit loss that was found on fc1/31 and the Windows server on fc1/62 on Fab_B_MDS_9396T_14 that originally reported the problem. This means the server on fc1/62 on Fab_B_MDS_9396T_14 was communicating with the same-path victim with FCID 0x220960 (fc7/12). This was conclusive evidence that the server reporting the problem, connected to interface fc1/62 on Fab_B_MDS_9396T_14, was an indirect victim of the same-path victim connected to interface fc7/12 on Fab_B_MDS_9718_02.

Example 4-64 *Details of the frames dropped due to timeout on Fab_B_MDS_9718_02*

```
slot 2 show hardware internal fcmac port 19
tmm_timeout_stat_buffer
+----------------------------------------------------
-----------------------+
| PORT:18 ASIC PORT: 9 PG:2 PG PORT:1 START: 4 END:
7 WR:7 RD:3 NUM PKTS:4 |
+-------+--------+--------+---+----+------+------+---
-+----+----+----+-+-----+
|Delay |  Chip  |Vegashdr|TS | FC | Src  | Dest
|RCTL| CTL| SI | DI |A|OFF |
|(msec)|time(0x)|time(0x)|VLD|TYPE|  ID  | ID   |
(0x)|(0x)|(0x)|(0x)|T|LINE|
+-------+--------+--------+---+----+------+------+---
-+----+----+----+-+-----+
|   940|    1d00|    1ca2|  1|   8|220960|6a0160|
1|1800| 15f|    9|0|    0|
|    940|     1d00|     1ca2|  1|    8|220960|6a0160|
1|1800| 15f|    9|0|    0|
|    940|     1d00|     1ca2|  1|    8|220300|6a0440|
1|1800|  ed|    9|0|    0|
|    940|     1d00|     1ca2|  1|    8|220300|6a0440|
1|1800|  ed|    9|0|    0|
+------+--------+--------+---+----+------+------+---
-+----+----+----+-+-----+
```

Note that Example 4-63 shows an earlier form of the **show logging onboard flow-congestion-drops** command. Later releases of MDS NX-OS have changed this command.

Interfaces fc7/12 and fc8/46 on switchname Fab_B_MDS_9718_02 are included in the graph shown in Figure 4-27. The vertical (time) alignment of the egress and ingress events demonstrates that all of these events were related.

At this point, the complete set of same-path and indirect victims could have been determined. The set of indirect victims was really the entire set of initiators logged into Fab_B_MDS_9396T_14, which could be determined by looking at the active zoneset to determine the set of initiators the targets on fc7/12 and fc8/46 were zoned with. If all the victims were identified, it would become apparent the extent that congestion spreading affected the fabric.

Case Study 2—Summary

This case study illustrates that the adverse effects of congestion can be reported even on indirect victims (Windows server connected to fc1/62 on Fab_B_9396T_14). The culprit (Linux server connected to fc1/31 on Fab_B_9396T_14) must also have been affected but wasn't reported when the investigation started.

There might have been other victims and they would have reported problems as well. But until the symptoms are investigated, it is not known if the reported device is a culprit or a victim. Following the troubleshooting methodology and workflow helps in clarifying the difference.

To summarize the culprit and victims in this case study, refer to Figure 4-28. The server on Fab_B_MDS_9396T_14 interface fc1/31 was the culprit because it withheld B2B credits from Fab_B_MDS_9396T_14 for 1 full second causing credit loss. This congestion spread, causing Fab_B_MDS_9396T_14 to withhold B2B credits on ISL fc1/50 (connected to fc2/19 on Fab_B_MDS_9718_02), a member of port-channel 143. Since Fab_B_MDS_9718_02 fc2/19 could not transmit for 1 second, it dropped 535 frames. Some of these frames were for fc1/31 but some were also destined to the server on fc1/62, making it an indirect victim. Recall that the original problem was reported for this server.

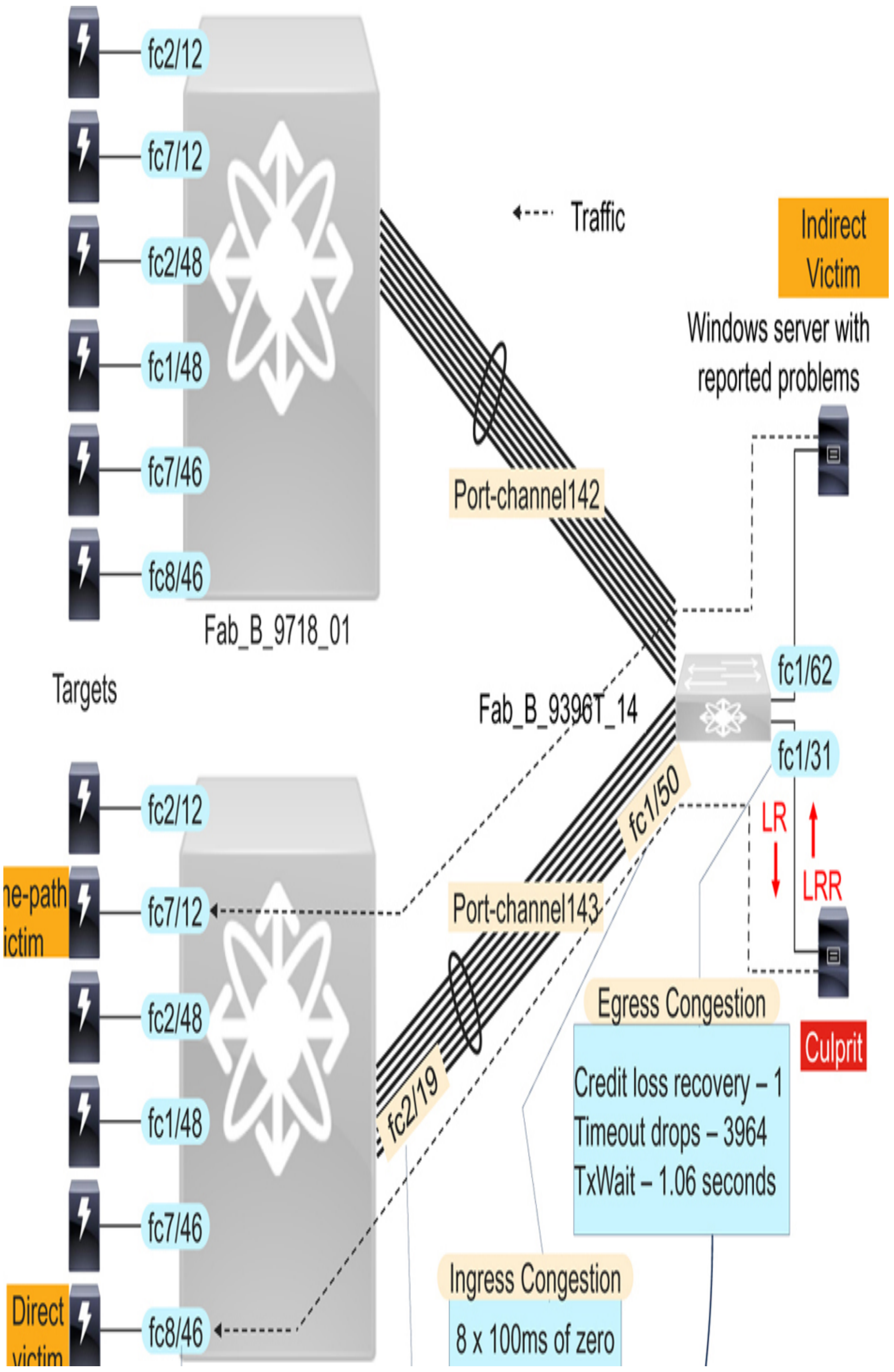


Figure 4-28 *Fabric B topology showing congestion spreading resulting in different types of victims*

The target on fc8/46 on Fab_B_MDS_9718_02 was the direct victim since it was in direct communication with the culprit. The target on fc7/12 was a same-path victim because it was transmitting to the server connected to Fab_B_MDS_9396T_14 fc1/62 across the congested ISL between Fab_B_MDS_9718_02 and Fab_B_MDS_9396T_14.

Any other hosts that were in communication with those two targets would have been indirect victims as well even if they were not connected to Fab_B_MDS_9396T_14 since those targets were prevented from transmitting to any hosts for 700ms each.

Finally, regardless of the size of the fabric, focus on the culprit and victim devices and their traffic paths. Do not be overwhelmed by the size of the fabric. As this case study illustrates, among the 49 switches in the fabric, only three had to be investigated.

Case Study 3—Over Utilization on a Single Device Causing Massive Congestion Problems

A large financial services company reported that multiple hosts experienced disk connectivity error messages from approximately midnight to the early hours of the morning on Oct 25. They suspected a slow drain device somewhere in the storage network and wanted to verify if congestion was the root cause of the problem.

Their fabric topology is a ‘semi-mesh’ (Figure 4-29) of seven switches made up of three MDS 9710, three MDS 9513, and an MDS 9396S. The MDS 9513 switches have 8GFC ports, whereas MDS 9710 and MDS 9396S have 16GFC ports.

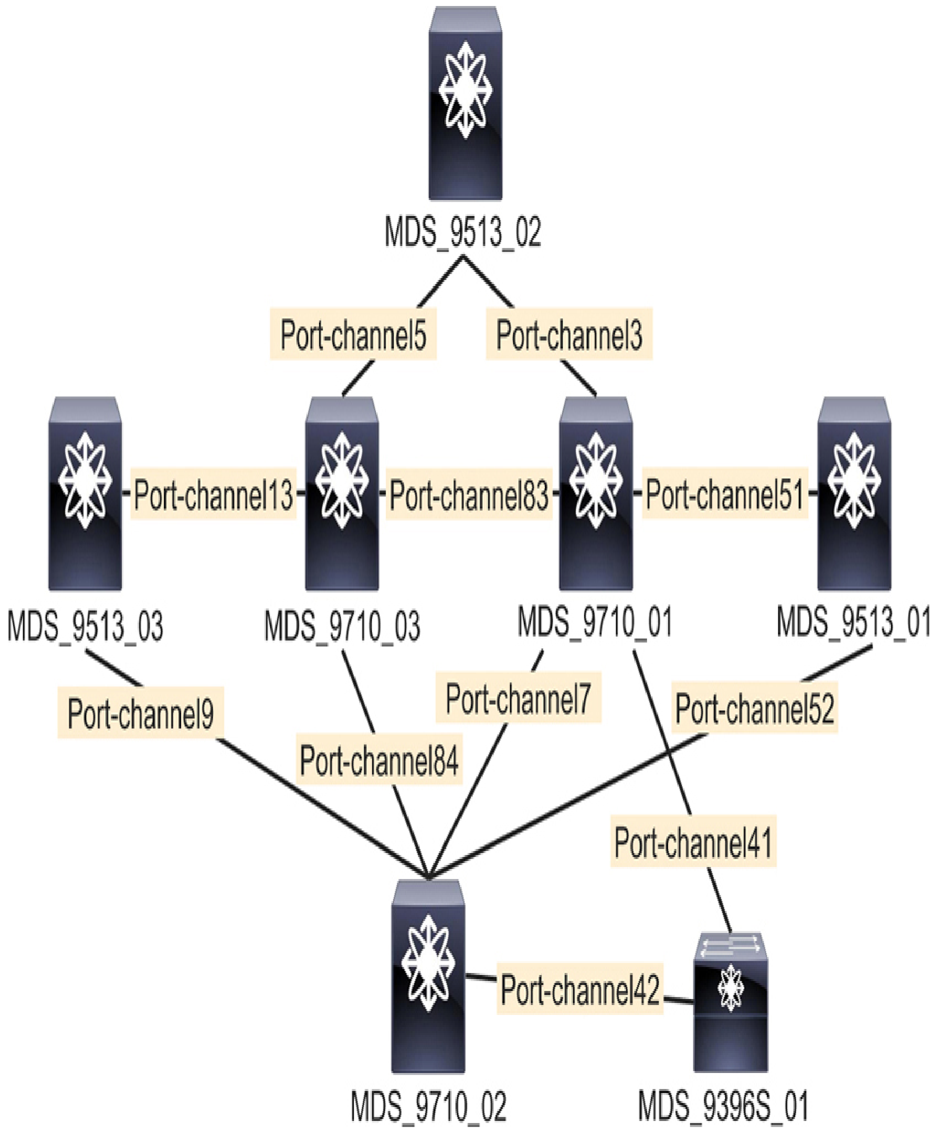


Figure 4-29 *Storage network topology*

They gathered output of **show tech-support slowdrain** command from all the switches in the fabric around 3 AM. This gave a complete view of all the switches in the fabric.

Not much information was provided so the entire fabric needed to be investigated starting with severe (Level 3) indications followed by moderate (Level 2) and Mild (Level 1.5 and Level 1) indications.

Level 3

The **show logging onboard error-stats** *command across all seven switches did not show any instances of CREDIT_LOSS*.

Level 2

The **show logging onboard error-stats** command from MDS_9513_03 and MDS_9710_03 showed timeout drops in the same period when the problem was reported. No other switches had any Level 2 indication.

MDS_9513_03

MDS_9513_03 had two instances of timeout drops on fc1/8 and fc1/9 (Example 4-65).

Example 4-65 *Timeout drops history in OBFL on MDS_9513_03*

```
show logging onboard error-stats

---------------------------------
 Module: 1 error-stats
---------------------------------



----------------------------------------------------
--------------------------
 ERROR STATISTICS INFORMATION FOR DEVICE DEVICE:
FCMAC
----------------------------------------------------
--------------------------
 Interface  |                                     |
|    Time Stamp
   Range    |       Error Stat Counter Name       |
Count   |MM/DD/YY HH:MM:SS
----------------------------------------------------
--------------------------
fc1/9       |THB_TMM_TOLB_TIMEOUT_DROP_CNT
|453      |10/25/17 02:04:33
fc1/8       |THB_TMM_TOLB_TIMEOUT_DROP_CNT
|644      |10/25/17 00:01:32
```

MDS_9513_03 also had TxWait of 2 and 3 seconds on fc1/9 and fc1/8 during the time of the timeout-drops (Example 4-

66).

Example 4-66 *TxWait history on OBFL on MDS_9513_03*

```
show logging onboard txwait
---------------------------------
 Module: 1 txwait
---------------------------------
 Notes:
     - Sampling period is 20 seconds
     - Only txwait delta >= 100 ms are logged

----------------------------------------------------
-------------------------
| Interface | Delta TxWait Time     | Congestion |
Timestamp                 |
|           | 2.5us ticks | seconds |            |
|
----------------------------------------------------
-------------------------
|   fc1/9   |   961842    |    2    |      12%   |
Wed Oct 25 02:04:33 2017 |
|   fc1/8   |  1566044    |    3    |      19%   |
Wed Oct 25 00:01:32 2017 |
```

MDS_9513_03 also had slowport-monitor configured at admin delay of 25ms so there were slowport-monitor-events as well, again on fc1/8 and fc1/9 (Example 4-67).

Example 4-67 *Slowport-monitor history in OBFL on MDS_9513_03*

```
show logging onboard slowport-monitor-events

...

---------------------------------
 Module: 1 slowport-monitor-events
---------------------------------


----------------------------------------------------
```

```
----------------------
| admin  | slowport  | txwait|           Timestamp
| Interface
| delay  | detection | oper  |
|
| (ms)   | count     | delay |
|
|        |           |  (ms) |
|
----------------------------------------------------
----------------------
|  25    |      7835 |  100  | 10/25/17 02:04:33.203
| fc1/9
|  25    |      7834 |   99  | 10/25/17 02:04:33.102
| fc1/9
|  25    |      7833 |   98  | 10/25/17 02:04:33.002
| fc1/9
|  25    |      7832 |  100  | 10/25/17 02:04:32.902
| fc1/9
|  25    |      7831 |   98  | 10/25/17 02:04:32.802
| fc1/9
|  25    |      7830 |   99  | 10/25/17 02:04:32.702
| fc1/9
|  25    |      7829 |  100  | 10/25/17 02:04:32.602
| fc1/9
|  25    |      7828 |   98  | 10/25/17 02:04:32.502
| fc1/9
|  25    |      7827 |   99  | 10/25/17 02:04:32.403
| fc1/9
|  25    |      7826 |   99  | 10/25/17 02:04:32.302
| fc1/9
...
|  25    |      7450 |  100  | 10/25/17 00:01:26.184
| fc1/8
|  25    |      7449 |   98  | 10/25/17 00:01:26.083
| fc1/8
|  25    |      7448 |   56  | 10/25/17 00:01:25.984
| fc1/8
|  25    |      7447 |   64  | 10/25/17 00:01:23.573
| fc1/8
|  25    |      7446 |   98  | 10/25/17 00:01:23.473
| fc1/8
|  25    |      7445 |  100   | 10/25/17 00:01:23.373
| fc1/8
|  25    |      7444 |   98   | 10/25/17 00:01:23.273
| fc1/8
|  25    |      7443 |  100   | 10/25/17 00:01:23.174
| fc1/8
|  25    |      7442 |   98   | 10/25/17 00:01:22.773
| fc1/8
|  25    |      7441 |  100   | 10/25/17 00:01:22.674
| fc1/8
```

Example 4-68 shows all this information together, with many indications of egress congestion on fc1/8 and fc1/9. In the 20-second intervals with the timeout-drops there were 3 seconds of TxWait on fc1/8 and 2 seconds of TxWait on fc1/9. Those were comprised of multiple 100ms (or near 100ms) continuous delays. That delay caused the timeout-drops when frames did not exit the egress switchport within 500 ms.

Example 4-68 *Egress congestion on fc1/8 and fc1/9 on MDS_9513_03*

```

-------------------------------T i m e o u t  D r o
p s-----------------------
| Intf  | Counter                | Count   | Delta
| Timestamp
 ---------------------------------------------------
-------------------------
| fc1/8 | <snip>TIMEOUT_DROP_CNT |     644 |    409
| 2017/10/25 00:01:32
| fc1/8 | TxWait Congestion 19%  | 3(secs) |1566044
|
| fc1/8 | Slowport_monitor_event |    7450 |  100ms
| 2017/10/25 00:01:26.184
| fc1/8 | Slowport_monitor_event |    7449 |   98ms
| 2017/10/25 00:01:26.083
| fc1/8 | Slowport_monitor_event |    7448 |   56ms
| 2017/10/25 00:01:25.984
| fc1/8 | Slowport_monitor_event |    7447 |   64ms
| 2017/10/25 00:01:23.573
| fc1/8 | Slowport_monitor_event |    7446 |   98ms
| 2017/10/25 00:01:23.473
| fc1/8 | Slowport_monitor_event |    7445 |  100ms
| 2017/10/25 00:01:23.373
| fc1/8 | Slowport_monitor_event |    7444 |   98ms
| 2017/10/25 00:01:23.273
| fc1/8 | Slowport_monitor_event |    7443 |  100ms
| 2017/10/25 00:01:23.174
| fc1/8 | Slowport_monitor_event |    7442 |   98ms
| 2017/10/25 00:01:22.773
| fc1/8 | Slowport_monitor_event |    7441 |  100ms
| 2017/10/25 00:01:22.674
| fc1/9 | <snip>TIMEOUT_DROP_CNT |     453 |    366
| 2017/10/25 02:04:33
| fc1/9 | TxWait Congestion 12%  | 2(secs) | 961842
|
| fc1/9 | Slowport_monitor_event |    7835 |  100ms
| 2017/10/25 02:04:33.203
| fc1/9 | Slowport_monitor_event |    7834 |   99ms
| 2017/10/25 02:04:33.102
| fc1/9 | Slowport_monitor_event |    7833 |   98ms
| 2017/10/25 02:04:33.002
| fc1/9 | Slowport_monitor_event |    7832 |  100ms
| 2017/10/25 02:04:32.902
| fc1/9 | Slowport_monitor_event |    7831 |   98ms
| 2017/10/25 02:04:32.802
| fc1/9 | Slowport_monitor_event |    7830 |   99ms
| 2017/10/25 02:04:32.702
| fc1/9 | Slowport_monitor_event |    7829 |  100ms
| 2017/10/25 02:04:32.602
| fc1/9 | Slowport_monitor_event |    7828 |   98ms
| 2017/10/25 02:04:32.502
| fc1/9 | Slowport_monitor_event |    7827 |   99ms
| 2017/10/25 02:04:32.403
| fc1/9 | Slowport_monitor_event |    7826 |   99ms
| 2017/10/25 02:04:32.302
 ---------------------------------------------------
-------------------------
```

Figure 4-30 shows a plot of events shown in Example 4-68 on fc1/8 and fc1/9. There were two timeout drop events (the orange dots in the Tx-Lvl-2 row, one for each interface fc1/8 and fc1/9. These were at different times but were both early in the morning. Figure 4-30 also shows almost continuous yellow dots in the Tx-Lvl-1 row indicating TxWait < 30% indications.

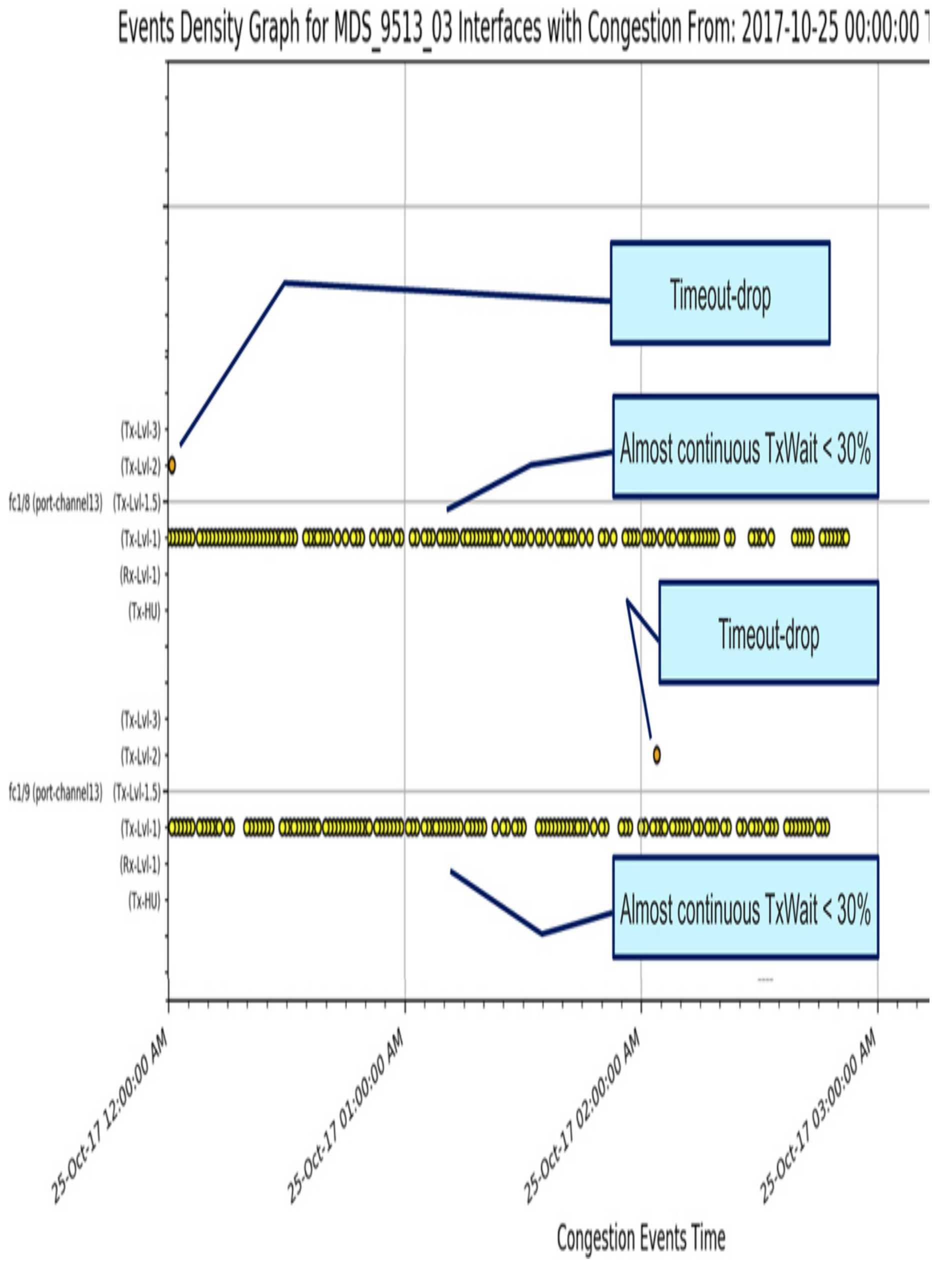


Figure 4-30 *Graph of TxWait and Timeout-drops for fc1/8 and fc1/9 on MDS_9513_03 on Oct 25 between midnight and 4AM*

As Example 4-69 shows, fc1/8 and fc1/9 both belonged to port-channel13 on MDS_9513_03.

Example 4-69 *Members of a port-channel on MDS_9513_03*

```
show interface

fc1/8 is trunking
    Hardware is Fibre Channel, SFP is short wave
laser w/o OFC (SN)
...
    Belongs to port-channel13

fc1/9 is trunking
    Hardware is Fibre Channel, SFP is short wave
laser w/o OFC (SN)
...
    Belongs to port-channel13
```

Port-channel13 connected to MDS_9710_03 (Example 4-70).

Example 4-70 *Neighbors of MDS_9513_03*

```
show topology

FC Topology for VSAN 1 :
-----------------------------------------------------
---------------------------
       Interface  Peer Domain Peer Interface
Peer IP Address(Switch Name)
-----------------------------------------------------
---------------------------
...
   port-channel13   0x06(6)   port-channel13
x.x.x.x(MDS_9710_03)
```

This investigation gave a good picture of the congestion on MDS_9513_03, and also that the culprit was towards MDS_9710_03 because of various egress congestion symptoms on MDS_9513_03.

Since port-channel13 was connected to MDS_9710_03, following the methodology, the troubleshooting continued to MDS_9710_03 (Figure 4-31).

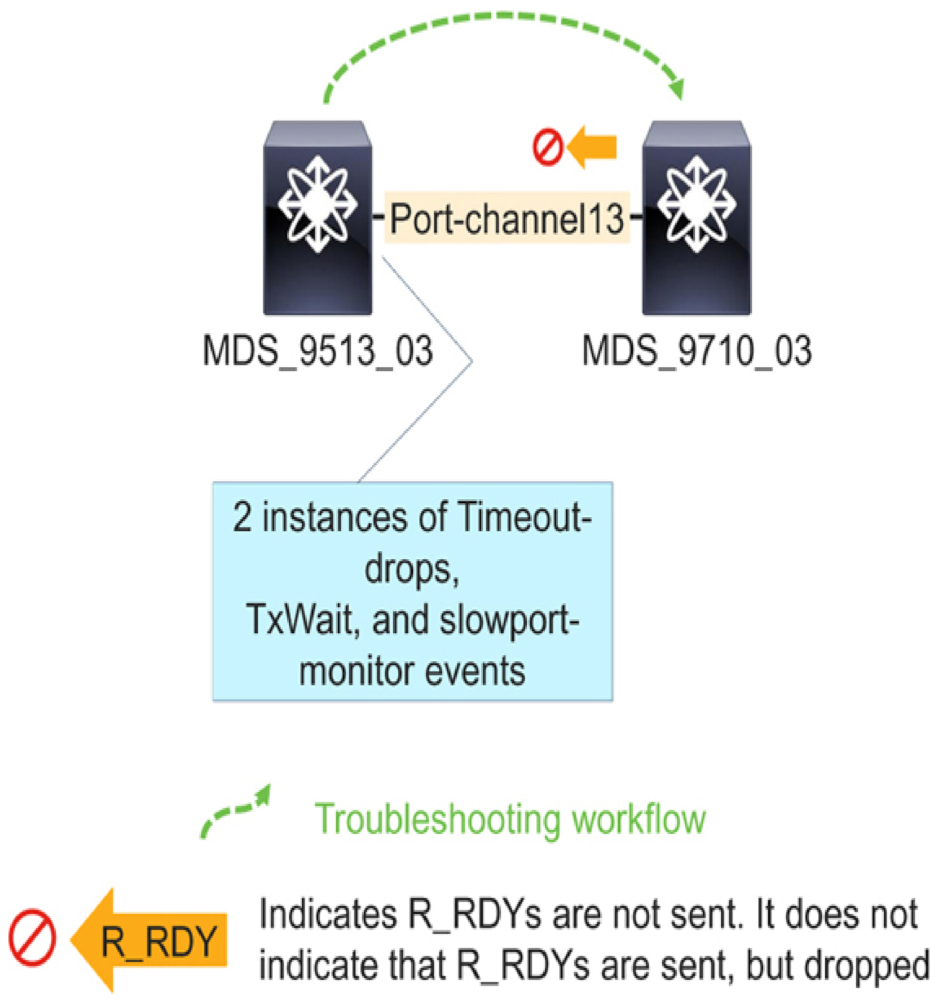

Port-channel13
MDS_9513_03
MDS_9710_03
2 instances of Timeout-drops, TxWait, and slowport-monitor events
Troubleshooting workflow
R_RDY
Indicates R_RDYs are not sent. It does not indicate that R_RDYs are sent, but dropped

Figure 4-31 *Congestion troubleshooting methodology and workflow*

MDS_9710_03

Even on this switch, the same methodology was followed starting with severe (Level 3) indications followed by moderate (Level 2) and Mild (Level 1.5 and Level 1) indications. For this, the **show logging onboard error-stats** command was executed. No Level 3 indications were seen. But Level 2 indications were seen because of 92 instances of timeout-drops on interfaces fc2/3 and fc2/5 (Example 4-71).

Example 4-71 *Timeout drops history in OBFL on MDS_9710_03*

```
show logging onboard error-stats
--------------------------------
 Module: 2 error-stats
```

```
---------------------------------
-----------------------------------------------------
--------------------------
 ERROR STATISTICS INFORMATION FOR DEVICE DEVICE:
FCMAC
-----------------------------------------------------
--------------------------
 Interface  |                                      |
|    Time Stamp
   Range    |       Error Stat Counter Name        |
Count  |MM/DD/YY HH:MM:SS
            |                                      |
|

fc2/5        |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|9803     |10/25/17 02:41:06
fc2/5        |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|9802     |10/25/17 02:34:46
fc2/5        |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|9760     |10/25/17 02:26:26
fc2/5        |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|9644     |10/25/17 02:23:06
fc2/3        |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|10429    |10/25/17 02:22:26
fc2/5        |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|9534     |10/25/17 02:18:46
fc2/5        |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|9281     |10/25/17 02:18:26
fc2/5        |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|9174     |10/25/17 02:17:46
fc2/5        |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|8726     |10/25/17 02:09:46
fc2/5        |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|8682     |10/25/17 02:08:26
fc2/3        |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|9209     |10/25/17 02:08:06
fc2/5        |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|8372     |10/25/17 02:04:46
fc2/5        |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|7989     |10/25/17 02:03:46
fc2/5        |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|7741     |10/25/17 01:55:26
```

```
fc2/3          |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|8952      |10/25/17 01:53:46
fc2/3          |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|8941      |10/25/17 01:51:26
fc2/5          |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|7717      |10/25/17 01:51:06
fc2/5          |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|7657      |10/25/17 01:47:46
fc2/3          |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|8583      |10/25/17 01:47:46
...65 lines deleted...
fc2/3          |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|270       |10/25/17 00:19:46
fc2/3          |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|233       |10/25/17 00:19:26
fc2/5          |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|340       |10/25/17 00:15:06
fc2/5          |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|293       |10/25/17 00:12:46
fc2/5          |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|125       |10/25/17 00:08:26
fc2/3          |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|218       |10/25/17 00:05:06
fc2/5          |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|104       |10/25/17 00:04:46
fc2/3          |F16_TMM_TOLB_TIMEOUT_DROP_CNT
|33        |10/24/17 23:47:26
```

Next, Level 1 indications were checked using the **show logging onboard txwait** command. During the period of timeout-drops, huge amounts of TxWait were seen on fc2/3 and fc2/5 and 8 other ports (Example 4-72).

Example 4-72 *TxWait drops history in OBFL on MDS_9710_03*

```
show logging onboard txwait

----------------------------
Module: 1 show clock
----------------------------
2017-10-25 03:09:25

---------------------------------
```

```
 Module: 1 txwait
---------------------------------
 Notes:
     - Sampling period is 20 seconds
     - Only txwait delta >= 100 ms are logged

-----------------------------------------------------
------------------------
| Interface | Delta TxWait Time     | Congestion |
Timestamp                 |
|           | 2.5us ticks | seconds |            |
|
-----------------------------------------------------
------------------------
|   fc1/3   |  3798017    |    9    |      47%   |
Wed Oct 25 02:52:29 2017 |
|   fc1/3   |  2904887    |    7    |      36%   |
Wed Oct 25 02:52:09 2017 |
|   fc1/1   |  2819801    |    7    |      35%   |
Wed Oct 25 02:51:49 2017 |
|   fc1/2   |  3138216    |    7    |      39%   |
Wed Oct 25 02:50:49 2017 |
|   fc1/2   |  2951451    |    7    |      36%   |
Wed Oct 25 02:50:09 2017 |
|   fc1/3   |  2472858    |    6    |      30%   |
Wed Oct 25 02:48:29 2017 |
|   fc1/1   |  3957727    |    9    |      49%   |
Wed Oct 25 02:47:49 2017 |
...
|   fc1/3   |  3092111    |    7    |      38%   |
Wed Oct 25 00:04:49 2017 |
|   fc1/2   |  3700355    |    9    |      46%   |
Wed Oct 25 00:04:29 2017 |
|   fc1/1   |  2821552    |    7    |      35%   |
Wed Oct 25 00:03:09 2017 |
|   fc1/2   |  2517849    |    6    |      31%   |
Wed Oct 25 00:02:49 2017 |
|   fc1/3   |  3342971    |    8    |      41%   |
Wed Oct 25 00:02:29 2017 |
|   fc1/3   |  2779333    |    6    |      34%   |
Wed Oct 25 00:02:09 2017 |
|   fc1/3   |  2726953    |    6    |      34%   |
Wed Oct 25 00:00:09 2017 |

---------------------------------
 Module: 2 txwait
---------------------------------
 Notes:
     - Sampling period is 20 seconds
     - Only txwait delta >= 100 ms are logged

-----------------------------------------------------
------------------------
| Interface | Delta TxWait Time     | Congestion |
Timestamp                |
|           | 2.5us ticks | seconds |             |
|
-----------------------------------------------------
------------------------
|    fc2/5    |  3156225    |    7    |       39%    |
Wed Oct 25 02:51:46 2017 |
|    fc2/3    |  3103348    |    7    |       38%    |
Wed Oct 25 02:51:26 2017 |
|    fc2/3    |  2868378    |    7    |       35%    |
Wed Oct 25 02:51:06 2017 |
|    fc2/1    |  2525762    |    6    |       31%    |
Wed Oct 25 02:51:06 2017 |
|    fc2/1    |  2824821    |    7    |       35%    |
Wed Oct 25 02:50:46 2017 |
|    fc2/3    |  2812005    |    7    |       35%    |
Wed Oct 25 02:49:46 2017 |
|    fc2/1    |  2804691    |    7    |       35%    |
Wed Oct 25 02:49:46 2017 |
|    fc2/3    |  2979742    |    7    |       37%    |
Wed Oct 25 02:49:06 2017 |
|    fc2/1    |  2520536    |    6    |       31%    |
Wed Oct 25 02:48:26 2017 |
|    fc2/3    |  2912907    |    7    |       36%    |
Wed Oct 25 02:48:06 2017 |
|    fc2/1    |  4053960    |   10    |       50%    |
Wed Oct 25 02:47:06 2017 |
...
|    fc2/3    |  2421106    |    6    |       30%    |
Wed Oct 25 00:00:46 2017 |
|    fc2/3    |  2678443    |    6    |       33%    |
Wed Oct 25 00:00:26 2017 |
|    fc2/5    |  3126338    |    7    |       39%    |
Wed Oct 25 00:00:06 2017 |
|    fc2/3    |  2464619    |    6    |       30%    |
Wed Oct 25 00:00:06 2017 |
```

The ten interfaces that showed high TxWait (fc1/1-5 and fc2/1-5) were all part of port-channel83 (Example 4-73).

Example 4-73 *Members of a port-channel on MDS_9710_03*

```
show interface
port-channel83 is trunking
    Port description is To MDS_9710_01
    Hardware is Fibre Channel
...
    Trunk vsans (admin allowed and active)
(1,3,5,7,9,23,1001)
    Trunk vsans (up)
```

```
(1,3,5,7,9,23,1001)
    Trunk vsans (isolated)                  ()
    Trunk vsans (initializing)              ()
    5 minutes input rate 1568708192
bits/sec,196088524 bytes/sec, ...
    Member[1] : fc1/1
    Member[2] : fc1/2
    Member[3] : fc1/3
    Member[4] : fc1/4
    Member[5] : fc1/5
    Member[6] : fc2/1
    Member[7] : fc2/2
    Member[8] : fc2/3
    Member[9] : fc2/4
    Member[10] : fc2/5
```

Figure 4-32 shows a plot of events shown in Example 4-71 and Example4-72. Timeout-drops (orange dots in the Tx-Lvl-2 row) on the bottom two interfaces fc2/3 and fc2/5 both in port-channel83. fc2/3 and fc2/5 also have lots of TxWait < 30 % (yellow dots in the Tx-Lvl-1 row) and TxWait >=30% (kakhi dots in the Tx-Lvl-1.5 row). The TxWait < 30% and TxWait >=30% are similar on the other 4 interfaces in the same port-channel83. Between both levels of TxWait it shows that the congestion was almost completely continuous on those six interfaces. Interestingly, timeout-drops were on just two of the ten interfaces: fc2/3 and fc2/5. This must be due to the traffic pattern at that time, that is, the limited number of exchanges and how the exchanges were load-balanced.

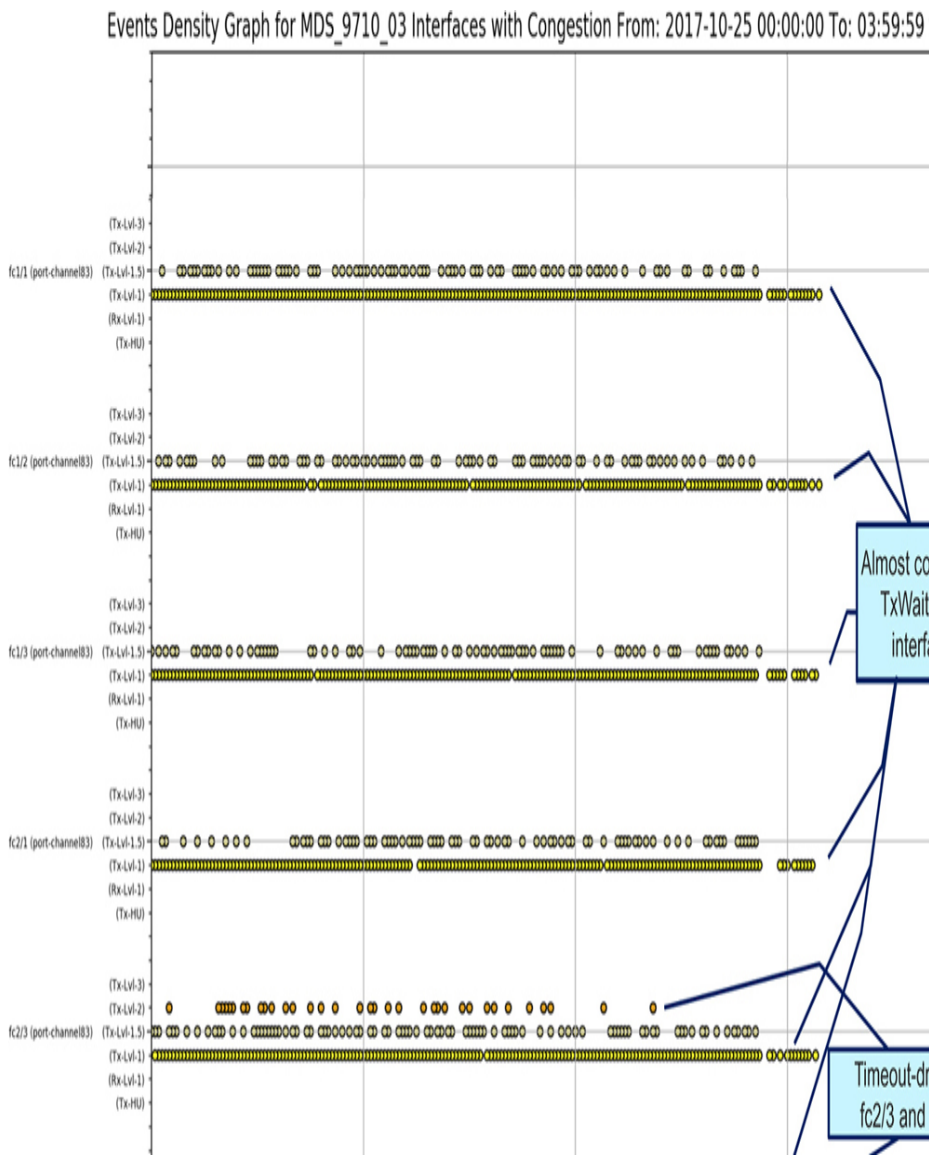


Figure 4-32 *Graph of TxWait and Timeout-drops on MDS_9710_03*

As Example 4-74 shows, Port-channel83 was connected to MDS_9710_01.

Example 4-74 *Neighbor of MDS_9710_03*

```
show topology

FC Topology for VSAN 1 :
        Interface  Peer Domain Peer Interface
Peer IP Address(Switch Name)
-----------------------------------------------------
-------------------------
```

```
   port-channel83    0x02(2)    port-channel83
x.x.x.x(MDS_9710_01)
```

Since port-channel83 was connected to MDS_9710_01, following the methodology, the troubleshooting continued on that switch (Figure 4-33).

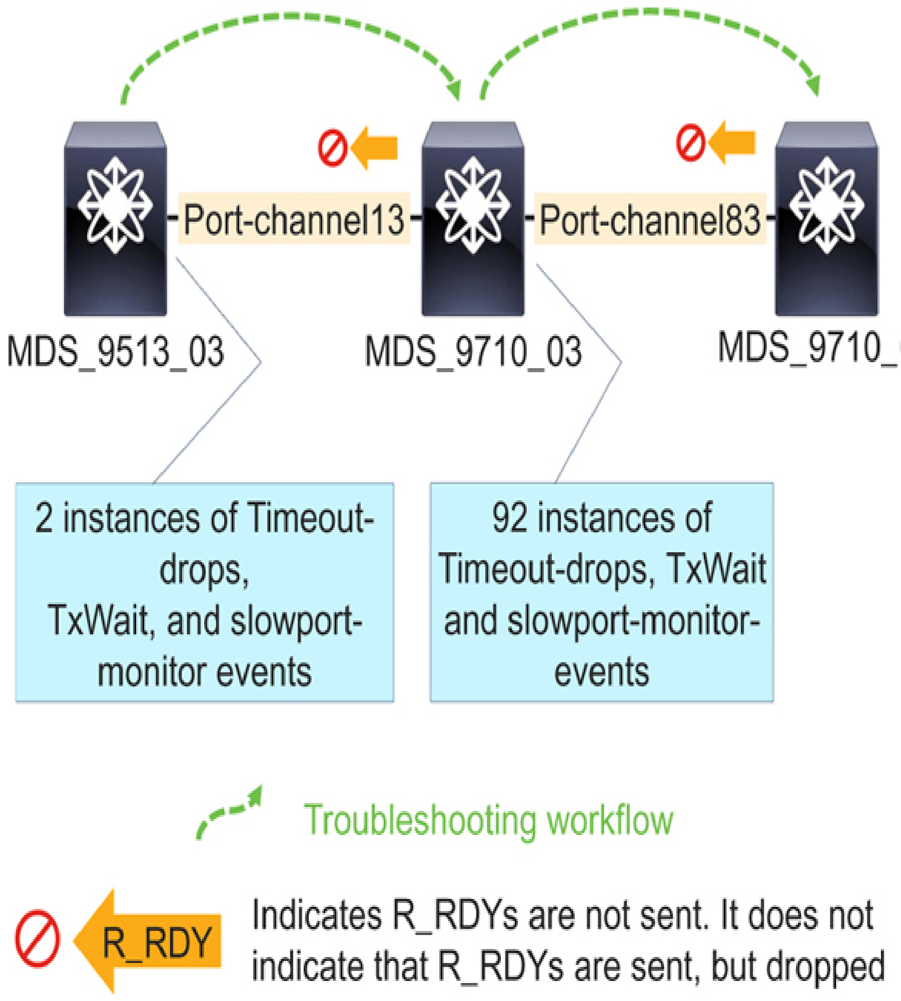

Port-channel13
Port-channel83
MDS_9513_03
MDS_9710_03
MDS_9710_01
2 instances of Timeout-drops, TxWait, and slowport-monitor events
92 instances of Timeout-drops, TxWait and slowport-monitor-events
Troubleshooting workflow
R_RDY
Indicates R_RDYs are not sent. It does not indicate that R_RDYs are sent, but dropped

Figure 4-33 *Congestion troubleshooting methodology and workflow*

MDS_9710_01

The same methodology of higher to lower severity events was followed even on this switch. The **show logging onboard error-stats** command did not show any Level 3 (credit loss) and Level 2 (timeout drops). But Level 1.5 TxWait (up to 85%) was shown on 8 interfaces: fc1/14-18 fc2/14-18 (Example 4-75).

Example 4-75 *TxWait history in OBFL on MDS_9710_01*

```
show logging onboard txwait

---------------------------------
 Module: 1 txwait
---------------------------------
 Notes:
     - Sampling period is 20 seconds
     - Only txwait delta >= 100 ms are logged

-----------------------------------------------------
-------------------------
| Interface | Delta TxWait Time     | Congestion |
Timestamp                |
|           | 2.5us ticks | seconds |            |
|
-----------------------------------------------------
-------------------------
|   fc1/18  |  2707816    |    6    |      33%   |
Wed Oct 25 03:08:57 2017 |
|   fc1/18  |  2598225    |    6    |      32%   |
Wed Oct 25 03:08:17 2017 |
|   fc1/15  |  2438351    |    6    |      30%   |
Wed Oct 25 03:07:57 2017 |
|   fc1/18  |  2652023    |    6    |      33%   |
Wed Oct 25 03:07:37 2017 |
|   fc1/15  |  3074730    |    7    |      38%   |
Wed Oct 25 03:07:17 2017 |
|   fc1/18  |  2971620    |    7    |      37%   |
Wed Oct 25 03:06:57 2017 |
|   fc1/18  |  2580904    |    6    |      32%   |
Wed Oct 25 03:05:57 2017 |
|   fc1/15  |  2881755    |    7    |      36%   |
Wed Oct 25 03:04:17 2017 |
|   fc1/18  |  2496238    |    6    |      31%   |
Wed Oct 25 03:03:37 2017 |
...
|   fc1/18  |  6024093    |   15    |      75%   |
Wed Oct 25 00:00:56 2017 |
|   fc1/15  |  4516534    |   11    |      56%   |
Wed Oct 25 00:00:56 2017 |
|   fc1/14  |  2495824    |    6    |      31%   |
Wed Oct 25 00:00:56 2017 |
|   fc1/18  |  4823198    |   12    |      60%   |
Wed Oct 25 00:00:36 2017 |
|   fc1/15  |  4533361    |   11    |      56%   |
Wed Oct 25 00:00:36 2017 |
|   fc1/14  |  2832063    |    7    |      35%   |
Wed Oct 25 00:00:36 2017 |
|   fc1/18  |  5567551    |   13    |      69%   |
Wed Oct 25 00:00:16 2017 |
|   fc1/15  |  4723513    |   11    |      59%   |
Wed Oct 25 00:00:16 2017 |

---------------------------------
 Module: 2 txwait
---------------------------------
 Notes:
     - Sampling period is 20 seconds
     - Only txwait delta >= 100 ms are logged

----------------------------------------------------
-------------------------
| Interface | Delta TxWait Time     | Congestion |
Timestamp                |
|           | 2.5us ticks | seconds |            |
|
----------------------------------------------------
-------------------------
|   fc2/15  |  3030792    |    7    |      37%   |
Wed Oct 25 03:09:16 2017 |
|   fc2/17  |  2919441    |    7    |      36%   |
Wed Oct 25 03:08:36 2017 |
|   fc2/14  |  2599293    |    6    |      32%   |
Wed Oct 25 03:08:36 2017 |
|   fc2/17  |  2466769    |    6    |      30%   |
Wed Oct 25 03:08:16 2017 |
|   fc2/18  |  2898939    |    7    |      36%   |
Wed Oct 25 03:07:56 2017 |
|   fc2/17  |  2861591    |    7    |      35%   |
Wed Oct 25 03:07:56 2017 |
|   fc2/17  |  2676722    |    6    |      33%   |
Wed Oct 25 03:07:36 2017 |
|   fc2/15  |  2716944    |    6    |      33%   |
Wed Oct 25 03:07:36 2017 |
|    fc2/17   |  2564469     |    6     |        32%    |
Wed Oct 25 03:07:16 2017 |
|    fc2/14   |  2631287     |    6     |        32%    |
Wed Oct 25 03:06:36 2017 |
...
|    fc2/18   |  4107915     |   10     |        51%    |
Wed Oct 25 00:00:55 2017 |
|    fc2/17   |  4347737     |   10     |        54%    |
Wed Oct 25 00:00:55 2017 |
|    fc2/15   |  3392306     |    8     |        42%    |
Wed Oct 25 00:00:55 2017 |
|    fc2/14   |  4711821     |   11     |        58%    |
Wed Oct 25 00:00:55 2017 |
|    fc2/18   |  4992766     |   12     |        62%    |
Wed Oct 25 00:00:35 2017 |
|    fc2/17   |  4743705     |   11     |        59%    |
Wed Oct 25 00:00:35 2017 |
|    fc2/15   |  4505854     |   11     |        56%    |
Wed Oct 25 00:00:35 2017 |
|    fc2/14   |  5441407     |   13     |        68%    |
Wed Oct 25 00:00:35 2017 |
|    fc2/18   |  4173179     |   10     |        52%    |
Wed Oct 25 00:00:15 2017 |
|    fc2/17   |  5624645     |   14     |        70%    |
Wed Oct 25 00:00:15 2017 |
|    fc2/15   |  3510757     |    8     |        43%    |
Wed Oct 25 00:00:15 2017 |
|    fc2/14   |  5516526     |   13     |        68%    |
Wed Oct 25 00:00:15 2017 |
```

Figure 4-34 shows a plot of events shown in Example 4-75. All interfaces show both Level 1.5 (TxWait > 30%) and Level 1 (TxWait < 30%) congestion.

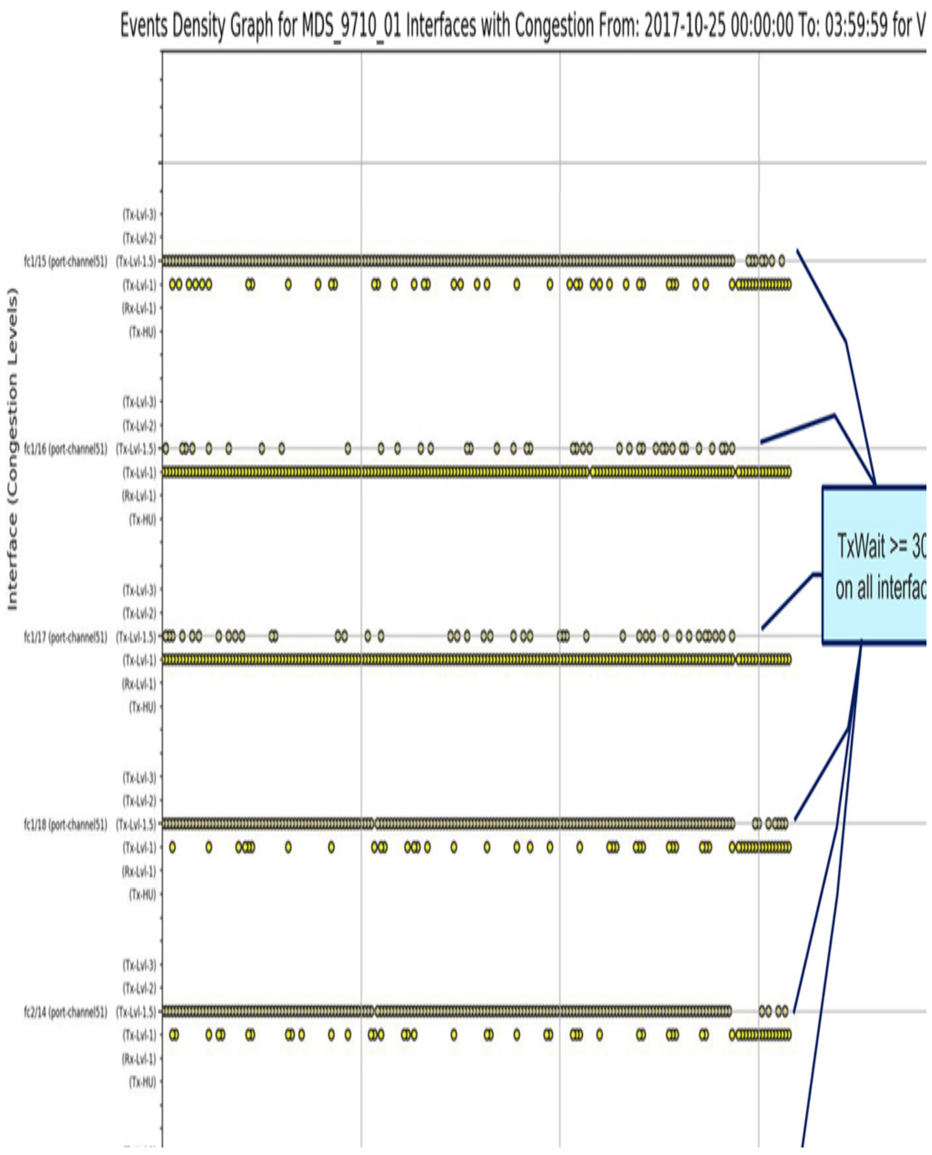


Figure 4-34 *Graph of TxWait on MDS_9710_01*

Interfaces: fc1/14-18 fc2/14-18 were all part of port-chanel51 as evident from **show interface** command (Example 4-76)

Example 4-76 *Members of a port-channel on MDS_9710_01*

```
show interface

port-channel51 is trunking
    Port description is To TXMDSA4
    Hardware is Fibre Channel
    Port WWN is 24:33:00:de:fb:35:3d:80
    Admin port mode is E, trunk mode is on
    snmp link state traps are enabled
    Port mode is TE
    Port vsan is 1001
    Speed is 80 Gbps
    admin fec state is down
    oper fec state is down
    Trunk vsans (admin allowed and active)
(1,3,5,7,9,23,1001)
    Trunk vsans (up)
(1,3,5,7,9,23,1001)
    Trunk vsans (isolated)                ()
    Trunk vsans (initializing)            ()
...
    Member[1] : fc1/14
    Member[2] : fc1/15
    Member[3] : fc1/16
    Member[4] : fc1/17
    Member[5] : fc1/18
    Member[6] : fc2/14
    Member[7] : fc2/15
    Member[8] : fc2/16
    Member[9] : fc2/17
    Member[10] : fc2/18
```

From the **show topology** command, it was found that port-channel51 was connected to MDS_9513_01 (Example 4-77).

Example 4-77 *Neighbor of MDS_9710_01*

```
show topology

FC Topology for VSAN 1 :
-----------------------------------------------------
----------------------------
       Interface  Peer Domain Peer Interface
Peer IP Address(Switch Name)
-----------------------------------------------------
----------------------------
...
   port-channel51   0x07(7)   port-channel51
x.x.x.x(MDS_9513_01)
```

Since port-channel51 was connected to MDS_9513_01, following the methodology, the troubleshooting continued on that switch (Figure 4-35).

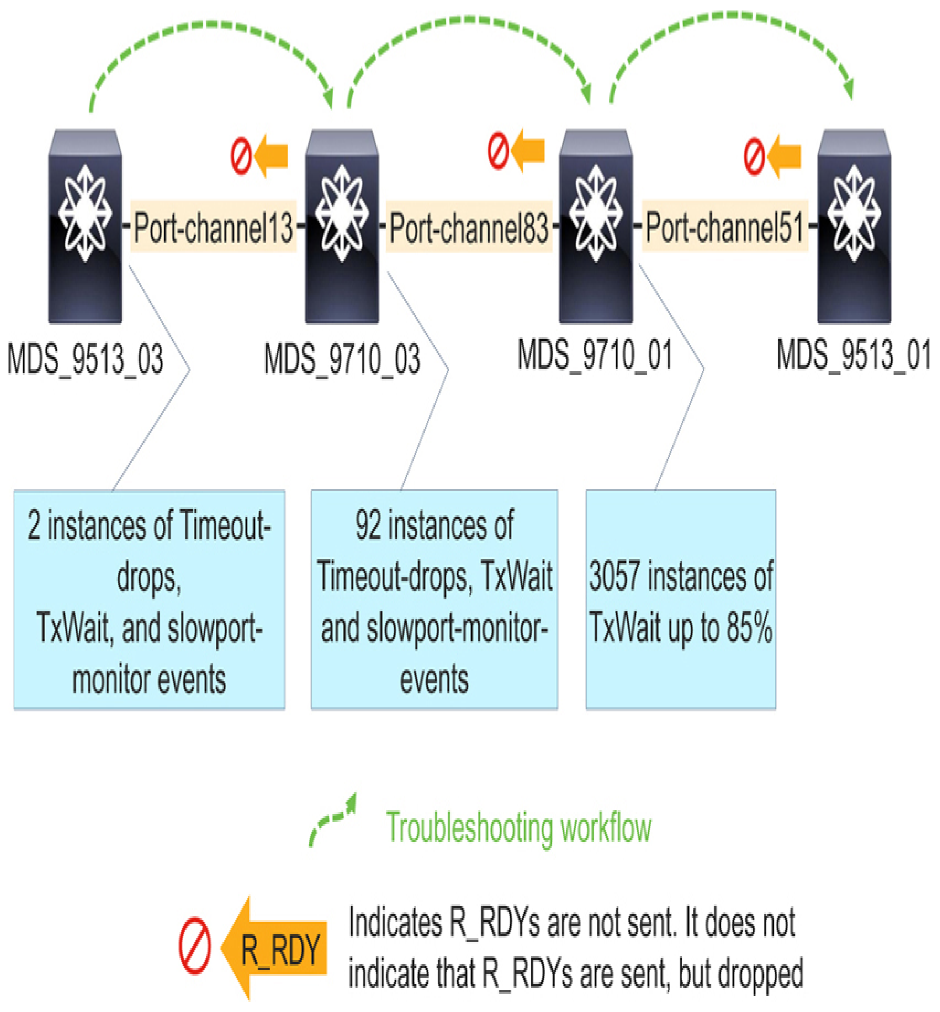

Port-channel13
Port-channel83
Port-channel51
MDS_9513_03
MDS_9710_03
MDS_9710_01
MDS_9513_01
2 instances of Timeout-drops, TxWait, and slowport-monitor events
92 instances of Timeout-drops, TxWait and slowport-monitor events
3057 instances of TxWait up to 85%
Troubleshooting workflow
R_RDY
Indicates R_RDYs are not sent. It does not indicate that R_RDYs are sent, but dropped

Figure 4-35 *Congestion troubleshooting methodology and workflow*

MDS_9513_01

The same methodology of higher to lower severity events was followed even on this switch using the **show logging onboard error-stats** command. On MDS_9513_01 there were three ports with Tx congestion indications as well as Rx congestion indicators on some of the interfaces in port-channel51.

First, Rx congestion events were analyzed followed by Tx congestion events.

Rx Congestion on MDS_9513_01

Other switches in this topology did not have any indications of Rx congestion despite having ingress congestion because the Rx congestion indications on Cisco MDS switches are not as prevalent as Tx congestion indications. As was previously mentioned, the RxWait counter with a granularity of 2.5 microseconds is available only in 64G MDS switches and onwards. The only timestamped Rx congestion indication is Rx-Credit-Not-Available (RX_WT_AVG_B2B_ZERO counter) which indicates continuous 100ms when an interface was at zero remaining-Rx-B2B-credits. Although these events indicate significant ingress congestion, the gap is they may not be seen at all or may only be seen sporadically. This is because they indicate complete continuous blocks of time where the switchport was withholding credits from the adjacent device. If, for example, the switch withheld credits for 99ms out of 100ms (99% ingress congestion) then this indication is not captured. So, it is not unusual for the Rx-Credit-Not-Available indications to not match up with the TxWait congestion indications on the adjacent side of the link.

That was clearly the case here as they did not show a pattern of almost continuous Rx-Credit-Not-Available. Example 4-78 shows Rx-Credit-Not-Available (RX_WT_AVG_B2B_ZERO counter) indications for the interfaces in port-channel51 on MDS_9513_01, which was connected to MDS_9710_01 but still did not match with Example 4-75.

Example 4-78 *Rx Credit Not Available on MDS_9513_01*

```
show logging onboard error-stats

----------------------------
Module: 1 show clock
----------------------------
2017-10-25 03:05:31

---------------------------------
 Module: 1 error-stats
---------------------------------



----------------------------------------------------
--------------------------
 ERROR STATISTICS INFORMATION FOR DEVICE DEVICE:
FCMAC
```

```
----------------------------------------------------
--------------------------
 Interface  |                                      |
|    Time Stamp
   Range    |       Error Stat Counter Name        |
Count  |MM/DD/YY HH:MM:SS
            |                                      |
|
----------------------------------------------------
--------------------------
fc1/3        |FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO
|164      |10/25/17 03:01:06
fc1/3        |FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO
|163      |10/25/17 02:55:26
fc1/5        |FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO
|199      |10/25/17 02:51:46
fc1/3        |FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO
|162      |10/25/17 02:51:46
fc1/4        |FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO
|169      |10/25/17 02:49:46
fc1/3        |FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO
|161      |10/25/17 02:38:06
fc1/4        |FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO
|168      |10/25/17 02:22:46
fc1/2        |FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO
|189      |10/25/17 02:14:25
fc1/3        |FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO
|160      |10/25/17 02:01:26
fc1/1        |FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO
|178      |10/25/17 02:00:06
fc1/8        |FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO
|208      |10/25/17 01:58:46
fc1/5        |FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO
|198      |10/25/17 01:56:26
fc1/1        |FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO
|177      |10/25/17 01:53:06
...
fc1/1        |FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO
|170      |10/25/17 00:19:45
fc1/3        |FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO
|156      |10/25/17 00:17:45
```

```
fc1/3          |FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO
|155        |10/25/17 00:16:05
```

Figure 4-36 shows a plot of events shown in Example 4-78. Even this graph does not match with Figure 4-34.

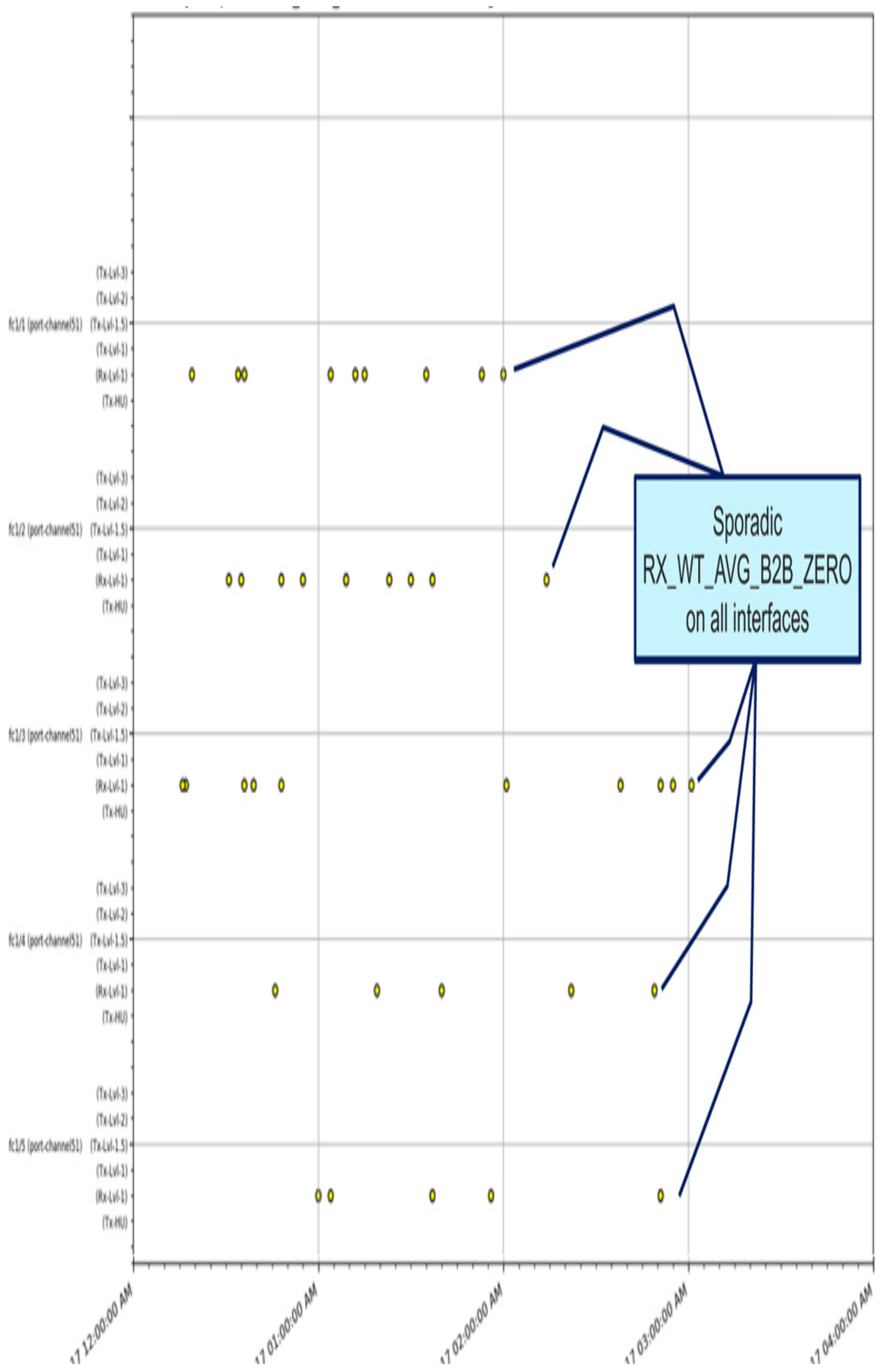


Figure 4-36 *Graph of Rx Credit Not Available on MDS_9513_01*

Next, let’s analyze the Tx congestion events.

Tx Congestion on MDS_9513_01

The following three interfaces showed Tx congestion indications: fc6/2, fc6/27, and fc11/31. We analyzed them in that order.

Interface fc6/2 had just a single instance of a 1ms slowport-monitor event (Example 4-79). This congestion was so minor, it couldn’t be the cause and source of congestion spreading.

Example 4-79 *Slowport-monitor history in OBFL for fc6/2 on MDS_9513_01*

```
show logging onboard slowport-monitor-events

---------------------------------
 Module: 3 slowport-monitor-events
---------------------------------


-----------------------------------------------------
-------------------
| admin   | slowport   | txwait|           Timestamp
| Interface
| delay   | detection | oper   |
|
|  (ms)    | count      | delay |
|
|         |            |   (ms)  |
|
-----------------------------------------------------
-------------------
|   1     |     215084 |    1   | 10/25/17 00:09:45.170
| fc6/2
```

Interface fc6/27 had multiple TxWait indications (Example 4-80) but they were all minor congestion percentages (1%-4%). Also, they didn’t even start until 1:47 AM which means they didn’t match the time when the problem started (around midnight). This couldn’t be the cause and the source of congestion spreading either.

Example 4-80 *TxWait history in OBFL for fc6/27 on MDS_9513_01*

```
show logging onboard txwait

---------------------------------
 Module: 6 txwait
---------------------------------
 Notes:
     - Sampling period is 20 seconds
     - Only txwait delta >= 100 ms are logged

----------------------------------------------------
------------------------
| Interface | Delta TxWait Time     | Congestion |
Timestamp                 |
|           | 2.5us ticks | seconds |            |
|
----------------------------------------------------
------------------------
|   fc6/27  |    87320    |    0    |       1%   |
Wed Oct 25 03:09:14 2017 |
|   fc6/27  |    75549    |    0    |       0%   |
Wed Oct 25 03:08:54 2017 |
|   fc6/27  |   111778    |    0    |       1%   |
Wed Oct 25 03:08:34 2017 |
|   fc6/27  |    64251    |    0    |       0%   |
Wed Oct 25 03:08:14 2017 |
|   fc6/27  |   118043    |    0    |       1%   |
Wed Oct 25 03:07:54 2017 |
|   fc6/27  |    62075    |    0    |       0%   |
Wed Oct 25 03:07:34 2017 |
|   fc6/27  |    82418    |    0    |       1%   |
Wed Oct 25 03:07:14 2017 |
...
|   fc6/27  |    41235    |    0    |       0%   |
Wed Oct 25 01:49:53 2017 |
|   fc6/27  |    68550    |    0    |       0%   |
Wed Oct 25 01:49:33 2017 |
|   fc6/27  |    68865    |    0    |       0%   |
Wed Oct 25 01:48:53 2017 |
|   fc6/27  |    51904    |    0    |       0%   |
Wed Oct 25 01:48:13 2017 |
|   fc6/27   |    47946    |    0    |        0%    |
Wed Oct 25 01:47:53 2017 |
|   fc6/27   |   344090    |    0    |        4%    |
Wed Oct 25 01:47:33 2017 |
|   fc6/27   |   109314    |    0    |        1%    |
Wed Oct 25 01:47:13 2017 |
```

Finally, fc11/31 showed almost continuous high Tx datarate indications right up until the time the **show tech-support slowdrain** command was taken. The model of the MDS_9513_01 switch did not support the **show logging onboard datarate** command, although the high Tx-datarate indications were in the Syslog messages (Example 4-81). It's the same information just in a different non-dedicated location. Remember that the interval of time the port was at a high Tx-datarate starts with the time of the rising-threshold alert and ends at the time of the falling-threshold alert (actually, 10 seconds earlier to be precise if Port-Monitor datarate counter is configured with 10 seconds poll-interval).

Example 4-81 *High Tx-datarate alerts in system messages on MDS_9513_01*

```
2017 Oct 25 00:04:16 MDS_9513_01 %PMON-SLOT11-3-
RISING_THRESHOLD_REACHED: TX
Datarate has reached the rising threshold
(port=fc11/31 [0x151e000],
value=448109400)
2017 Oct 25 01:13:29 MDS_9513_01%PMON-SLOT11-5-
FALLING_THRESHOLD_REACHED: TX
Datarate has reached the falling threshold
(port=fc11/31 [0x151e000],
value=313607272)

2017 Oct 25 01:13:40 MDS_9513_01%PMON-SLOT11-3-
RISING_THRESHOLD_REACHED: TX
Datarate has reached the rising threshold
(port=fc11/31 [0x151e000],
value=447487470)
2017 Oct 25 01:16:54 MDS_9513_01%PMON-SLOT11-5-
FALLING_THRESHOLD_REACHED: TX
```

```
Datarate has reached the falling threshold
(port=fc11/31 [0x151e000],
value=143534627)

2017 Oct 25 01:17:16 MDS_9513_01%PMON-SLOT11-3-
RISING_THRESHOLD_REACHED: TX
Datarate has reached the rising threshold
(port=fc11/31 [0x151e000],
value=447029444)
2017 Oct 25 01:28:23 MDS_9513_01%PMON-SLOT11-5-
FALLING_THRESHOLD_REACHED: TX
Datarate has reached the falling threshold
(port=fc11/31 [0x151e000],
value=114416022)

2017 Oct 25 01:28:44 MDS_9513_01%PMON-SLOT11-3-
RISING_THRESHOLD_REACHED: TX
Datarate has reached the rising threshold
(port=fc11/31 [0x151e000],
value=441156207)
2017 Oct 25 01:38:14 MDS_9513_01%PMON-SLOT11-5-
FALLING_THRESHOLD_REACHED: TX
Datarate has reached the falling threshold
(port=fc11/31 [0x151e000],
value=256458788)

...

2017 Oct 25 02:35:59 MDS_9513_01%PMON-SLOT11-3-
RISING_THRESHOLD_REACHED: TX
Datarate has reached the rising threshold
(port=fc11/31 [0x151e000],
value=440426069)
2017 Oct 25 02:43:32 MDS_9513_01%PMON-SLOT11-5-
FALLING_THRESHOLD_REACHED: TX
Datarate has reached the falling threshold
(port=fc11/31 [0x151e000],
value=145847454)

2017 Oct 25 02:43:53 MDS_9513_01%PMON-SLOT11-3-
RISING_THRESHOLD_REACHED: TX
Datarate has reached the rising threshold
```

```
(port=fc11/31 [0x151e000],
value=437953228)
2017 Oct 25 02:44:47 MDS_9513_01%PMON-SLOT11-5-
FALLING_THRESHOLD_REACHED: TX
Datarate has reached the falling threshold
(port=fc11/31 [0x151e000],
value=251593255)

2017 Oct 25 02:55:01 MDS_9513_01%PMON-SLOT11-3-
RISING_THRESHOLD_REACHED: TX
Datarate has reached the rising threshold
(port=fc11/31 [0x151e000],
value=426934341)
```

Looking at the first rising-threshold/falling-threshold pair shown (Oct 25 00:04:16 to 01:13:29) the duration of the high utilization was approximately 69 minutes and 13 seconds. Other rising-threshold/falling-threshold pairs have different durations, but they end up being almost continuous high utilization.

Notice the last Syslog was a ‘rising-threshold’ message. Since there was no accompanying ‘falling-threshold’ alert, the conclusion was that the high Tx Utilization that started ten seconds before 02:55:01 was continuing. In fact, the **show tech-support slowdrain** command captured the interface, and it showed a high Tx-datarate condition occurring when this command was executed (Example 4-82).

Example 4-82 *Ingress and Egress throughput under* ***show interface*** *command on MDS_9513_01*

```
fc11/31 is up
    Hardware is Fibre Channel, SFP is short wave
laser w/o OFC (SN)
...
    Speed is 4 Gbps
...
    5 minutes input rate 24763680 bits/sec,3095460
bytes/sec, 8549 frames/sec
    5 minutes output rate 3322102016
```

```
bits/sec,415262752 bytes/sec, 206491
frames/sec
```

The 5-minutes average output rate on fc11/31 was 3322102016 bits/sec and it operated at 4 GFC. Although the bit rate of the 4 GFC link is 4.25 Gbps, because of the 8b/10b encoding, 20% is the overhead. So, the max relevant data rate is 80% of 4.25 Gbps, which is 3.4 Gbps. This relationship between bit rate and data rate is explained in Chapter 2, the section on *Difference Between Fibre Channel Speed and Bit Rate*. With 3322102016 bits/sec throughput, the 5-minute average percentage utilization was higher than 95% (3,322,102,016 / 3,400,000,000).

Figure 4-37 shows the plot of congestion events on these three interfaces under investigation on MDS_9513_01.

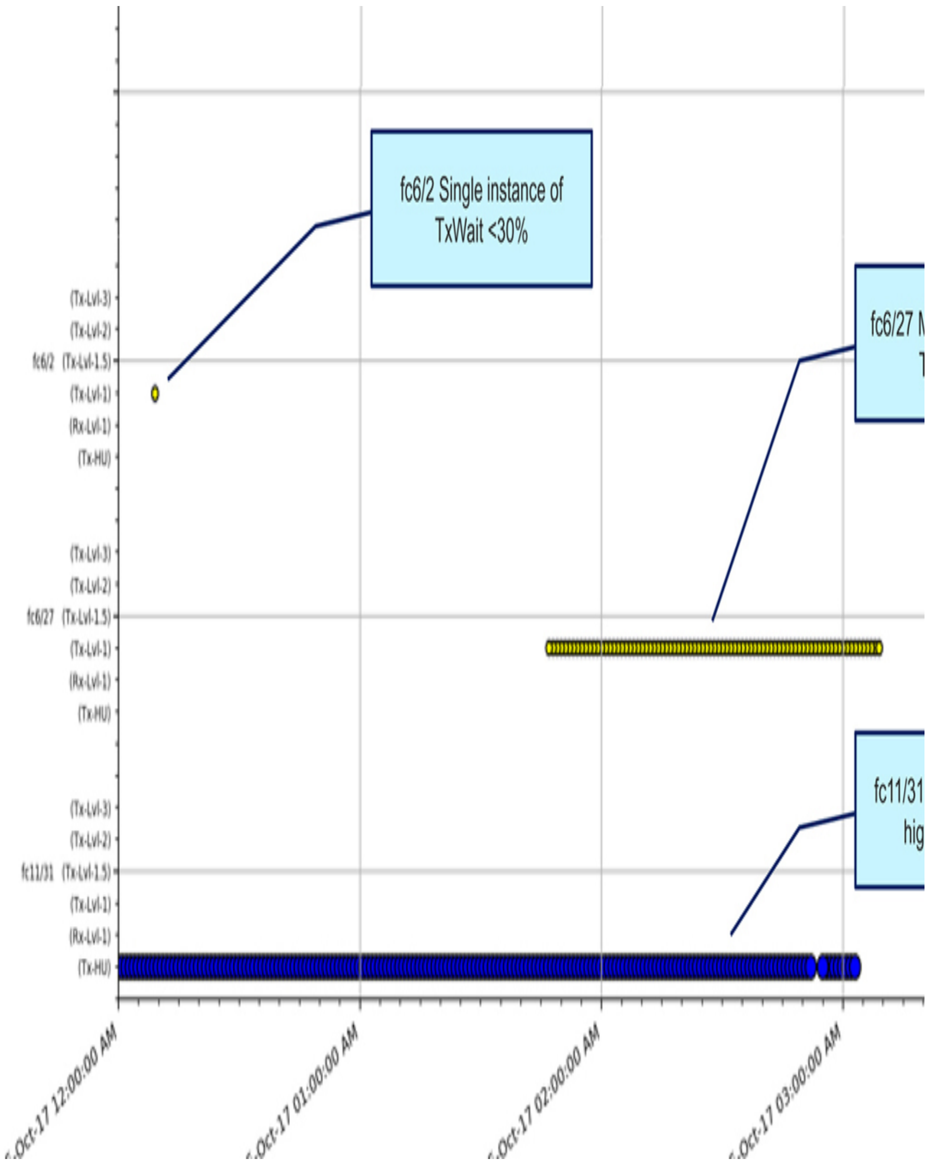


Figure 4-37 *Graph of congestion events on MDS_9513_01*

Culprit Analysis

At this point, the culprit was identified to be the host connected to interface fc11/31 on MDS_9513_01 (Figure 4-38). The time correlation of the almost continuous Tx-datarate indications on fc11/31 on MDS_9513_01 matched the times of the almost continuous high TxWait and timeout-drops on MDS_9513_03, MDS_9710_03, and MDS_9710_01 port-channels going to MDS_9513_01. Only the congestion indications on fc11/31 correlated with the times of the congestion indications on the other three switches.

Victim Analysis

As explained in earlier case studies, victim devices should be evaluated in the following order:

1. Direct victims.
2. Same-path victims.
3. Indirect victims.

Direct Victims

The switchport (fc11/31 on MDS_9513_01) that was connected to the culprit operated at 4 GFC. The active zoneset showed that this server was zoned with four targets—two on MDS_9710_03 and two on MDS_9710_01:

- MDS_9710_03 Interface(s): fc9/8 (8G) and fc10/31 (8G).
- MDS_9710_01 Interface(s): fc9/14 (8G) and fc10/29 (8G).

As Figure 4-38 shows, these are the direct victims.

The direct victims would have their traffic rate reduced by the congestion. From the switch's perspective, this was ingress congestion. The only real-time and date-stamped indications of ingress congestion are the 100ms of continuous zero remaining-Rx-B2B-credits counter RX_WT_AVG_B2B_ZERO. This would affect any of the servers they were communicating with.

Looking at the four target interfaces in order:

1. Switchname MDS_9710_03, interface fc9/8 did not show any indications of RX_WT_AVG_B2B_ZERO in the four-hour timeframe under investigation.
2. Switchname MDS_9710_03, interface fc10/31 showed only a single indication of RX_WT_AVG_B2B_ZERO in the four-hour timeframe under investigation, as shown in Example 4-83.

Example 4-83 *Ingress congestion symptom on direct victim-connected switchport fc10/31 on MDS_9710_03*

```
show logging onboard module 10 error-stats
-----------------------------------------------------
--------------------------
 ERROR STATISTICS INFORMATION FOR DEVICE DEVICE:
FCMAC
-----------------------------------------------------
--------------------------
 Interface  |                                       |
|    Time Stamp
   Range    |       Error Stat Counter Name       |
Count   |MM/DD/YY HH:MM:SS
            |                                       |
|
'''
fc10/31     |FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO       |2
|10/25/17 00:14:25
```

3. As Example 4-84 shows, switchname MDS_9710_01, interface fc9/14 showed three indications of RX_WT_AVG_B2B_ZERO in the four-hour timeframe in question. Each time it incremented only by one. This indicated just 3 instances of a continuous zero remaining-Rx-B2B-credits.

Example 4-84 *Ingress congestion symptom on direct victim-connected switchport fc9/14 on MDS_9710_01*

```
show logging onboard module 9 error-stats
-----------------------------------------------------
--------------------------
 ERROR STATISTICS INFORMATION FOR DEVICE DEVICE:
FCMAC
-----------------------------------------------------
--------------------------
 Interface  |                                       |
|    Time Stamp
   Range    |       Error Stat Counter Name       |
Count   |MM/DD/YY HH:MM:SS
            |                                       |
|
-----------------------------------------------------
```

```
--------------------------
...
fc9/14      |FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO
|21        |10/25/17 02:56:32
fc9/14      |FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO
|20        |10/25/17 01:05:10
fc9/14      |FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO
|19        |10/25/17 00:38:49
```

4. As Example 4-85 shows, switchname MDS_9710_01, interface fc10/29 showed two indications of RX_WT_AVG_B2B_ZERO in the four-hour timeframe in question. Each time it incremented only by one. This indicated just 2 instances of a continuous zero remaining-Rx-B2B-credits.

Example 4-85 *Ingress congestion symptom on direct victim-connected switchport fc10/29 on MDS_9710_01*

```
show logging onboard module 10 error-stats
----------------------------------------------------
--------------------------
 ERROR STATISTICS INFORMATION FOR DEVICE DEVICE:
FCMAC
----------------------------------------------------
--------------------------
 Interface  |                                      |
|    Time Stamp
   Range    |        Error Stat Counter Name       |
Count   |MM/DD/YY HH:MM:SS
            |                                      |
|
----------------------------------------------------
--------------------------
...
fc10/29     |FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO
|18        |10/25/17 03:00:39
fc10/29     |FCP_SW_CNTR_RX_WT_AVG_B2B_ZERO
|17        |10/25/17 02:55:39
```

So, as can be seen by these displays for the direct victims, there were not a significant number of the

RX_WT_AVG_B2B_ZERO entries to show a strong correlation to ingress congestion. This was almost certainly due to the limitations of the RX_WT_AVG_B2B_ZERO counter itself. Consequently, all four targets should be considered direct victims in this case study.

Same-Path Victims

All devices communicating along the path would be same-path victims. Since so many switches were involved the number of same-path victims would be huge.

This also explains that congestion seen on MDS_9513_03 even when fc11/31 on MDS_9513_01 wasn't communicating with any interfaces on that switch. Most likely there were devices on MDS_9513_03 that were communicating with devices on MDS_9710_03, MDS_9710_01 and/or MDS_9513_03. Because the path between those switches was congested, other devices utilizing the same path would be affected as well, making them the same-path victims.

Indirect Victims

The four direct victims (the four targets) were zoned with a total of eight other servers besides the culprit.

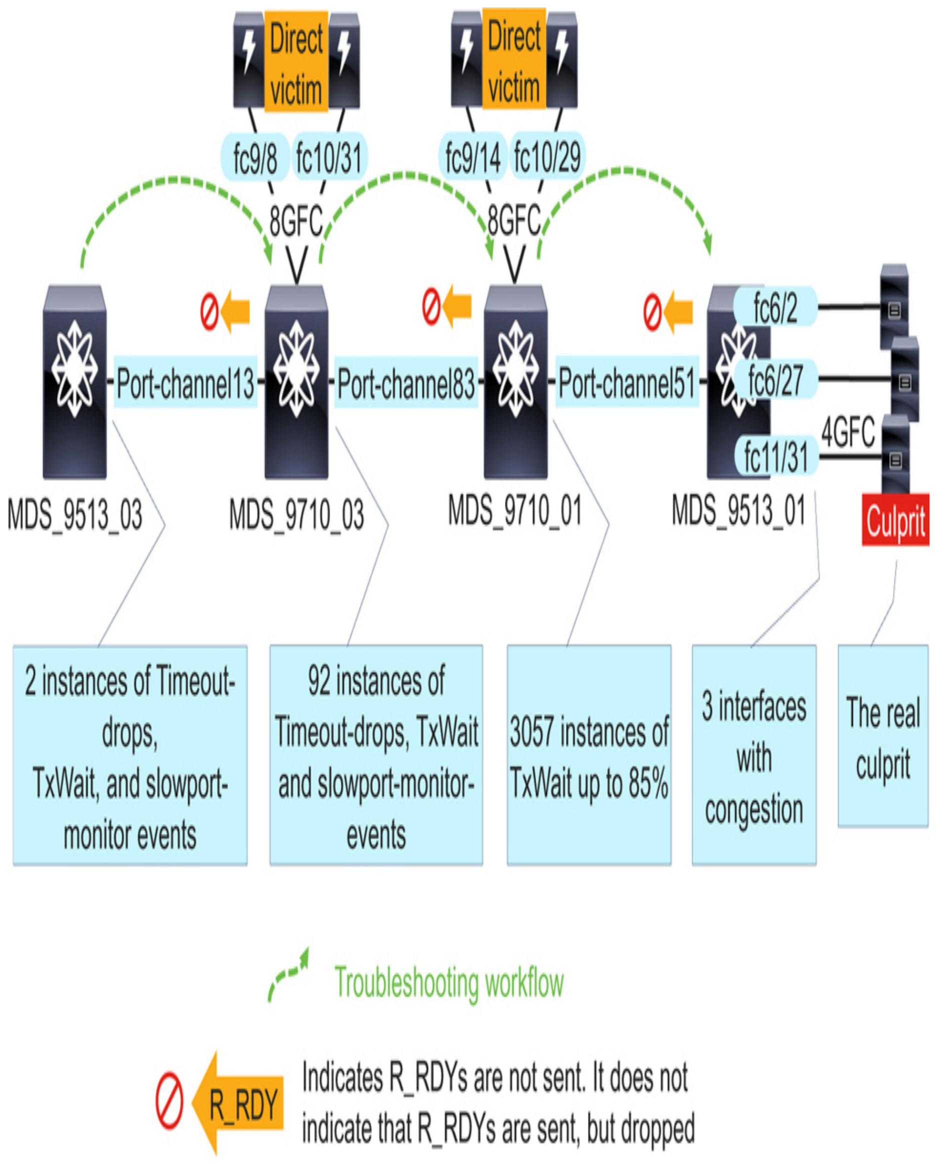
- Four were on MDS_9513_01 (including the culprit on fc11/31)
- Two were on MDS_9513_02
- Three were on MDS_9513_03

Likewise, the end devices communicating with same-path victims could also be found using zoning database as explained in Case Study 1 and 2.

Indirect victims typically do not have any indications of ingress or egress congestion on their connected switchports, although their I/O performance gets degraded.

Case study 3—Summary

Figure 4-38 summarizes the workflow and troubleshooting methodology. Starting with the congestion indications on MDS_9513_03, the real culprit is the device that connects to fc11/31 on MDS_9513_01. This was the source of congestion.



Direct victim
Direct victim
fc9/8
fc10/31
fc9/14
fc10/29
8GFC
8GFC
Port-channel13
Port-channel83
Port-channel51
fc6/2
fc6/27
fc11/31
4GFC
Culprit
MDS_9513_03
MDS_9710_03
MDS_9710_01
MDS_9513_01
2 instances of Timeout-drops, TxWait, and slowport-monitor events
92 instances of Timeout-drops, TxWait and slowport-monitor-events
3057 instances of TxWait up to 85%
3 interfaces with congestion
The real culprit
Troubleshooting workflow
R_RDY
Indicates R_RDYs are not sent. It does not indicate that R_RDYs are sent, but dropped

Figure 4-38 *Finding the culprit and victims using the congestion troubleshooting methodology and workflow*

The cause of congestion was due to over-utilization of fc11/31 because the connected host was requesting more data via Read I/O operations than the switch could transmit. Only Read operations can generate a large amount of data going to the host. Interface fc11/31 was operating at 4 GFC and it was zoned with 4 x 8 GFC target interfaces. That means if there were concurrent Read I/O operations to all 4 targets, there was a potential for 8 times (total 32G vs 4G) more data arriving from the targets than the 4 GFC server link could transmit and receive. This could easily cause an over-utilization condition.

The solution would be to increase the HBA speed or number of HBAs, so the amount of data being requested by the server can actually be delivered to it by the switchport(s). Other congestion prevention mechanisms explained in Chapter 6 can also be used.

This case study illustrates the following:

1. Victims can be anywhere in the fabric. In this scenario, the victims of various types would be very numerous.

2. It is important to pay attention to the alerts. Although the switches generated the alerts, the users did not pay attention to them until a detailed investigation.

3. Production fabrics typically always have some indications of congestion. It’s not practically feasible, and even worthwhile, to completely eliminate congestion. The real goal should be to lower the severity so that it doesn’t cause application degradation. In Figure 4-38, fc6/27 and fc6/2 on MDS_9513_01 have some indications of congestion but fc11/31 is the only interface that victimizes other devices.

4. As the earlier section on Timeout Drop Anomaly explains, in a multi-switch environment, the instances and amounts of timeout-drops reduce on the switchports that are closer to the culprit device in the data path. As Figure 4-38 shows, MDS_9710_03 reported timeout drops but MDS_9710_01 and MDS_9513_01 did not report timeout drops, although they reported TxWait. MDS_9513_03 reported just two instances of timeout drops which was less than MDS_9710_03 but remember the direct victims were not on MDS_9513_03 so those drops were not drops to the culprit host. This was some other traffic affected by the congestion spreading along the path.

Case Study 4— Long Distance ISLs Causing congestion

An electric power transmission company reported slow data transfer over a three-switch MDS 9148S Fibre Channel fabric (Figure 4-39). The two initiators (Server 1 and Server 2) communicated with two tape libraries with multiple LTO7 drives. No specific dates/times were described as it was a problem that was reported to happen “all the time”.

The admins of this environment gathered **show tech-support slowdrain** outputs for all the switches in the fabric. This gave a complete view of all the switches in the fabric.

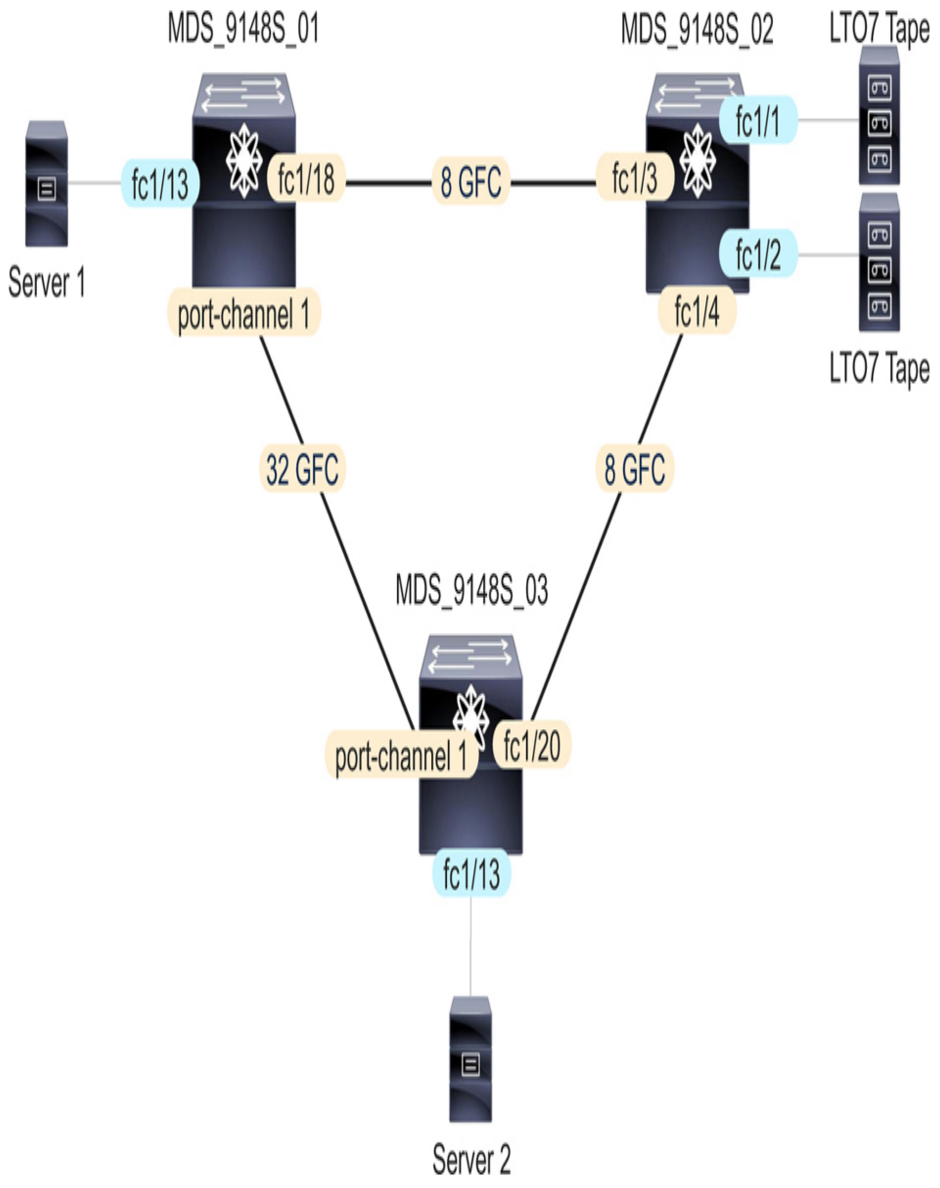


Figure 4-39 *Three-switch topology with servers and tape libraries*

Since the problem description was not very detailed, the troubleshooting started at the highest severity of congestion (Level 3) followed by the lower levels (Level 2, 1.5, and 1).

Level 3

The **show logging onboard error-stats** command across all three switches did not show any instances of CREDIT_LOSS.

Level 2

The **show logging onboard error-stats** command across all three switches did not show any instances of timeout-drops.

Level 1.5

The **show logging onboard txwait** command on MDS_9148S_01 and MDS_9148S_03 switches shows TxWait > 30%. MDS_9148S_02 did not show high TxWait.

MDS_9148S_01

MDS_9148S_01 was the first switch that was checked. Server 1 was connected to this switch on interface fc1/13. Checking **show logging onboard txwai**t, it was found that interface fc1/18 had consistently high TxWait of 37% to 39% (Example 4-86). This is classified as Level 1.5 congestion. The entries started at 11:55 on Jan 19, 2022 and ended at 13:03 the same day. This is the exact time the **show tech-support slowdrain** was issued so it was undetermined how long the high TxWait condition would have continued. The OBFL TxWait log file only went back to 21:05:18 on January 18th (less than one day) due to lots of low-level TxWait on other interfaces. Because of this reason, it was not possible to determine any kind of daily historical pattern.

Example 4-86 *TxWait history in OBFL*

```
show logging onboard txwait
----------------------------
Module: 1 show clock
----------------------------
2022-01-19 13:03:49

---------------------------------
Module: 1 txwait
---------------------------------
Notes:
     - Sampling period is 20 seconds
     - Only txwait delta >= 100 ms are logged

----------------------------------------------------
-------------------------
| Interface | Delta TxWait Time     | Congestion |
Timestamp                 |
|           | 2.5us ticks | seconds |            |
|
----------------------------------------------------
-------------------------
|   fc1/18  |  2992094    |    7    |      37%   |
Wed Jan 19 13:03:44 2022 |
|   fc1/18  |  2981426    |    7    |      37%   |
Wed Jan 19 13:03:24 2022 |
|   fc1/18  |  2893653    |    7    |      36%   |
Wed Jan 19 13:03:04 2022 |
|   fc1/18  |  3026084    |    7    |      37%   |
Wed Jan 19 13:02:44 2022 |
|   fc1/18  |  3002741    |    7    |      37%   |
Wed Jan 19 13:02:24 2022 |
|   fc1/18  |  3002238    |    7    |      37%   |
Wed Jan 19 13:02:04 2022 |
...
|   fc1/18  |  3121981    |    7    |      39%   |
Wed Jan 19 11:57:02 2022 |
|   fc1/18  |  3135681    |    7    |      39%   |
Wed Jan 19 11:56:42 2022 |
|   fc1/18  |  3129080    |    7    |      39%   |
Wed Jan 19 11:56:22 2022 |
|   fc1/18  |  2553667    |    6    |      31%   |
Wed Jan 19 11:56:02 2022 |
|    fc1/18    |   3007817     |     7     |       37%    |
Wed Jan 19 11:55:42 2022 |
```

Figure 4-40 shows a plot of the events shown in Example 4-86.

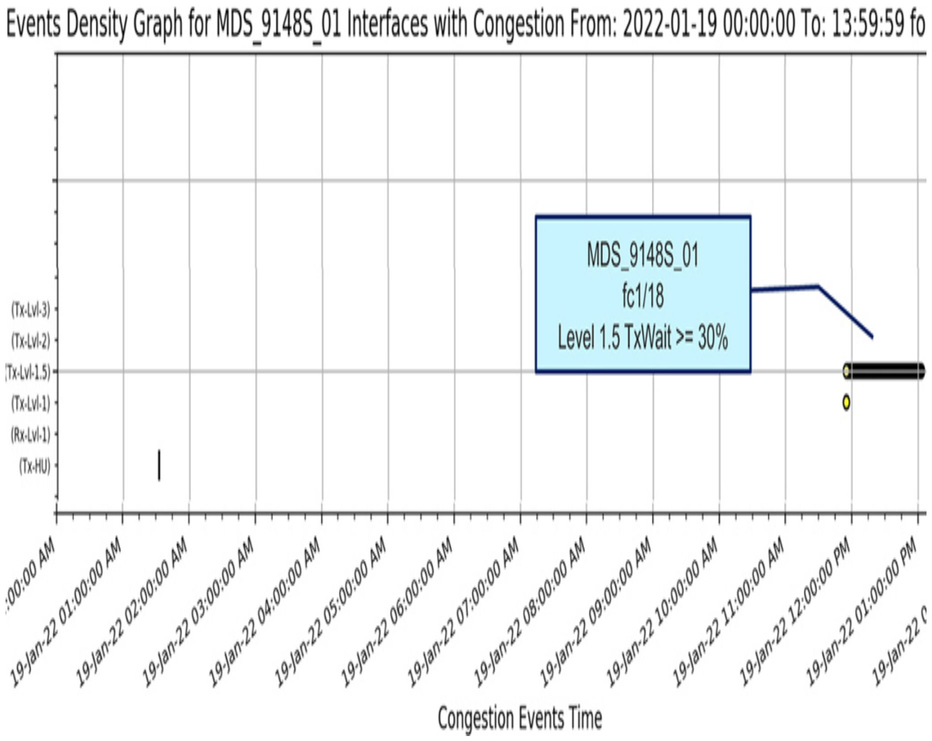


Figure 4-40 *Graph of congestion events on MDS_9148S_01*

Example 4-87 shows that fc1/18 on MDS_9148S_01 was connected to fc1/3 on MDS_9148S_02.

Example 4-87 *Neighbor of MDS_9148S_01*

```
show topology

FC Topology for VSAN 102 :
----------------------------------------------------
---------------------------
       Interface  Peer Domain Peer Interface
Peer IP Address(Switch Name)
----------------------------------------------------
---------------------------
           fc1/18  0x18(24)           fc1/3
x.x.x.x(MDS_9148S_02)
```

```
    port-channel1 0x83(131)    port-channel1
x.x.x.x(MDS_9148S_03)
```

Since fc1/18 on MDS MDS_9148S_01 was connected to MDS_9149S_02, as per the methodology, the troubleshooting continued on MDS_9148S_02 (Figure 4-41).

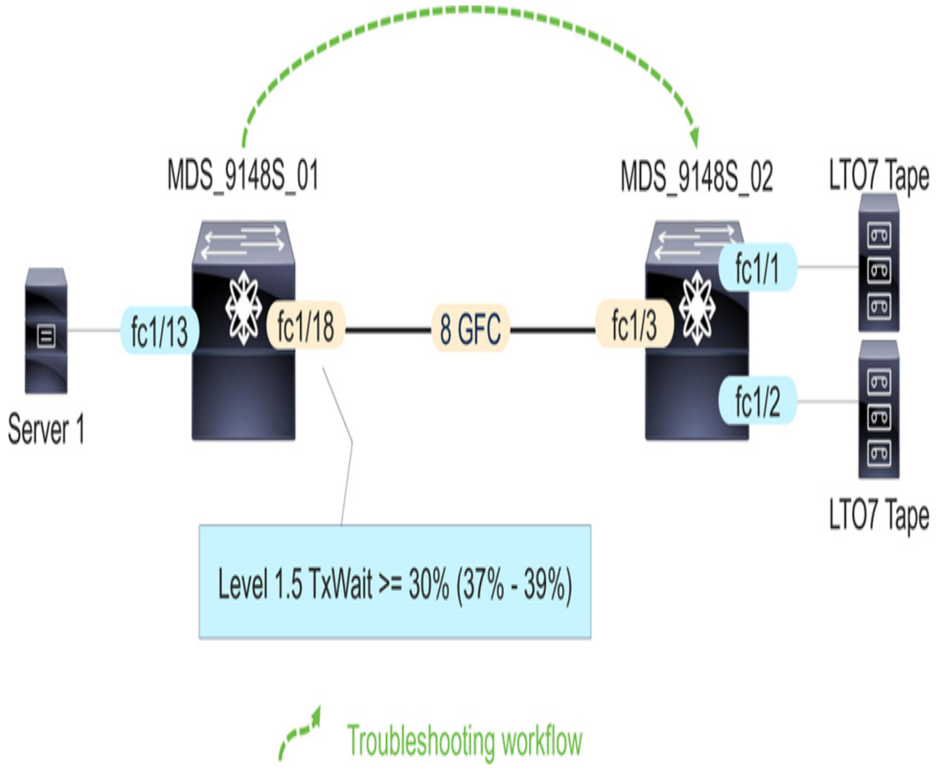


Figure 4-41 *Congestion troubleshooting methodology and workflow*

MDS_9148S_02

MDS_9148S_02 did not have any indications of TxWait or high utilization. This was puzzling at first. Additionally, there were no instances of the Rx-Credit-Not-Available indications (counter xxx_RX_WT_AVG_B2B_ZERO) in **show logging onboard error-stats**. So, even though there was TxWait >= 30% (Level 1.5) indications on the adjacent switch (MDS_9148S_01), there must not have been continuous 100 ms periods of zero remaining-Rx-B2B-credits because counter xxx_RX_WT_AVG_B2B_ZERO has a minimum granularity of 100ms on the Cisco MDS switches.

Since there was another switch to check, troubleshooting continued to MDS_9148S_03.

MDS_9148S_03

Server 2 was connected to MDS_9148S_03 on interface fc1/13. Checking **show logging onboard txwait** it was found that interface fc1/20 had consistently high TxWait between 22% to 39%, but most entries were between 37% to 39% (Example 4-88). This is classified as Level 1.5 congestion. The entries started at 11:53 on Jan 19th and ended at 13:04 the same day. This was the exact time the **show tech-support slowdrain** was taken for this switch so it is undetermined how long the high TxWait condition would have continued. The OBFL TxWait log file only went a bit further to 02:26:12 on January 17th (approximately 2 1/2 days). The TxWait started suddenly on January 19th and no historical pattern was noted.

Example 4-88 *TxWait history in OBFL*

```
show logging onboard txwait
----------------------------
Module: 1 show clock
----------------------------
2022-01-19 13:04:28

---------------------------------
Module: 1 txwait
---------------------------------
Notes:
     - Sampling period is 20 seconds
     - Only txwait delta >= 100 ms are logged

----------------------------------------------------
-------------------------
| Interface | Delta TxWait Time     | Congestion |
Timestamp                 |
|           | 2.5us ticks | seconds |            |
|
|   fc1/20  |  3022813    |    7    |      37%   |
Wed Jan 19 13:04:12 2022 |
|   fc1/20  |  3051467    |    7    |      38%   |
Wed Jan 19 13:03:52 2022 |
|   fc1/20  |  2875843    |    7    |      35%   |
Wed Jan 19 13:03:32 2022 |
|    fc1/20    |   3090374     |     7     |        38%     |
Wed Jan 19 13:03:12 2022 |
|    fc1/20    |   3136709     |     7     |        39%     |
Wed Jan 19 13:02:52 2022 |
|    fc1/20    |   3142457     |     7     |        39%     |
Wed Jan 19 13:02:32 2022 |
...
|    fc1/20    |   3009680     |     7     |        37%     |
Wed Jan 19 11:54:51 2022 |
|    fc1/20    |   3013953     |     7     |        37%     |
Wed Jan 19 11:54:31 2022 |
|    fc1/20    |   3019068     |     7     |        37%     |
Wed Jan 19 11:54:11 2022 |
|    fc1/20    |   2918146     |     7     |        36%     |
Wed Jan 19 11:53:51 2022 |
|    fc1/20    |   2915259     |     7     |        36%     |
Wed Jan 19 11:53:31 2022 |
|    fc1/20    |   1806037     |     4     |        22%     |
Wed Jan 19 11:53:11 2022 |
```

Figure 4-42 shows a plot of events shown in Example 4-88.

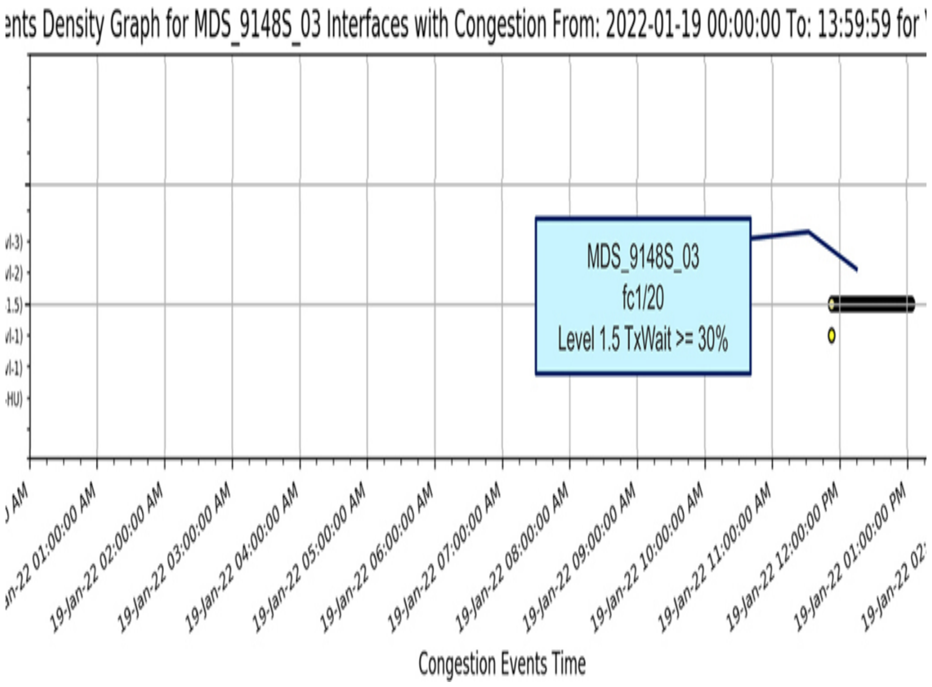


Figure 4-42 *Graph of congestion events on MDS_9148S_03*

Example 4-89 shows that fc1/20 on MDS_9148S_03 was connected to fc1/4 on MDS_9148S_02.

Example 4-89 *Neighbor of MDS_9148S_03*

```
show topology

FC Topology for VSAN 102 :
-----------------------------------------------------
---------------------------
       Interface  Peer Domain Peer Interface
Peer IP Address(Switch Name)
-----------------------------------------------------
--------------------------
           fc1/20  0x18(24)           fc1/4
x.x.x.x(MDS_9148S_02)
    port-channel1  0x32(50)    port-channel1
x.x.x.x(MDS_9148S_01)
```

At this point, Level 1.5 congestion was found on both MDS_9148S_01 and MDS_9148S_03 pointing to a problem at MDS_9148S_2 where the two LTO7 tape drives were connected. However, there were absolutely no congestion indications recorded in MDS_9148S_02. The situation looks like Figure 4-43.

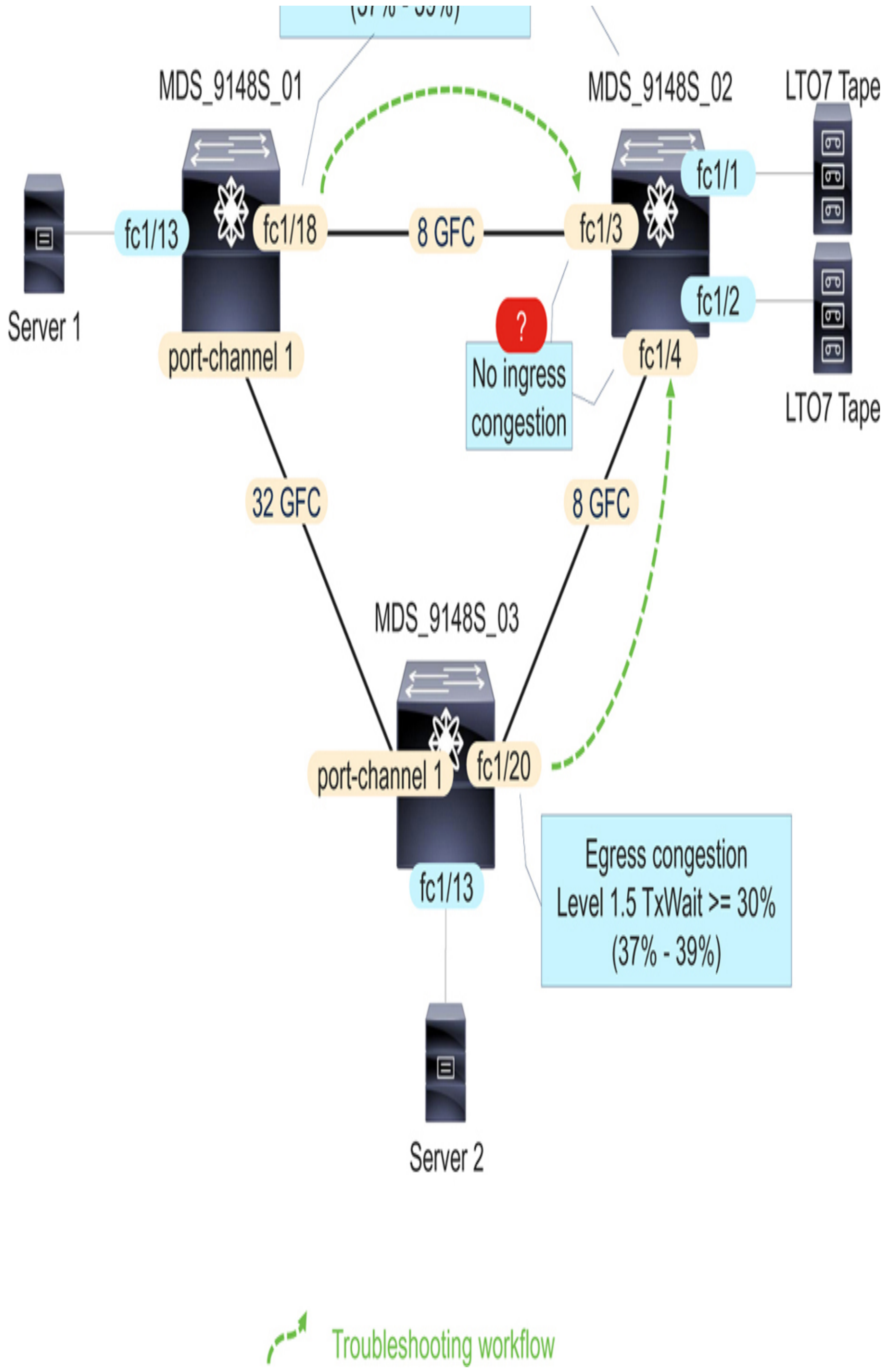


Figure 4-43 *Troubleshooting congestion with no obvious indications*

At this point, we ruled out the two main reasons for congestion in MDS_9148S_02, namely, slow drain and over-utilization. However, there was one other thing we could check on the two ISLs. On the ISL between MDS_9148S_01 and MDS_9148S_02, on MDS_9148S_01, there were over 42 billion ‘Transmit B2B credit transitions to zero’ on fc1/18

(Example 4-90), but on the adjacent interface on MDS_9148S_02, there were just ‘14 Receive B2B credit transitions to zero’ (Example 4-91). The high number of ‘Transmit B2B credit transitions to zero’ seems expected since there was such a high amount of TxWait on this interface. The low number of receive credit transition to zero indicated that MDS_9148S_02 did not withhold B2B credits from MDS_9148S_01. In other words, MDS_9148S_02 wasn’t experiencing any Tx congestion on other ports where the traffic received from fc1/3 was sent, and therefore it did not withhold Rx B2B credits to reduce the rate of ingress traffic from MDS_9148S_01.

Example 4-90 *B2B credit transitions to zero on MDS_9148S_01*

```
show interface counters
fc1/18
...
     42871190441 Transmit B2B credit transitions to
zero
     1049830 Receive B2B credit transitions to zero
      309974175814 2.5us TxWait due to lack of
transmit credits
    Percentage TxWait not available for last
1s/1m/1h/72h: 37%/37%/32%/0%
...
    Last clearing of "show interface" counters:
never
```

Example 4-91 *B2B credit transitions to zero on MDS_9148S_02*

```
show interface counters
fc1/3
...
     6394 Transmit B2B credit transitions to zero
     14 Receive B2B credit transitions to zero
      43552 2.5us TxWait due to lack of transmit
credits
    Percentage TxWait not available for last
```

```
1s/1m/1h/72h: 0%/0%/0%/0%
...
    Last clearing of "show interface" counters:
never
```

Next, we investigated the ISL between MDS_9148S_03 and MDS_9148S_02. On MDS_9148S_03 there were over 41 billion ‘Transmit B2B credit transitions to zero’ (Example 4-92), but on the adjacent interface on MDS_9148S_02 there were just ‘13 Receive B2B credit transitions to zero’ (Example 4-93). The low number of receive credit transitions to zero indicated that MDS_9148S_02 did not withhold B2B credits from MDS_9148S_03, just like it did not withhold B2B credits from MDS_9148S_01.

Example 4-92 *B2B credit transitions to zero on MDS_9148S_03*

```
show interface counters
fc1/20
...
     41123552317 Transmit B2B credit transitions to
zero
     23528564 Receive B2B credit transitions to zero
      335244600329 2.5us TxWait due to lack of
transmit credits
    Percentage TxWait not available for last
1s/1m/1h/72h: 38%/38%/37%/0%
...
    Last clearing of "show interface" counters:
never
```

Example 4-93 *B2B credit transitions to zero on MDS_9148S_02*

```
show interface counters
fc1/4
...
     6170 Transmit B2B credit transitions to zero
     13 Receive B2B credit transitions to zero
      47400 2.5us TxWait due to lack of transmit
credits
```

```
    Percentage TxWait not available for last
1s/1m/1h/72h: 0%/0%/0%/0%
...
    Last clearing of "show interface" counters:
never
```

As Chapter 2 explains, the common reasons for ISLs to experience congestion in a Fibre Channel fabric are the following:

1. The existence of congestion due to an edge device, such as slow-drain or over-utilization.
2. Over-utilization of an ISL.
3. Bit errors on ISL.
4. Lack of B2B credits for the distance, frame size, and speed of an ISL.

In this case, there were no indications for #1, #2, and #3. However, #4 was yet to be investigated.

There are three main ways of transporting FC traffic over long distances:

1. FCIP
2. Long Wave/Extended reach SFPs across the dedicated fiber.
3. Dense wavelength-division multiplexing (DWDM)

There is no FCIP here so we can rule out #1.

The SFPs were Short Wave (multimode fiber) (verified using the **show interface transceiver details** command), which only go short distances so #2 can be ruled out as well.

The use of DWDM links (#3) can't be checked from the switch side. Typically, Fibre Channel switches connect to the DWDM equipment using Short Wave SFPs (but they don't have to). Then the local DWDM equipment transports traffic to the far-side DWDM equipment, which sends traffic on the SW SFP connection to the distant Fibre Channel switch. The long-distance connection is almost entirely transparent to the switches except for the additional delay.

At this point, the admins of this environment were asked about the possibility of these being long-distance DWDM connections. They did not readily know the answer to this question because they were new in their roles. Later, they were able to find that these DWDM connections had the following distances.

- Fiber MDS_9148S_01 <-> MDS_9148S_02: 60km.
- Fiber MDS_9148S_03 <-> MDS_9148S_02: 53 km.

So, there were two sites involved here with distances between 60 km and 53 km. MDS_9148S_01 and MDS_9148S_03 are at one site and MDS_9148S_02 is at another site. The different distances in the long-distance ISLs were because of the different geographical paths taken by the physical cables. Figure 4-44 shows the topology including these distances.

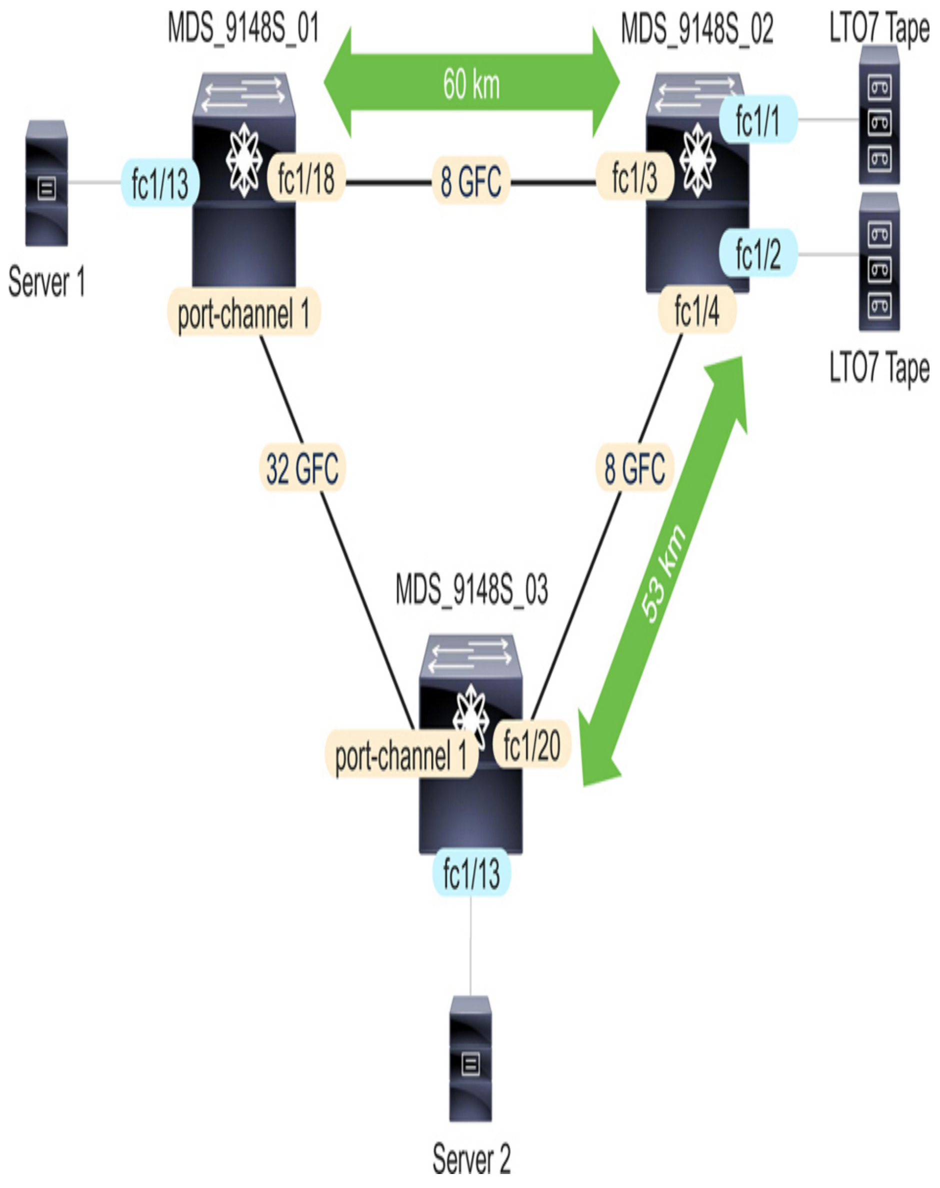


Figure 4-44 *Topology with long-distance native FC links over DWDM*

As Chapter 2 explains, the required B2B credits on a link depend on the following factors:

1. Link Speed: This was known: 8 GFC.
2. Link Distances: These were known as 60 km and 53 km.
3. Average frame size in each direction. This was determined next.

From MDS_9148S_01, the average frame sizes could be calculated from the **show interface** command (Example 4-94).

Example 4-94 *Calculating average frame size from* ***show interface*** *command on MDS_9148S_01*

```
show interface

fc1/18 is up
    Port description is Link to MDS_9148S_02
    Hardware is Fibre Channel, SFP is short wave
laser w/o OFC (SN)
...
    Operating Speed is 8 Gbps
...
    Transmit B2B Credit is 64
    Receive B2B Credit is 64
...
      116274935916 frames input,233832433383848
bytes
...
      90952411260 frames output,178262775692496
bytes
```

Average input frame size = 233832433383848 bytes / 116274935916 frames = 2011 bytes/frame

Average output frame size = 178262775692496 bytes / 90952411260 frames = 1960 bytes/frame

Since the average output frame size was smaller than the average input frame size, the smaller value was used because it would require a larger number of B2B credits.

This ISL was 60 km long. An 8 GFC link needs a minimum of 4 B2B credits per km for full-sized FC frames (Refer to Chapter 2, the section on *Required Number of B2B Credits on an FC Port*). For 60 km, the link needs a minimum of 240 B2B credits (4 x 60 = 240). Since the average output frame size was only about 91% of the full-size FC frame size, it resulted in a multiplier of 1.1 (2148 bytes / 1960 bytes = 1.1) for calculating the number of B2B credits. Ultimately, it led to 264 (240 x 1.1 = 264) minimum required B2B credits to allow the ISL to perform at its maximum capacity.

This value was the absolute minimum required to allow the link to operate at its full capacity. It didn't include any additional buffers for minor congestion, switching latency, or even times when the average frame might temporarily be smaller than the overall average. Hence, we could round up a bit and get to approximately a minimum of 300 B2B credits.

Example 4-94 shows that transmit and receive B2B credits on fc1/18 on MDS_9148S_01 were 64, which was the default value on these switches (Cisco MDS 9148S). This was clearly insufficient for a 60 km link operating at 8 GFC for transporting an average frame size of 1960 bytes. The B2B credits must be increased if performance was to be improved.

Similarly, fc1/20 on MDS_9148S_03 needed its minimum B2B credits calculated. Example 4-95 shows the output of the **show interface** command for fc1/20.

Example 4-95 *Calculating average frame size from* ***show interface*** *command on MDS_9148S_03*

```
show interface

fc1/20 is up
    Port description is Link to MDS_9148S_02
...
    Operating Speed is 8 Gbps
...

    Transmit B2B Credit is 64
    Receive B2B Credit is 64
...
       10156973193 frames input,16881692260304 bytes
...
      85156522487 frames output,173513471956560
bytes
```

Average input frame size = 16881692260304 bytes / 10156973193 frames = 1662 bytes/frame

Average output frame size = 173513471956560 bytes / 85156522487 frames = 2037 bytes/frame

Since the average input frame size was smaller than the average output frame size, the smaller value was used because it would require a larger number of B2B credits.

As mentioned, an 8 GFC link needs a minimum of 4 B2B credits per km (one way) for full-sized FC frames. For 53 km, the link needed a minimum of 212 B2B credits (4 x 53 = 212). Since the average output frame size was only about 77% of the full-size FC frame size, it resulted in a multiplier of 1.29 (2148 bytes / 1662 bytes = 1.29) for calculating the number of B2B credits. Ultimately, it led to 273 minimum required B2B credits (240 x 1.29 = 273).

We could round up a bit and get to approximately a minimum 300 B2B credits to accommodate minor congestion and a smaller frame size, as explained earlier.

Example 4-95 shows that fc1/20 on MDS_9148S_03 had 64 transmit and receive B2B credits. Again, this count was clearly insufficient for 8 GFC links that was 53 km long for transporting an average frame size of 1662 bytes. The B2B credits must be increased if performance was to be improved.

Culprit Analysis

This case study is somewhat unusual because there was no specific culprit device. The sources of congestion are the ISLs between MDS_9148S_01 and MDS_9148S_02, and MDS_9148S_03 and MDS_9148S_02.

Victim Analysis

Just because there was no culprit device, it doesn't mean there weren't any victims. There were no direct victims but all the end devices, both hosts and targets, utilizing the two congested ISLs would be same-path victims. These same-patch victims might have led to many indirect victims.

Note that the TxWait was occurring only on MDS_9148S_01 and MDS_9148S_03, which indicated that traffic was primarily flowing from those two switches to MDS_9148S_02. Looking into the active zoning database would show end devices that were allowed to communicate across the ISL. This was done by comparing the domain-ids within the VSAN. Any end device in a zone with an FCID containing a domain on MDS_9148S_01 or MDS_9148S_03 that also contains an end device with an FCID containing a domain on MDS_9148S_02 would be a potential same-path victim.

In this case, there was just one interface on MDS_9148S_01, fc1/13, and one device on MDS_9148S_03, fc1/22, zoned with any end device on MDS_9148S_02. These would be the same-path victims. For these two interfaces, the only indication that might be expected in OBFL is the Rx-credit-not-available (100ms continuous) counter in OBFL error-stats. There were no indications of those. Another place to look was in **show interface counters**, as shown in Example 4-96 and Example 4-97.

Example 4-96 *Ingress congestion indications using show interface on MDS_9148S_01*

```
show interface counters

fc1/13
    5 minutes input rate 632419712 bits/sec,
79052464 bytes/sec, 38482 frames/sec
    5 minutes output rate 51441408 bits/sec, 6430176
bytes/sec, 3807 frames/sec
...
     1411979644 Transmit B2B credit transitions to
zero
     9729898792 Receive B2B credit transitions to
zero
```

Example 4-97 *Ingress congestion indications using show interface on MDS_9148S_03*

```
show interface counters

fc1/22
  5 minutes input rate 233262624 bits/sec, 29157828
bytes/sec, 14541 frames/sec
  5 minutes output rate 76302272 bits/sec, 9537784
bytes/sec, 5578 frames/sec
...
```

```
28113 Transmit B2B credit transitions to zero
438334747 Receive B2B credit transitions to zero
```

Example 4-96 and Example 4-97 show a large number of Receive B2B credit transitions to zero on both interfaces. This indicated ingress congestion being applied by the switchport but since there were no instances of Rx-credit-not-available, the duration must have been less than 100ms each time (Figure 4-45). This would make sense in this situation since there was no culprit end device causing large amounts of congestion. B2B credits were being returned by MDS_9148S_02 just not at a rate that would allow both MDS_9148S_01 and MDS_9148S_03 to achieve line rate.

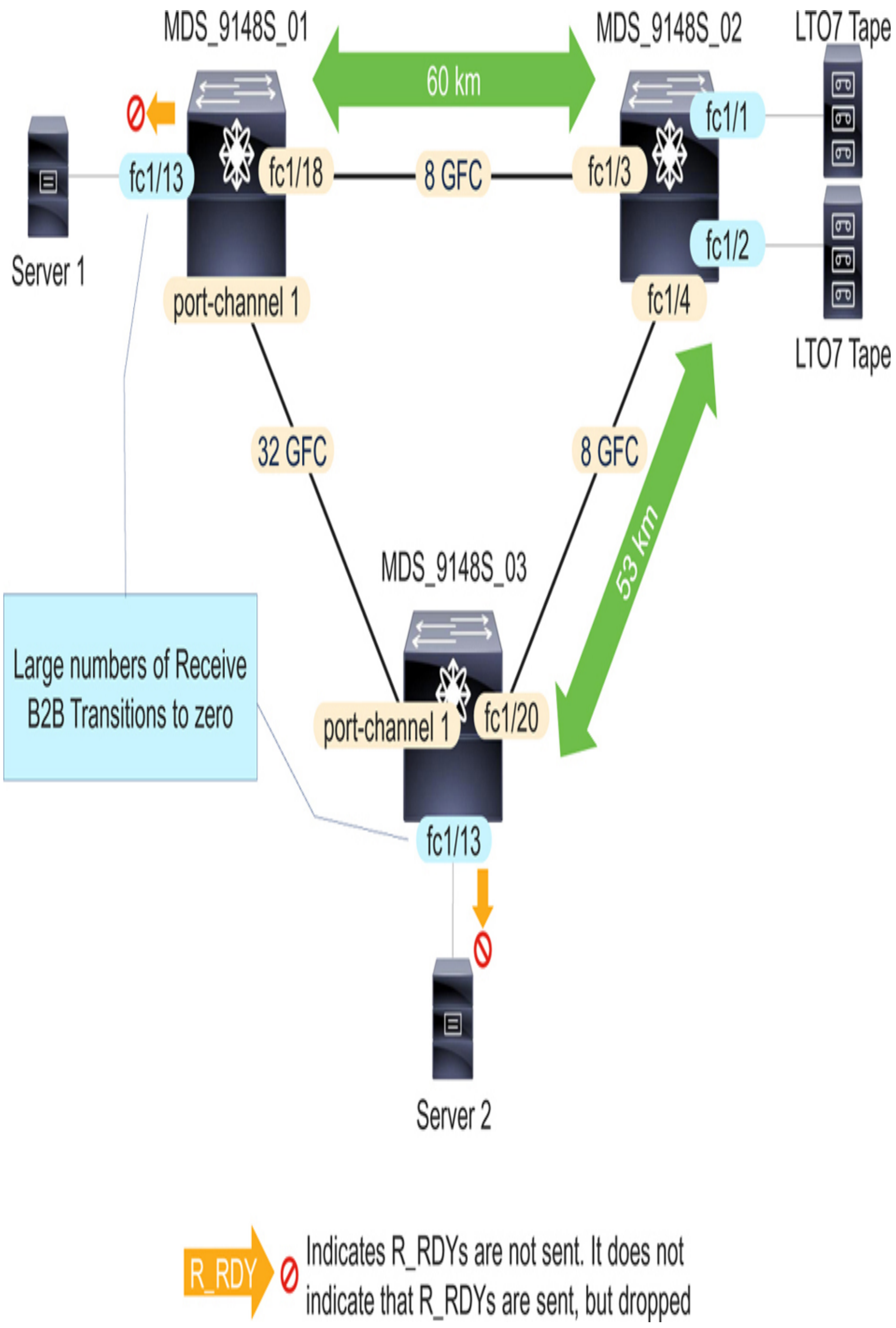


Figure 4-45 *Receive B2B credit transitions to zero on the server-connected switchports*

Case Study 4—Summary

While troubleshooting congestion, there may not be a typical culprit end device(s). An FC link may be the source of congestion as well. Regardless of the cause and the source of congestion, the troubleshooting methodology and workflow should remain the same.

In this case study, the cause of congestion was fewer than needed B2B credits on the ISLs. Both MDS_9148S_01 fc1/18 and MDS_9148S_03 fc1/20 were hitting zero remaining-Tx-B2B-credits and TxWait was incrementing not because of any slow-drain or over-utilized device(s) on MDS_9148S_02 but because there were an insufficient number of B2B credits on the ISLs.

The solution was to increase the number of B2B credits. On this switch model (Cisco MDS 9148S) the maximum number of credits configurable per port is 253. Once that was increased in both ISLs, in both directions, the amount of TxWait dropped and performance significantly improved.

As a follow-up, the maximum number of B2B credits allowed on the MDS 9148S was still below the minimum values calculated of approximately 300. The customer reported that performance, although much improved, was still not as much as they expected. At that point, another type of switch would need to be used with greater B2B credit capabilities.

Summary

Congestion in Fibre Channel fabrics can have the following severity levels:

1. **Mild Congestion (Level 1 and Level 1.5)**: Increased Latency but no frame drops and no Link Reset protocol.

   a. Level 1 when TxWait < 30%

   b. Level 1.5 when TxWait >= 30%

2. **Moderate Congestion (Level 2)**: Increased Latency and Frame drops but no Link Reset protocol
3. **Severe Congestion (Level 3)**: Increased Latency, Frame drops, and Link Reset protocol.

We recommend starting the troubleshooting with Level 3 issues followed by Level 2, Level 1.5, and Level 1 issues. While using this methodology, follow the traffic destination path to find the source of congestion.

Cisco MDS switches provide real-time and historic views of Level 3, Level 2, and Level 1 congestion events. On other types of switches, use what is available on those platforms. If the other switches connect to the MDS switches, use the reverse direction events or metrics on the MDS switches.

The primary focus of this chapter is to explain troubleshooting congestion using the on-switch commands and logs. The workflow can be more efficient if a remote monitoring platform, such as Cisco NDFC/DCNM, continuously monitors the network and automatic alerting is enabled on the switches and someone actively watches those alerts.

With the help of four case studies, this chapter explains the methodology for troubleshooting congestion in production environments. Following this methodology helps in knowing where to start and where to proceed because in all the case studies not much information was available in the beginning.

The first case study explains finding the culprit of congestion and victims in a single-switch fabric.

The second case study explains how an ‘Event 153’ error on a Windows server was because this server was a victim of congestion in the storage network.

The third case study explains the troubleshooting methodology of how you should move across multiple switches towards the destination of traffic to find the source of congestion (culprit).

Finally, the fourth case study explains how the performance degradation was correlated with high TxWait even though a slow-drain device and over-utilized edge link did not exist. The reason was diagnosed to be long-distance ISLs that did not have enough B2B credits for their distance, speed, and frame size.

The focus of this chapter is on Fibre Channel transport. But the same congestion troubleshooting methodology and workflow applies to lossless Ethernet storage networks as well. The metrics and the symptoms may be different, but the overall workflow remains the same. Refer to Chapter 7, “Congestion Management in Ethernet Storage Networks,” for details on lossless Ethernet storage networks. Instead of repeating everything there, we will refer back to this chapter for the same basic concepts.

One final takeaway is the importance of detailed, time-stamped congestion logs that show things like TxWait, RxWait, frame drops, errors, etc. These go way beyond just simple counters where it is impossible to determine when and how much the counter incremented. Without this information, a detailed investigation to arrive at the culprit and victims of various categorizations would simply be impossible. Since, congestion is a reality in almost all lossless networks, ensuring the switch fabric has this kind of diagnostic capability should be a high priority.

References

- The methodology, workflow, and case studies that are explained in this chapter are based on the years of experience of the authors in troubleshooting congestion in the production Fibre Channel fabrics across the world.
- Matplotlib: Visualization with Python: https://matplotlib.org/
- Cisco MDS 9000 NX-OS Software Configuration Guides.

Chapter 5. Solving Congestion by Storage I/O Performance Monitoring

This chapter covers the following topics:

- Why monitor storage I/O performance?
- How and where to monitor storage I/O performance?
- Cisco SAN Analytics—A network-centric approach for storage I/O performance monitoring.
- I/O flows in a storage network and metrics.
- I/O operations and network traffic patterns.
- Case studies on detecting, eliminating, and predicting congestion using Cisco SAN Analytics.

Why Monitor Storage I/O Performance?

Storage I/O performance monitoring provides advanced insights into network traffic, which can then be used to accurately solve network congestion. This information is in addition to what the network ports already provide by counting the number of packets sent and received, the number of bytes sent and received, and link errors. Additionally, storage I/O performance monitoring brings visibility to the upper layers of the stack and can explain why the network has or lacks traffic by providing the following information:

- The upper-layer protocol—SCSI or NVMe that generated the network traffic

- Upper Layer Protocol errors like SCSI queue full, reservation conflict, NVMe namespace not ready, etc.
- IOPS, throughput, I/O size, etc.
- How long I/O operations take to complete, the delay caused by storage arrays, and the delay caused by hosts.

This performance can also be monitored for every flow giving granular insights into the traffic on a network port. This flow-level performance monitoring is extremely useful because most production environments are virtualized. When a host causes congestion due to over-utilization of its link, the network can detect this condition as explained in the earlier chapters. Additionally, storage I/O performance monitoring can detect the cause of the high amount of traffic and which virtual machine (VM) is asking for it.

Likewise, when a host causes congestion due to slow-drain, investigating the SCSI- and NVMe-level performance and error metrics can explain why the host has become slower in processing the traffic. It can also be found if a particular VM has caused the entire host to slow down. Additionally, storage I/O performance monitoring can also predict the likeliness of network congestion. These and many more benefits of storage I/O performance monitoring are explained with the help of case studies later in this chapter.

Storage I/O performance monitoring is a detailed subject. Its use cases involve application and storage performance insights, storage provisioning recommendations, infrastructure optimization, change management, audits, reporting, etc. The scope of this book, however, is limited only to congestion use cases. We recommend continuing your education on this topic even outside this book.

The chapter focuses only up to the SCSI and NVMe protocols in the block-storage stack for performance monitoring. But these protocols initiate I/O operations only when an application wants them to read or write data. Therefore, monitoring higher layers in the stack, up to the applications, can provide even more insights into why the network has traffic. Application-level monitoring, however, such as that provided by the Cisco AppDynamics observability platform is outside the scope of this book. This is another area that we recommend to continue your education even outside this book.

How and Where to Monitor Storage I/O Performance

On a high level, storage I/O performance can be monitored within a host, storage arrays, or the network. These three are viable options because an I/O operation passes through many layers within the initiator (host), the target (storage array), and multiple switches in the network. This section explains these approaches briefly but the primary focus of this chapter is on monitoring storage I/O performance in the network.

Storage I/O Performance Monitoring in the Host

Most operating systems, such as Linux, Windows, and ESXi, monitor storage I/O performance. Example 5-1 shows monitoring storage I/O performance in Linux using the **iotop** command.

Example 5-1 *Storage I/O performance monitoring in Linux*

```
[root@stg-tme-lnx-b200-7 ~]# iotop

Total DISK READ :      36.30 M/s | Total DISK WRITE
:      36.85 M/s
Actual DISK READ:      36.31 M/s | Actual DISK
WRITE:      36.80 M/s
  TID  PRIO  USER     DISK READ  DISK WRITE  SWAPIN
IO>    COMMAND
  941 be/3 root        0.00 B/s    0.00 B/s  0.00 %
3.31 % [jbd2/dm-101-8]
46303 be/4 root        6.42 M/s    6.37 M/s  0.00 %
1.93 % fio config_fio_1
  542 be/3 root        0.00 B/s    0.00 B/s  0.00 %
1.89 % [jbd2/dm-22-8]
26496 rt/4 root        0.00 B/s    0.00 B/s  0.00 %
```

```
1.26 % multipathd
46383 be/4 root       7.13 M/s    7.11 M/s  0.00 %
0.42 % fio config_fio_1
46284 be/4 root      11.96 M/s   12.34 M/s  0.00 %
0.00 % fio config_fio_1
46384 be/4 root       5.19 M/s    5.40 M/s  0.00 %
0.00 % fio config_fio_1
46402 be/4 root       5.61 M/s    5.63 M/s  0.00 %
0.00 % fio config_fio_1
```

For the purpose of solving network congestion, monitoring storage I/O performance within the hosts has the following considerations.

1. Per-path storage I/O performance should be monitored because although multiple paths that perform at different levels exist between the host and the storage array, the host may only report a cumulative performance, by default.
2. Metrics from thousands of hosts should be collected and presented in a single dashboard for early detection of congestion.
3. Collecting the metrics from hosts may require dedicated agents, which have the overhead of maintaining them.
4. Different implementations on different operating systems, such as Linux, Windows, and ESXi, may have a non-uniform approach to collecting the same metrics.
5. Be aware that measuring the performance within the hosts makes it prone to issues on that host itself. Is the "monitored" end-device "monitoring" itself? What happens when it gets congested or becomes a slow-drain device?
6. Because of organizational silos, hosts and storage arrays may be managed by different teams.

Storage I/O Performance Monitoring in the Storage Array

Most arrays monitor storage I/O performance. For example, Figure 5-1 shows I/O performance on a Dell EMC PowerMax storage array.

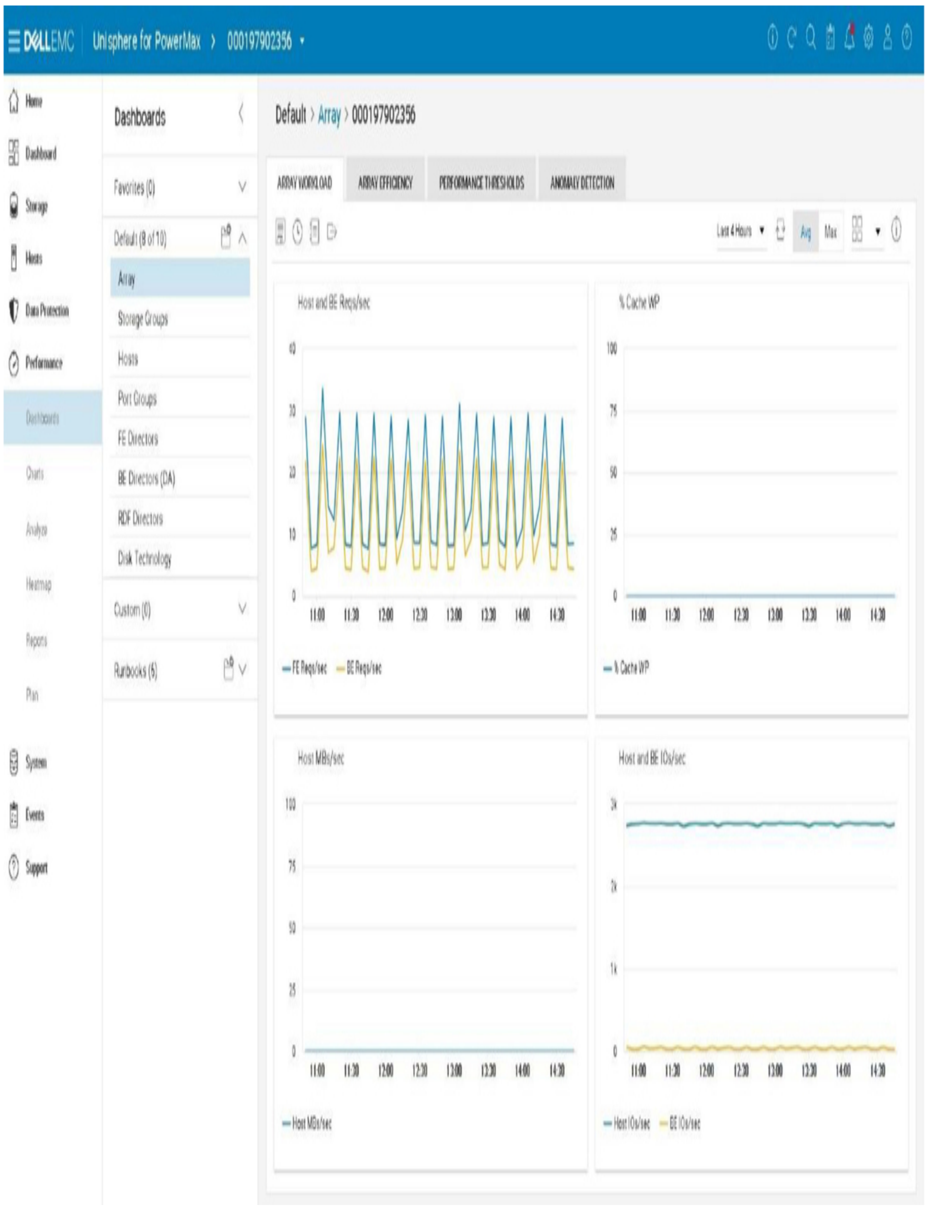


Figure 5-1 *Storage I/O performance monitoring on a Dell EMC PowerMax storage array.*

The metrics collected by the storage arrays can be used for monitoring the I/O performance, but this approach has similar challenges as the host-centric approach, as explained in the previous section.

Storage I/O Performance Monitoring in the Network

I/O operations are encapsulated within the frames for transporting them via a storage network. The network switches only need to look up the headers to send the frames toward their destination. In other words, a network, for its typical function of frame forwarding, need not know what's inside the frame. However, monitoring storage I/O performance in the network requires advanced capability on the switches for inspecting the transport (such as Fibre Channel) header and upper-layer protocol (such as SCSI and NVMe) headers.

Cisco SAN Analytics is an example of such a capability. It allows storage I/O performance monitoring natively within the network because it is integrated-by-design with Cisco MDS switches. As Fibre Channel frames are switched between the ports of an MDS switch, the port-ASICs inspect the FC and NVMe/SCSI headers and analyze them to collect I/O performance metrics such as the number of I/O operations per second, how long the I/O operations are taking to complete, how long the I/O operations are spending within the storage array, how long the I/O operations are spending within the hosts, and so on. Cisco SAN Analytics does not inspect the frame payload because there is no need for it as the metrics can be calculated by inspecting only the headers.

Cisco SAN Analytics, because of its network-centric approach and unique architecture, has the following merits for monitoring storage I/O performance:

1. **Vendor Neutral:** Cisco SAN Analytics is not dependent on server vendor (HPE, Cisco, Dell, etc.), host OS vendor (Red Hat, Microsoft, VMware, etc.), storage array vendor (Dell EMC, HPE, IBM, Hitachi, Pure, NetApp, etc.).
2. **Not dependent on end-device type:** Cisco SAN Analytics is not dependent on

   a. Server architecture — Rack-mount, blade, etc.

   b. OS type — Linux, Windows, or ESXi.

c. Storage architecture — All-Flash, Hybrid, non-flash, etc.

Legacy end devices can also benefit because no changes are needed on them, such as installing an agent or updating the firmware.

3. **No dependency on the monitoring architecture of end devices:** Different products have different logic to collect similar metrics. For example, some storage arrays collect I/O completion time on the front-end ports, whereas other storage arrays may collect it on the back-end ports. Different host operating systems may collect I/O completion time at different layers in the host stack. Cisco SAN Analytics doesn’t have this dependency.
4. **Flow level monitoring:** Cisco SAN Analytics monitors performance for every flow separately. When a culprit switchport is detected, flow-level metrics help in pin-pointing the issue to an exact initiator, target, virtual machine, or LUN/Namespace.
5. **Flexibility of location of monitoring:** Cisco SAN Analytics can monitor storage I/O performance at any of the following locations.

   a. Host-connected switchports — Close to apps and servers

   b. Storage-connected switchports — Close to Storage arrays

   c. ISL ports — Flow-level granularity in the core of the network.
6. **Granular:** Cisco SAN Analytics monitors storage I/O performance at a low granularity—Microseconds for on-switch monitoring and seconds for exporting metrics from the switch.

This chapter focuses on Cisco SAN Analytics for solving congestion in storage networks, although the education and case studies can be used with host-centric and storage array-centric approaches as well.

Cisco SAN Analytics Architecture

Cisco SAN Analytics architecture can be divided into three components (Figure 5-2).

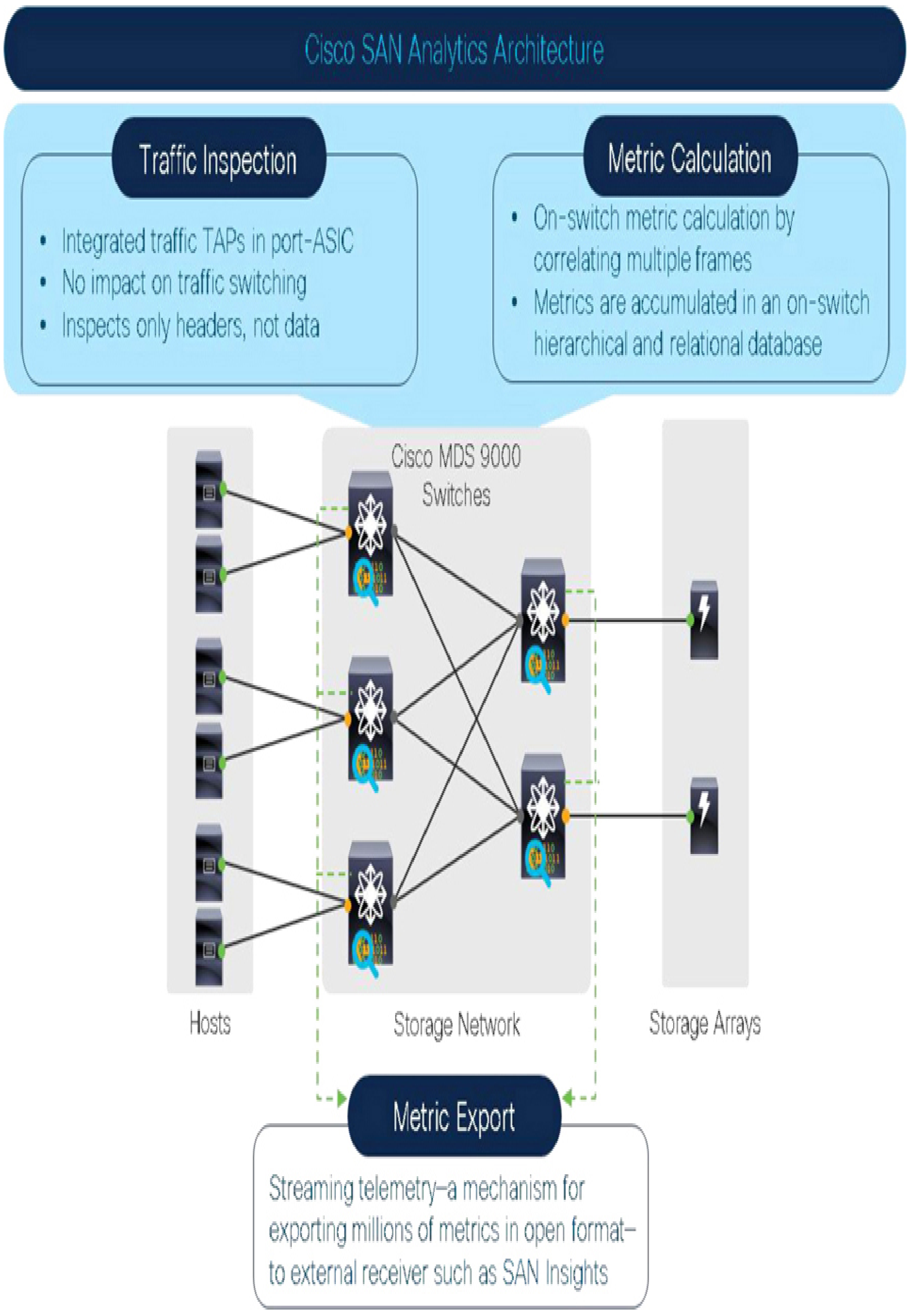

Cisco SAN Analytics Architecture
Traffic Inspection
• Integrated traffic TAPs in port-ASIC
• No impact on traffic switching
• Inspects only headers, not data
Metric Calculation
• On-switch metric calculation by correlating multiple frames
• Metrics are accumulated in an on-switch hierarchical and relational database
Cisco MDS 9000 Switches
Hosts
Storage Network
Storage Arrays
Metric Export
Streaming telemetry–a mechanism for exporting millions of metrics in open format–to external receiver such as SAN Insights

Figure 5-2 *Cisco SAN Analytics Architecture.*

- Traffic inspection by port ASICs on Cisco MDS switches.
- Metric calculation by an onboard Network Processing Unit (NPU) or by the port ASIC.

- Streaming of flow metrics to an external analytics and visualization engine for end-to-end visibility.

Traffic Inspection

Traffic inspection is integrated by design within the Fibre Channel port ASICs. In addition to switching the frames between the switchports, these ASICs can inspect the traffic in ingress and egress directions without any performance or feature penalty. In other words, Traffic Access Points (TAPs) are inbuilt into the port ASICs.

This approach is secure because the port ASICs inspect only the Fibre Channel and SCSI/NVMe headers of the relevant frames. The frame payload (application data) is not inspected.

These ASICs are custom-designed by Cisco and they are exclusively used only in MDS switches. Cisco Nexus switches and UCS Fabric Interconnects, despite supporting FC ports on selective models, use a different ASIC, and thus don't offer SAN Analytics.

Metric Calculation

After inspecting the frame headers, Cisco MDS switches calculate the metrics by correlating multiple frames with common attributes, for example, frames belonging to the same I/O operation and belonging to the same flow.

The metric calculation logic in the 32 Gbps MDS switches resides in an onboard network processing unit (NPU), which is a powerful packet processor. In the 64 Gbps MDS switches, the metric calculation logic resides within the port-ASIC itself, although the NPU continues to exist on the switches. Regardless of this architectural detail, the overall metric calculation logic remains the same.

Cisco MDS switches accumulate the metrics in a hierarchical and relational database for on-switch visibility or exporting them to a remote receiver.

At the time of this writing, Cisco SAN Analytics does not collect I/O flow metrics in FICON environments.

Metric Export

Cisco SAN Analytics is designed to inspect every flow that passes through a storage network in an always-on fashion. As a result, it collects millions of metrics per second. A traditional approach (such as SNMP) for exporting a large number of metrics may not work at this scale, and thus, Cisco introduced streaming telemetry for this purpose. In addition to being efficient, streaming telemetry exports metrics in open format, which simplifies 3rd party integrations.

The receiver of streaming telemetry can use I/O flow metrics from multiple switches for providing fabric-wide and end-to-end visibility into a single pane of glass, long-term metric retention, trending, correlation, predictions, etc. SAN Insights is an example of such a receiver, which is a feature in Cisco Nexus Dashboard Fabric Controller (NDFC), formerly known as Cisco Data Center Network Manager (DCNM). Figure 5-3 shows SAN Insights Dashboard. It provides many ready-made use cases, such as automatic learning, baselining, and deviation calculations for up to 1 million I/O flows per NDFC server as of release 12.1.2. This high scale gives visibility into the issues anywhere in the fabric.

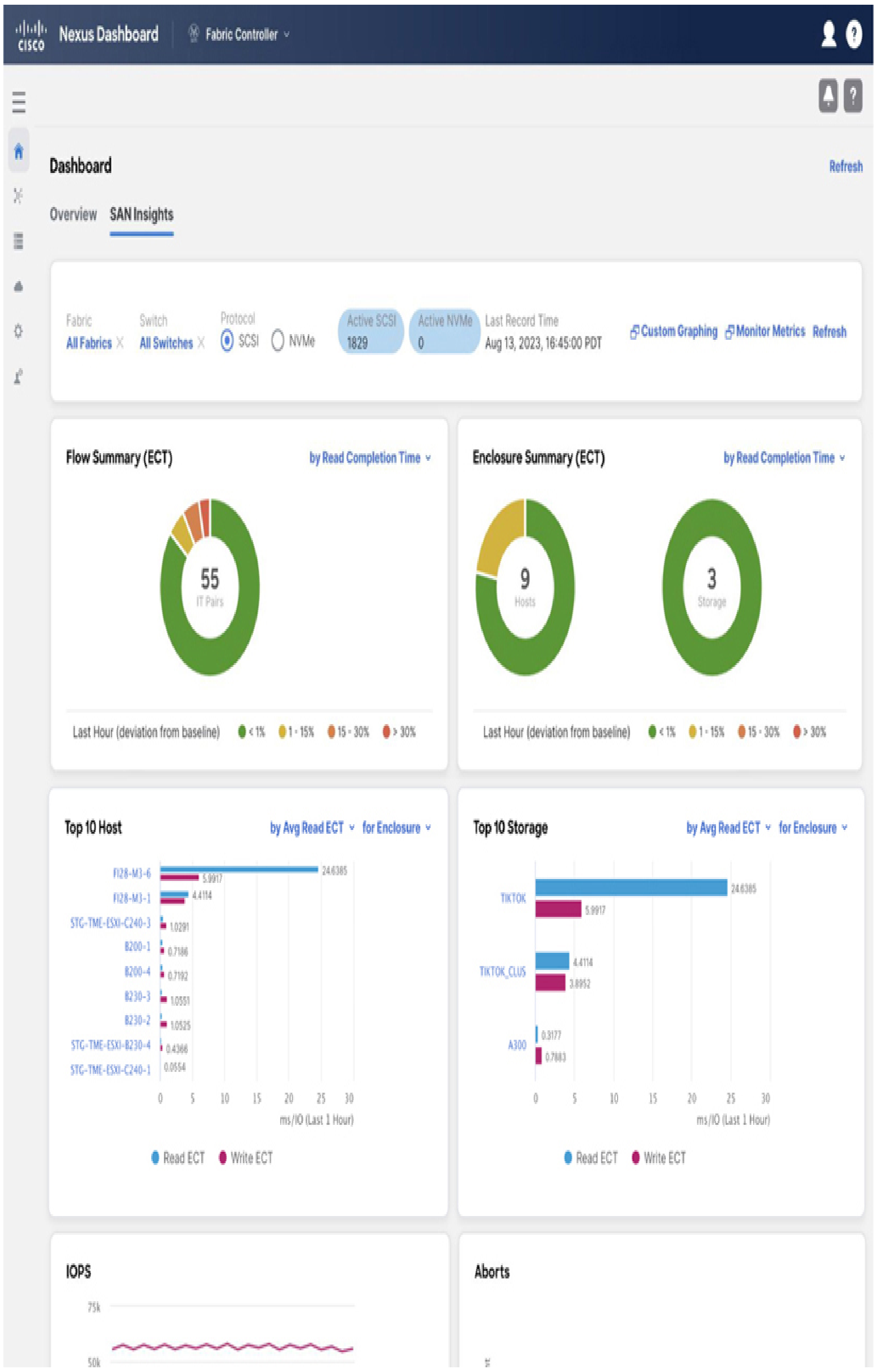


Figure 5-3 *SAN Insights Dashboard in Cisco NDFC*

Understanding I/O Flows in a Storage Network

Without considering I/O flows, a network is only aware of the frames in ingress and egress directions. Categorizing the network traffic into I/O flows helps in correlating it with initiators, targets, and the logical units (LUN) for SCSI I/O operations and namespaces for NVMe I/O operations. Additionally, storage performance can be monitored for every I/O flow individually to get detailed insights into the traffic. For example, when a switchport is 90% utilized, throughput per I/O flow can tell which initiator, target, and LUN/Namespace are the top consumers.

I/O Flows in Fibre Channel Fabrics

The following can be the types of I/O flows in a Fibre Channel Fabric

1. **Port flow:** Traffic belonging to all the I/O operations that pass through a network port makes a port flow. It can be SCSI port flow for SCSI traffic and NVMe port flow for NVMe traffic.
2. **VSAN flow:** A port of a Cisco Fibre Channel Switch may carry traffic in one or more VSANs. Hence, a port flow can be further categorized into one or more VSAN flows.
3. **Initiator flow:** Traffic belonging to all the I/O operations that are initiated by an initiator makes an Initiator flow.
4. **Target flow:** Traffic belonging to all the I/O operations that are destined for a target makes a Target flow.
5. **Initiator-Target (IT) flow:** Traffic belonging to all the I/O operations between a pair of initiator and target makes an IT flow.
6. **Initiator-Target-LUN (ITL) flow:** Traffic belonging to all the I/O operations between an initiator, target, and a logical unit (LUN) makes an ITL flow. An ITL flow is applicable only for SCSI I/O operations.
7. **Initiator-Target-Namespace (ITN) flow:** Traffic belonging to all the I/O operations between an initiator, target, and a namespace makes an ITN flow. An ITN flow is applicable only for NVMe I/O operations.

8. **Target-LUN (TL) flow:** Traffic belonging to all the I/O operations that are destined for a target port and a specific logical unit (LUN) makes a TL flow. A TL flow is applicable only for SCSI I/O operations.
9. **Target-Namespace (TN) flow:** Traffic belonging to all the I/O operations that are destined to a target port and a specific namespace makes a TN flow. A TN flow is applicable only for NVMe I/O operations.

The definition of an I/O flow can also be extended to a virtual entity (VE), such as a virtual machine (VM) on the host. When combined with an ITL or ITN flow, the end-to-end flow becomes a VM-ITL or a VM-ITN flow. There are at least two approaches for achieving this visibility into the VMs.

The first approach needs support from hosts, and in some cases even from storage arrays, for tagging the VM identifier in the frame header. Although Cisco SAN Analytics on MDS switches supports VM-ITL and VM-ITN flows, because of the dependency on the end devices most production deployments are not ready for it at the time of this writing.

The second approach uses the APIs from VMware vCenter to provide the correlation between the VM and the initiator and LUN (or namespace) from the ITL (or ITN) flow. The benefit of this approach, unlike the first approach, is that upgrading the end devices is not mandatory. Cisco SAN Insights uses this approach in NDFC 12.1.2 onwards.

In environments where even the read-only access to VMware vCenter cannot be added to NDFC, this approach can still be used for manually correlating ITL or ITN flows with the VMs. The use of this approach is demonstrated further in the section on *Case Study 3 — An Energy Company Eliminated Congestion Issues*.

This chapter focuses only on ITL flows which are natively available on the Cisco MDS switches without any dependency on the end devices and NDFC. The environments with VM-ITL flows made available using any of the two approaches mentioned earlier can benefit by expanding ITL flows in the same way as port flows are expanded to IT flows and ITL flows.

To understand the I/O flows and how they help in gaining granular details of the network, consider the example in Figure 5-4. Two initiators, I-1 and I-2, connect to two targets, T-1, and T-2, via a fabric of Switch-1 and Switch-2. The ISL port on Switch-1 (Port-3) reports an ingress throughput of 800 MB/s. After enabling SAN Analytics, Port-3 can categorize network traffic into multiple types of I/O flows and monitors the performance of every flow.

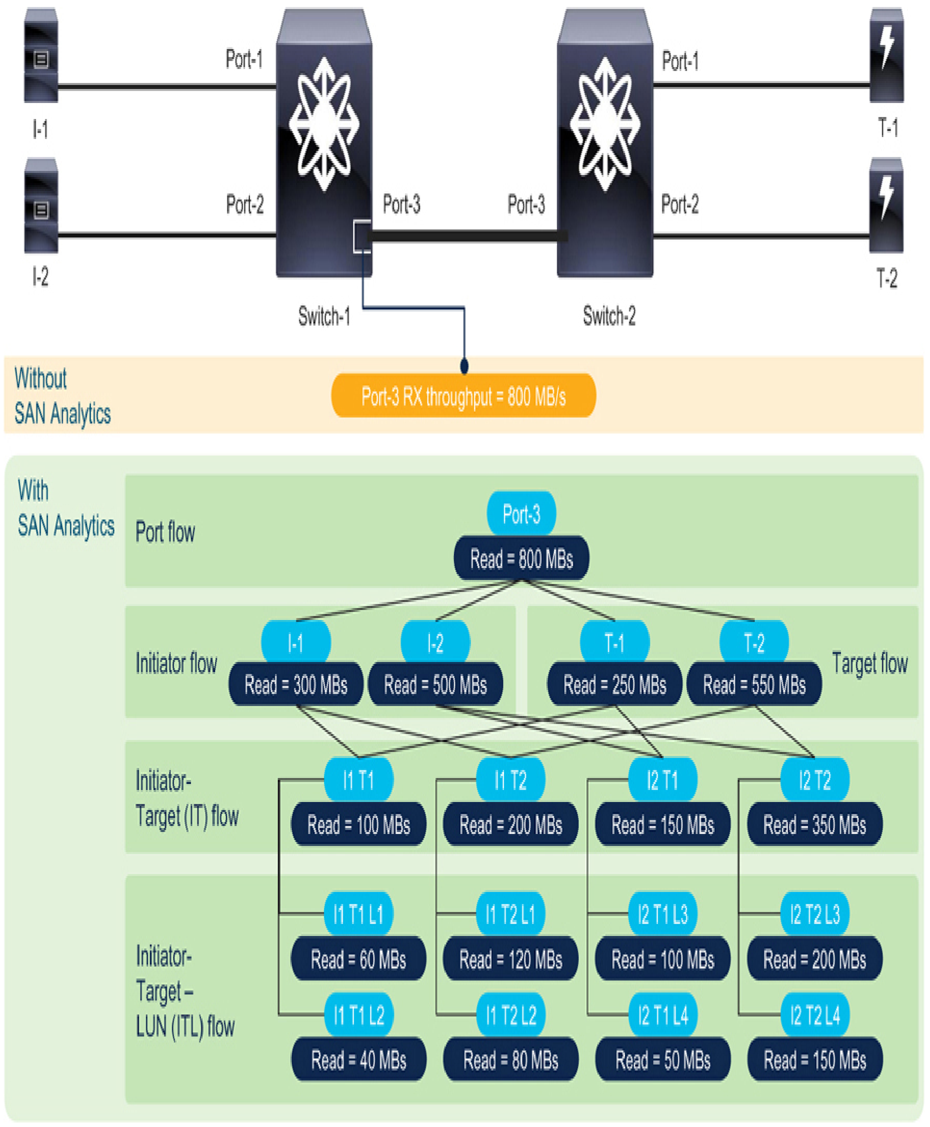


Figure 5-4 *I/O flows and flow-level metrics using Cisco SAN Analytics*

SAN Analytics can find the following details.

- The 800 MB/s throughput on Port-3 on Switch-1 is because of SCSI read I/O operations.
- Port-3 may have two VSANs—VSAN 100 and VSAN 200 (Not shown in Figure 5-4). The VSAN flows provide a further breakdown of the port flow throughput, for example, a read throughput of 600 MB/s for VSAN 100 and 200 MB/s for VSAN 200.
- I-1's read throughput via Port-3 is 300 MB/s, whereas I-2's read throughput via Port-3 is 500 MB/s.
- T-1's read throughput via Port-3 is 250 MB/s, whereas T-2's read throughput via Port-3 is 550 MB/s.
- Port-3 has four IT flows—I1-T1, I1-T2, I2-T1, and I2-T2. Their read-throughput is as follows.
  - I1-T1 is 100 MB/s.
  - I1-T2 is 200 MB/s.
  - I2-T1 is 150 MB/s.
  - I2-T2 is 350 MB/s.
- Port-3 has 8 ITL flows—I-1 uses LUN-1 and LUN-2, whereas I-2 uses LUN-3 and LUN-4. Their respective read throughput is as follows.
  - I1-T1-L1 — 60 MB/s
  - I1-T1-L2 — 40 MB/s
  - I1-T2-L1 — 120 MB/s
  - I1-T2-L2 — 80 MB/s
  - I2-T1-L3 — 100 MB/s
  - I2-T1-L4 — 50 MB/s
  - I2-T2-L3 — 200 MB/s
  - I2-T2-L4 — 150 MB/s

As evident from this example, the hierarchical and relational definitions of I/O flows help in a precise breakdown of traffic on a switchport. During congestion, the per-flow metrics, such as throughput, help in pinpointing the root cause of the exact entity, such as initiator, target, LUN, or namespace. Without per-flow storage I/O performance monitoring, as provided by Cisco SAN Analytics, such detailed insights are not possible.

I/O Flows versus I/O Operations

I/O flows shouldn't be confused with I/O operations. An I/O flow is identified by the end-to-end tuples such as initiator, target, LUN, or namespace (ITL or ITN flows). In contrast, I/O operations transfer data within an I/O flow. For example, when Initiator-1 initiates 100 read I/O operations per second to LUN-1 on Target-1, the ITL flow is identified as Initiator-1 - Target-1 - LUN-1, whereas there were 100 I/O operations per second.

An I/O flow is created only after an initial exchange of I/O operations between the identifying tuples. Later, if the initiator doesn't read or write data, the I/O flows may still exist, but no I/O operations flow through it, which results in zero IOPS for these I/O flows.

I/O Flow Metrics

The I/O flow metrics collected by Cisco SAN Analytics can be classified into the following categories.

- **Flow identity metrics:** These metrics identify a flow, such as switchport, initiator, target, LUN, or Namespace.
- **Metadata metrics:** The metadata metrics provide additional insights into the traffic. For example:
  - **VSAN count:** Number of VSANs carrying traffic on a switchport.
  - **Initiator count:** Number of initiators that are exchanging I/O operations behind a switchport.
  - **Target count:** Number of targets that are exchanging I/O operations behind a switchport.
  - **IT flow count:** Number of pairs of initiators and targets that are exchanging I/O operations via a switchport.

  - **TL and TN flow count:** Number of pairs of Targets and LUNs/Namespace behind a switchport that are exchanging I/O operations.
  - **ITL and ITN flow count:** Number of pairs of initiators, targets, and LUN/Namespace that are exchanging I/O operations via a switchport.
  - **Metric collection time:** The start time and the end time for I/O flow metrics during a specific export. This metric helps in knowing the precise duration when a metric was calculated at the link.
- **Latency metrics:** Latency metrics identify the total time taken to complete an I/O operation and the time taken to complete various steps of an I/O operation. For example:
  - **Exchange Completion Time (ECT):** Total time taken to complete an I/O operation.
  - **Data Access Latency (DAL):** Time taken by a target to send the first response to an I/O operation. DAL is one component of ECT that's caused by the target.
  - **Host Response Latency (HRL):** Time taken by an initiator to send the response after learning that the target is ready to receive data for a write I/O operation. HRL is one component of ECT that's caused by the initiator.
- **Performance metrics:** These metrics measure the performance of I/O operations. For example:
  - **IOPS:** Number of read and write I/O operations completed per second.
  - **Throughput:** Amount of data transferred by read and write operations in bytes per second.
  - **Outstanding I/O:** The number of read and write I/O operations that are yet to be completed.
  - **I/O size:** The amount of data requested by a read or write I/O operation.
- **Error metrics:** The error metrics indicate errors in read and write I/O operations. For example, Aborts, Failures,

Check condition, Busy condition, Reservation Conflict, Queue Full, LBA out of range, Not ready, and Capacity exceeded.

An exhaustive explanation of all these metrics is a detailed subject. This chapter is just a starting point for using end-to-end I/O flow metrics in solving congestion and other storage performance issues.

Latency Metrics

Latency is a generic term to convey storage performance. But as Figure 5-5 and Figure 5-6 show, there are multiple latency metrics each conveying a specific meaning. Latency metrics are measured in time (microseconds, milliseconds, etc.).

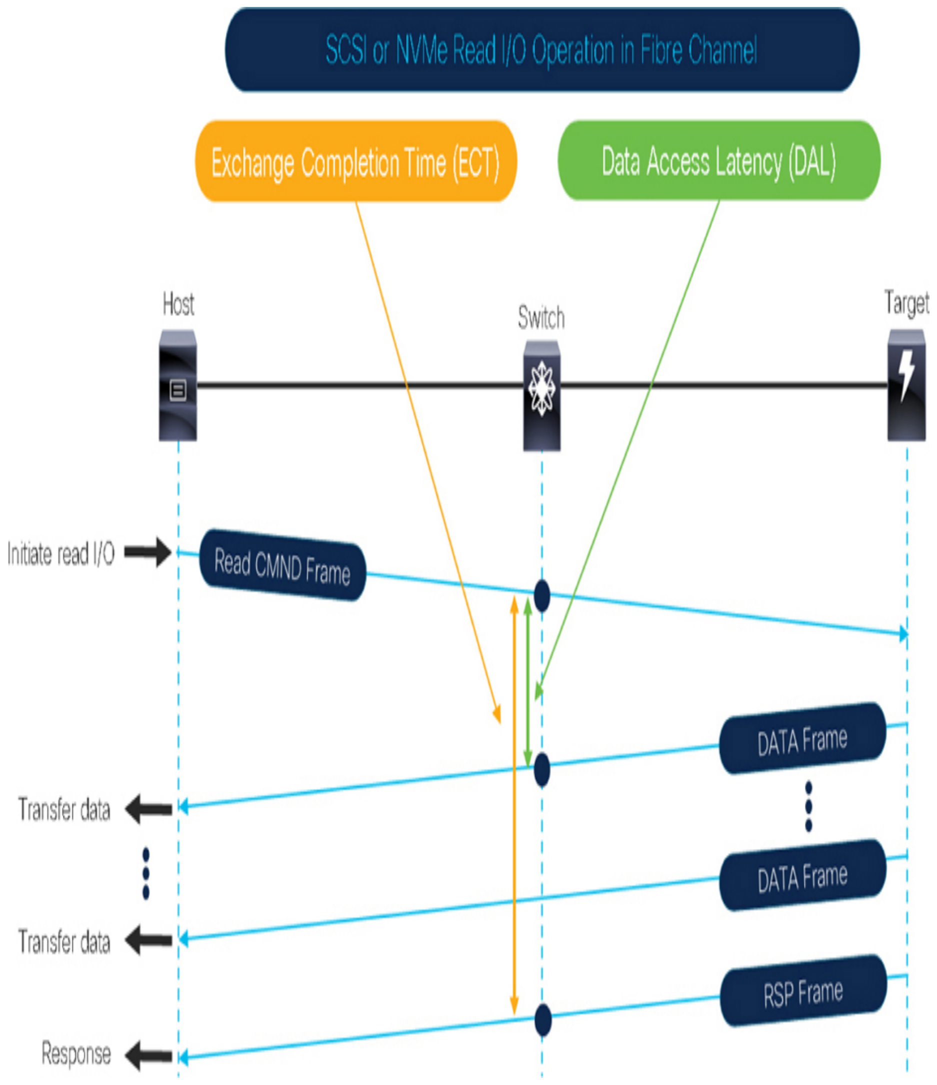


Figure 5-5 *Latency metrics for a read I/O operation*

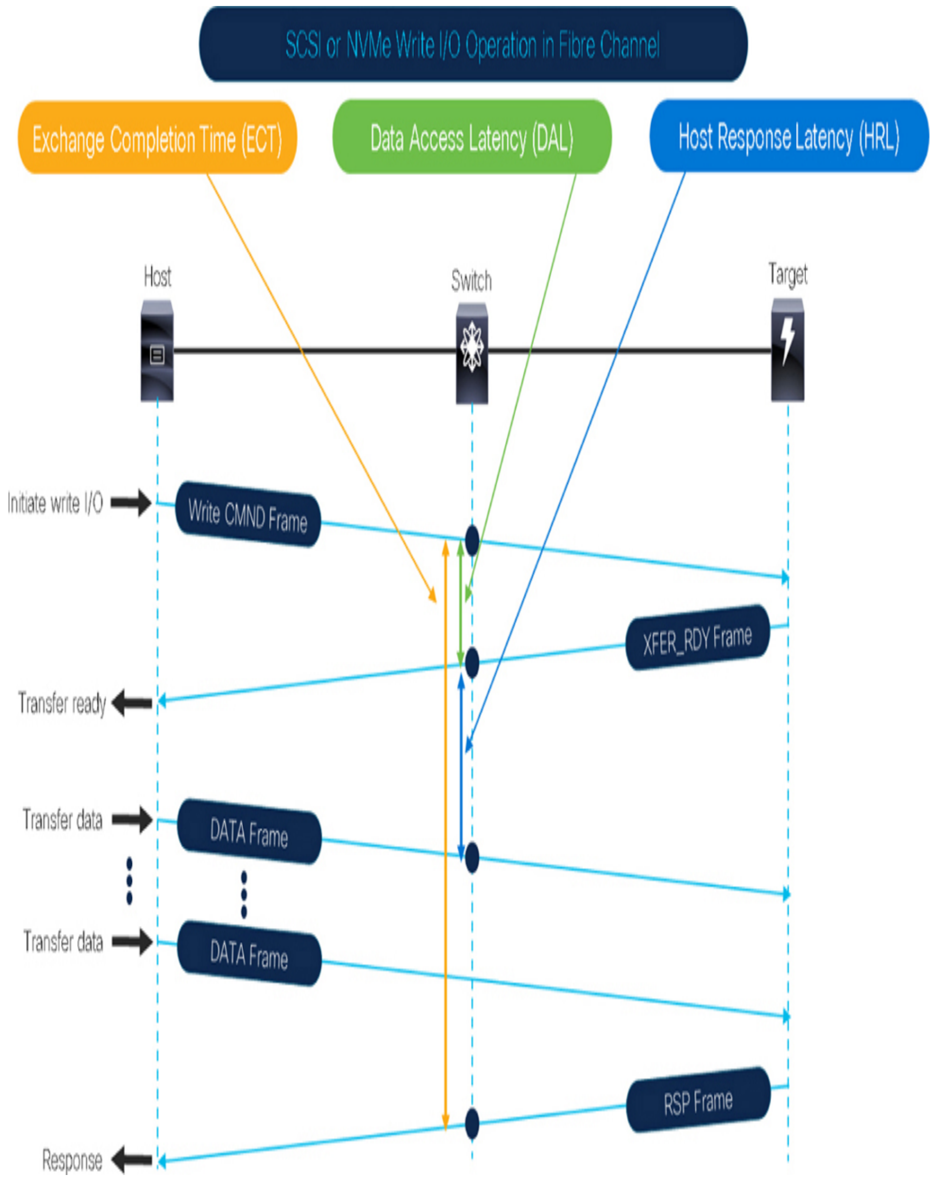


Figure 5-6 *Latency metrics for a write I/O operation*

Exchange Completion Time

Exchange Completion Time (ECT) is the time taken to complete an I/O operation. It is measured by the time difference between the command (CMND) frame and the response (RSP) frame. In Fibre Channel, an I/O operation is carried by an Exchange, and hence it's called Exchange Completion Time, but ECT can also be known as I/O completion time.

ECT is an overall measure of storage performance. In general, the lower the ECT the better. This is because lower ECTs result in improved application performance.

At the same time, a direct correlation between ECT and application performance is not straightforward because it's dependent on the application I/O profile. In general, when application performance degrades and if ECT increases (degrades) at the same time, the reason for the performance degradation is the slower I/O performance.

Data Access Latency

Data Access Latency (DAL) is the time taken by a storage array in sending the first response after receiving a command (CMND) frame. For a read I/O operation, DAL is calculated by the time difference between the command (CMND) frame and the first-data (DATA) frame. For a write I/O operation, DAL is calculated by the time difference between the command (CMND) frame and the transfer-ready (XFER_RDY) frame.

When a target receives a read I/O operation, if the data requested is not in cache, the target must first read the data from the storage media, which takes time. The amount of time it takes to retrieve the data from the media depends on several factors such as overall system utilization and the type of storage media being used. Likewise, when a target receives a write I/O operation, it must process all the other operations ahead of this operation, which takes time. An increase in these time values lead to large DAL.

In most cases, it's best to investigate DAL while troubleshooting higher ECT because DAL may tell why ECT increased. When ECT increases and DAL also increases, it indicates a slowdown within the storage array.

Host Response Latency

Host Response Latency (HRL), for a write I/O operation, is the time taken by a host in sending the data after receiving the transfer ready. It is calculated by a time difference between the transfer-ready frame and the first data frame.

Because read I/O operations do not have transfer ready, HRL is not calculated for them.

In most cases, it’s best to investigate HRL while troubleshooting higher write ECTs because HRL may tell why ECT increased. When write ECT increases and HRL also increases, it indicates a slowdown within the host.

Using Latency Metrics

The following are important details to remember about latency metrics, such as ECT, DAL, and HRL while solving congestion in a storage network.

1. A good way of using ECT is to monitor it for a long duration and find any deviations from the baseline. For example, consider two applications with an average ECT of 200 µs and 400 µs over a week. The I/O flow path of the first application gets congested resulting in an increased ECT of 400 µs. At this moment, although both applications have the same ECT, only the first application may be degraded, whereas the second application remains unaffected, although their ECT values are the same
2. ECT measures the overall storage performance, but it doesn’t convey the source of the delay, which can be the host, network, or storage array. The delay caused by the host is measured by HRL, whereas the delay caused by the storage array is measured by DAL.
3. The delay caused by the network may be the direct result of congestion. For example, when a host-connected switchport has high TxWait, the frames can’t be delivered to it in a timely fashion. As a result, the time taken to complete the I/O operations (ECT) increases.
4. Although an increase in TxWait (or a similar network congestion metric) increases ECT, the reverse may not be correct. ECT may increase even when the network isn’t congested. ECT is an end-to-end metric. It may increase due to delays caused by hosts, network, or storage. The block I/O stack within a host involves multiple layers. Similarly, an I/O operation undergoes many steps within a storage array. The delay caused by any of these layers increases ECT.

5. Network congestion is one of the reasons for higher ECT. However, it’s not the only reason. Other network issues may increase ECT even without congestion, for example, network traffic flowing through suboptimal paths, long-distance links, or poorly designed networks.
6. All latency metrics increase under network congestion. This increase is seen in all the I/O flows whose paths are affected by congestion.
7. While considering dual fabrics with active-active multipath, if only one fabric is congested, only the I/Os using the congested fabric report increase in ECT. The average increase in the ECT as reported by the host may or may not show this difference depending on how much ECT degrades. For example, consider an application that measures I/O completion time (ECT) as 200 μs. The application access storage via Fabric-A and Fabric-B. ECT over Fabric-A is 180 μs whereas ECT over Fabric-B is 220 μs. If fabric-A becomes congested and results in the increase of ECT from 180 to 270 μs (50% deviation), the average ECT as measured by the application increases to 245 μs, which is only 22% increase.

How can you verify if an increase in ECT for an application is because of congestion or not:

- Check the metrics for the ports (like TxWait) in the end-to-end data path.
- Check the ECT of the I/O flows that use the same network path as the switchport. If ECT increases just for one I/O flow but the rest of the I/O flows don’t show an increase, it is not a network congestion issue because the network doesn’t do any preferential treatment for I/O flows. A fabric just understands the frames and all frames are equal for it.
- Investigate other metrics, like I/O size, IOPS, etc. A common example is an increase in I/O size because larger I/O size operations take longer to complete. Also, find any SCSI and NVMe errors and link-level errors.

The Location of Measuring Latency Metrics

Cisco SAN Analytics calculates latency metrics by taking the time difference between relevant frames on the analytics-enabled switchports on MDS switches. As a result, the absolute value of these metrics may differ by a few microseconds based on the exact location of the measurement. For example, ECT reported by a storage-connected switchport may be a few microseconds lower than the ECT reported by a host-connected switchport. This is because the storage-connected switchport sees the command frame a few microseconds after the host-connected switchport and, it sees the response frames a few microseconds earlier than the host-connected switchport. When the time difference is taken between the command frame and the response frame on the storage port, it comes out to be less as compared to the time difference between the command frame and the response frame on the host-connected switchport.

This difference in the value of latency metrics based on the location of measurement is marginal. It may be a matter of discussion in an academic exercise but for any real-world production environment, this difference is very small, increases complexity, makes it hard for various teams to understand the low-level details, and doesn’t change the end result.

What is more important is to understand that in lossless networks congestion spreads end-to-end quickly. If this congestion increases ECT by 50% on the storage-connected switchport, the same percentage increase will be seen on the host-connected port also, although the absolute values may differ.

It can be argued that what happens if the congestion is only so severe that the effect is limited to storage ports or host ports? In production environments, the spread of congestion can’t be predicted. More importantly, if the congestion has not spread end-to-end, it’s not severe enough to act upon. In such cases, it is best to monitor and use the metrics for future planning but without an end-to-end spread, the effect of congestion is limited to a small subset of the fabric.

Performance Metrics

Performance metrics convey the rate of I/O operations, their pattern, and the amount of data transferred.

I/O Operations Per Second (IOPS)

IOPS, as its name suggests, is the number of read or write I/O operations per second. Typically, IOPS is a function of the application I/O profile and the type of storage. For example, transactional applications have higher IOPS requirements as compared to backup applications. Also, SSDs provide higher IOPS as compared to HDDs.

Inferring the network traffic from IOPS is not directly possible. An I/O operation may result in a few or many frames depending upon the data transferred by that I/O operation. Likewise, the throughput caused by I/O operations depends on the amount of data transferred by those I/O operations. Hence, it’s difficult to predict the effect of higher IOPS on network congestion without accounting for I/O size, explained next.

On the other hand, network congestion typically results in reduced IOPS because the network is unable to deliver the frames to their destinations in a timely fashion or can transfer fewer frames.

I/O Size

The amount of data transferred by an I/O operation is known as its I/O size.

I/O size is the function of the application’s I/O profile. For example, a transactional application may have an I/O size of 4 KB, whereas backup jobs may use an I/O size of 1 MB.

This I/O size metric under the context of storage I/O performance monitoring or SAN Analytics is different from the amount of data that an application wants to transfer as part of an application-level transaction or operation. For example, an application wants to transfer 1MB of data but the host may decide to request this data using four I/O operations each of size 256K. This difference is worth understanding, especially while investigating various layers within a host.

I/O size is encoded in the command frame of I/O operations. It has no dependency on network health. As a result, I/O size doesn't change with or without congestion.

Large I/O size results in a higher number of frames, which in turn leads to higher network throughput. For example, a 2 KB read I/O operation results in just one Fibre Channel data frame of size 2 KB, whereas a 64 KB read I/O operation results in 32 Fibre Channel frames of size 2 KB. Because of this reason, I/O size directly affects the network link utilization, and thus provides insights into why a host port or a host-connected switchport may be highly utilized. For example, a host link may not be highly utilized with an I/O size of 16 KB. But the same link may get highly utilized, and thus becomes the source of congestion when the I/O size spikes to 1 MB.

To understand the effect of I/O size on link utilization, consider the example in Figure 5-7. Two hosts, Host-1, and Host-2 connect to the switchports at 8GFC for accessing storage from multiple arrays. Both servers are doing 10,000 read I/O operations per second (IOPS). However, the I/O sizes used by the two servers are different. Host-1 uses an I/O size of 4K, whereas Host-2 uses an I/O size of 128K.

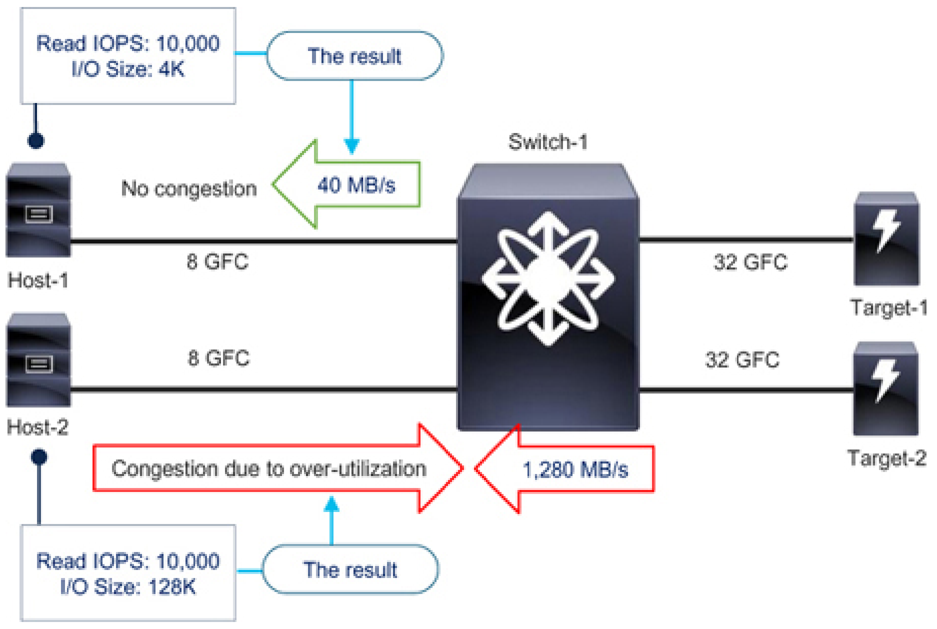


Figure 5-7 *Detecting and predicting the cause of congestion using I/O size*

Host-1 with 10,000 IOPS and 4K I/O size results in a throughput of 40 MB/s, whereas Host-2 with 10,000 IOPS and 128K I/O size results in a throughput of 1,280 MB/s. As evident, 1,280 MB/s can't be transported via 8GFC link because its maximum data rate is 800 MB/s. As a result, Host-2's read I/O traffic causes congestion due to over-utilization. Host-1 doesn't cause congestion, although its read IOPS is the same as Host-2. As evident, I/O size is the differentiating factor.

Throughput

Throughput is a generic term that has different meanings for different people. For measuring storage performance, throughput is measured in the amount of data transferred by I/O operations in Megabytes per second (MB/s). On the other hand, for measuring network performance, throughput is measured in frames transferred per second and the amount of data transferred by those frames in gigabits per second (Gbps).

Pay attention to measuring storage performance in bytes (B) per second and network performance in bits (b) per second. So don't forget to convert from bytes to bits or vice versa.

Another important detail to remember is that the read and write I/O throughput may have a marginal difference when measured on the end devices versus the network. Applications measure the total amount of data that they exchange with the storage volumes. However, the network throughput differs slightly because I/O operations have headers, such as Fibre Channel headers and SCSI/NVMe headers. For all practical purposes, this marginal difference can be ignored. Be aware that the throughput reported by various entities may differ, but don't get carried away by these marginal differences.

Outstanding I/O

Outstanding I/O is the number of I/O operations that are yet to be completed. In other words, an initiator sent a command frame, but it hasn’t received a response frame yet. Outstanding I/O is also known as Open I/O or Active I/O.

In production environment, there are always new I/Os being originated while the previous I/Os are being completed because the applications may be multithreaded or multiprocessing. Also, keeping some I/O operations open helps in a performance boost.

Outstanding I/O is directly related to the queue-depth value on a host as well as similar values on storage arrays. Different entities have different thresholds for outstanding I/O. For example, a host may stop initiating new I/O operations when the outstanding I/O reaches a threshold, such as 32. Likewise, a target may reject new incoming I/O operations when a large number of I/O operations (such as 2048) are already open (or outstanding), and the target is still processing them.

Congestion in a storage network may be a side effect of a large number of outstanding I/O operations.

I/O Operations and Network Traffic Patterns

Traffic in a storage network is the direct result of an application initiating a read or a write I/O operation. Because of this reason, network traffic patterns can be better understood by analyzing the application I/O profile, such as the timing, size, type, and rate of I/O operations. Essentially, the application I/O profile helps in understanding why the network has traffic.

Read I/O Operation in a Fibre Channel Fabric

Refer to Figure 5-8 for a SCSI or NVMe read I/O operation in a Fibre Channel fabric. A host initiates a read I/O operation using a read command, which the host encapsulates in a Fibre Channel frame and sends out of its port. The host-connected switchport receives the frame and sends them to the next hop based on the destination in the frame header. The network of switches, in turn, delivers this frame to the target. Such a frame that carries a read command is called a read command frame (CMND).

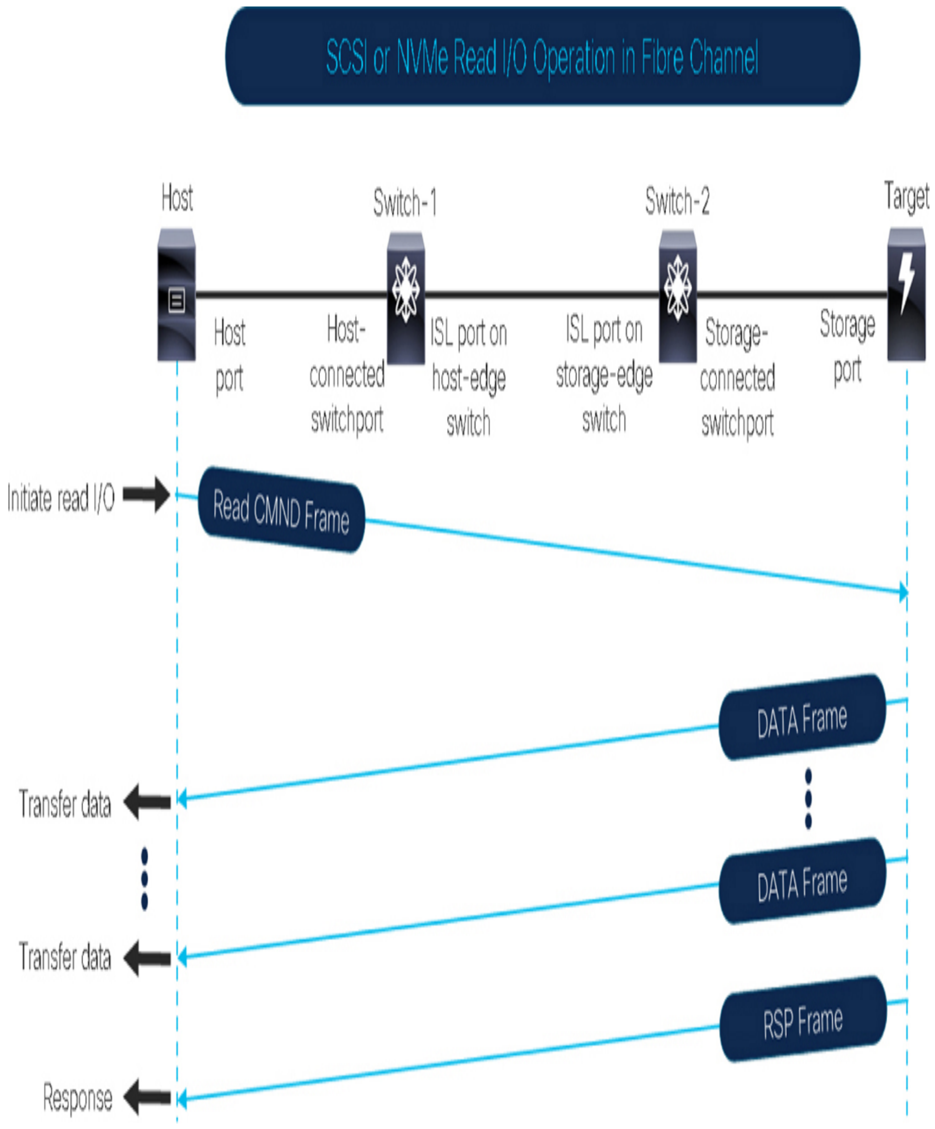

SCSI or NVMe Read I/O Operation in Fibre Channel
Host
Switch-1
Switch-2
Target
Host port
Host-connected switchport
ISL port on host-edge switch
ISL port on storage-edge switch
Storage-connected switchport
Storage port
Initiate read I/O
Read CMND Frame
DATA Frame
Transfer data
DATA Frame
Transfer data
RSP Frame
Response

Figure 5-8 *SCSI or NVMe Read I/O Operation in a Fibre Channel Fabric*

The target, after receiving the read-command frame, sends the data to the host in one or more FC frames. These frames that carry data are called data frames (DATA). The exact number of data frames returned by the target depends upon the I/O size of the read command. A full-size FC frame can transfer up to 2048 bytes (2 KB) of data. Hence, the target sends one data frame if the read I/O size is less than or equal to 2 KB. The size of this frame depends on the data carried by it plus the overhead of the header. However, when the I/O size is larger than 2 KB, the target sends the data in multiple frames. Typically, all these frames are full-size FC frames carrying 2 KB worth of data. If the size requested is not a multiple of 2KB then the last frame is smaller than 2KB. For example, an I/O size of 4 KB results in two full-size FC frames. But if the I/O size is 5 KB, the target may send two full-size FC frames each carrying 2 KB, and the third frame carrying any remaining data, which is 1 KB.

After sending all the data to the host, the target indicates the completion of the I/O operations by sending a response, which carries the status. A frame that carries a response is called a response frame (RSP).

Some implementations can optimize the read I/O operations by sending the last data and the response in the same frame if their combined size is below 2 KB. These optimized read I/O operations may not always have a dedicated response frame. Regardless of the type of read I/O operation, their result on network traffic remains the same.

Write I/O Operation in a Fibre Channel Fabric

Refer to Figure 5-9 for a SCSI or NVMe write I/O operation in a Fibre Channel fabric. A host initiates a write I/O operation using a write command, which the host encapsulates in a Fibre Channel frame and sends out of its port. The host-connected switchport receives the frame and sends them to the next hop based on the destination in the frame header. The network of switches, in turn, delivers this frame to the target. Such a frame that carries a write command is called a write command frame (CMND).

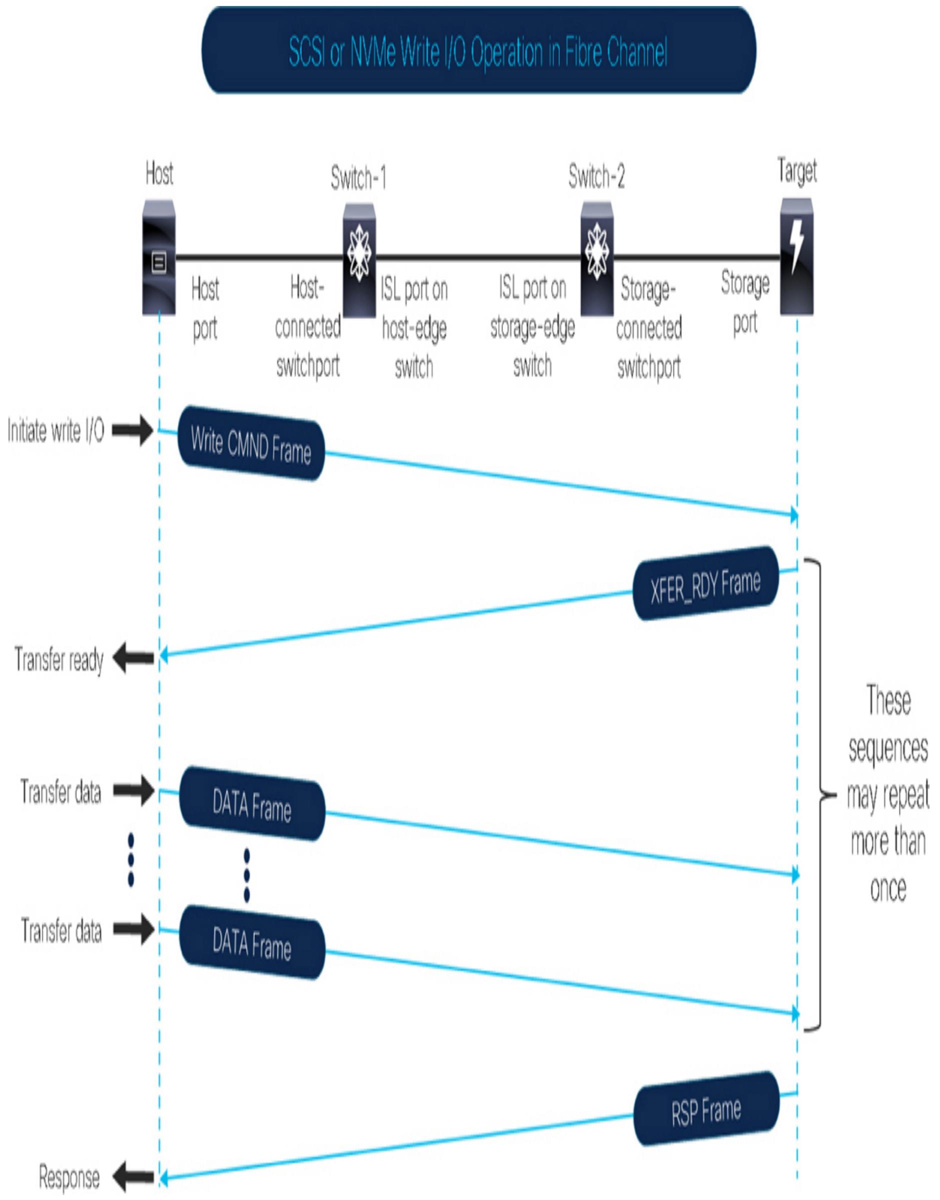

SCSI or NVMe Write I/O Operation in Fibre Channel
Host
Switch-1
Switch-2
Target
Host port
Host-connected switchport
ISL port on host-edge switch
ISL port on storage-edge switch
Storage-connected switchport
Storage port
Initiate write I/O
Write CMND Frame
XFER_RDY Frame
Transfer ready
Transfer data
DATA Frame
Transfer data
DATA Frame
These sequences may repeat more than once
RSP Frame
Response

Figure 5-9 *SCSI or NVMe Write I/O Operation in a Fibre Channel Fabric*

The target, after receiving the write-command frame, prepares to receive the data and sends a frame to the host indicating it is ready to receive all or some of the write data. This is called a transfer-ready frame (XFER_RDY). A transfer ready carries the amount of data that the target is ready to receive in one sequence or burst. Refer to Chapter 2, "Understanding Congestion in Fibre Channel Fabrics," the section on "Transforming an I/O Operation to FC frames" for more details on a Fibre Channel sequence. Typically, this size is the same as the size requested by the write-command frame. But sometimes, the target may not have the resources to receive all the data that the host wants to write in a single sequence. For example, a host may want to write 4 MB data, which it specifies in the write-command frame. The target, however, may have the resources to accept only 1 MB of data at a time. Hence, the target sends 1 MB as the burst length in the transfer-ready frame.

The host, after receiving the transfer-ready frame, sends the data to the host in one or more FC frames. These frames are called data frames (DATA). The exact number of data frames returned by the host depends upon the burst size of the transfer-ready frame. It follows the same rules as explained previously for the read I/O operations. The difference for write I/O operation is that multiple sequences of transfer-ready may be involved if the target chooses to return a burst size that is less than the write command IO size.

After receiving all the data that the host requested to write in this I/O operation (which may have been in multiple sequences due to the target sending one or multiple transfer-ready frames), the target indicates the completion of the I/O operations by sending a response, which carries the status. A frame that carries a response is called a response frame (RSP).

Some implementations can optimize the write I/O operations by eliminating the transfer ready. In such cases, the target informs the initiator during the Process Login (PRLI) state that it will always keep the resources ready to receive a minimum size (First Burst) of data. The initiator sends the data frames immediately after sending the write-command frames, without waiting for the transfer-ready frames to arrive. Regardless of the type of the write I/O operation, their result on network traffic remains the same.

Network Traffic Direction

Table 5-1 explains the direction of traffic as a result of a read I/O operation in Figure 5-8. Refer to Figure 5-10.

a

Table 5-1 *Traffic direction in a storage network because of read I/O operation*

Frame type	Host port	Host-connected switchport	ISL port on host-edge switch	ISL port on storage-edge switch	Storage-connected switchport	Storage port
Read I/O command frame	Egress	Ingress	Egress	Ingress	Egress	Ingress
Read I/O data frame	Ingress	Egress	Ingress	Egress	Ingress	Egress
Read I/O response frame	Ingress	Egress	Ingress	Egress	Ingress	Egress

Table 5-2 explains the direction of traffic because of a write I/O operation in Figure 5-9. Refer to Figure 5-10.

Table 5-2 *Traffic direction in a storage network because of write I/O operation*

Frame type	Host port	Host-connected switchport	ISL port on host-edge switch	ISL port on storage-edge switch	Storage-connected switchport	Storage port
Write I/O command frame	Egress	Ingress	Egress	Ingress	Egress	Ingress
Write I/O transfer ready	Ingress	Egress	Ingress	Egress	Ingress	Egress
Write I/O data frame	Egress	Ingress	Egress	Ingress	Egress	Ingress
Write I/O response frame	Ingress	Egress	Ingress	Egress	Ingress	Egress

As is clear from Table 5-1 and Table 5-2, egress traffic on the host port, which is the same as the ingress traffic on the host-connected switchport is because of:

- Read I/O command frames.
- Write I/O command frames.
- Write I/O data frames.

Similarly, ingress traffic on the host port, which is the same as the egress traffic on the host-connected switchport is because of:

- Read I/O data frames.
- Read I/O response frames.
- Write I/O transfer-ready frames
- Write I/O response frames.

Figure 5-10 shows the traffic directions on various network ports because of different sequences of read and write I/O operations.

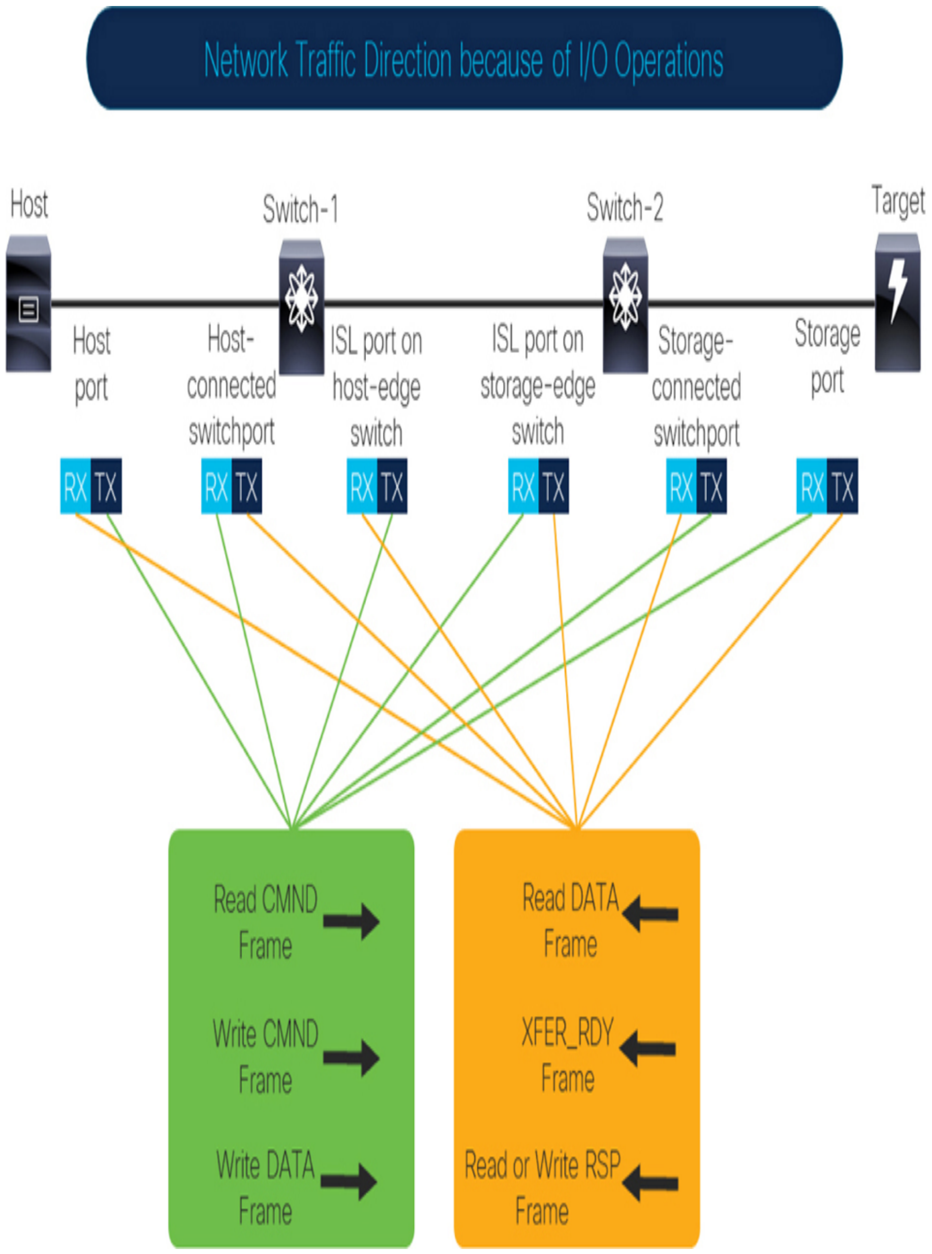


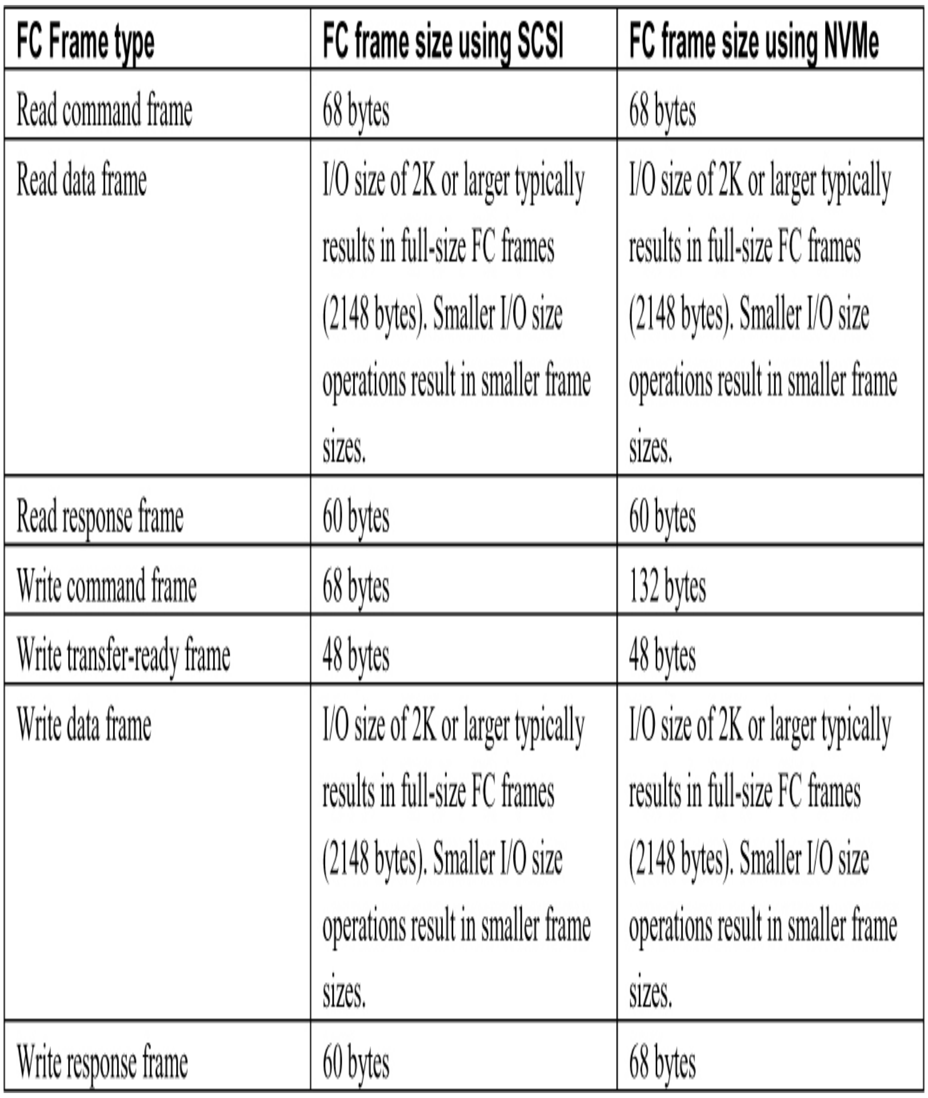
Figure 5-10 *Network traffic direction because of read and write I/O operations*

Typically a network switch doesn’t need to know the type of a frame (command, data, transfer ready, or response) to send the frames toward destination. However, without knowing the type of the frame, the real cause of throughput can’t be explained. This is another reason for monitoring the storage I/O performance using SAN Analytics.

Network Traffic Throughput

The previous section explains the direction of traffic of read and write I/O operations. But not all the frames are of the same size. Read and write I/O data frames are large and usually occur in larger quantities. Hence, they are the major contributors to link utilization. Other frames, such as read and write I/O command frames, response frames, and write I/O transfer ready frames are small and relatively few. Hence, they cause much lower link utilization. Refer to Table 5-3 for the typical size of different frame types for SCSI and NVMe I/O operations.

Table 5-3 *Typical size of frames for SCSI and NVMe I/O Operations*



FC Frame type	FC frame size using SCSI	FC frame size using NVMe
Read command frame	68 bytes	68 bytes
Read data frame	I/O size of 2K or larger typically results in full-size FC frames (2148 bytes). Smaller I/O size operations result in smaller frame sizes.	I/O size of 2K or larger typically results in full-size FC frames (2148 bytes). Smaller I/O size operations result in smaller frame sizes.
Read response frame	60 bytes	60 bytes
Write command frame	68 bytes	132 bytes
Write transfer-ready frame	48 bytes	48 bytes
Write data frame	I/O size of 2K or larger typically results in full-size FC frames (2148 bytes). Smaller I/O size operations result in smaller frame sizes.	I/O size of 2K or larger typically results in full-size FC frames (2148 bytes). Smaller I/O size operations result in smaller frame sizes.
Write response frame	60 bytes	68 bytes

Correlating I/O Operations, Traffic Patterns, and Network Congestion

The direction and size of various frames in a storage network lead to the following conclusions:

1. Read and write data frames are the major cause of link utilization. Other frames, such as command frames and response frames, are small, and hence their throughput is negligible as compared to the data frames.
2. Read and write data frames flow only after (or as the result of) command frames.
3. A command frame, based on the size of the requested data (called I/O size), can generate many data frames.
4. Most data frames of an I/O operation are full-sized, except the last frame in the sequence.
5. Read data frames flow from storage (target) to hosts (initiators), whereas write data frames flow from hosts to storage.
6. When a host-connected switchport is highly utilized in the egress direction, it’s mostly due to read data frames. Likewise, when a storage-connected switchport is highly utilized in the egress direction, it’s mostly due to write data frames.
7. The key reason for congestion due to slow-drain from hosts and due to over-utilization of the host link are the multiple concurrent large-size read I/O command frames from the host. In other words, the host is asking for more data than it can process or can be sent to it on its link.
8. The key reason for congestion due to slow-drain from a storage port or due to over-utilization of the storage link is the total amount of data being requested by the storage array via multiple concurrent write I/O transfer ready frames. In other words, the storage array is asking for more data than it can process or can be sent to it on its link.

These conclusions are extremely useful in understanding the reason for congestion caused by a culprit device or the effect of congestion on the victim devices. These conclusions also explain that monitoring a host port or a switchport can detect congestion, whereas storage I/O performance monitoring can give insights into why the congestion exists.

For example, refer to Figure 5-11, which illustrates congestion due to over-utilization of the host links because of large-size read I/O operations. The host connects at 32GFC. It initiates 5000 read I/O operations per second (IOPS), each requesting to read 1 MB of data from various targets. To initiate these I/O operations, the host sends 5000 command frames per second, each of 68 bytes, which leads to the host port's egress throughput of 2.8 Mbps (5000 x 68 B x 8 bits per byte), which is the same as the ingress throughput on the host-connected switchport. Because the maximum data rate of a 32GFC port is 28.025 Gbps, these command frames result in 0.01% utilization, which is negligible.

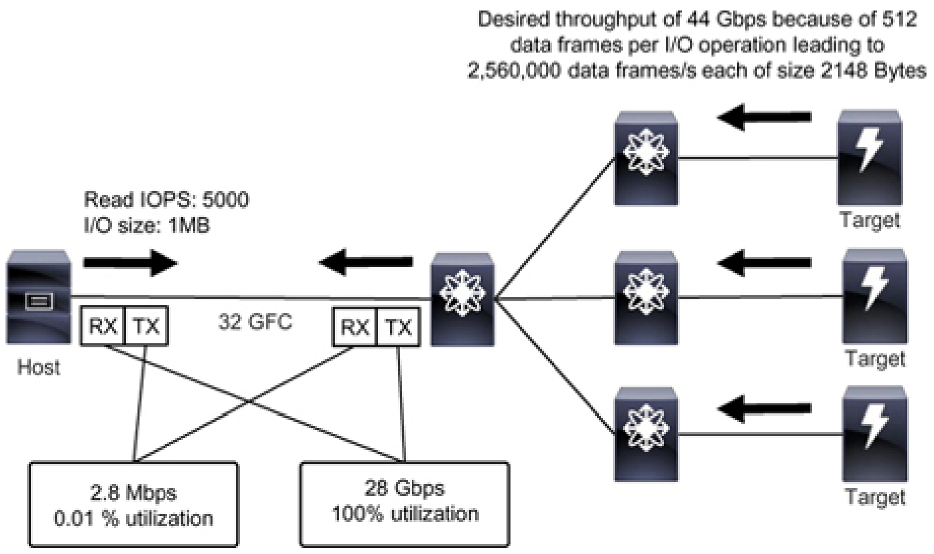

Desired throughput of 44 Gbps because of 512 data frames per I/O operation leading to 2,560,000 data frames/s each of size 2148 Bytes
Read IOPS: 5000
I/O size: 1MB
Host
RX
TX
32 GFC
RX
TX
Target
Target
Target
2.8 Mbps
0.01 % utilization
28 Gbps
100% utilization

Figure 5-11 *Congestion due to over-utilization because of large-size read I/O operations.*

The targets, after receiving these command frames, send the data for every I/O operation in approximately 512 full-size frames (2048 bytes per frame). For 5000 IOPS, the targets send 2,560,000 frames/sec (5000 x 512), each of size 2148 bytes (including header). These data frames lead to a throughput of 44 Gbps (2,560,000 x 2148 B x 8 bits per byte). But the host can receive only 28.025 Gbps on the 32GFC link.

This condition results in congestion due to over-utilization of the host link. The key point to understand is that the ingress utilization of the host-connected switchport is negligible yet this minimal throughput results in 100% egress utilization. From the perspective of the network, these are just the percentage utilization of the links. Only after getting an insight into the I/O operations, the real reason for the link utilization can be explained.

Although the read I/O data frames make the most of the egress traffic on a host-connected switchport, these data frames are just a consequence of the read I/O command frames which were sent by the host port. Because limiting the rate of read I/O command frames can lower the rate of read I/O data frames, limiting the rate of ingress traffic on the host-connected switchport can lower the rate of egress traffic on this port. This logic makes the foundation of Dynamic Ingress Rate Limiting, which is a congestion prevention mechanism explained in Chapter 6, "Preventing Congestion in Fibre Channel Fabrics."

Case Study 1 — A Trading Company Predicted Congestion Issues using SAN Analytics

A trading company has thousands of devices connected to a Fibre Channel fabric, and they have multiple such fabrics. Because of the large scale, they always had minor congestion issues. However, the severity and number of such issues increased as they deployed all-flash storage arrays. After an investigation, they found that the newer congestion issues were due to the over-utilization of the host links. Most hosts were connected to the fabric at 8GFC. The older storage arrays were connected at 16GFC. But the newer all-flash arrays were connected at 32GFC, which increased the speed mismatch between the hosts and the storage. As explained in Chapter 1, "Introduction to Congestion in Storage Networks," the section on *Congestion Due to Over-Utilization of a Link*, this speed mismatch combined with the high performance of all-flash arrays was the root cause of the increased occurrences of congestion issues.

The trading company understood the problem and its root cause. They also understood that the real solution was to upgrade the hosts because this would eliminate the speed mismatch with the all-flash storage arrays, essentially removing one major cause of congestion due to over-utilization of the host links. But, due to finite human resources, they could only upgrade a few hundred hosts every month. At this pace, it would take them many years to upgrade all the hosts and they would be subjected to congestion issues during this time. While they could not speed up this change, they wanted to have a prioritized list of the hosts with more likeliness of causing congestion. Instead of upgrading the host randomly or in another order that doesn’t consider the likeliness of congestion, following this methodology would allow them to minimize congestion issues.

Background

The trading company uses storage arrays from two major vendors. Their hosts include almost all kinds of servers (such as blade and rack-mount servers) from all major vendors. They use all major operating systems for hosting hundreds of applications.

The trading company uses Cisco MDS switches (mostly modular directors) in their Fibre Channel fabrics. Most connections were capable of running at 16GFC. However, while deploying all-flash arrays, they upgraded the storage connections to 32GFC. For management and monitoring of the fabric, they use Cisco Data Center Network Manager (DCNM), later re-branded as Nexus Dashboard Fabric Controller (NDFC).

Initial Investigation — Finding the Cause and Source of Congestion

The trading company used the following tools for detecting and investigating congestion issues.

1. **Alerts from Cisco MDS Switches:** They had enabled alerts for Tx B2B credit unavailability using TxWait counter and alerts for high link utilization using Tx-datarate counter. As they deployed all-flash arrays, the number of alerts generated due to TxWait didn't change but the number of alerts due to Tx-datarate increased.
2. **Traffic trends, seasonality, and peak utilization using DCNM:** After receiving the alerts from the MDS switches, the trading company used the historic traffic patterns in DCNM. The host ports that generated Tx-datarate alerts showed an increased peak utilization. This increased utilization coincided with the time when they deployed all-flash storage arrays.

These two mechanisms are explained in detail in Chapter 3, "Detecting Congestion in Fibre Channel Fabrics."

A Better Host Upgrade Plan

They designed the host upgrade plan using the following two steps.

1. Step-1: Detect the hosts that were already causing congestion and upgrade them first.
2. Step-2: Predict the hosts that were more likely to cause congestion and upgrade them next.

Step-1: Detect Congestion

The trading company detected the hosts that needed urgent attention as explained in the earlier section on *Initial Investigation — Finding the Cause and Source of Congestion*. These ports were the first to be upgraded. Among these ports, they prioritized upgrading the ports with slower speeds.

But only a small percentage of the hosts made it to this list and they still wanted a prioritized list of the other hosts.

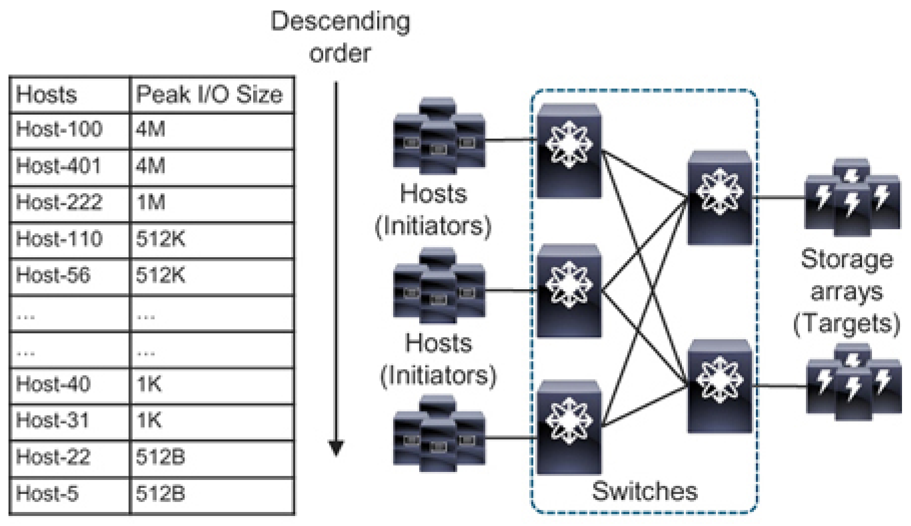
Step-2: Predict Congestion

The next step in designing a host upgrade plan (a priority list of hosts) was finding the hosts that were more likely to cause congestion due to over-utilization of their links.

Additionally, they wanted to find the hosts that were the cause of congestion but could not be detected using the mechanism of step-1. Any detection approach has a minimum time granularity. Events that sustain for a shorter duration than this minimum time granularity often remain undetected. For example, even if congestion is detected at a granularity of 1 second, many congestion issues that sustain microseconds (sometimes called micro congestion) can’t be detected. This is common with the all-flash storage arrays that have response times in microseconds. Because of this reason, the usual detection mechanisms of the previous step can’t predict the likelihood of congestion.

This is where the insights obtained by SAN Analytics help. The trading company enabled SAN Analytics on all their storage ports. Although only the storage ports inspected the traffic, the visibility from SAN Analytics was end-to-end at a granularity of every Initiator, Target, and logical unit (LUN) or ITL flow.

After collecting I/O flow metrics for a week, they took the following steps (Figure 5-12):



Hosts	Peak I/O Size
Host-100	4M
Host-401	4M
Host-222	1M
Host-110	512K
Host-56	512K
...	...
...	...
Host-40	1K
Host-31	1K
Host-22	512B
Host-5	512B

Hosts with larger read I/O size are more likely to cause congestion due to over-utilization of their links

Figure 5-12 *Sorted list of hosts as per their peak read I/O size for predicting congestion due to over-utilization.*

1. Extracted the read I/O size, write I/O size, read IOPS, and write IOPS for all the hosts.
2. Made sorted lists of the hosts as per their read I/O size and read IOPS. In other words, they found the hosts with the largest read I/O size and highest read IOPS. Write I/O size and write IOPS were not considered because as mentioned in the section on *Correlating I/O Operations, Traffic Patterns, and Network Congestion*, most traffic due to write I/O operations flows from hosts to targets and do not lead to congestion due to over-utilization of host link.
3. The hosts at the top of this list were more likely to cause congestion of their links. Such hosts were upgraded before the hosts with smaller read I/O sizes and lower IOPS. This is how they predicted congestion.

For predicting congestion, a key consideration is to focus on the peak values instead of the average values of the I/O flow metrics. This is because when the average values are high, they indicate that the real-time values sustain for a while. In this case, sustained traffic could have been detected by Tx-datarate alert in step-1, which has a granularity of 10 seconds. But Tx-datarate could miss occasional spikes in traffic that sustain only for a few milliseconds or even seconds. Such conditions can be found or even predicted by focusing on the peak values of the I/O flow metrics.

Another consideration is to prioritize the I/O size metric over the IOPS metric because of two key reasons. First, as explained earlier in the section on *I/O size*, I/O size is decided by the application or the host and it is not affected by network congestion. In contrast, IOPS is reduced during network congestion. The second reason is that I/O size is an absolute metric which means it is directly collected from the frame headers. As a result, its peak value is not affected by averaging. In contrast, IOPS is a derived metric from the average number of I/O operations over a duration, such as 30 seconds. Even the most granular value of IOPS must be calculated over a duration, which makes it an average value. This goes against the benefit of the peak values explained earlier.

For collecting data, the trading company used a custom-developed collector that polled the metrics for initiator flows every 30 seconds from the MDS switches and then used the peak values in 6-hour ranges. It was a custom development because their use case was very specific and it was unavailable ready-made at that time on the MDS switches or SAN Insights. The raw metrics were still available, just that they were not available in an easy-to-interpret format. This is what their custom development achieved.

Example 5-2 shows the output of a similar custom development, which is based on the **ShowAnalytics** command on the MDS switches. It shows a sorted list of initiators as per their read I/O sizes. The **ShowAnalytics** command is a presentation layer for the raw flow metrics. It is written in Python. Many use cases are available ready-made, and their functionality can be enhanced even further by the users. More details are available at https://github.com/Cisco-SAN/ShowAnalytics-Examples/tree/master/004-advanced-top-iosize.

Example 5-2 *Finding I/O size of hosts using SAN Analytics*

```
MDS# python bootflash:analytics-top-iosize.py --top
--key RIOSIZE

+--------+-----------------------------------------
+-------------------+
|  PORT  |        VSAN|Initiator|Target|LUN
|      IO SIZE      |
+--------+-----------------------------------------
+-------------------+
|        |
|   Read   |  Write   |
| fc1/35 |  20|0x320076|0x050101|002c-0000-0000-0000
|  1.2 MB |32.0 KB   |
| fc1/34 |  20|0x320076|0x050041|000c-0000-0000-0000
|  1.1 MB |32.0 KB   |
| fc1/33 | 20|0x320076|0x050021|002f-0000-0000-0000
|  1.0 MB |25.6 KB  |
| fc1/35 | 20|0x320076|0x050101|001b-0000-0000-0000
|  1.0 MB |48.0 KB  |
| fc1/33 | 20|0x320076|0x050021|0017-0000-0000-0000
| 992.0 KB|27.4 KB  |
| fc1/33 | 20|0x320076|0x050021|0026-0000-0000-0000
| 992.0 KB|32.0 KB  |
| fc1/33 | 20|0x320076|0x050021|0022-0000-0000-0000
| 960.0 KB|32.0 KB  |
| fc1/34 | 20|0x320076|0x050041|0025-0000-0000-0000
| 960.0 KB|28.0 KB  |
| fc1/35 | 20|0x320076|0x050101|001a-0000-0000-0000
| 960.0 KB|32.0 KB  |
| fc1/34 | 20|0x320076|0x050041|0014-0000-0000-0000
| 928.0 KB|32.0 KB  |
+--------+------------------------------------------
+------------------+
```

Case Study 1 — Summary

A trading company reduced congestion issues by designing a two-step host upgrade plan. In step-1, they used the congestion detection capabilities of Cisco MDS switches and DCNM (NDFC). In step-2, they used the predictive capabilities of SAN Analytics. Instead of upgrading the hosts randomly, they prioritized upgrading the hosts that were more likely to cause congestion based on the peak values of read I/O size. By following this plan, they lowered the severity of congestion and the number of such issues was only a fraction compared to the beginning of their upgrade cycle when they started deploying all-flash arrays.

Case Study 2 — A University Avoided Congestion Issues by Correcting Multipathing Misconfiguration

A university observed congestion issues in their storage networks. After enabling alerting on the MDS switches, the university concluded that the congestion was due to the over-utilization of a few host links.

They monitored the read and write I/O throughput on these hosts using the host-centric approach as explained earlier in the section on *Storage I/O Performance Monitoring in the Host*. The throughput reported by the operating system (Linux) was much lower than the combined capacity of the host ports. This made them believe that ample network capacity was still available.

The university wanted to know why these hosts caused congestion due to over-utilization even though the I/O throughput was less than the available capacity. Finding the reason for the congestion would pave the path to the solution.

Background

The university used the Port-Monitor feature for automatically detecting congestion and generating alerts on Cisco MDS switches. They also enabled SAN Analytics and exported the metrics to DCNM/NDFC SAN Insights for long-term trending and end-to-end correlation of the I/O flow metrics.

Investigation

The university measured the host I/O throughput at the operating system, which was the combined throughput, but they had not measured the per-path I/O throughput. This was important because their hosts were connected to the storage arrays via two independent and redundant Fibre Channel fabrics (Fab-A and Fab-B). Most of their hosts have two HBAs, each with two ports (a total of 4 ports). The first port on both HBAs connects to Fab-A, whereas the second port on both HBAs connects to Fab-B (Figure 5-13).

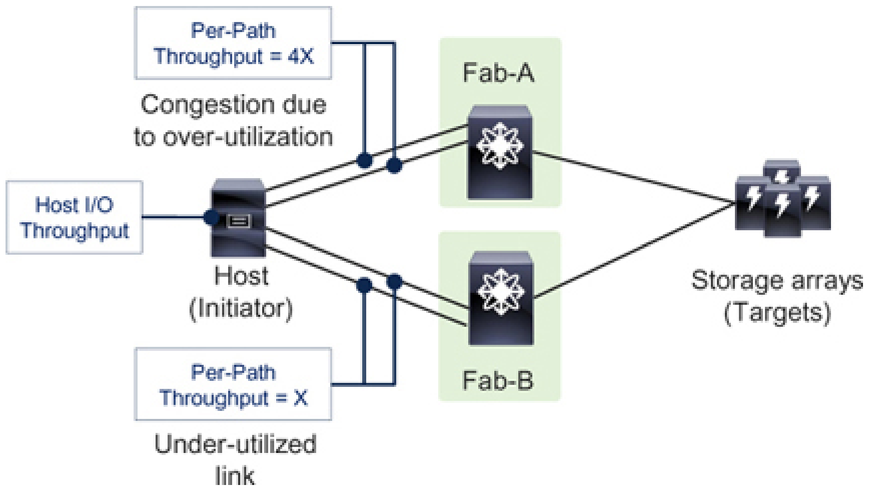
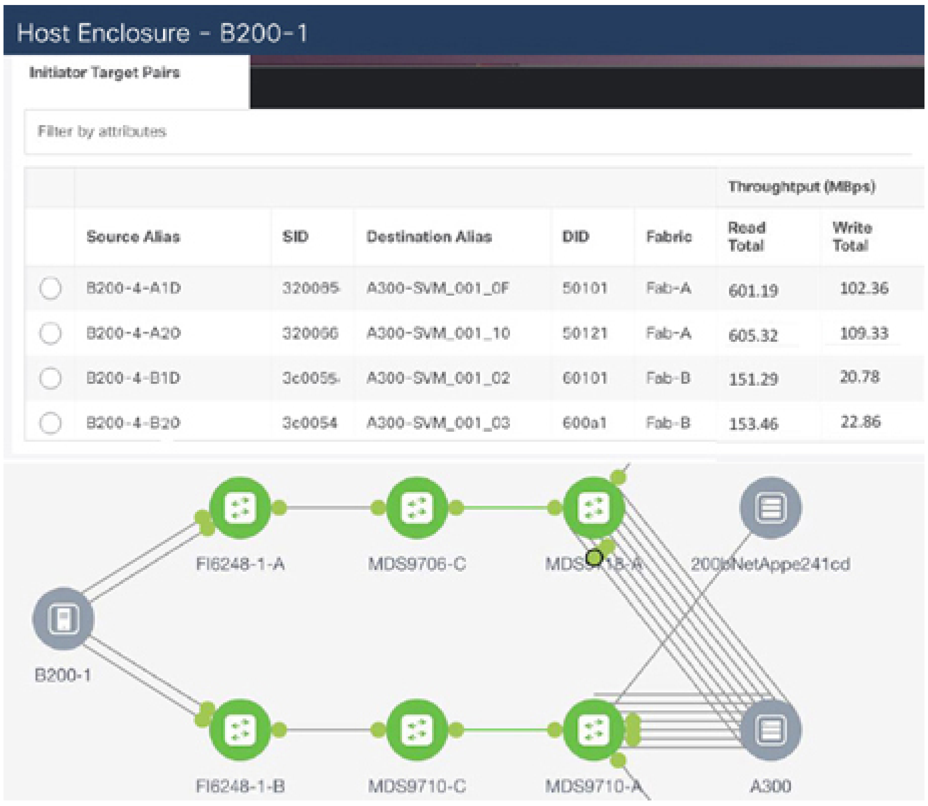


Figure 5-13 *Per-path throughput monitoring helps in finding multipathing misconfiguration.*

They used SAN Analytics to find the throughput per path, which is already available in DCNM SAN Insights. They found that although the combined throughput reported by SAN Insights is the same as the throughput they had measured at the operating system, the per-path throughput is not uniformly balanced. The ports connected to Fab-A were up to four times more utilized than the ports connected to Fab-B. When the host I/O throughput used to spike, the increase seen on the ports connected to Fab-A was up to four times as compared to the increase seen on the ports connected to Fab-B. During this duration of the spike, the ports connected to Fab-A would operate at full capacity while the ports connected to Fab-B would be underutilized. This was the reason for congestion due to the over-utilization of host links in Fab-A.

In Figure 5-13, traffic imbalance among the four host links can also be detected by measuring the utilization of host ports or their connected switchports. But if the hosts are within a blade server chassis, finding this traffic imbalance is not possible just by measuring port utilization. For example, in Cisco UCS architecture, the links that connect to the MDS switches can carry traffic for up to 160 servers, each with multiple initiators. Finding the throughput per initiator is possible only after getting flow-level visibility as provided by SAN Analytics.

Figure 5-14 shows per-path throughput for the host and an end-to-end topology in DCNM/NDFC.


Host Enclosure - B200-1

Initiator Target Pairs

Filter by attributes

						Throughput (MBps)	
	Source Alias	SID	Destination Alias	DID	Fabric	Read Total	Write Total
◯	B200-4-A1D	320065	A300-SVM_001_0F	50101	Fab-A	601.19	102.36
◯	B200-4-A20	320066	A300-SVM_001_10	50121	Fab-A	605.32	109.33
◯	B200-4-B1D	3c0055	A300-SVM_001_02	60101	Fab-B	151.29	20.78
◯	B200-4-B20	3c0054	A300-SVM_001_03	600a1	Fab-B	153.46	22.86

Figure 5-14 *Ready-made view of the per-path throughput of hosts in NDFC/DCNM SAN Insights*

The root cause of this congestion was the misconfiguration of multipathing on these hosts. The university solved this congestion issue by correcting the multipathing misconfiguration on these hosts. SAN Analytics played a key role in finding the root cause because it was able to show a host's combined throughput as well as the per-path throughput.

Case Study 2 — Summary

Using SAN Analytics, a university was able to find non-uniform traffic patterns which led to congestion due to over-utilization of a few links while other links were under-utilized. The insights given by SAN Analytics pinpointed a problem at the host multipathing layer. They solved the congestion issues by correcting the multipathing misconfiguration that resulted in uniform utilization of the available paths.

Case Study 3 — An Energy Company Eliminated Congestion Issues

An energy company observed high TxWait on their storage-connected switchports, which means that the storage arrays had a slower processing rate as compared to the rate at which the frames were being delivered to them (slow-drain). Thus, the storage ports slowed down the sending of R_RDY primitives leading to zero remaining-Tx-B2B-credits on the connected switchports, which led to high TxWait.

They observed the high TxWait across all their storage ports. No specific storage array stood out. Also, the TxWait spikes were observed throughout the peak business hours. They couldn't pinpoint the high TxWait to any specific hour.

The energy company wanted to know the reason for the high TxWait on their storage-connected switchport. Knowing the root cause of this problem would allow them in finding a solution before it becomes business-impacting.

Background

The energy company uses storage arrays from a few major vendors. Their hosts include almost all kinds of servers (such as blade and rack-mount servers) from all major vendors. Most of their servers are virtualized using a leading hypervisor. They use Cisco MDS switches in their Fibre Channel fabrics. They used the Port-Monitor feature for automatically detecting congestion and generating alerts for TxWait and other counters. However, not many alerts were generated because the measured TxWait by the switchports was lower than the configured thresholds.

The energy company polls the TxWait value from all switchport every 30 seconds using the MDS Traffic Monitoring (MTM) App. Refer to Chapter 3 for more details on it. Cisco NDFC/DCNM Congestion Analysis also provides this information.

Investigation

The energy company needed more details to proceed with the investigation of high TxWait on the storage-connected switchport because the existing data points were not conclusive. There were no specific time patterns or locations to pinpoint. TxWait was observed throughout the business hours randomly across all the storage-connected switchports. Also, some team members suspected issues within storage arrays. However, this possibility was ruled out because high TxWait on the connected switchport was seen from all the storage arrays that had different vendors and different architectures.

The energy company took the following steps for investigating this issue.

1. They enabled SAN Analytics on the storage-connected switchports and allowed the I/O flow metrics to be collected for a week.
2. Next, they correlated TxWait with ECT on the storage ports. The ECT pattern matched with the TxWait pattern, which was expected because high TxWait causes a delay in frame transmission which in turn leads to longer Exchange Completion Times.
3. They also tried matching the pattern of IOPS and throughput but that didn't lead to any new revelations.
4. Next, they correlated TxWait with I/O size. They didn't observe any matching patterns with read I/O size. However, they noticed that the time pattern of the spikes in write I/O size was an exact match with the spikes in TxWait.
5. The spikes in write I/O size could explain the spikes in TxWait on the storage ports. The following is the explanation.
    a. Typically, the write I/O size was in the range of 512 bytes – 64 KB. During the spikes, the write I/O size increased to 1 MB. A 64 KB write I/O operation results in 32 full-size Fibre Channel frames, whereas a 1 MB write I/O operation results in 512 full-size Fibre Channel frames.

b. Most traffic due to a write I/O Operation flows from hosts to storage ports.

c. As evident, the spike in write I/O size caused a burst of frames towards the storage arrays.

d. It was possible that the storage arrays could not process the burst of the frames in a timely manner and used the B2B flow control mechanism to slow down the ingress frame rate. The storage arrays reduced the rate of sending of R_RDY primitives leading to zero remaining-Tx-B2B-credits on the connected switchport, which led to high TxWait.

6. After knowing that the large size write I/O operations were the reason for TxWait on storage-connected switchports, they wanted to resolve this issue. They had to find which hosts (initiators) and possibly which applications used the large-size write I/O operations.

7. They used SAN Analytics to find the write I/O size for every Initiator-Target-LUN (ITL) flow on the storage-connected switchports. This detailed information was enough to find the hosts (initiators) that initiated the large-size write I/O operations.

8. Using SAN Analytics they found that these ITL flows had been active, and they had been doing write I/O operations with typical I/O sizes in the range of 512 bytes – 64 KB. The write I/O size spiked to 1 MB just before these ITL flows stopped showing any I/O activity. In other words, the IOPS and throughput of these ITL flows dropped to zero right after their spike in write I/O size to 1 MB. It was not only an interesting pattern, but it was also commonly seen on all the ITL flows that showed a spike in write I/O size to 1 MB.

9. Next, they located the servers using the initiator value from the ITL flows. Because these servers were virtualized, they used the LUN value from the ITL flow to locate the datastore and a virtual disk on the hypervisor. However, to their disappointment, they

couldn’t find any data store or a virtual disk that was associated with the LUN value.

10. Because the data from SAN Analytics showed them non-zero IOPS for the ITL flows, they were confident that these hosts used the storage volume associated with the LUN. Initially, they thought that they were not seeing the entire information from the hosts. But later it was suspected that probably all these hosts stopped using the LUN. Not using the LUN coincided with the traffic pattern where the ITL flows showed no I/O activity right after a spike in the write I/O size.
11. This led them to suspect any clean-up mechanism before freeing up the disks. When this question was raised to the application and virtualization teams, they found that as per their compliance guidelines, they write explicit (eager) zeros before freeing up the volumes.
12. They found that many applications were short-lived. When such applications are provisioned, they create virtual machines and allocate storage. As soon as the application is shut down, the virtual machine resources are freed. During this process, they wipe all the data followed by writing (eager) zeros on the volumes.
13. Next, they found the disk clean-up process. The hypervisor documentation made it clear that this cleanup process of writing zeros used an I/O size of 1 MB. This value matched with the write I/O size value shown by SAN Analytics on the storage-connected switchport that reported spikes in TxWait. This also explained why no I/O activity was seen right after the write I/O size spiked.
14. They concluded that the disk clean-up process was the root cause of the spike in write I/O size, which in turn caused the spikes in TxWait on the storage-connected switchports. To test it, they followed the same sequence of deploying an application followed by shutting it down. When the virtual machine was freed, they could match the timestamps on the hypervisor with the spike in write I/O size for the corresponding ITL flow on the storage port as reported by SAN Analytics. Connecting these

end-to-end dots between the storage network and the application gave them a clear understanding of the root cause of the problem.

15. However, the problem was yet to be solved. Because of the compliance guidelines, they couldn’t stop the disk cleanup process. Also, changing the default write I/O size of the disk cleanup process was perceived to be risky. Their final approach, which aligned with their compliance guidelines and was agreed upon by all the teams, was not to clean up the virtual machines during peak business hours. They changed their workflow to not free up the virtual machine immediately after the application was shut down. Rather, they delayed the cleanup process until the off-peak (late-night) hours.
16. They verified this change using the TxWait values on switchports and write I/O size as reported by SAN Analytics. They didn’t see spikes in TxWait anymore. They saw spikes in write I/O size but now TxWait didn’t increase probably because the overall load on the storage arrays was low during the off-peak hours, and thus, the spike of the write I/O size for some flows didn’t cause processing delay with the storage arrays.

Figure 5-15 shows TxWait graph in NDFC/DCNM Congestion Analysis. This graph has a granularity of 60 seconds. TxWait of 30 seconds in this graph translates to 50% TxWait.

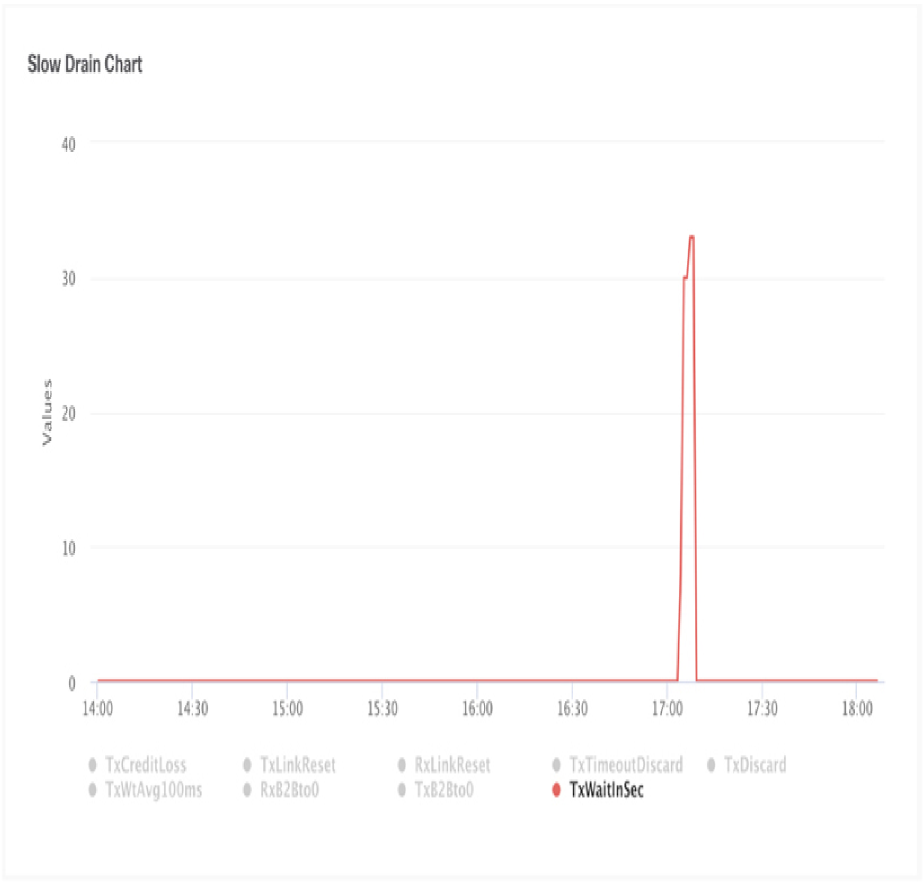
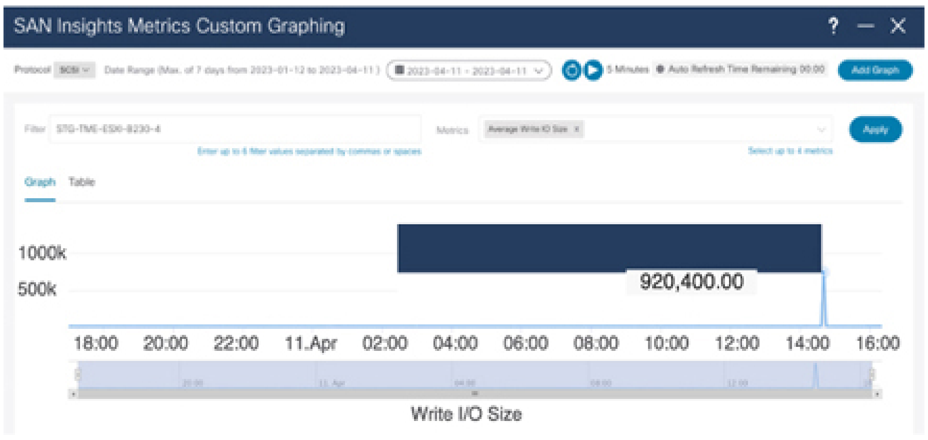


Figure 5-15 *TxWait in NDFC/DCNM Congestion Analysis*

Figure 5-16 shows write I/O size time-series graph in NDFC/DCNM SAN Insights. Notice the sudden spike and timestamp.


SAN Insights Metrics Custom Graphing

Protocol SCSI Date Range (Max. of 7 days from 2023-01-12 to 2023-04-11) 2023-04-11 - 2023-04-11 5 Minutes Auto Refresh Time Remaining 00:00

Filter STG-TME-ESXI-8230-4 Metrics Average Write IO Size

Enter up to 6 filter values separated by commas or spaces

Select up to 4 metrics

Figure 5-16 *Write I/O size spike as shown in NDFC/DCNM SAN Insights*

Figure 5-15 and Figure 5-16 are close representations but they are not sourced from the environment of the energy company.

These are shown here to explain how the spike in TxWait and I/O size can be found and used.

Case Study 3 — Summary

Using SAN Analytics, an energy company was able to find the root cause of high TxWait on the storage-connected switchports and eliminated this congestion issue. First, they found that the spike in TxWait was caused by the spike in write I/O size. Then they found the culprit ITL flows and used the initiator and LUN values to locate the hosts and the virtual machine. Finally, they used the traffic pattern—zero I/O activity just after a spike in write I/O size—to conclude that the disk cleanup process was the root cause of the spike in write I/O size. Based on this conclusion, they solved the problem by delaying the disk cleanup until off-peak hours. This simple step eliminated congestion (TxWait) from their storage-connected switchports which essentially led to better overall storage performance. This performance optimization wasn’t possible without the insights provided by SAN Analytics.

Case Study 4 — A Bank Eliminated Congestion by Infrastructure Optimization

A bank has an edge-core design in a storage network that connects thousands of devices. They often receive a high egress utilization alert from a switchport, which connects to Host-1. The high-utilization condition persists for a few minutes, and it happens a few times every day. While this switchport reports high egress utilization, congestion is seen on the ISL ports as confirmed using TxWait on the ISL ports of the upstream switch.

The bank has a large server farm and many servers are underutilized. They believe that high egress utilization on the switchport that connects to Host-1 can be eliminated by moving some of the workloads to another server. However, instead of randomly moving a workload to another server (hit-and-miss approach), they wanted to make a data-driven decision to make the right change in one attempt because every change was expensive, and the cost multiplies quickly in large environments.

Background

The bank uses storage arrays from a few major vendors. Their hosts deployment includes almost all kinds of servers (such as blade and rack-mount servers). Most of their servers are virtualized using a leading hypervisor. They use Cisco MDS switches in their Fibre Channel fabrics. They enabled automatic monitoring and alerting using the Port-Monitor feature on MDS switches.

Using the high egress utilization (Tx-datarate) alerts, the bank was able to find the following information:

- **When the congestion starts:** Reported by the time stamp of the Port-Monitor alerts.
- **How long the congestion lasts:** Reported by the difference in timestamps of the rising and falling threshold events.
- **Where was the source of congestion:** Port-Monitor alert reports the switch and the switchport that is highly utilized. The FLOGI database (NX-OS command **show flogi database**) shows that the affected switchport connects to Host-1.
- **How bad was the congestion (Congestion Severity):** Reported by Tx-datarate on the switchport that connects Host-1 and TxWait on the ISL ports of the upstream switch. Refer to Chapter 4, "Troubleshooting Congestion in Fibre Channel Fabrics," the section on *Congestion Severities and Levels* for more details.

Investigation

The bank needed more details to make a data-driven change for reducing the high ingress utilization of the Host-1 port, which is the same as the egress utilization of the connected switchport. Although the metrics from the switchport and the alerts from Port-Monitor show high utilization, granular flow level details were not available.

They wanted to move some workload from Host-1 to the other underutilized servers. But they didn't know which workload to move and to which server.

The bank went through the following steps for investigating this issue. For the sake of simplicity, this explanation limits the scope to only four servers (Host-1 – Host-4).

1. The bank enabled SAN Analytics on the host-connected switchports and ran it for a week while the same pattern of over-utilization and congestion repeated. This exercise helped in collecting end-to-end I/O flow metrics.
2. Using SAN Analytics, they found the number of targets (using IT flows) and the number of logical units (storage volume or LUN) (using ITL flows) that each server was doing I/O operations with. Table 5-4 shows their findings.

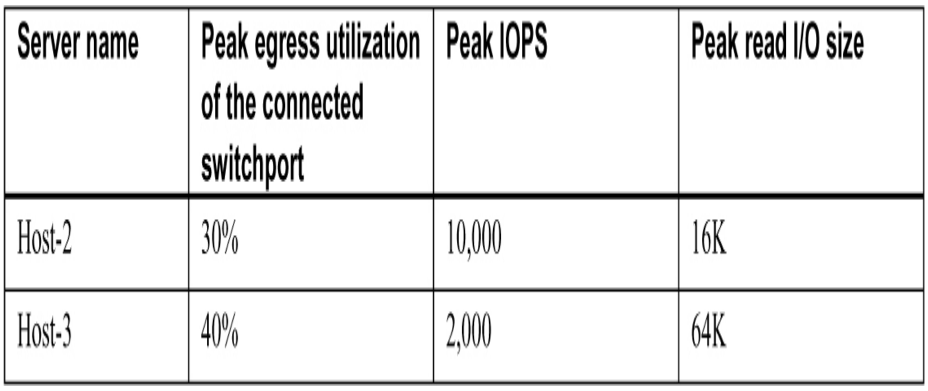
Table 5-4 *Distribution of IT and ITL flows of the servers*

Server name	# IT flows	# ITL flows	# LUN (ITL flows ÷ IT flows)
Host-1	4	40	10
Host-2	4	20	5
Host-3	4	12	3
Host-4	4	80	20

Dividing the number of ITL flows by the number of IT flows gave them the number of LUNs that each server was doing I/O operations with. This information indicated that Host-1 was accessing a higher number of LUNs as compared to Host-2 and Host-3. But Host-4's LUN number was double that of Host-1 yet it didn't cause high utilization as Host-1.

3. Next, they found the throughput for every ITL flow. They focused only on read I/O throughput because most egress traffic on host-connected switchports is the result of read I/O operations. After sorting the ITL flows on the Host-1 connected switchport as per the read I/O throughput they found an ITL flow that had a throughput much higher than the other ITL flows. Also, the pattern of spikes and dips of the read I/O throughput of this ITL flow matched with the egress utilization on the Host-1 connected switchport. Clearly, this ITL flow was the major cause of the high utilization of the switchport and consequently the reason for congestion on the ISL.
4. Further, they wanted to find the workload that was using this ITL flow. Host-1 was virtualized with many virtual machines. They used the LUN value of the ITL flow to find the datastore. Then they found the virtual disk that was created using this datastore and found the virtual machines using that virtual disk. To verify that they located the correct virtual machine, they used the I/O throughput as reported by the operating system of the VM and matched it with the throughput reported by SAN Analytics for the detected ITL flow.
5. After locating the high-throughput virtual machine on Host-1, they wanted to find the best server to which this virtual machine could be moved. Was it Host-2, Host-3, or Host-4?
6. Host-4 was ruled out because it already had a greater number of ITL flows. The possible options were Host-2 with 20 ITL flows, and Host-3 with 12 ITL flows.
7. They found more metrics as reported by SAN Analytics. Table 5-5 shows their findings.

Table 5-5 *I/O flow metrics from SAN Analytics for Host-2 and Host-3*



Server name	Peak egress utilization of the connected switchport	Peak IOPS	Peak read I/O size
Host-2	30%	10,000	16K
Host-3	40%	2,000	64K

It was important to use the peak values for making the right decisions because congestion issues are more severe under peak load. Based on this data, the bank decided to move the high-throughput virtual machine from Host-1 to Host-2 because of its lower utilization and lower read I/O size. Had they made the decision based on the number of ITL counts alone, they would have chosen Host-3, which was not the best choice. By using the insights provided by SAN Analytics, the bank was able to make a better data-driven decision.

8. The bank continued to monitor the servers and repeated these steps for further optimization.

Case Study 4 — Summary

A bank received high egress utilization alerts from one of the host-connected switchports which led to congestion on the ISL. They resolved this issue by moving a high-throughput workload/VM from this host to other under-utilized hosts. To make this change, they used SAN Analytics to find the number of IT flows and ITL flows. Then they found the throughput per flow and sorted the flows as per their throughput to find the culprit flow. Next, they located the virtual machine using the LUN value from the ITL flow and correlated it with the datastore and virtual disk on the hypervisor. Finally, they analyzed the peak throughput, IOPS, and I/O sizes of the other servers to find the best host for the high-throughput workload.

The insights provided by SAN Analytics helped the bank in resolving this issue in one change.

Summary

Storage I/O performance monitoring provides advanced insights into network traffic, which can be used to accurately solve network congestion. Cisco SAN Analytics, which is a network-centric approach for storage I/O performance monitoring, provides end-to-end visibility into I/O operations between virtual machines, initiators, targets, and LUN/Namespace. The per-flow performance metrics from SAN Analytics help in knowing the network traffic patterns. For example, the throughput on a port can be predicted using the I/O size of the read and the write operations. Also, most throughput because of a read I/O operation is in the direction from storage (target) to hosts (initiators), whereas most throughput because of a write I/O operation is in the direction from hosts to storage. Although the read and write I/O data frames make the most of the traffic, these data frames are just a consequence of the read and write I/O command frames which are sent from the hosts to the target. These details not just help in detecting and predicting congestion issues, but they also help in preventing them using mechanisms like Dynamic Ingress Rate Limiting, as explained in Chapter 6.

This chapter explains the practical usage of SAN Analytics via four case studies. The steps explained in these case studies can be reused in other environments for detecting and predicting congestion issues.

Last but not least, storage I/O performance monitoring and SAN Analytics are detailed subjects, which achieve a lot more than detecting and predicting congestion in storage networks. We recommend continuing your education on this topic even outside this book.

References

- Cisco SAN Analytics and SAN Telemetry Streaming Solution Overview: https://www.cisco.com/c/en/us/products/collateral/storage-networking/mds-9700-series-multilayer-directors/solution-overview-c22-740197.html

- Cisco MDS 9000 Series SAN Analytics and SAN Telemetry Streaming Configuration Guide: https://www.cisco.com/c/en/us/td/docs/dcn/mds9000/sw/9x/configuration/san-analytics/cisco-mds-9000-san-analytics-telemetry-streaming-configuration-guide-9x.html
- DCNM SAN Insights - Next Generation Network Visibility - BRKDCN-2271. Cisco Live 2019, San Diego.
- DCNM SAN Insights - Next Generation Network Visibility - BRKDCN-3645. Cisco Live 2022, Las Vegas.
- Detecting, Alerting,, Identifying and Preventing, SAN Congestion - BRKDCN-3241, Cisco Live 2022, Las Vegas.
- SAN Congestion! Understanding, Troubleshooting, Mitigating in a Cisco Fabric - BRKSAN-3446, Cisco Live 2017, Las Vegas.
- ISO/IEC 14165-226:2020 FIBRE CHANNEL SINGLE-BYTE COMMAND CODE SETS MAPPING PROTOCOL – 6 (FC-SB-6)
- https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml
- NVMe over fabrics Specification
- NVM Express Base specification, Revision 2.0. Available from http://www.nvmexpress.org.
- INCITS 514-2014, Information technology – SCSI Block Commands - 3 (SBC-3). Available from http://webstore.ansi.org.
- NVM Express RDMA Transport Specification, Revision 1.0. Available from https://www.nvmexpress.org
- NVM Express TCP Transport Specification, Revision 1.0. Available from https://www.nvmexpress.org.

Chapter 6. Preventing Congestion in Fibre Channel Fabrics

This chapter covers the following topics:

- An overview of reducing or eliminating congestion in storage networks.
- Congestion recovery by disconnecting culprit devices.
- Congestion recovery by early dropping of frames.
- Traffic segregation to dedicated ISLs or virtual links.
- Congestion prevention by using initiator rate limiters on storage arrays.
- Congestion prevention by using Dynamic Ingress Rate Limiting on switches.
- Congestion prevention by notifying end devices.
- Network design considerations.

Network congestion can be prevented by eliminating the causes or the source of congestion. For example, increasing the speed of an over-utilized link eliminates the cause of congestion. Likewise, disconnecting a slow-drain device eliminates the source (and cause) of congestion.

But in production environments, such changes are easier said than done because finding the cause and source of congestion may take a long time. Even once identified, making a change to resolve the problem may not be easy or it may take a long time. For example, a faulty HBA, which causes congestion due to slow-drain, continues to victimize many devices until it is replaced. However, replacing the HBA on a production server requires approvals from multiple teams for a maintenance window, which may take a few days or even weeks. During this duration, it is much better to automatically prevent or recover from congestion until a permanent solution is in place.

When the cause and the source of congestion can be predicted, making a permanent change is the best approach. For example, when the utilization of a host link has been trending higher in the last few months, upgrading the host-link speed well in advance is the best approach. Think of this scenario as an old car that has been showing its check engine light. The congestion caused by this car, which would occur if it broke down in the middle of a highway, can be prevented without waiting for it to actually break down.

In contrast, a healthy car can still cause congestion on the highway because of unpredictable events (e.g., a flat tire). Likewise, in a storage network, an end device may suddenly start causing congestion. Such cases need an automatic approach so that culprits don't victimize other devices.

Other times, although the cause and the source of congestion are known, finding the corrective action takes a long time. For example, when a healthy host causes slow-drain randomly without any obvious reasons, it's worth investigating this host before making a change. Replacing the host entirely without being sure of the corrective action increases the cost and may not resolve the problem. On the other hand, letting this host victimize the other devices while it is under investigation is not the best approach. Automatic isolation or segregation of this under-investigation host buys enough time for its diagnosis and applying a solution.

This chapter explains many approaches for reducing or eliminating congestion in a storage network. Many of these approaches are not a permanent solution and should be used only until

- The source of congestion is determined.

- The cause of congestion is determined.
- The corrective action is known.
- Resources are available to implement a permanent corrective action.

This chapter also presents the trade-offs with various approaches, which help in deciding the best approach as per the constraints of an environment and how long that approach may be used.

Another consideration is to set realistic expectations. Most production storage networks have some level of congestion. Eliminating congestion completely may not be possible and may not be worth the effort. The aim, therefore, should be to lower the severity of congestion so that application performance is acceptable.

Eliminating or Reducing Congestion — An Overview

Recall that a culprit device is any device that causes congestion in a storage network. A victim device is any device that is adversely affected by network congestion. Refer to Chapter 2, "Understanding Congestion in Fibre Channel Fabrics," for more details.

To allow a storage network to automatically eliminate or reduce congestion as it occurs, the following are the high-level approaches:

- **Disconnecting a culprit device**: Disconnecting a culprit device eliminates the source of congestion and therefore results in recovering from congestion. This is typically done by shutting down the link to the culprit device. Many production networks use this approach today. But a disconnected device can't serve its purpose, and hence this is a big-hammer approach.
- **Congestion recovery by early dropping of frames**: Dropping frames frees up the buffers and makes them available for re-use leading to recovery from congestion.

This reduces the spread of congestion allowing the victims to function normally. The effectiveness of this approach depends on factors like when to drop the frames, where to drop the frames, and when to stop dropping the frames after congestion abates. There are two common approaches to dropping the frames. First, a switch can drop a frame if the frame doesn't go out of the egress port within a timeout duration. Second, the switch can proactively monitor slow-drain and drop only those frames that are destined for the slow-drain device. Although both these approaches help in recovering from congestion, dropping frames is a deviation away from the basic principle of lossless networks.

- **Traffic Segregation**: Segregating the traffic going to a culprit device from the other traffic can avoid the effect of congestion on the other devices. This approach is like creating multiple lanes on a highway and enforcing the slow-moving vehicles to stay in their lanes. Similarly, a network link can be virtualized into multiple virtual links, and traffic going to a culprit device can be segregated into a virtual link, which is different from the virtual link used by the traffic going to other devices. Alternatively, traffic can be segregated into dedicated links, which is like creating separate highways altogether.
- **Notifying the end devices about congestion**: End devices (culprits and victims) may not be aware of congestion in the network, and thus, they continue sending more traffic, which may worsen the severity of congestion or extend the duration of congestion. If, however, the end devices can learn about congestion in the network, they can take preventive actions, such as lowering their traffic rate or using alternative paths. But for any preventive action by end devices, first, they must learn about congestion in the network. This can be achieved by congestion notifications from the network switches. Examples of this approach are Explicit Congestion Notification (ECN) in TCP/IP networks, Forward Explicit Congestion Notification (FECN) and Backward Explicit Congestion Notification (BECN) in

frame relay networks, Quantized Congestion Notification (QCN) in Ethernet networks, and Fabric Performance Impact Notifications (FPIN) in Fibre Channel fabrics. Although such approaches are promising, their effectiveness depends on support for the notifications and the coordination between the network switches and the end devices.

- **Limiting the traffic going to a congested device**: The simple idea behind this approach is—Congestion disappears/reduces when traffic disappears/reduces. But obviously, disappearing/reducing the traffic in a network would defy its whole purpose. So, the success of this approach depends on detecting congestion and dynamically rate-limiting the traffic to reduce or eliminate congestion. Some storage arrays can apply per-initiator rate limiters. But the challenge is to decide when and how to enable and disable these rate limiters. An alternative approach can be for the network switches to detect congestion from an end device and dynamically limit the traffic sent by that device. This approach is highly effective because it uses the behavior of traffic in storage networks, which is, an end device receives traffic only when it requests for it, such as initiating a Read I/O operation. Instead of limiting the data frames, rate-limiting is applied to the command frames that solicit the data frames. This innovative approach is extremely successful in preventing congestion in storage networks. It is available on Cisco MDS switches, and it's called Dynamic Ingress Rate Limiting (DIRL).
- **Re-designing the network**: Re-designing a network can eliminate or reduce the severity of congestion. For example, converting large storage fabrics having thousands of devices to smaller islands limits the effect of congestion only within the island.
- **Upgrade**: Many times, upgrading a culprit device is the ultimate solution to congestion. For example, when a link is highly utilized (which may be causing congestion due to over-utilization), the ultimate solution is to increase the link speed or add additional links. Typically, greenfield

environments report fewer congestion issues because of newer and faster servers, storage, and end-to-end high-speed links with no speed mismatches. In contrast, in a brownfield environment, when an end device causes slow-drain and the investigation indicates that the problem is with the older, slower, and low-capacity device, many times (not all), upgrading this device is the real solution.

In addition to understanding the implementation of these approaches, it is important to understand their results, pros, and cons. Although terms like reducing, mitigating, avoiding, or eliminating congestion convey the general ideas involved, all these approaches have some caveats. These caveats are the real reasons why even some of the promising approaches, which are effective in a lab environment, are not commonly adopted in production networks.

The following section categorizes these approaches based on their outcomes. Later, this chapter has dedicated sections explaining each approach, their results, and caveats.

Defining the Outcome of an Approach

Based on their effectiveness, this book categorizes various approaches into the following three categories:

- **Congestion Recovery:** These approaches result in recovering the victim devices from the effect of congestion by severely affecting the culprit devices. Examples of the approaches that result in Congestion Recovery are:
  - Disconnecting a congested/culprit device.
  - Dropping frames based on their age in the switch.
  - Dropping frames based on slow-drain on an edge switchport.
- **Congestion Segregation**: These approaches segregate the traffic destined for a culprit device from other traffic in the network. The behavior of the culprit device remains unchanged, and it may continue to cause congestion, but

it no longer victimizes as many other devices as before segregation. Examples of the approaches that result in Congestion Segregation are:

  - Segregating the traffic to dedicated links.
  - Segregating the traffic to virtual links.

- **Congestion Prevention**: These approaches attempt to eliminate or lower the severity of congestion, typically by reducing the traffic (without dropping frames) destined for the culprit device. When devices stop being culprits, there will not be any victims. Examples of the approaches that result in Congestion Prevention are:

  - Notifying the end-devices about congestion, such as FPIN and Congestion Signals in Fibre Channel.
  - Host I/O rate limiters on storage arrays.
  - Dynamic Ingress Rate Limiting on Cisco MDS switches.

Manual versus Automatic Approaches

In addition to what these approaches achieve, they differ in how they are enabled.

- **Manual approach**: These approaches notify an admin and rely on a manual action to be taken, which often involves delay. Another use of manual intervention is for finding the thresholds for enabling automatic actions, explained next.

- **Automatic approach**: These approaches get activated automatically when an event is detected. For example, when an FC port has zero remaining-Tx-B2B-credits for a timeout threshold, all the frames going out of this port can be dropped automatically. Such automatic actions are the key to creating the first line of defense against the congestion issues until the root cause is investigated and solved by an admin. The real success of automatic actions, however, depends on how well they are implemented, their granularity, configured thresholds, and adaptiveness.

Note that it is not possible for the network to actually fix the root cause of a device causing congestion in the network. Even the best of these approaches, falling under the category of congestion prevention, only attempt to eliminate or lower the severity of congestion caused by a culprit device. These approaches do not guarantee that the problem will go away. Such assurance is possible only after solving the original reason for a device causing congestion in the first place. For example, if a host causes slow-drain at 10 PM every day because of a scheduled job, the real solution depends on altering what that job does. Likewise, if a virtualized host causes congestion because the hosted virtual machines or containers are asking for too much data from the storage arrays that result in the over-utilization of the directly connected link, the real solution may be to increase the link speed, add more links or reduce the number of VMs on this host or similar. In such cases, the congestion prevention approaches may benefit the victim devices but don't really provide a way for the VMs on the culprit to operate at the most optimum rate. Because of these reasons, as mentioned earlier, use the congestion recovery, segregation, and prevention approaches to buy time until the root cause of the problem is investigated and resolved.

Link Capacity

When network links are highly utilized, adding more capacity to them is the only solution. The earlier chapters already explain this topic in detail. Without repeating everything, the following is the summary:

1. An over-utilized edge link is one of the main causes of congestion in Fibre Channel fabrics. Treat all occurrences of high utilization as over-utilization because differentiating between them is extremely difficult in most production environments. Refer to Chapter 1, the section on *Defining Full Utilization vs High Utilization vs*

Over-Utilization and *Monitoring Full Utilization vs High Utilization vs Over-Utilization* for more details.

2. While designing a network, it may be difficult to predict the required throughput on any ISLs. Start with an educated guess and monitor using Tx-datarate, Tx-datarate-burst, and remote monitoring platforms, such as Cisco NDFC. Be ready to add more capacity to the ISLs as their utilization increases.

3. In Fibre Channel environments, the purpose of dual fabrics is that each fabric should be able to function normally even if the second fabric fails in some fashion. Even though this is a rare occurrence, the general rule is that ISLs in a fabric should not run at more than 50% utilization in either direction. It is generally true that even edge links should not run at greater than 50% utilization either. But maintaining less than 50% utilization on ISLs is much more important to handle the load of even the other fabric if that fails.

4. On Cisco MDS switches, use the Port-Monitor counters Tx/Rx-datarate and Tx/Rx-datarate-burst to automatically detect and alert on high-utilization at 10-second and 1-second granularity respectively. ISLs should have much lower alerting thresholds than edge links to ensure one fabric can properly handle the load of both fabrics. Refer to Chapter 3, "Detecting Congestion in Fibre Channel Fabrics," in section P*ort-Monitor* for more details.

5. ISLs must be the fastest links in the fabric. While aggregating physical links in a port-channel, the speed of the member links must be as fast or faster than the fastest edge links in the fabric. Refer to Chapter 2, the section on *An ISL Not Being the Fastest Link* for more details.

6. A faster speed link is better than multiple slower links aggregated together to provide the same capacity. For example, a 64GFC ISL is better than having two links each operating at 32GFC because of the faster serialization on the 64GFC port and no dependency on the load-balancing on the two links.

Congestion Recovery by Disconnecting the Culprit Device

A storage network can recover from congestion due to slow-drain by disconnecting the slow-drain device from the network. This way, the damage caused by it can be avoided while an admin investigates the root cause and applies a permanent solution.

The approach of disconnecting a culprit device is used in many production networks, not just for recovering from congestion but also under the following conditions:

- Disconnect a device if it receives (or its peer receives) too many bit errors, such as CRC-errored frames. Cisco MDS switches, by default, automatically disable a switchport that exceeds bit error rate thresholds. The default threshold is 15 error bursts in a minimum interval of 45 seconds and a maximum of 5 minutes with a sampling interval of 3 seconds Alternatively, Port-Monitor feature allows automatically disabling of switchports based on user-configurable thresholds.
- Disconnect a device if its link flaps too many times in a short duration.
- Disconnect a device if its transceiverreports issues such as excessively low receive power.
- Disconnect a device if it runs into a credit-loss condition resulting in triggering Link Reset Protocol multiple times in a short duration.

Although this approach recovers the victim devices, the culprit device itself is severely affected. As a result, some people call this a big-hammer approach. A device in a disconnected state can’t be monitored. What if the device stops being a culprit after some time or after changing the application workload? This changed behavior can’t be verified until the device is connected again. This, of course, is risky because if the change isn’t effective, it will once again cause congestion.

Considerations for Disconnecting a Culprit

The success of disconnecting a device from the network depends upon the event which caused it to disconnect from the network and the conditions at that time.

Typically, in production Fibre Channel fabrics, hosts access storage arrays via two separate fabrics, known as SAN A and SAN B. A multipathing I/O (MPIO) software ensures that all paths are used as per the defined policy. Disconnecting a host port may signal the multipathing software to move the traffic to the other available ports, which may, in turn, cause side effects.

Consider the following examples.

1. **Example 1**: If one port of a dual-port HBA on a host reports bit errors, disabling it is a viable option assuming that the host multipathing layer moves all the I/O traffic to other healthy paths, therefore preventing CRC corrupted frames. However, if both the ports were 60% utilized, disconnecting one port may result in excessive traffic on the other port triggering a potential congestion problem, which did not exist before.
2. **Example 2**: When the root cause of a slow-drain problem lies at the lower layer on an HBA port (such as physical layer issues like bit errors), the symptoms may be seen only on that port. If, however, the root cause of a slow drain problem lies at a higher layer within the application or the operating system, such as requesting an excessive amount of data by the use of large-size read I/O operations, the symptoms may be seen on both the HBA ports and their connected switchports. If the switches are configured to disconnect the ports based on a threshold, but only one of these ports exceeds the threshold, that port may be disconnected triggering the host multipathing layer to move all the I/O traffic to the other available port. This remaining port now may see slow-drain above the threshold and then be automatically disconnected. The end result is the host is completely disconnected from the

network. This may be the desired result, but it should be understood when implementing this approach.

Does that mean disconnecting a culprit host completely from both fabrics is a better approach? The answer depends on the condition of disconnecting the device and the operational practices in place. Remember that the culprit host might be running a business-critical application. So, disconnecting it entirely may not be the best approach. However, if the application is moved to another healthy host, then disconnecting the host entirely can be considered.

Overall, be aware of the side effects of disconnecting culprit ports from a fabric. Without solving the original problem, this action may just move the problem to the other fabric.

How to Disconnect?

The simplest way to disconnect a device (or port) is to manually shut down the connected switchport. But manual actions are often delayed.

Alternatively, threshold-based policies can automatically shut down switchports without any delay. For example, Port-Monitor feature on Cisco MDS switches can errordisable (shutdown) a port when the counters exceed their configured thresholds. Refer to the *Port Monitor* section in Chapter 3 for more details.

Some users have automated this action. For example, an SNMP trap to a remote server triggers an API that disconnects the culprit devices as per the configured rules and thresholds.

Congestion Recovery by Dropping Frames

During congestion, the time spent by a frame within a switch is much longer than the typical port-to-port switching latency. Instead of allowing the frames to remain within a switch forever, the frames are dropped after a timeout duration or if the egress switchport is not ready to transmit frames due to congestion for an extended duration. Dropping these frames make the buffers available for re-use, and thus, helps in recovering from congestion.

The effectiveness of congestion recovery by dropping frames depends on how, when, and where these frames are dropped.

Dropping Frames Based on Their Age in the Switch

Using this approach for congestion recovery, a switch drops a frame if it does not go out of the egress port within a timeout duration. This timeout is called Frame Discard Timeout Value (F_D_TOV) in the Fibre Channel standards.

Cisco MDS switches call it the congestion-drop timeout, and it is enabled by default at 500 ms. Each frame is timestamped when it is received on the ingress port. A frame that is unable to be transmitted out of the egress port within 500 ms is dropped and the timeout discards (drops) counter is incremented on the egress port. Refer to Chapter 3 the section on *Timeout-discards or Timeout-drops* for detecting these counters on Cisco MDS switches.

Configuring Congestion-drop Timeout on Cisco MDS switches

Example 6-1 shows configuring congestion-drop timeout on Cisco MDS switches. The values can be between 200 ms and 500 ms with 10 ms increments. Although measuring the benefit of a congestion-drop timeout value in a production storage network is difficult, we still recommend lowering it to 200 ms. This is because if a frame is unable to go out of the egress port within 200 ms, the probability of this frame going out of the egress port in an additional 300 ms is very low. In other words, not changing the congestion-drop timeout to 200 ms from the default value of 500 ms delays the recovery action by an additional 300 ms.

Example 6-1 *Changing congestion-drop timeout on Cisco MDS switch*

```
MDS(config)# system timeout congestion-drop 200
logical-type core
MDS(config)# system timeout congestion-drop 200
logical-type edge
```

Details on congestion-drop timeout

The following are important details about congestion-drop timeout on Cisco MDS switches.

- Congestion-drop is enabled by default on all ports for a value of 500 ms, and it cannot be disabled.
- Earlier Cisco MDS NX-OS versions used the terminology of F_Ports and E_Ports. Later versions of NX-OS use logical-type edge ports and core ports instead. This is simpler terminology because the ports on an NPIV parent switch (TF_Ports) that connect to an NPV switch are functionally similar to ISLs (E_Ports). Refer to Cisco MDS Series Interfaces Configuration Guide for more details.
- As Example 6-1 shows, congestion-drop can be specified with different values on logical-type edge and core ports.

A key attribute of congestion-drop timeout is that it doesn't differentiate between the frames that are destined for a culprit device and the frames that are destined for a healthy device (a device that is victimized by the culprit device). In other words, frames destinated for culprits as well as victims are dropped. For example, in Figure 6-1, Host-1, which is a slow-drain device, receives frames from Target-1. Host-2 receives frames from Target-2 and it is a victim of congestion caused by Host-1.

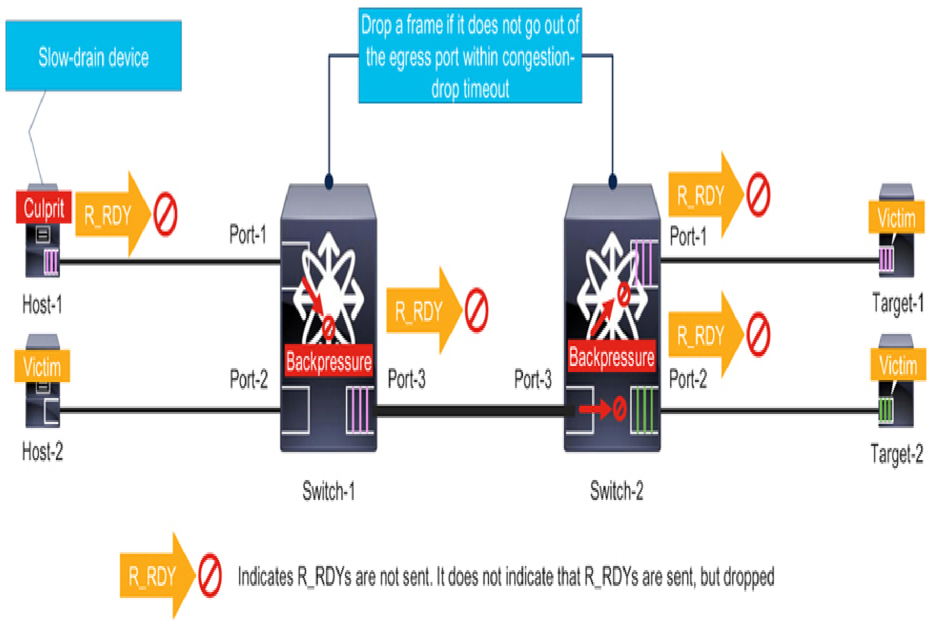


Figure 6-1 *Congestion-drop timeout on Cisco MDS switch*

Switch-1 drops a frame if it doesn't go out of its egress port within the congestion-drop timeout. Dropping a frame makes an ingress buffer available for re-use on Port-3, which results in sending a R_RDY from Switch-1/Port-3 to Switch-2/Port-3.

After receiving a R_RDY, Switch-2 sends one frame to Switch-1. But other frames continue waiting in the switch buffers. Switch-2 drops a frame if it enters Port-1 or Port-2 but doesn't go out of Port-3 in the congestion-drop timeout. The frames that enter Port-1 are destined for the culprit (Host-1), whereas the frames that enter Port-2 are destined for the victim (Host-2). This explains why congestion-drop timeout doesn't differentiate between frames destined for culprits and victims.

This approach of dropping any frame after a timeout (such as congestion-drop timeout) is not a strong recovery approach because of the following reasons.

1. Although dropping the frames frees up buffers and helps the network in recovering from congestion, frame drops seriously affect the end devices because they reinitiate the entire I/O operation even if just one frame of this I/O operation is dropped.
2. The timeout value is in the order of hundreds of milliseconds which is long.

3. A frame must already be in the switch for congestion-drop timeout which means that the frame is already affected by congestion.
4. As explained, congestion-drop timeout doesn’t differentiate between frames destined for culprits and victims.

Because of these reasons, congestion-drop timeout is not enough for congestion recovery by itself. Additional features must complement it.

Dropping the Frames based on Slow-Drain on an Edge Port

Using this approach for congestion recovery, a switch drops frames that are destined for a slow-drain edge device based on a continuous duration of Tx B2B credit unavailability. The device must maintain its slow-drain state continuously for a timeout duration before the switch starts dropping the frames that are destined for it.

This approach is different from the previous approach of dropping frames based on their age in the switch (congestion-drop timeout) because frames going to a healthy or victim device are not dropped. Only the frames that are destined for a slow-drain device are dropped. Also, this approach is more effective in recovering from congestion because it frees up buffers more quickly than the congestion-drop timeout.

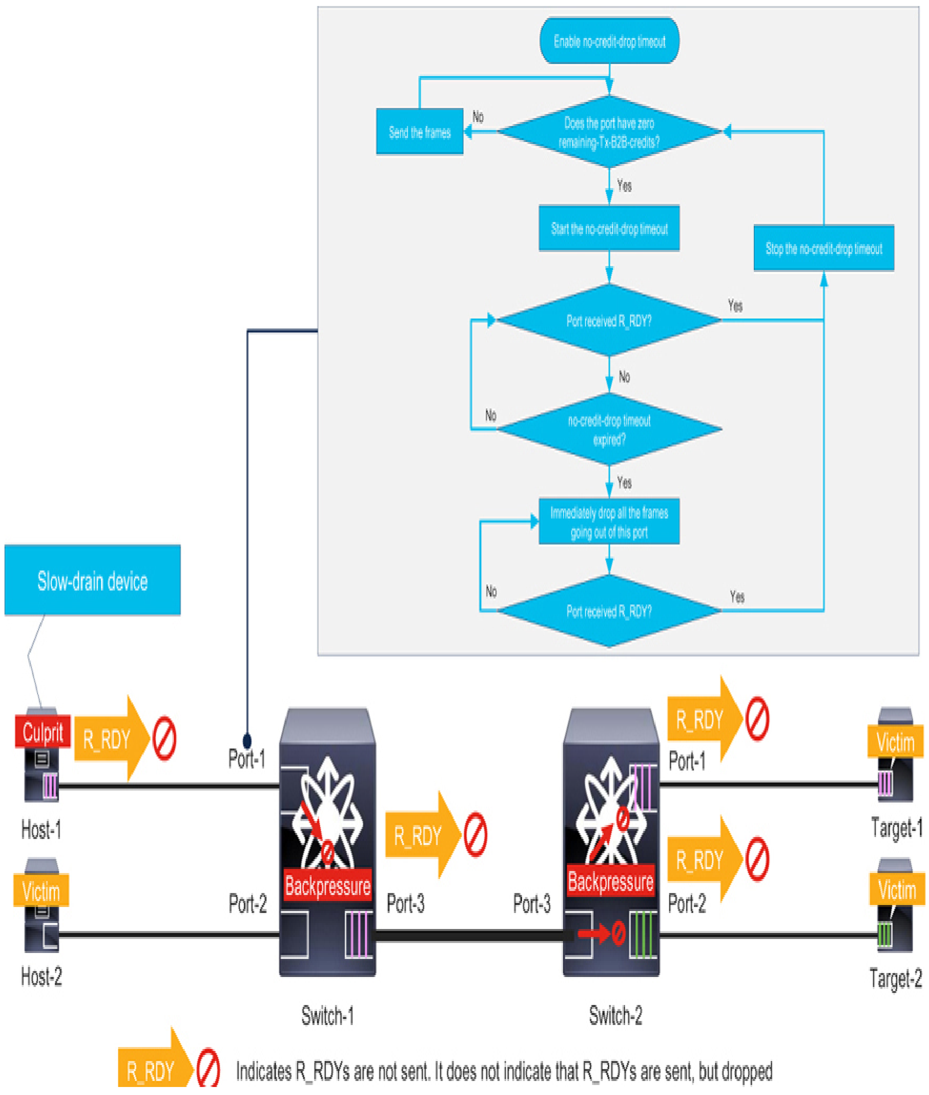
Let’s understand this better using the example of a Cisco MDS switch where this approach is called no-credit-drop timeout. A slow-drain device does not send R_RDY (or sends them at a reduced rate) because it is not ready to receive frames. If the connected switchport has zero remaining-Tx-B2B-credits continuously for a duration longer than the no-credit-drop timeout, the switch takes the following actions:

- Immediately drops all frames in the switch buffers that are waiting to go out of the switchport that is connected to the culprit slow-drain device, and

- While that switchport remains at zero remaining-Tx-B2B-credits, drop any new frames received on other ports on the switch to be sent out of this switchport. These frames are dropped immediately without waiting any significant amount of time in the switch.

Frames dropped because of no-credit-drop timeout results in incrementing the timeout discard (drop) counter on the egress port. Refer to Chapter 3 the section on *Timeout-discards or Timeout-drops* for detecting these counters on Cisco MDS switches.

The slow-drain device sends a R_RDY when it is ready to receive a frame. The connected switchport immediately resumes sending frames upon receiving this R_RDY. If the slow-drain device is not ready again to receive more frames, the culprit switchport may stay at zero remaining-Tx-B2B-credits continuously for the no-credit-drop timeout. If that occurs, the no-credit-drop timeout is triggered again, and frames are once again dropped (Figure 6-2). These actions of frame-drop and resume-sending-the-frames are automatically invoked.



Enable no-credit-drop timeout
Does the port have zero remaining-Tx-B2B-credits?
No
Send the frames
Yes
Start the no-credit-drop timeout
Stop the no-credit-drop timeout
Port received R_RDY?
Yes
No
no-credit-drop timeout expired?
No
Yes
Immediately drop all the frames going out of this port
Port received R_RDY?
No
Yes
Slow-drain device
Culprit
R_RDY
Host-1
Port-1
Victim
Host-2
Port-2
Backpressure
Port-3
R_RDY
Switch-1
Port-3
Backpressure
Switch-2
R_RDY
Port-1
R_RDY
Port-2
Victim
Target-1
Victim
Target-2
R_RDY
Indicates R_RDYs are not sent. It does not indicate that R_RDYs are sent, but dropped

Figure 6-2 *No-credit-drop timeout on Cisco MDS switch*

Figure 6-3 shows the results of no-credit-drop timeout. Switch-1 drops the frames that are destined for the slow-drain device (Host-1). This results in making the buffers available for re-use which eases up the upstream congestion. Switch-1 drops the frames as fast as they are received from Switch-2, and thus, Switch-1/Port-3 doesn’t reduce the rate of sending of R_RDY to Switch-2/Port-3. Consequently, Switch-2 doesn’t observe congestion, and hence, it sends frames to Switch-1 at the same rate as received from Target-1 and Target-2. Frames destined for Host-1 are dropped by Switch-1, whereas other frames reach their destinations (such as Host-2). This explains the congestion recovery by no-credit-drop timeout.

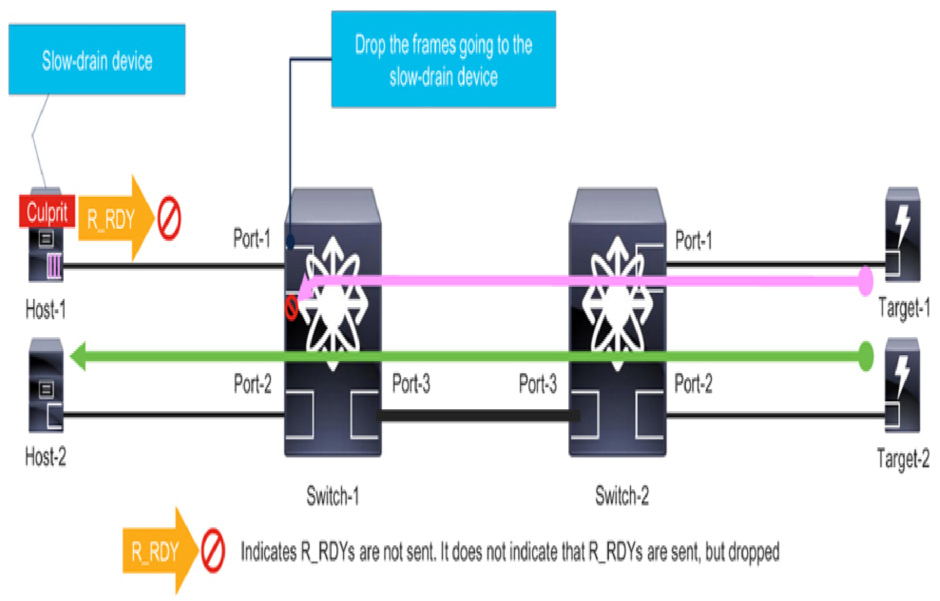

Slow-drain device
Drop the frames going to the slow-drain device
Culprit
R_RDY
Host-1
Port-1
Port-1
Target-1
Host-2
Port-2
Port-3
Port-3
Port-2
Target-2
Switch-1
Switch-2
R_RDY
Indicates R_RDYs are not sent. It does not indicate that R_RDYs are sent, but dropped

Figure 6-3 *Congestion recovery by no-credit-drop timeout*

Enabling no-credit-drop timeout on Cisco MDS Switches

For enabling no-credit-drop timeout on Cisco MDS switches, use the **system timeout no-credit-drop <ms> logical-type edge** command. No-credit-drop can only be enabled for edge ports, not for core ports. This is because enabling it for the core port will result in dropping the frames for many devices that may not be the culprits.

The unit of timeout value is in milliseconds, and it can be configured from 1 ms to 500 ms in 1 ms increments.

Details on no-credit-drop timeout

The following are the important points about no-credit-drop timeout on Cisco MDS switches:

- No-credit-drop is disabled by default.
- When enabled, the no-credit-drop timeout should be lower than the congestion-drop timeout.
- As previously mentioned, the earlier Cisco MDS NX-OS versions used the terminology of F_Ports and E_Ports. The later versions of NX-OS use logical-type edge ports and core ports.

- Cisco MDS switch can report the continuous duration of zero remaining-Tx-B2B-credits using Slowport-monitor. Refer to Chapter 3, section *Tx Credit Unavailability at Milliseconds — Slowport-monitor*, for more details on it. When enabling Slowport-monitor and no-credit-drop timeout together, the Slowport-monitor admin-delay must be lower than the no-credit-drop timeout. Because no-credit-drop timeout drops the frames, if it's configured for a timeout value that's equal to or lower than the Slowport-monitor admin-delay, the frames would be dropped before Slowport-monitor can detect. Therefore, Slowport-monitor threshold will never be reached.
- No-credit-drop does not apply to congestion due to over-utilization of edge links because the edge port has non-zero credits. This type of congestion may cause TxWait on the ISL port of the upstream switch, but no-credit-drop timeout does not apply to core (ISL) ports.
- No-credit-drop timeout can be considered too harsh on the slow-drain device itself. When enabled and the device doesn't return a R_RDY for the configured timeout, for example, 80 ms, all the frames going to this device are dropped until it returns a R_RDY. It is possible that the slow drain device may be running a tier-1 app and may not handle the frame drops gracefully. The counter argument in such scenarios is that if such a device is unable to return R_RDY in 80 ms, it is severely congested, not receiving traffic anyway, and probably affecting many other devices. At least by dropping the frames, you can prevent the adverse effect on the other devices. To make it balanced, you can increase the no-credit-drop timeout value to 100 ms. But don't increase it too much otherwise, it won't bring any benefit.

No-credit-drop timeout in Action

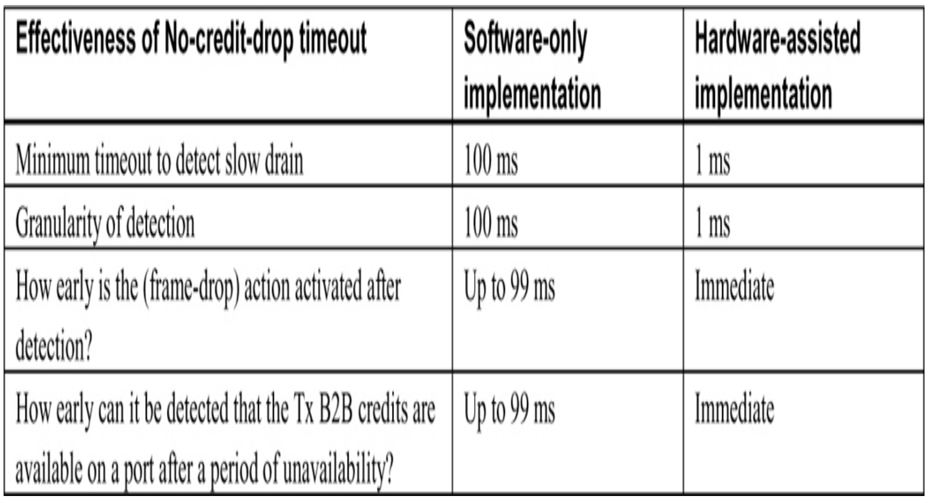
The effectiveness of automatic recovery by no-credit-drop timeout depends on its implementation, such as software-only and hardware-assisted. The following factors define its effectiveness:

- The minimum timeout value at which slow-drain can be detected?
- The granularity of detection.
- How early can the (frame-drop) action activate after detecting congestion?
- How soon can it be detected that the Tx B2B credits are available on a port after a period of unavailability? This determines how soon the Tx traffic can resume on the port.

Cisco MDS 9500 (end of support in 2022) and earlier platforms used a software-only implementation. In 2015, the 16-Gbps MDS platforms and later started using a hardware-assisted implementation. Hence, if you are using the recent MDS switches, you already have the best implementation available to you. This section compares the benefits of this hardware-assisted implementation with the software-based implementations and demonstrates how the same idea can result in better outcomes with different implementations.

Table 6-1 shows the benefits of a hardware-assisted approach. A software-only approach must wait for polling by a process every 100 ms. This results in a detection delay of 100 ms and a frame-drop action delay of up to 99 ms. In contrast, the hardware-assisted approach is natively implemented in the port-ASIC. This results in the detection of congestion as early as 1 ms followed by immediate action.

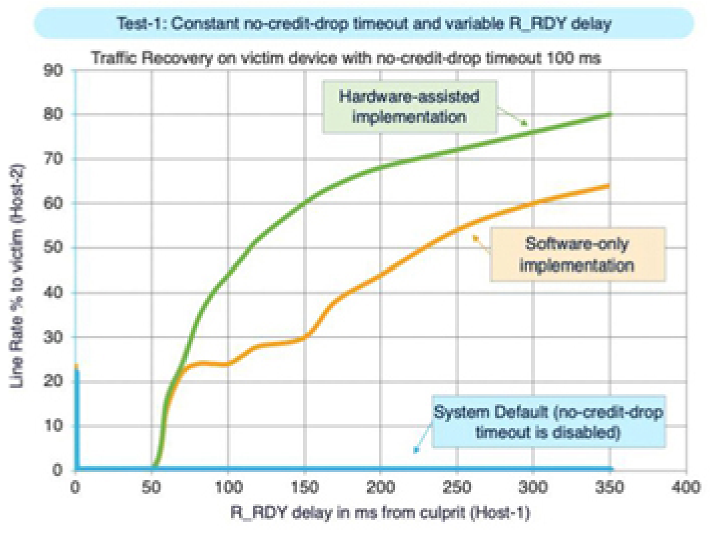
Table 6-1 *Comparing the effectiveness of no-credit-drop timeout*



Effectiveness of No-credit-drop timeout	Software-only implementation	Hardware-assisted implementation
Minimum timeout to detect slow drain	100 ms	1 ms
Granularity of detection	100 ms	1 ms
How early is the (frame-drop) action activated after detection?	Up to 99 ms	Immediate
How early can it be detected that the Tx B2B credits are available on a port after a period of unavailability?	Up to 99 ms	Immediate

Figure 6-4 explains the benefits of the hardware-assisted implementation of no-credit-drop timeout in a test environment as shown in Figure 6-3. Target-1 sends traffic to Host-1 and Target-2 sends traffic to Host-2. All edge links operate at the same speed and the ISL operates at a higher capacity than the edge links so that ISLs are not bottlenecked. Traffic is sent by the targets and received by the hosts at the full capacity of their edge links. Switch-1 and Switch-2 are Cisco MDS switches. To enable hardware-assisted implementation of no-credit-drop timeout, Switch-1 runs the later version of NX-OS that supports it. The models of Switch-1 and Switch-2 remain unchanged in all the tests.

Next, Host-1 simulates a slow-drain device by delaying the sending of R_RDY. The tests measure the traffic received by Host-2 when Host-1 causes congestion.



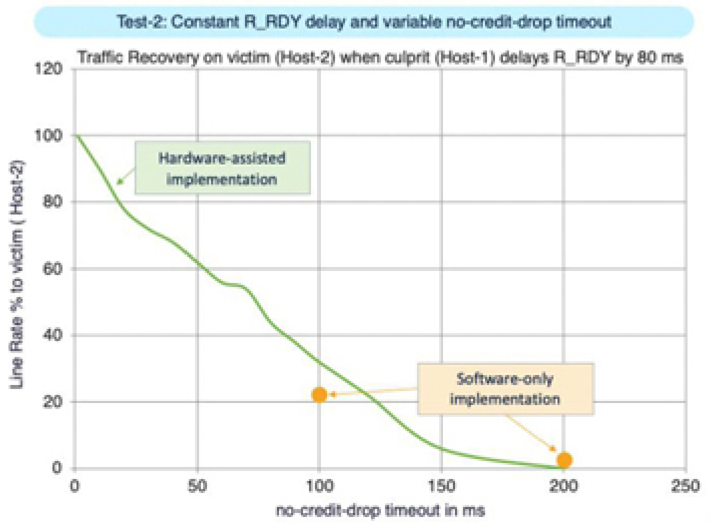


Figure 6-4 *Increased efficiency of no-credit-drop timeout with hardware-assisted implementation.*

The following are the conclusions from Test-1:

- When Host-1 causes congestion and no-credit-drop timeout is not enabled, the traffic received by Host-2 remains at 0%.
- **Software-only implementation**: When no-credit-drop timeout is enabled at 100 ms using a software-only implementation, the traffic received by Host-2 remains at 0% for shorter R_RDY delays from Host-1. But traffic recovers when Host-1's R_RDY delay is greater than 50 ms. Host-2 achieves up to 65% recovery at larger R_RDY delays from Host-1.
- **Hardware-assisted implementation**: Traffic recovers quickly when the R_RDY delay is greater than 50 ms

from Host-1. Host-2 achieves up to 80% recovery at larger R_RDY delays from Host-1. More importantly, the traffic ramp is much better at smaller R_RDY delays.

Test-2 measures the traffic received by Host-2 at a constant R_RDY delay of 80 ms from Host-1 and at various no-credit-drop timeout values on Switch-1. The following are the conclusions from Test-2:

- **Software-only implementation**: The software-only implementation of no-credit-drop timeout recovers the traffic received by Host-2 to 20%. This software-only implementation allows a minimum no-credit-drop timeout value of 100 ms with a granularity of 100 ms which makes the next possible value 200 ms. As shown, 0% traffic recovers with no-credit-drop timeout of 200 ms when Host-1's R_RDY delay is 80 ms.
- **Hardware-assisted implementation**: The hardware-assisted implementation allows a minimum possible value of no-credit-drop timeout to 1 ms with a granularity of 1 ms. As shown, 100% of traffic recovers for Host-2 at smaller no-credit-drop timeout values. The traffic recovery reduces as the value of no-credit-drop timeout increases.

This explains why the success of this approach— Dropping the Frames based on Slow-Drain on an Edge Port—relies heavily on the implementation.

Finding the optimum no-credit-drop timeout Value

A complicating factor for no-credit-drop timeout is choosing its value. The results of Test-2 in Figure 6-4 show that lower timeout values are more effective in recovering from congestion. Although this is true, the timeout value shouldn't be too low in a production storage network because lower values may be very aggressive in dropping frames that may be legitimately delivered when a device causes congestion only momentarily. On the other hand, the timeout values shouldn't be too high, so the action doesn't even get activated.

It is best to use a data-driven approach to find an optimum timeout value to increase the effectiveness of no-credit-drop timeout. As explained in Chapter 3, the section on *Continuous Tx Credit Unavailability in Milliseconds — Slowport-monitor*, Slowport-monitor feature on the Cisco MDS switches reports the continuous duration of zero remaining-Tx-B2B-credits on the ports, which can be used to find a baseline, peak, and variance in the values. Use the sum of baseline and variance to calculate an optimum timeout value for no-credit-drop. A good timeout value should drop the frames only during a sustained and peak congestion event, not when a device is only momentarily busy. For example, most ports on a switch report a peak value of slowport-monitor oper-delay as 60 ms when monitored for a month. Only two ports report this value to 110 ms. In this case, no-credit-drop timeout of 100 ms should be used. This assumes that occurrences of 110 ms of slowport-monitor oper-delay were investigated and found to cause severe congestion spreading in the network resulting in the victimization of many other devices.

Slowport-monitor instead of TxWait should be used for deriving the value of no-credit-drop timeout because no-credit-drop timeout activates only on a continuous duration of zero remaining-Tx-B2B-credits, which is reported by slowport-monitor. In this case, TxWait should not be used because it reports accumulated duration with a granularity of 2.5 microseconds, and hence, it reports continuous as well as non-continuous durations of zero remaining-Tx-B2B-credits.

If you need a generic value to start with, in most Fibre Channel fabrics built using Cisco MDS switches, no-credit-drop timeout of 100 ms should be a safe value. Well-functioning fabrics can lower the timeout to 50 ms. But because no two production environments are the same, giving a blanket recommendation is not possible.

Traffic Segregation

Traffic segregation leads to congestion segregation because traffic for a culprit device no longer interacts with traffic for other end devices.

In lossless networks (such as Fibre Channel), congestion caused by a culprit device can victimize many other devices because they share the same network path. Traffic destined for the culprit may consume all the buffers on the network ports, which doesn't leave enough buffers for other traffic.

This condition is like a single-lane highway where fast cars get slowed down behind slow-moving trucks. Many highways solve this problem by the following steps:

1. Create multiple lanes on the highway.
2. Segregate fast cars and slow trucks into separate lanes.
3. Prevent cars, and especially the slow trucks from changing lanes under congestion.

After making these changes, as Figure 6-5 shows, fast cars don't have to wait behind slow trucks. Note that only the segregation of cars and trucks into separate lanes is not enough. Equally important is to prevent the trucks from changing lanes when their lanes are congested. Only then, even if the lane used by the slow trucks is congested, the fast cars are not victimized.

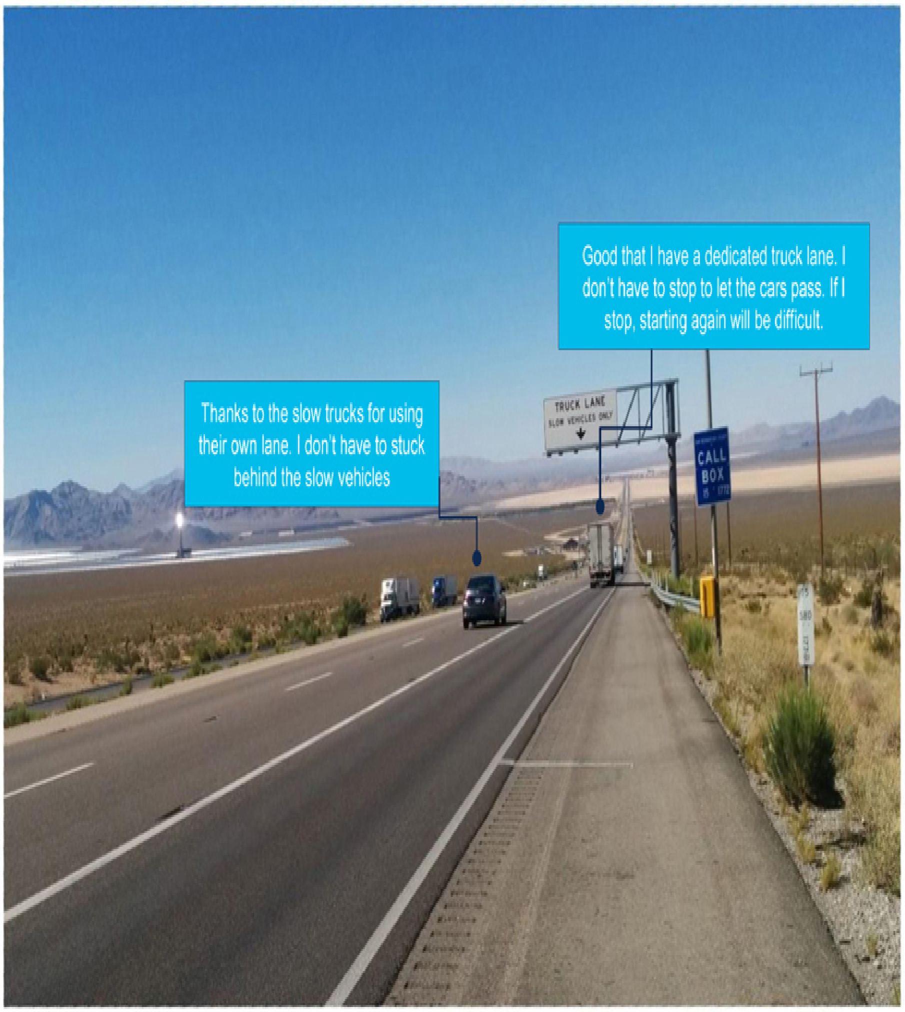

Good that I have a dedicated truck lane. I don't have to stop to let the cars pass. If I stop, starting again will be difficult.
Thanks to the slow trucks for using their own lane. I don't have to stuck behind the slow vehicles
TRUCK LANE
SLOW VEHICLES ONLY
CALL BOX

Figure 6-5 *Traffic and congestion segregation to dedicated lanes*.

Like the approach of multiple lanes on a highway, storage networks can segregate traffic to different virtual links on a physical link, with each virtual link having its dedicated buffers and flow control.

Another traffic segregation mechanism can be to create dedicated roads. For example, in some places, truck routes are dedicated only to slow-moving vehicles (Figure 6-6) or trucks may be prohibited from the city routes. Like this approach, storage networks can segregate traffic into dedicated links.

Figure 6-6 *Traffic and congestion segregation to dedicated routes*

The common idea behind both these approaches (virtual links or dedicated links) is that culprit traffic doesn't consume all the resources or buffers, and thus doesn't victimize the traffic in other lanes or links.

The next consideration for traffic segregation is when to do it. Some highways may enforce traffic segregation during peak hours, whereas other highways may enforce traffic segregation permanently. Likewise, storage networks may segregate the traffic when needed or permanently based on the environment and their use cases.

The approach of changing the traffic assignment to different virtual links or physical links is another important consideration. A permanent assignment is better when the source and cause of congestion are already known, for example, a backup server or an end device that connects at a slower speed and is mostly highly utilized. In contrast, dynamic and automatic assignment of traffic is useful against unexpected congestion events, for example, servers that randomly cause slow-drain.

Overall, the effectiveness of traffic segregation depends on the following factors:

- What to segregate: This is explained further in the next section on *Categorizing Traffic for Segregation*.
- When to segregate: Temporary or permanent.
- How to segregate: Automatic or manual. Typically, temporary traffic segregation is better achieved using an automatic approach, whereas permanent traffic segregation can be done manually.

The rest of this section explains these factors in detail.

Categorizing Traffic for Segregation

Traffic must be categorized before segregating it. For example, trucks as slow traffic, and cars as fast traffic.

In a storage network, when the aim of traffic segregation is congestion segregation, it is best to categorize devices with similar congestion profiles in the same category, for example, the following categories.

- **Application tiering**: Assign tier-1 application traffic to a category, tier-2 application traffic to another category, and so on.
- **Storage protocol tiering (NVMe and SCSI)**: Assign NVMe traffic to a category and SCSI traffic to another category.
- **Storage performance tiering**: Assign all-flash storage traffic to a category, hybrid storage traffic to another category, and a different category for HDD storage traffic.
- **Host speed tiering**: Assign different categories for the traffic from and to the hosts that connect at different speeds, such as 8GFC, 16GFC, 32GFC, and 64GFC.
- **I/O profile tiering:** Assign traffic with different I/O profiles in different categories. For example, large I/O size, medium I/O size, and small I/O size.

The premise of traffic categorization is that congestion may affect the traffic only within the same category. Traffic in a different category remains unaffected and doesn't fall victim to a culprit device, although this is not entirely true as explained later in the section on *Scope of Congestion Segregation Using Virtual Links*.

Traffic Segregation to Dedicated ISLs

As explained in Chapter 2, when using B2B flow control, congestion caused by an end device spreads to the ISLs resulting in the victimization of many other devices that share the same ISLs for receiving traffic. However, other ISLs that are not used by the traffic destined for a culprit device are not affected by this congestion. Hence, segregating traffic to dedicated ISLs also segregates congestion.

This section explains using the Virtual SAN (VSAN) technology on Cisco FC and FCoE switches for segregating traffic to ISLs.

Before going any further, it's important to understand that by default VSANs provide a control-plane separation on ISLs, not data-plane separation. Each VSAN has dedicated fabric services, like its own routing table, name server database, and zone database. Configuration and control-plane changes in one VSAN don't affect the other VSANs. For example, zoning activation in one VSAN is limited to that VSAN only, and other VSANs have no clue about this change. With multiple switches in a fabric, shared ISLs carry traffic for all the VSANs while maintaining control-plane separation by adding a VSAN header (an extended header. Refer to Chapter 2 for Fibre Channel frame format) on the Fibre Channel frames traversing an ISL. By default, all VSANs are allowed on the ISLs, and hence, they don't provide data-plane separation.

If an ISL carries traffic for two VSANs, congestion in one VSAN affects traffic in the second VSAN as well. On the other hand, if the two VSANs have dedicated ISLs, congestion in one does not affect traffic in the other.

Using VSANs for Traffic Segregation on Dedicated ISLs

To achieve traffic segregation using VSANs, the following are necessary steps:

1. Assign devices in separate VSANs.
2. Dedicate physically separate ISLs to VSAN.

To assign an end device to a VSAN, change the VSAN membership of the connected switchport to the desired VSAN ID (Example 6-2). Changing the VSAN ID of a switchport is momentarily disruptive so plan accordingly. Also, an edge switchport can belong to only one VSAN at a time. Techniques of sharing resources from another VSAN are described later in the *Accessing Multiple VSANs* section.

Example 6-2 *Changing VSAN assignment on Cisco MDS*

```
MDS# configure
Enter configuration commands, one per line.  End
with CNTL/Z.
MDS(config)# vsan database
MDS(config-vsan-db)# vsan 20 interface fc3/47
Traffic on fc3/47 may be impacted. Do you want to
continue? (y/n)  [n] y
MDS(config-vsan-db)# end
MDS#
```

Next, allow only the desired VSANs on an ISL. Recall that all VSANs are allowed on the ISLs by default. As Example 6-3 shows, to allow or prevent a specific VSAN on an ISL, use the NX-OS command **switchport trunk allowed vsan**.

Example 6-3 *Changing the allowed VSAN on an ISL on Cisco MDS*

```
MDS# configure
MDS(config)# interface port-channel 3
MDS(config-if)# switchport trunk allowed vsan ?
  <1-4093>   VSAN id range
  add        Give VSAN id range to add to allowed
vsan list
```

```
  all       Add all the vsans to allowed vsan list

MDS(config-if)# switchport trunk allowed vsan 20
MDS(config-if)# end
MDS#
```

Consider the Fibre Channel fabric in Figure 6-7. All the hosts connect to Switch-1, whereas all the storage arrays connect to Switch-2. The switches interconnect via four links operating at 64GFC, which are aggregated in a port-channel providing a collective capacity of 256 Gbps.

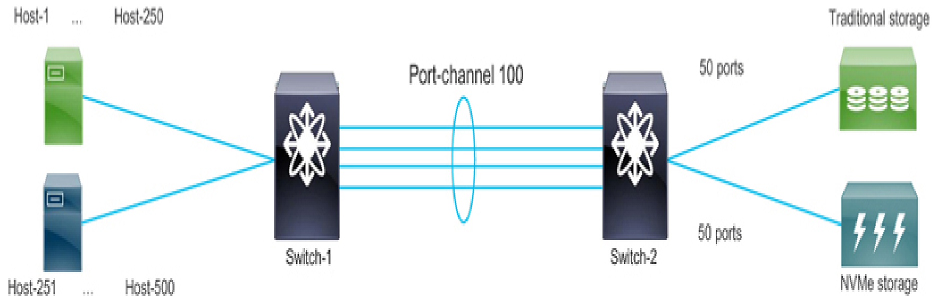


Figure 6-7 *A Fibre Channel fabric shared by NVMe and traditional storage arrays*

This fabric has 250 hosts (Hosts 1 – 250) that access storage from the traditional storage arrays (50 ports). The admins of this environment aim to add newer hosts (Hosts 251 – 500) and NVMe storage arrays (additional 50 ports) to this fabric. The newer hosts will run tier-1 apps and they will use NVMe over Fibre Channel (NVMe/FC) to access the storage arrays.

The admins want to use the same physical fabric to connect the newer devices because the switches have many available ports. At the same time, the admins want to avoid any adverse effects on the newer hosts and NVMe storage arrays when the existing devices are the source of congestion. They decide to achieve this goal by segregating the NVMe/FC and non-NVMe/FC traffic.

In this fabric without any traffic segregation, when an existing host (Hosts 1 – 250) is the source of congestion, it affects the newer hosts (Host 251 – 500). But segregating the traffic of the existing hosts from the newer hosts limits the effect of congestion among existing hosts only.

The following are the steps to segregate traffic in this fabric (Figure 6-8):

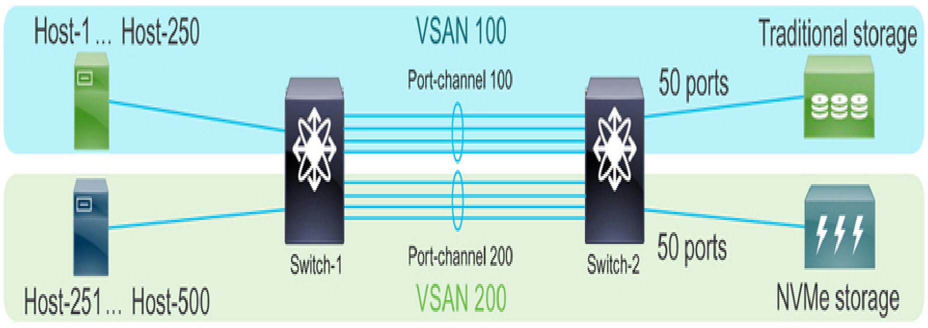


Figure 6-8 *Traffic segregation to dedicated links*

1. Assign Hosts 1-250 and Hosts 251-500 in different VSANs. For example, if Host 1-250 were in VSAN 100, add the new hosts (Host 251-500) in VSAN 200.
2. Leave the VSAN membership of traditional storage arrays unchanged (VSAN 100). However, assign NVMe storage array ports to the same VSAN as hosts 251-500 (VSAN 200).
3. Create a new port-channel 200 interface and add four new ISLs (or more depending upon the need) between Switch-1 and Switch-2.
4. Modify the VSAN allowed list of the two port-channel links. Allow only VSAN 100 to use port-channel 100 and allow only VSAN 200 to use port-channel 200.

After making the changes, as shown in Figure 6-8, when a host among Hosts 1 – 250 in VSAN 100 becomes the source of congestion, its effect is limited to the devices in VSAN 100 only. Hosts 251 – 500 that access NVMe storage arrays in VSAN 200 are not affected.

To summarize the traffic segregation of Figure 6-8:

- **What is segregated**: NVMe/FC and non-NVMe/FC.
- **When it is segregated**: Permanent.
- **How is it segregated:** Manually to dedicated ISLs by VSAN.

Traffic segregation shown in Figure 6-8 achieves the initial goal of congestion segregation. But after operating it for a while, the admins come up with a new requirement. They don't want to use the NVMe storage arrays for data backup because of the high cost. Rather, they prefer taking backups on the traditional storage arrays. The admins have 50 hosts (Host 251 – 300) that they would like to read data from the NVMe storage and write it to the traditional storage. But Host 251 – 500 can't access the traditional storage arrays because they are in different VSANs.

The admins can solve this challenge in two ways. The first way is selectively allowing the traffic between the VSANs using Inter-VSAN Routing (IVR). The second way is adding more HBAs to the hosts and dedicating the HBA ports to the specific VSANs. Both approaches have their trade-offs, as explained next.

Inter-VSAN Routing

Host 251 – 300 can continue to be in VSAN 200 for accessing NVMe storage arrays. To allow these hosts to access the traditional storage in VSAN 100, Inter-VSAN Routing (IVR) can be used as shown in Figure 6-9.

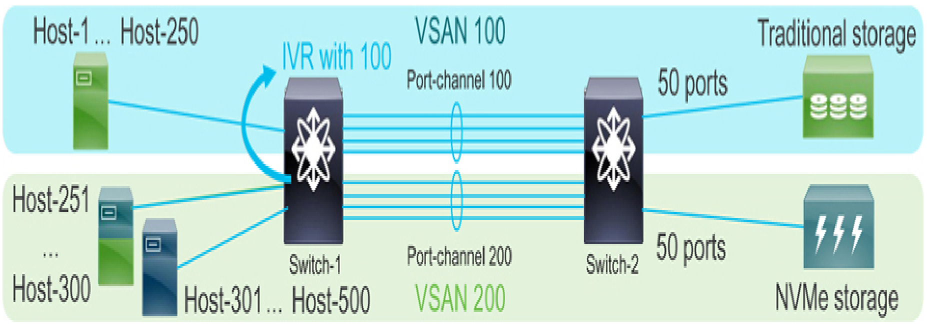


Figure 6-9 *Traffic segregation by selectively allowing traffic between VSANs*

Note that enabling IVR on just one switch is enough.

But the result of enabling IVR on Switch-1 is different from the result of enabling IVR on Switch-2. When IVR is enabled on Switch-1, Hosts 251 – 300 use port-channel 100 (VSAN 100) for accessing traditional storage. As a result:

- When Hosts 1 – 250 are the source of congestion, they victimize Hosts 251 – 300 (indirect victims) but do not affect Hosts 301 – 500.

- When traditional storage arrays are the source of congestion, they victimize Hosts 1 – 250 (direct victim) as well as Hosts 251 – 300 (direct victim), but do not affect Hosts 301 – 500.

When IVR is enabled on Switch-2, Hosts 251 – 300 use port-channel 200 (VSAN 200) for accessing traditional storage. As a result:

- When Hosts 1 – 250 are the source of congestion, they victimize Hosts 251 – 300 (indirect victim) because the traditional storage arrays are the direct victims themselves. But Hosts 301 – 500 are not affected.
- When the traditional storage arrays are the source of congestion, they victimize all three groups of hosts. This is because port-channels 100 as well as 200 are congested. Hosts 1 – 250 and Hosts 251 – 300 become direct victims, whereas Hosts 301 – 500 become same-path victims.

Refer to Chapter 4, "Troubleshooting Congestion in Fibre Channel Fabrics," the section on *Identify the Affected Devices (Victims)* for more details on direct, indirect, and same-path victims.

The key points to remember are:

1. In this topology, enabling IVR on Switch-1 is probably better because when the source of congestion is a traditional storage array, fewer hosts are affected as compared to enabling IVR on Switch-2.
2. Regardless of where IVR is enabled, the net result is that IVR is not a complete solution because congestion spreads to the other VSANs that the culprit device is communicating with.

Add Additional HBAs

Hosts 251 – 300 can continue to access NVMe storage in VSAN 200 using the existing HBA ports. To access the traditional storage, additional HBA ports can be added and assigned to VSAN 100. This approach doesn't require configuring IVR on the MDS switches and avoids the potential of congestion spreading as explained earlier. But it increases the cost due to additional HBAs, cabling, transceivers, etc. Also, this approach is not viable if the switches do not have ports available to connect the additional HBAs.

Considerations for Traffic Segregation to Dedicated ISLs using Multiple VSANs

This section explains additional detail for segregating traffic to dedicated ISLs using VSANs.

Fabric Management Practices

Get ready for a change in your fabric management practices, especially, if you have not been using multiple VSANs. Consider this approach as a new way of operating your environment. The commands on Cisco NX-OS follow a VSAN scope. For example, if you were using a single VSAN earlier, all the zoning changes were made only in that VSAN. After making the changes for VSAN-based traffic segregation, be careful about the VSAN in which zoning changes take place. While adding a new device to VSAN 100, make sure to apply the zoning changes to the same VSAN, not to VSAN 200. Don’t forget to change the automation framework accordingly.

Planning for the changes

Traffic segregation to VSANs and ISLs should be a permanent and long-term approach. Changing the VSAN assignment of an end device is a disruptive change because the device must re-login (FLOGI, PLOGI, PRLI, etc.) via the new VSAN and the zoning configuration must be added to the new VSAN.

The best time of making this change is while refreshing your storage arrays. This is when your fabric is already under a maintenance window, cables are disconnected, newer devices are added, and so on. Well-planned steps can help in minimizing the maintenance window and avoiding downtime.

Spread of Congestion

Segregating the traffic to VSANs, limits the spread of congestion only to the VSAN where the culprit is connected. The devices within that VSAN are still victimized. Consider a fabric with 500 devices. Congestion caused by one culprit device may adversely affect up to 500 devices. In comparison, with VSAN-based traffic segregation, if 250 devices each are assigned to two VSANs, congestion caused by one culprit device may affect up to 250 devices. In other words, VSAN-based traffic segregation doesn't pinpoint the congestion to the culprit device. Because of this reason, this approach is not sufficient by itself, and other congestion prevention approaches should complement it.

Using with the Other Approaches

VSAN-based traffic segregation complements and can co-exist with other approaches.

VSAN technology is at the core of Cisco Fibre Channel switches. It has been around for 20+ years, has been widely used, is stable, and has fewer limitations with other features. Because of these reasons, it is a valid approach to use for a multi-tiered congestion prevention plan. Use it as a long-term architectural approach and complement it with other approaches like DIRL and Congestion Isolation.

VSAN-based traffic segregation is a good candidate for introducing NVMe-oF storage arrays, as Figure 6-8 explains. The NVMe-oF storage arrays offer ultra-high performance which can be fully used without building dedicated fabrics. You can connect the NVMe-oF storage arrays and the associated hosts to the shared fabric and use the VSAN-based traffic segregation approach to keep the separation of control-planes and data-planes.

Traffic Categorization

The number of VSANs and dedicated ISLs per VSAN depend on the traffic categories. Refer to the earlier section on *Categorizing Traffic for Segregation*. Cisco supports up to 80

VSANs in a fabric. Although increasing the number of VSANs may limit the spread of congestion, it increases the management complexity. Find a balanced approach with as few categories as possible.

Case Study 1 — A Bank Avoided Congestion by Traffic Segregation

A bank uses converged infrastructure for hosting its applications. Figure 6-10 shows their design. AFA-1 is the best-in-class storage array that they use for tier-1 apps, whereas AFA-2 is an economical storage array that they use for tier-2 and other apps.

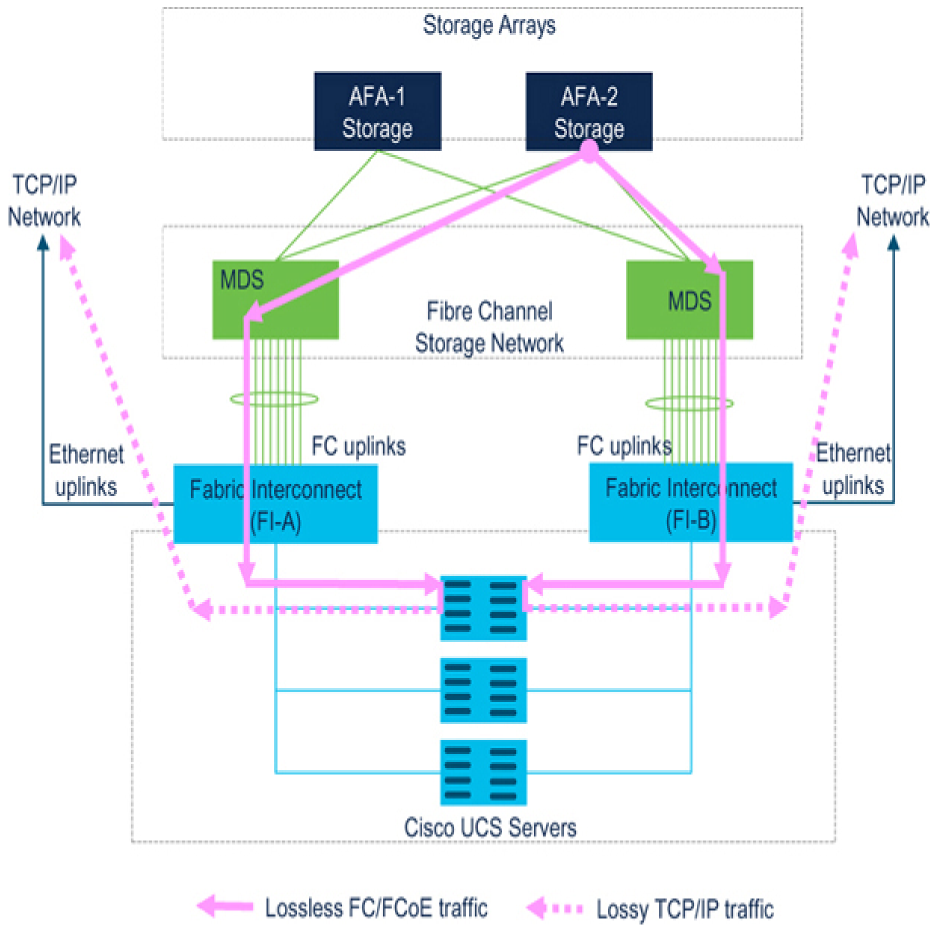


Figure 6-10 *Converged infrastructure design at a banking organization*

Every day at the end of business (5 PM) on the US east coast, the bank starts data backup from the storage arrays. AFA-1 storage array has a native IP-based replication mechanism (not shown in Figure 6-10). The backup traffic from AFA-1 goes to the TCP/IP network directly without passing over the storage network. Tier-2 storage arrays, however, rely on a host-based application for data backup. These hosts read data from AFA-2 via the Fibre Channel fabric and write data to the disaster recovery site via the TCP/IP network. When the backup applications start, the performance of tier-1 apps severely degrades as observed by the employees on the US west coast, who still have 3 hours left before the end of their day.

The bank wanted to avoid the adverse effect of congestion on tier-1 apps caused by the backup app without changing the backup schedule.

Background and Investigation

The following are more details about this environment.

1. AFA-1 connects at 32GFC.
2. AFA-2 connects at 16GFC.
3. The storage network is a single-switch fabric using Cisco MDS switch.
4. Their compute infrastructure is based on Cisco UCS servers. A pair of Fabric Interconnects connects to 4 to 12 chassis, and each chassis has eight servers.
5. The Fabric Interconnects connect to MDS switches with 8 links, each operating at 8GFC, and aggregated in a port-channel providing a combined capacity of 64 Gbps.

The following are the results of their investigation.

1. When the backup application starts, the MDS switch generates high TxWait alerts on the ports that connect to the Fabric Interconnects.
2. The host-based backup applications use large-size Read I/O operations to read data from AFA-2, as verified using SAN Analytics on MDS switches. Refer to Chapter 5, "Solving Congestion by Storage I/O Performance Monitoring," for more details on SAN Analytics.

3. While investigating within the UCS domain, they observe a high rate of PFC Pause frames from the servers that run the backup applications. This increase in the Pause frame rate was probably because the servers get busy reading and writing a large amount of data. Other servers don't show any significant increase in the Pause frame rate. For further details on detecting and troubleshooting congestion in this environment, refer to Chapter 7, "Congestion Management in Ethernet Storage Networks," and Chapter 9, "Congestion Management in Cisco UCS Servers," to know how the bank monitors Pause frames within a Cisco UCS domain. The aim of this case study in this chapter is to explain how traffic was segregated to achieve congestion segregation.

4. Eventually, as Figure 6-11 shows the backpressure applied by the servers that run the backup applications results in congestion, which spreads upstream from Fabric Interconnects to the MDS switches.

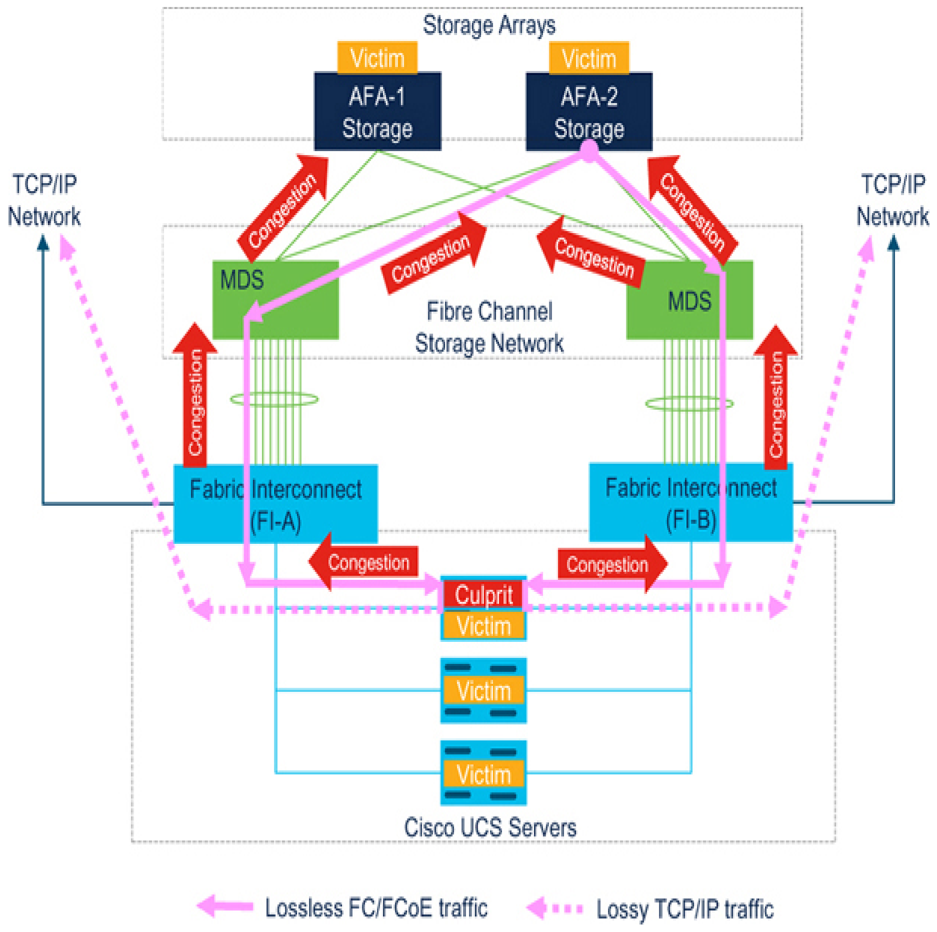
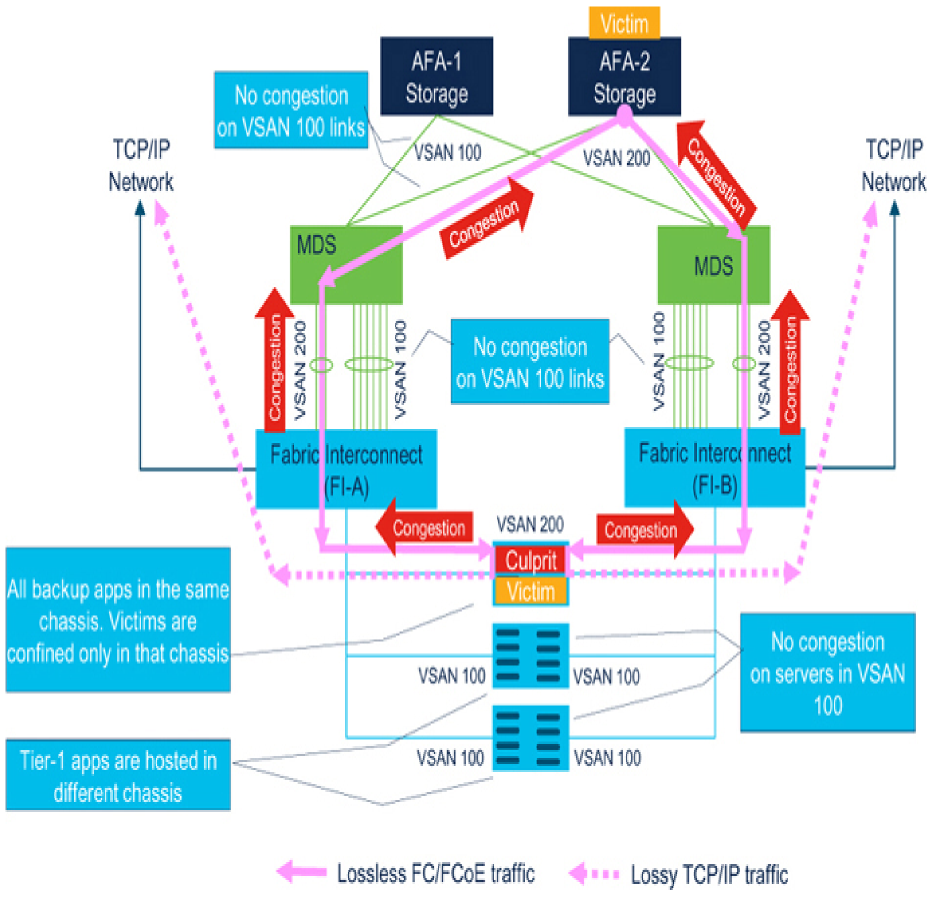


Figure 6-11 *Congestion caused by backup servers affects all other servers*

Overall, the hosts that run the backup applications are the source of congestion, and therefore they are the culprits. This congestion victimizes other servers running tier-1 apps as well as AFA-1 and AFA-2. Since AFA-1 and AFA-2 are victims, all servers communicating with them are victimized as well.

Solution – Traffic Segregation to Dedicated ISLs

The bank made the following changes to segregate tier-1 applications from the adverse effect of congestion caused by the backup apps (Figure 6-12).



Victim
AFA-1 Storage
AFA-2 Storage
No congestion on VSAN 100 links
TCP/IP Network
TCP/IP Network
VSAN 100
VSAN 200
Congestion
Congestion
MDS
MDS
Congestion
VSAN 200
VSAN 100
No congestion on VSAN 100 links
VSAN 100
VSAN 200
Congestion
Fabric Interconnect (FI-A)
Fabric Interconnect (FI-B)
Congestion
VSAN 200
Congestion
Culprit
Victim
All backup apps in the same chassis. Victims are confined only in that chassis
VSAN 100
VSAN 100
No congestion on servers in VSAN 100
Tier-1 apps are hosted in different chassis
VSAN 100
VSAN 100
Lossless FC/FCoE traffic
Lossy TCP/IP traffic

Figure 6-12 *Congestion caused by backup servers affects servers only with the same chassis*

1. Assign AFA-1 and AFA-2 to different VSANs. For example, AFA-1 to VSAN 100 and AFA-2 to VSAN 200.
2. Instead of a port-channel of eight physical links between MDS and Fabric Interconnects, they created two port-

channels. The first port-channel with six physical links and the second port-channel with two physical links. They derived the number of links in a port-channel based on the throughput requirements of the traffic from AFA-1 and AFA-2.

3. Next, they allowed only VSAN 100 on the port-channel with six members and only VSAN 200 on the port-channel with two members.
4. For accessing AFA-1, they assigned the vHBAs on the blade servers to VSAN 100. Likewise, for accessing AFA-2, they assigned the vHBAs to VSAN 200. Refer to Chapter 9 for more details about the use of vHBAs in Cisco UCS architecture.
5. For the servers that needed storage from AFA-1 and AFA-2, they created additional vHBAs and assigned them to VSAN 100 or VSAN 200 accordingly.
6. Within the UCS domain, they hosted the backup applications and tier-1 applications in different chassis. This was an important step because up to eight blade servers within a UCS chassis share the same links to connect to the Fabric Interconnects (server ports). Even if one server applies backpressure (culprit), all eight servers within that chassis become same-path victims.
7. Congestion on the links between Fabric Interconnects and MDS remained confined to the port-channel assigned to VSAN 200. Because congestion didn’t affect the traffic in the port-channel assigned to VSAN 100, none of the tier-1 apps were affected anymore.

Case Study 1 — Summary

A bank avoided the adverse effect of congestion on the tier-1 apps by segregating the backup traffic and tier-1 app traffic to separate links. For accessing storage in multiple VSANs, they created additional vHBAs on the Cisco UCS servers, which was a no-cost change. They limited the culprits and victims within a single UCS chassis (up to 8 servers). The UCS service profiles simplified moving the workloads across the servers. Tier-2 applications in VSAN 200 still experienced congestion during backup periods but this was deemed acceptable since the Tier-1 applications functioned without any impact.

Traffic Segregation Using Virtual Links

Until now this and the earlier chapters explain congestion spreading using B2B flow control, where backpressure exerted by a culprit device slows down all the traffic on a link. This happens because all the B2B credits are shared by all the traffic passing through that link regardless of its destination. This behavior can be changed by segregating the traffic using Virtual Links (VL) on an ISL. Each VL has dedicated B2B credits, runs its own dedicated flow control, and these credits are not shared among the VLs. As a result, congestion in one VL does not affect the other VLs on the same link.

The assignment of traffic to a VL can be automatic or permanent. The automatic assignment is useful when an end device randomly causes slow-drain. In contrast, a permanent assignment is preferential when the source of congestion is already known or if the aim is to permanently segregate the traffic among different categories, such as NVMe versus non-NVMe.

Note that this approach does not completely isolate the spread of congestion to the culprit device. This detail is explained in the later section on *Scope of Congestion Segregation Using Virtual Links*. But first, let’s understand the fundamentals.

Understanding Virtual Links

The B2B flow control mechanism operates at the link level.

In contrast, a Fibre Channel link with VLs uses the following approach (Figure 6-13):

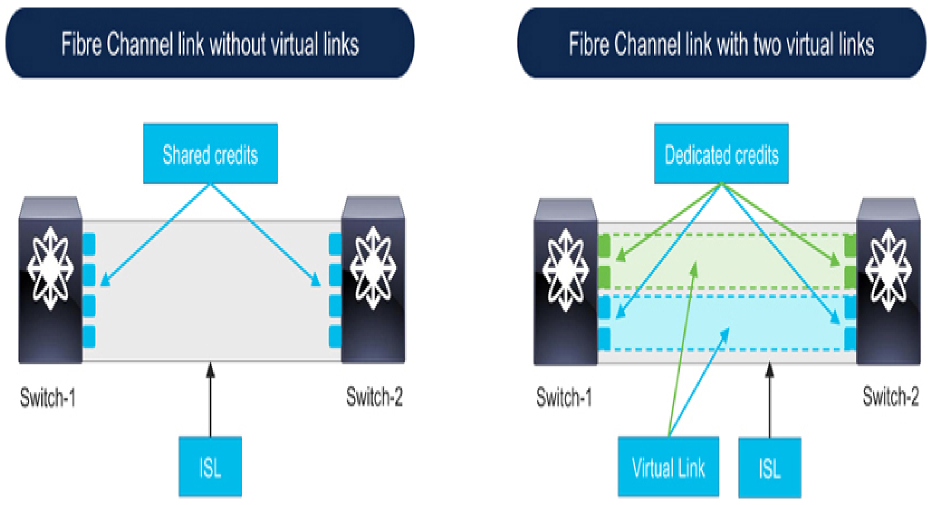

Fibre Channel link without virtual links
Shared credits
Switch-1
Switch-2
ISL
Fibre Channel link with two virtual links
Dedicated credits
Switch-1
Switch-2
Virtual Link
ISL

Figure 6-13 *Understanding virtual links*

1. The B2B credits of an FC port are divided and allocated among the VLs. For example, an FC port without VLs can have 500 B2B credits. The same FC port when using two VLs can allocate 250 B2B credits to each VL.
2. The B2B credits are not shared among the VLs.
3. Each VL has a dedicated flow control mechanism. As a result, an FC port can't send frames in a VL if this VL has zero remaining-Tx-B2B-credits. However, the same port can send frames in another VL if that VL has at least one remaining-Tx-B2B-credit.

The number of available and enabled VLs depends on the capability of a product.

Virtual Links and Virtual Channels

In Fibre Channel, Virtual Links (VL) and Virtual Channels (VC) are similar for all relevant purposes., although there are some terminology and implementation differences. For example, Fibre Channel standards use the VC terminology, whereas VL is a Cisco terminology. Also, VCs use VC_RDY primitive, whereas VLs use ER_RDY primitive. Regardless of these minor differences, the key point to remember is that VLs and VCs have dedicated buffers that cannot be shared.

Virtual Links (VL) and Virtual SAN (VSAN)

Don’t confuse VLs with VSANs in Fibre Channel fabrics because they are orthogonal technologies. VSANs don’t dedicate buffers on ISLs. Although control plane separation is available, traffic in different VSANs continues to share the buffers on the switchports. In contrast, traffic in different VLs does not share buffers among themselves.

Also, don’t confuse Cisco VSAN (upper-case V) with VMware vSAN (lower-case v) technology. Cisco VSAN virtualizes the Fibre Channel fabric, whereas VMware vSAN virtualizes storage.

Flow Control in a Virtual Link

The following are the steps for enabling VLs on a Fibre Channel link:

1. **Agree on using VLs**: Before sending frames in a VL, the neighboring FC port must agree on using VLs and the number of VLs during link initialization. Different devices support a different number of VLs. For example, the 16 Gbps and 32 Gbps Cisco MDS switches support up to four VLs. The number of VLs available on 64 Gbps MDS switches is 8, although only 4 VLs are enabled at the time of this writing.
2. **Communicate the number of B2B credits per VL**: Neighboring FC ports must communicate the number of Rx B2B credits per VL to the neighbor which becomes the neighbor’s Tx B2B credits. For example, an FC port with 500 buffers may agree on creating four VLs, and allocating 15 credits to VL0, 15 credits to VL1, 40 credits to VL2, and 430 credits to VL3. The allocated credits to a VL can’t be changed while the link is operational. Changing this number is possible by making a configuration change on the neighbor. But the change is effective only after the link flaps, which triggers link initialization, communicating the new number of B2B credits to the neighbor. Like a link without VLs, the number of B2B credits per VL is simply communicated (not negotiated upon) to the neighbor.

3. **Assign traffic to the VLs**: FC ports need a way for assigning traffic to a VL. Not just the sender, but the receiver must also be able to detect the VL in which a frame arrives. For example, the sender sends traffic destined for a culprit device in VL1, whereas it sends other traffic in VL3. When a receiver receives a frame, it must be able to differentiate between the traffic in VL1 and VL3. Only then the receiver can return the credit for a specific VL. The exact approach of assigning the traffic to a VL depends on the implementation. For example, Cisco MDS switches encode the VL information in the extended header in the Fibre Channel frames on ISLs. Refer to Chapter 2 for more details on *Fibre Channel Frame Format*.

4. **VL-aware B2B flow control**: Finally, the FC ports must use an enhanced version of the B2B flow control mechanism to achieve a dedicated flow control for each VL without affecting the other VLs. For this purpose, the typical R_RDY primitive is not enough because it doesn't carry the VL number. Hence, instead of R_RDY, there is Enhanced R_RDY (ER_RDY) (or VC_RDY when using the VC terminology). Because of the use of ER_RDY primitives, the links using VLs are also known to be operating in ER_RDY mode.

ER_RDY-based (VL-aware) flow control has the same principles as R_RDY-based flow control. In summary:

1. ER_RDYs flow in the opposite direction of the frames.
2. ER_RDYs do not contribute to the link utilization because they are Fibre Channel primitive signals.
3. ER_RDYs remain local to a link.

Congestion Segregation Using Virtual Links

Figure 6-14 shows a typical Fibre Channel fabric with congestion. Target-1 sends traffic to Host-1, whereas Target-2 sends traffic to Host-2. Host-1 becomes a slow-drain device, and hence it reduces the rate of sending of R_RDYs. The ISL between Switch-1 and Switch-2 does not use VLs, and thus, traffic destined for Host-1 consumes all the B2B credits on the ISL ports without leaving enough credits for traffic from Target-2 to Host-2. Finally, Host-1 becomes the culprit that victimized Host-2, Target-1, and Target-2.

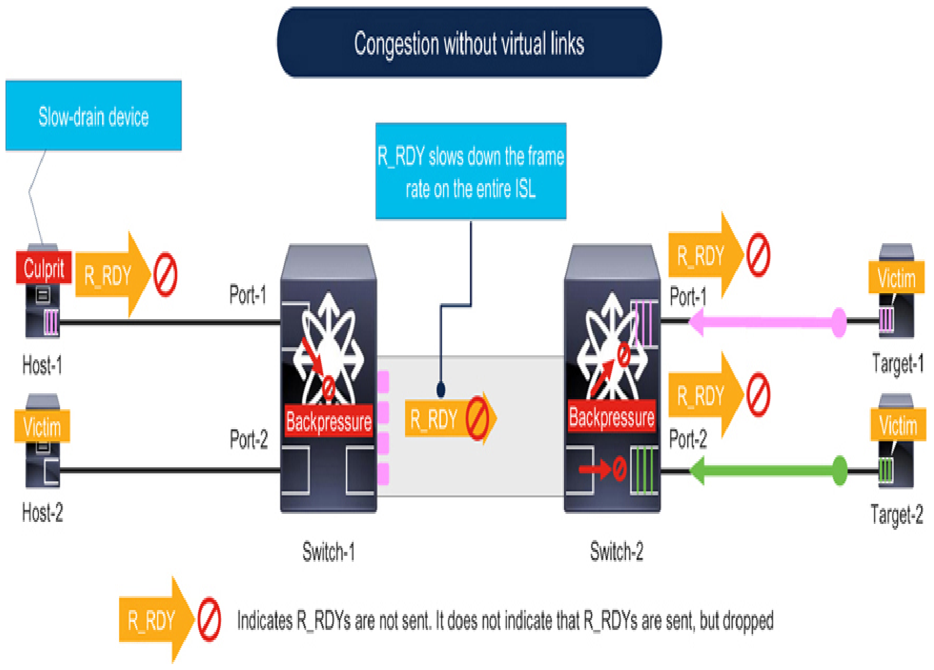

Congestion without virtual links
Slow-drain device
R_RDY slows down the frame rate on the entire ISL
Culprit
R_RDY
Port-1
Host-1
Victim
Port-2
Host-2
Backpressure
R_RDY
Backpressure
R_RDY
Port-1
R_RDY
Port-2
Victim
Target-1
Victim
Target-2
Switch-1
Switch-2
R_RDY
Indicates R_RDYs are not sent. It does not indicate that R_RDYs are sent, but dropped

Figure 6-14 *Congestion without using virtual links*

Now, let's enable VL on the ISL between Switch-1 and Switch-2 (Figure 6-15). There are two VLs and each VL has an equal allocation of B2B credits. Target-1 to Host-1 traffic uses VL0, whereas Target-2 to Host-2 traffic uses VL1. When Host-1 causes congestion due to slow-drain, frames destined for it consume all the B2B credits of VL0. Frames going from Target-2 to Host-2 continue to flow using the B2B credits of VL1. Because the credits are not shared among the VLs, VL1 is not congested, although VL0 is congested. Finally, Host-1 is the culprit, but it victimizes only Target-1. Host-2 and Target-2 are not victims anymore.

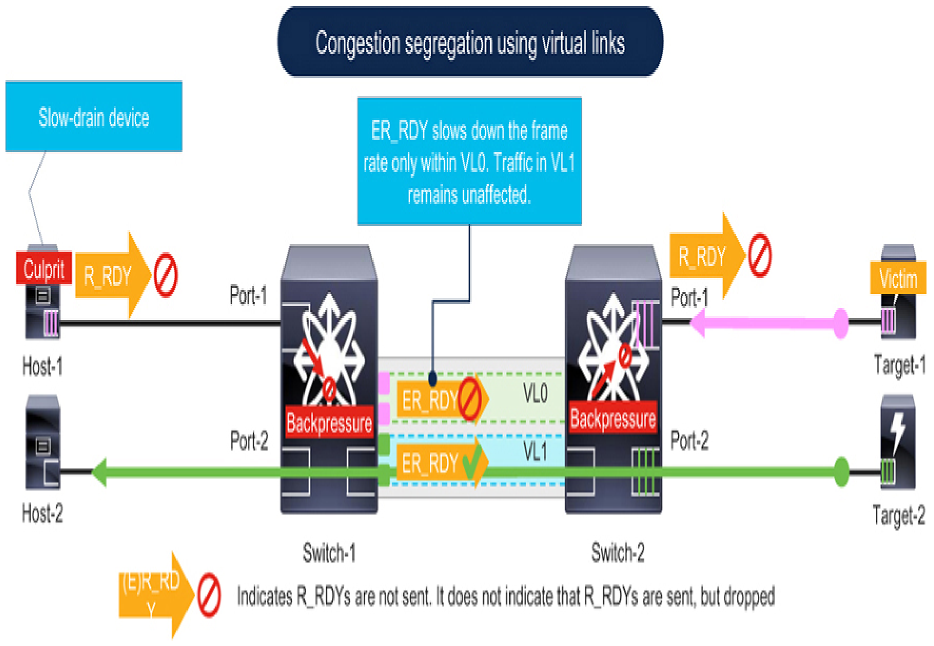

Congestion segregation using virtual links
Slow-drain device
ER_RDY slows down the frame rate only within VL0. Traffic in VL1 remains unaffected.
Culprit
R_RDY
Port-1
Host-1
Backpressure
ER_RDY
VL0
VL1
Port-2
Host-2
R_RDY
Port-1
Victim
Target-1
Port-2
Target-2
Switch-1
Switch-2
(E)R_RDY
Indicates R_RDYs are not sent. It does not indicate that R_RDYs are sent, but dropped

Figure 6-15 *Congestion segregation using virtual links*

Scope of Congestion Segregation Using Virtual Links

A common misunderstanding is that VLs (or VCs) isolate the congestion to a culprit device only. VLs segregate traffic and congestion on the ISLs, but which victim devices really benefit from it depends on the type of victim. As Chapter 4, the section on *Identify the Affected Devices (Victims)* explains, victims can be of the following three types:

1. **Direct Victims**: These are the devices that are in direct communication with the culprit device. In Figure 6-15, Target-1 is a direct victim of Host-1 because it is sending traffic to Host-1.

2. **Indirect Victims**: These are the devices that are communicating with the direct victims. No such device exists in Figure 6-15.

3. **Same-Path Victims**: These are the devices that are utilizing the same network path as the culprit and its direct victims. In Figure 6-15, Host-2 and Target-2 are same-path victims.

Traffic segregation to VLs helps only the same-path victims, not the direct and the indirect victims. Let's understand this further.

Figure 6-15, although helpful for a simple explanation, it is far from reality because production storage networks rarely have one-to-one traffic. If Target-2 also starts sending traffic to Host-1, it would become a direct victim, and that would make Host-2 an indirect victim. When Host-1 traffic is assigned to VL0, VL1 is not congested anymore, but congestion spreads via VL0 to both the targets, and hence, Host-2 does not benefit from traffic segregation to the VLs on the ISL.

Likewise, if Target-2 doesn’t send traffic to Host-1 but Target-1 starts sending traffic to Host-2, then Host-2 becomes an indirect victim. After changing the VL assignment of Host-1 traffic to VL0, Host-2 will remain partially affected because Target-1 continues to be a victim, although Host-2’s traffic from Target-2 will be fine.

Next, consider Figure 6-16 for understanding the scope of congestion segregation using virtual links. Host-1 – Host-4 connect to Switch-1, and Target-1 – Target-4 connect to Switch-2. The traffic pattern is one-to-one from the targets to the hosts, that is, Target-1 sends traffic only to Host-1, Target-2 sends traffic only to Host-2, and so on. Traffic destined for Host-1 flows through VL0 on the ISL, whereas other traffic flows through VL1. When Host-1 causes slow-drain, this congestion spreads to Target-1 via VL0. First, Host-1 reduces the rate of sending of R_RDYs to Switch-1. Then, Switch-1 reduces the rate of sending of ER_RDY to Switch-2 for VL0. Finally, Switch-2 reduces the rate of sending of R_RDYs to Target-1. Because the B2B flow control mechanism applies to the entire Target-1 link, all the egress traffic from Target-1 slows down. Overall, with one-to-one traffic, when one host becomes a culprit, there is just one direct victim and no indirect victim, hence six end devices remain unaffected.

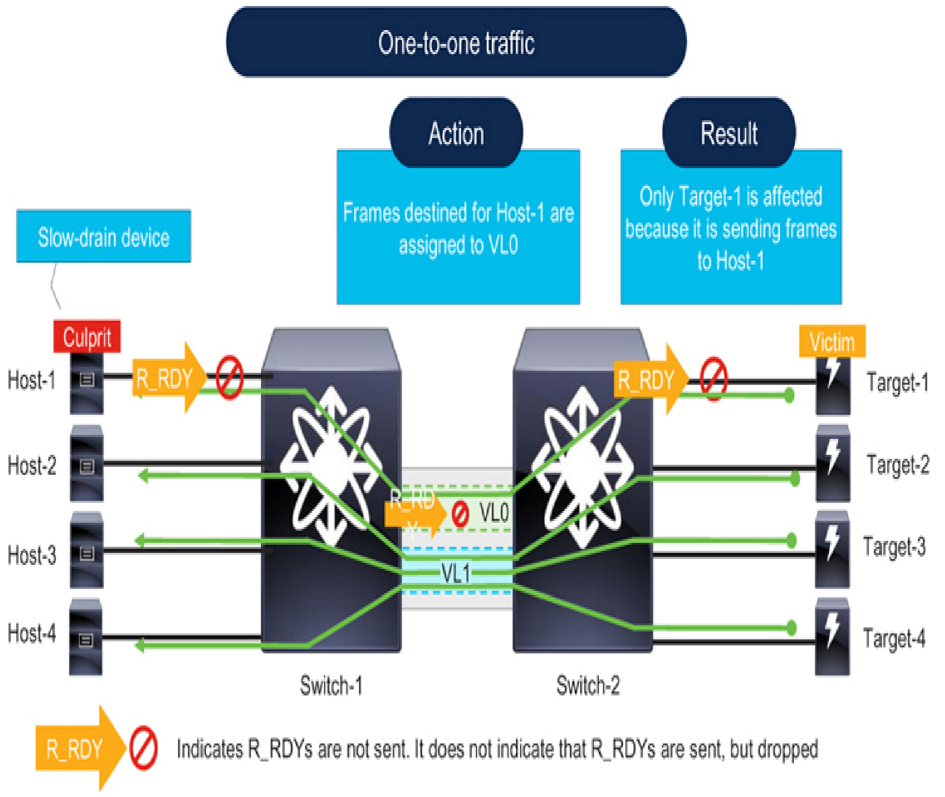

One-to-one traffic
Action
Frames destined for Host-1 are assigned to VL0
Result
Only Target-1 is affected because it is sending frames to Host-1
Slow-drain device
Culprit
Victim
Host-1
Host-2
Host-3
Host-4
R_RDY
VL0
VL1
Target-1
Target-2
Target-3
Target-4
Switch-1
Switch-2
Indicates R_RDYs are not sent. It does not indicate that R_RDYs are sent, but dropped

Figure 6-16 *Congestion segregation using virtual links with one-to-one traffic*

Next, consider another scenario in the same topology when all targets send traffic to all hosts, that is, Target-1 sends traffic to Host-1 – Host-4, Target-2 sends traffic to Host-1 – Host-4, and so on (Figure 6-17). Traffic destined for Host-1 flows through VL0 on ISL, whereas other traffic flows through VL1. Even in this scenario, when Host-1 causes slow-drain, congestion eventually spreads via VL0 on the ISL. However, the difference is that congestion spreads to the ultimate traffic source making Target-1 through Target-4 the direct victims. These direct victims slow down all the traffic regardless of their destination. Consequently, other hosts (Host-2 – Host-4) that receive traffic from Target-1 – Target-4 become indirect victims regardless of which VL is being used for their traffic on the ISL. Overall, with a many-to-many traffic pattern among eight end devices when one host becomes a culprit, it victimizes all the other seven devices either directly or indirectly. Clearly, congestion segregation is not effective in this case.

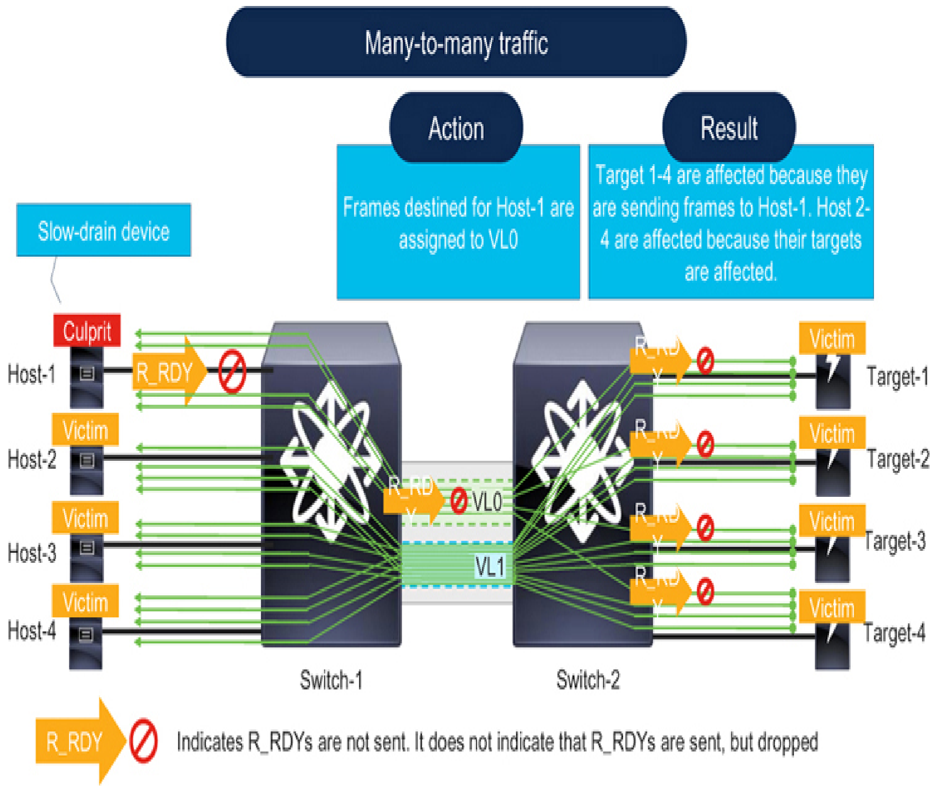

Many-to-many traffic
Action
Frames destined for Host-1 are assigned to VL0
Result
Target 1-4 are affected because they are sending frames to Host-1. Host 2-4 are affected because their targets are affected.
Slow-drain device
Culprit
Host-1
R_RDY
Victim
Host-2
Victim
Host-3
Victim
Host-4
VL0
VL1
Switch-1
Switch-2
Victim
Target-1
Victim
Target-2
Victim
Target-3
Victim
Target-4
R_RDY
Indicates R_RDYs are not sent. It does not indicate that R_RDYs are sent, but dropped

Figure 6-17 *Congestion segregation using virtual links with many-to-many traffic*

Overall, as evident from Figure 6-16 and Figure 6-17, the effectiveness of congestion segregation by segregating traffic in VLs depends upon the traffic pattern in that network and this mechanism does not completely isolate congestion to a culprit device only.

Extending Virtual Links to the End Devices

In Figure 6-17, VLs exist only on the ISL. The links that connect the end devices (Hosts and Targets) do not use VLs and continue using the shared credits for all the traffic on those links.

The effectiveness of congestion segregation can increase if the VLs can extend to end devices also and the end devices can be notified of which specific devices are congested. When Host-1 becomes a slow-drain device, instead of slowing down the traffic on the entire link, it can slow down the traffic only within one VL. Likewise, the targets can slowdown sending the traffic only within one VL that is assigned for Host-1, while they continue sending traffic in other VLs to the rest of the hosts.

Extending VLs between each pair of devices—host and switch, switch and switch, switch and storage array—increases the effectiveness of congestion segregation. However, a device can only have a finite number of VLs, and increasing the number of VLs has side effects, as explained in the later section on *Too many VLs — The Hidden Side Effects*. More importantly, extending VLs to the end devices needs support from the hosts and the storage arrays. Although support on Cisco MDS switches and some vendors are available, most production environments are not ready at the time of this writing, hence VLs to end devices is not explained any further in this book.

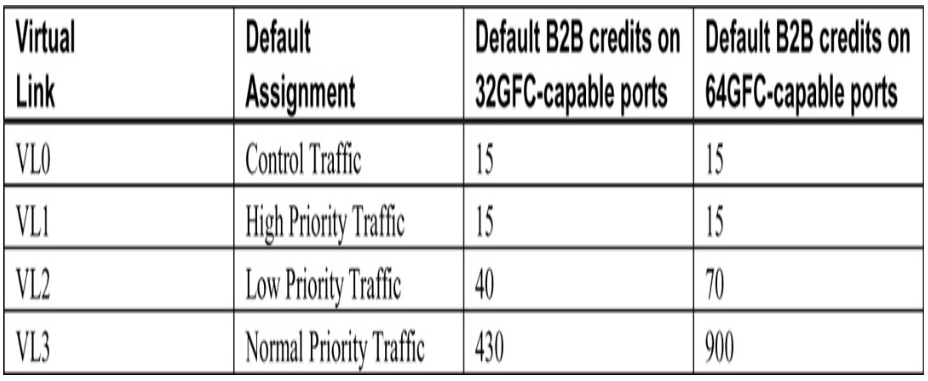
Enabling Virtual Links on ISLs on Cisco MDS Switches

By default, Cisco MDS Switches don't use VLs or the ER_RDY flow control mechanism. ISLs must be manually moved to ER_RDY flow-control mode to create VLs on them. The following are the high-level steps for enabling VLs:

1. Configure ER_RDY flow control on an MDS switch using the NX-OS config command **system fc flow-control er_rdy**. Note that this must be done on switches on both sides of the ISL.
2. To enable ER_RDY flow control on the ISLs, flap the links using NX-OS interface command **shutdown** followed by **no shutdown**. VLs are automatically created after the link flaps. Make this change non-disruptive by flapping one link at a time in a port-channel (link aggregation).

Table 6-2 shows the default number of B2B credits available for virtual links on Cisco MDS switches at the time of this writing.

Table 6-2 *Default credits for virtual links on ISLs on Cisco MDS switches*



Virtual Link	Default Assignment	Default B2B credits on 32GFC-capable ports	Default B2B credits on 64GFC-capable ports
VL0	Control Traffic	15	15
VL1	High Priority Traffic	15	15
VL2	Low Priority Traffic	40	70
VL3	Normal Priority Traffic	430	900

Do not be confused with calling the traffic assignment as low, normal, or high priority. Assuming that a switchport is not transmitting at line-rate and it is not receiving more frames on other ports than can be sent on this port, traffic in all the VLs still gets transmitted as long as VLs have the credits. The practical difference between the VLs is about having dedicated credits, and not the misnomer of traffic priority assigned to them.

The number of B2B credits for each VL is automatically set to the default values. If required, change the number of B2B credits for VLs using NX-OS command **switchport vl-credit**.

Traffic Assignment to Virtual Links

After enabling VLs on an ISL, the next questions are:

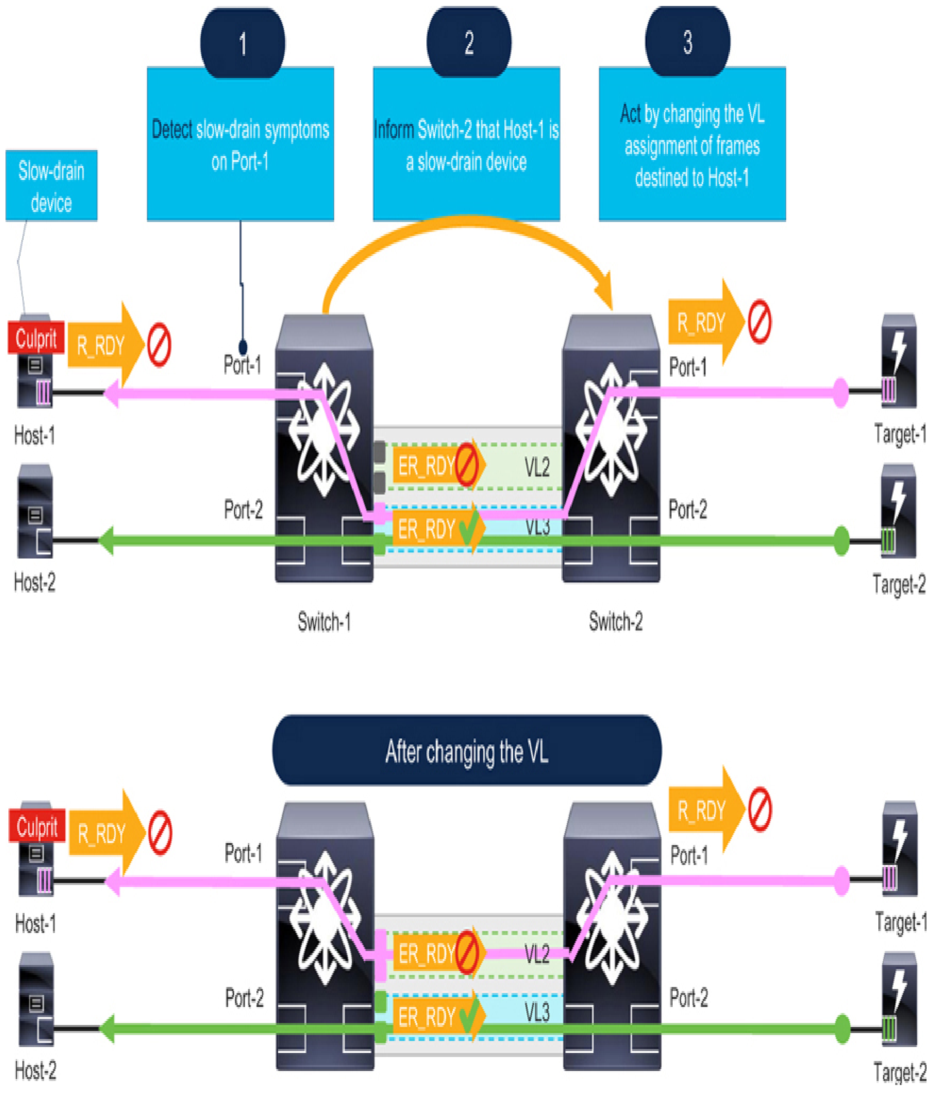
1. **What traffic to assign to a VL**: Traffic destined for a culprit device should be in a separate VL so that congestion is confined within that VL only. Refer to the earlier section on *Categorizing Traffic for Segregation* for more options.
2. **When to change the traffic assignment**: It can be temporary or permanent. Traffic to a slow-drain device can be temporarily assigned to a VL while the investigation continues, and a solution is applied. Alternatively, when the source of congestion is already known, such as backup servers connected at 4GFC, traffic destined for them can be permanently assigned to a separate VL.
3. **How to change the traffic assignment from one VL to another**: It can be automatic or manual. Automatic

approaches, such as using the Congestion Isolation feature on Cisco MDS switches, are better for the temporary assignment of traffic when an end device suddenly causes slow-drain. In contrast, for permanently changing the traffic assignment, a manual approach is better because the destination of traffic is already known and there is no need for automatic action.

Automatic Assignment of Traffic to Virtual Links — Congestion Isolation

The Congestion Isolation feature on Cisco MDS switches segregates congestion by automatically changing the VL assignment of the traffic going to an end device from VL3 to VL2 when that end device is detected to cause slow-drain.

For Congestion Isolation to work, you must explicitly enable it and configure the thresholds in the Port-Monitor feature for detecting Tx B2B credit unavailability on a port. For details on Port-Monitor, refer to Chapter 3, the section on *Port-Monitor on Cisco MDS Switches*. When the thresholds are exceeded (a slow drain device is detected), the following events take place (Figure 6-18):



1
Detect slow-drain symptoms on Port-1
2
Inform Switch-2 that Host-1 is a slow-drain device
3
Act by changing the VL assignment of frames destined to Host-1
Slow-drain device
Culprit
R_RDY
Port-1
Port-2
Host-1
Host-2
ER_RDY
VL2
VL3
Target-1
Target-2
Switch-1
Switch-2
After changing the VL

Figure 6-18 *Congestion Isolation on Cisco MDS switches*

1. **Detect**: The local switch (Switch-1) isolates the port (Port-1) on which it detects slow-drain. Then, it tags the FCIDs (or WWPNs) that are logged in (Host-1) via that switchport (Port-1).
2. **Inform**: The slow device information is shared with all the other switches in the fabric. At the time of this writing, this information is shared using Cisco Fabric Services, and not using the Fabric Performance Impact Notification (FPIN).
3. **Act**: The other switches (Switch-2) stop sending any new frames destined for the slow-drain device (Host-1) in VL3 (assigned for other traffic). Instead, they send the frames to the slow-drain device in VL2 (assigned for traffic destined for slow-drain devices).

Because the frames going to the slow drain device do not use VL3 anymore, its buffers eventually become available for other traffic.

Figure 6-18 shows only VL2 and VL3 for the sake of simplicity, although there may be more VLs on the ISL.

Granularity of Congestion Isolation

The Congestion Isolation feature on Cisco MDS switches is a fabric-wide mechanism involving the following three steps.

1. **Detection of a slow-drain device**: Typically, detecting a slow-drain device takes a minimum of one second.
2. **Informing other switches about the slow-drain device**: The local switch (Switch-1 in Figure 6-18) informs other switches (Switch-2) in the fabric about the identity of the slow-drain device. This step is needed because the local switch (Switch-1) receives frames destined for the slow-drain device on ISL. It can’t change the assignment of those frames to a different VL by itself. The sender (Switch-2) must send the frames in a different VL. But to change the VL assignment the sender must know about the slow drain device, which it learns from the switch (Switch-1) that detects congestion.

   This step involves fabric-wide notification and may take a few seconds depending on how busy the fabric is. In most fabrics, this step should not take longer than 2-3 seconds.
3. **Changing the VL assignment of frames destined for a slow-drain device**: The sender (Switch-2) changes the VL assignment of the new frames. This step should be fairly quick, although the in-flight frames still take the earlier VL.

Overall, from the time a device starts causing slow-drain till the time its traffic is isolated to a different VL on the ISL should be completed within 5 seconds. This is a relatively longer duration to act. Because of this reason, use higher Port-Monitor thresholds when using the congestion isolation feature to be certain that the device is causing severe congestion.

The Real Benefit of Congestion Isolation

As the earlier section on *Granularity of Congestion Isolation* explains, isolating a slow-drain device is a fabric-wide mechanism, which takes a few seconds to take effect. The slow-drain device continues to victimize other devices during this duration.

The real benefit of Congestion Isolation, however, can be seen when the device repeatedly causes slow-drain. When causing slow-drain for the first time, traffic segregation may take a few seconds. But now this device's traffic is already segregated to a different VL. When it causes slow-drain the next time, traffic utilizing other VLs will not be affected leading to instantaneous congestion segregation. To understand this better, refer to the later section on *No-credit-drop Timeout and Congestion Isolation in Action*, which demonstrates the real benefits of the Congestion Isolation feature on Cisco MDS switches.

Automatic De-isolation of a Previously Known Slow-Drain Device

Just like isolating a slow drain device can be automated, de-isolating a device can also be automated. During de-isolation, the traffic assignment changes from VL2 (assigned for traffic destined for slow-drain devices) to VL3 (assigned for all traffic) when a previously known slow-drain device ceases to cause slow-drain as per the configured thresholds.

In reality, automatic congestion de-isolation is not as commonly used because of the following reasons.

- **Operational Practices**: Before de-isolating an end device, some users prefer investigating it thoroughly. For example, they watch this device closely even for minor symptoms of congestion. They also watch it for other symptoms which may not be directly related to congestion, for example, Read and Write I/O response times, SFP TX and RX power, and CRC errors. If suspected, the end devices can be replaced by newer devices. After being sure that the previously identified device is not a slow drain device anymore, only then they allow this device to use VL3 (assigned to all traffic). Because of this operational practice, these users prefer a manual step for de-isolating a device.

- **All the VLs are the same**: There is no difference between using VL2 or VL3 for the traffic destined for a slow drain device. As explained in the earlier section on *Enabling Virtual Links on ISLs on Cisco MDS Switches*, VL2 is not inferior to VL3, although it carries traffic for slow-drain devices. The number of B2B credits may be lower in VL2 but that number should be enough for line-rate performance on the short-distance links within a data center. Because of this reason, some users don't rush this move and prefer taking a manual step instead.
- **Fabric stability**: Moving the traffic across VLs (isolation and de-isolation) is a fabric-wide mechanism, which many users prefer to minimize for increasing fabric stability. They prefer making these changes manually in a maintenance window.

Some users have a large number of slow-drain devices. With automatic isolation, VL2 gets crowded with many devices. Hence, they enable automatic de-isolation to keep the devices under control in VL2. But to make the feature more effective, they watch the logs carefully and if a device makes too many transitions between VL3 and VL2, they take extra measures like replacing the device.

As is evident, using automatic de-isolation depends on your operational practices. Regardless of changing the VL assignment automatically or manually, we strongly recommend investigating and solving the real cause of the congestion of the isolated devices. Without solving the root cause of congestion, isolating the traffic is nothing more than reducing the scope of the congestion.

Enabling Congestion Isolation on Cisco MDS switches

Before using Congestion Isolation on Cisco MDS switches, the ISLs must be using virtual links and ER_RDY flow-control mode. Refer to the earlier section on *Enabling Virtual Links on ISL on Cisco MDS Switches* for more details.

Earlier releases may use different commands. For example, enabling Congestion Isolation in earlier releases requires configuring **feature congestion-isolation** command. Refer to the Cisco MDS Series Interfaces Configuration Guide for details. This section uses the commands that are available in Cisco MDS NX-OS 8.5(1) onwards, which introduces a new process called Fabric Performance Monitoring (FPM).

For enabling automatic isolation of a slow-drain device, first, enable Congestion Isolation feature using NX-OS command **feature fpm**. Then, configure a Port-Monitor policy. In Example 6-4, any edge port on the switch is congestion-isolated and egress traffic to this port is assigned to VL2 on the ISL when the port has zero remaining-Tx-B2B-credits for 400 ms (TxWait rising-threshold of 40%) in 1-second polling interval. Once the port-monitor policy is created it needs to be activated. If there is another Port-Monitor policy of logical-type all or logical-type edge already active, then that policy must be deactivated first.

Example 6-4 *Enabling Congestion Isolation on Cisco MDS switches.*

```
!
feature fpm
!
port-monitor name EdgePorts
  logical-type edge
  no monitor counter all
  monitor counter txwait
  counter txwait poll-interval 1 delta rising-
threshold 40 event 4 falling-
threshold 0 event 4 portguard cong-isolate
!
port-monitor activate EdgePorts
```

Note that the **no monitor counter all** command expands into multiple individual **no monitor counter *counter-name*** commands. This turns off monitoring of the various counters and then **monitor counter txwait** turns on monitoring of just the txwait counter. This is just an example policy and normally other counters should be monitored as well.

At the time of this writing, congestion isolation (portguard action on cong-isolate) is supported for the following Port-Monitoring counters:

- credit-loss-reco
- tx-credit-not-available
- tx-slowport-oper-delay
- txwait

These counters are explained in Chapter 3.

Example 6-5 shows the output of **show port-monitor** command. It shows the configuration for the TxWait counter. This is a wide output, and it has been split into multiple lines.

Example 6-5 *Verifying the Port-Monitor configuration on Cisco MDS switches.*

```
switch# show port-monitor active
----------------------------------------------------
----------------------------


Policy Name   : EdgePorts
Admin status  : Active
Oper status   : Active
Port type     : All Edge Ports
-----------------------------------
| Counter | Threshold  | Interval |
|         |    Type    |  (Secs)  |
|         |            |          |
-----------------------------------
| TXWait  | Delta      | 1        |
-----------------------------------


        |-------------------------------------------
```

```
         |      Warning        |     Thresholds       |
         |--------------------|---------------------|
         | Threshold | Alerts |  Rising   | Falling  |
         |-------------------------------------------
         | none      | n/a    | 40%       | 0%       |
         |-------------------------------------------

                ------------------------------------
-----------------------
                |       Rising/Falling actions
| Congestion-signal |
                |-----------------------------------
--------------|--------
                | Event |   Alerts     |  PortGuard
| Warning | Alarm    |
                ------------------------------------
-----------------------
                | 4     | syslog,rmon  | Cong-
isolate | n/a      | n/a      |
                ------------------------------------
-----------------------
On falling threshold portguard actions FPIN, DIRL,
Cong-Isolate-Recover will
initiate auto recovery of ports.
```

Example 6-6 shows the isolated devices. In this output, the Event type of credit-stall is the same as slow-drain.

Example 6-6 *Isolated devices on Cisco MDS switches*

```
switch# show fpm congested-device database local
VSAN: 1
-----------------------------------
No congested devices found

VSAN: 50
----------------------------------------------------
--------------------------
PWWN                    | FCID      | Event type    |
Detect type | Detect Time
----------------------------------------------------
----------------------------------------------
```

```
21:00:f4:e9:d4:54:ac:f8 | 0x7d0000 | credit-stall |
local-pmon  | Thu Jan 28
05:08:31 2021
```

Port-Monitor generates an SNMP trap and a Syslog message when a device is congestion isolated.

An isolated end device can be automatically de-isolated if **portguard cong-isolate-recover** is used instead of **portguard cong-isolate** in Port-Monitor.

Manual Assignment of Traffic to Virtual Links

Besides automatic action, traffic going to an end device can also be manually assigned to a different virtual link.

Example 6-7 shows how the virtual link assignment of the traffic going to an end device with WWPN 11:11:11:11:11:11:11:11 can be changed from VL3 to VL2. The result of this command is the same as that of the Congestion Isolation feature explained earlier.

Example 6-7 *Manually changing the VL assignment of traffic from VL3 to VL2*

```
switch(config)# fpm congested-device static list
switch(config-congested-dev-static)# member pwwn
11:11:11:11:11:11:11:11 vsan 20
credit-stall
```

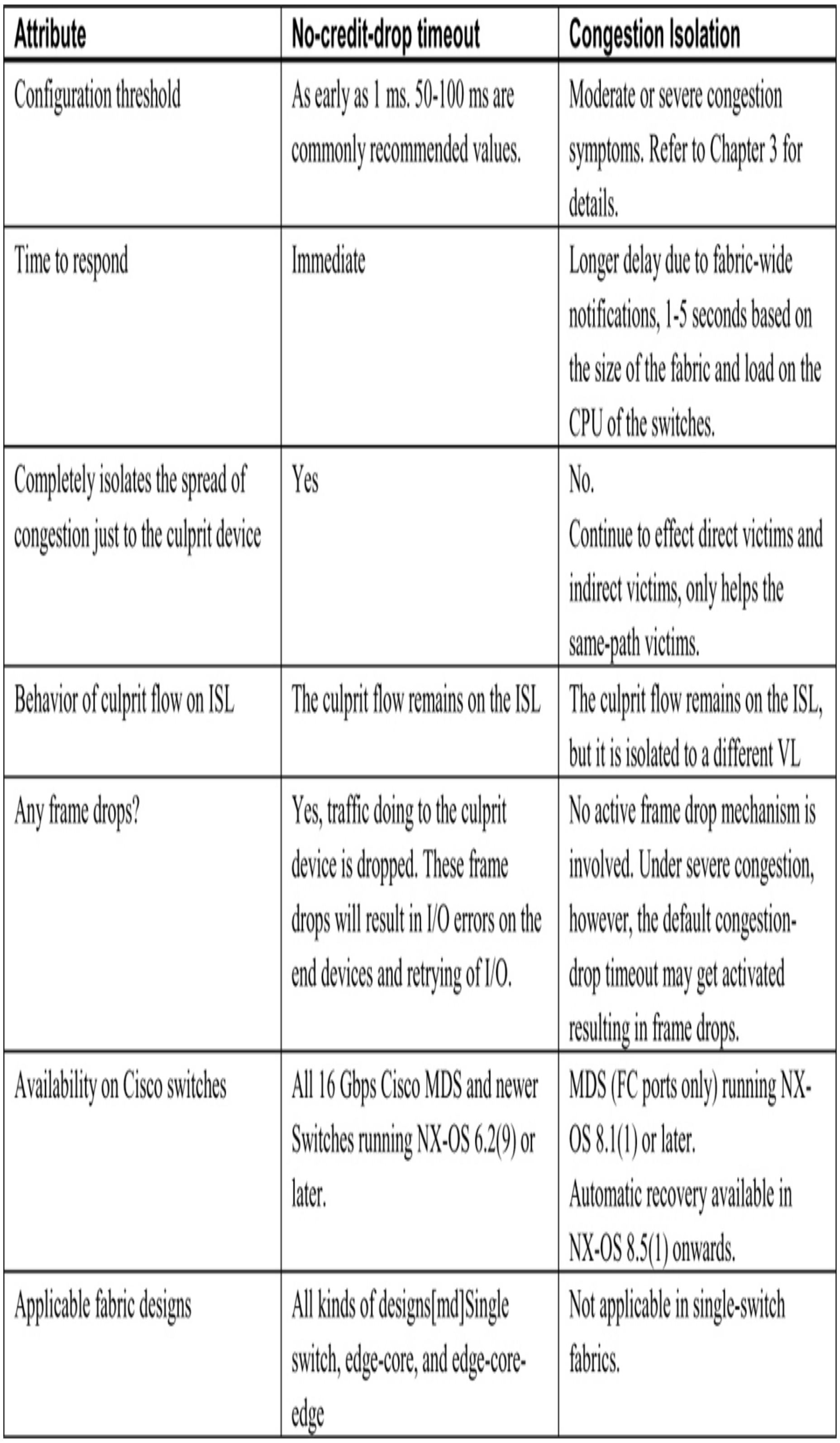
To manually change the traffic assignment of a device back from VL2 to VL3, use the command **fpm congested-device recover pwwn <> vsan <>** on Cisco MDS switches. The result of this command is the same as the automatic de-isolation of a device.

Manual assignment of traffic to a virtual link should be considered a permanent change or at least a change that's not as dynamic as automatic Congestion Isolation.

Comparing no-credit-drop timeout with Congestion Isolation

Table 6-3 compares Congestion Isolation and no-credit-drop timeout side-by-side.

Table 6-3 *Comparison of no-credit-drop timeout with Congestion Isolation*



Attribute	No-credit-drop timeout	Congestion Isolation
Configuration threshold	As early as 1 ms. 50-100 ms are commonly recommended values.	Moderate or severe congestion symptoms. Refer to Chapter 3 for details.
Time to respond	Immediate	Longer delay due to fabric-wide notifications, 1-5 seconds based on the size of the fabric and load on the CPU of the switches.
Completely isolates the spread of congestion just to the culprit device	Yes	No. Continue to effect direct victims and indirect victims, only helps the same-path victims.
Behavior of culprit flow on ISL	The culprit flow remains on the ISL	The culprit flow remains on the ISL, but it is isolated to a different VL
Any frame drops?	Yes, traffic doing to the culprit device is dropped. These frame drops will result in I/O errors on the end devices and retrying of I/O.	No active frame drop mechanism is involved. Under severe congestion, however, the default congestion-drop timeout may get activated resulting in frame drops.
Availability on Cisco switches	All 16 Gbps Cisco MDS and newer Switches running NX-OS 6.2(9) or later.	MDS (FC ports only) running NX-OS 8.1(1) or later. Automatic recovery available in NX-OS 8.5(1) onwards.
Applicable fabric designs	All kinds of designs[md]Single switch, edge-core, and edge-core-edge	Not applicable in single-switch fabrics.

Table 6-3 compares the two approaches but remember that they complement each other when used together in production environments.

No-credit-drop Timeout and Congestion Isolation in Action

This section shows no-credit-drop timeout and Congestion Isolation in action by demonstrating the following scenarios.

1. Effect of a slow-drain (culprit) host on a victim host.
2. Recovery of the victim host after enabling no-credit-drop timeout.
3. Recovery of the victim host after enabling Congestion Isolation.
4. Recovery of victim host after enabling no-credit-drop timeout and Congestion Isolation together.

Test Setup

Figure 6-19 shows the test setup. The slow drain culprit host (Host-1) is simulated using a Xgig traffic generator, which can delay the sending of R_RDY primitives. The victim host (Host-2) is a Cisco UCS C240 M4 rack server running VMware ESXi 6.5 as a hypervisor and CentOS 7.0 as a VM. FIO generates traffic from the Linux/CentOS VM. Target-1 is another Xgig port. Target-2 is a Dell EMC Unity All Flash Array. Switch-1 and Switch-2 are Cisco MDS 9710 Switches running NX-OS 8.1(1). All links operate at 16GFC. The switches connect via multiple links aggregated in a port-channel.

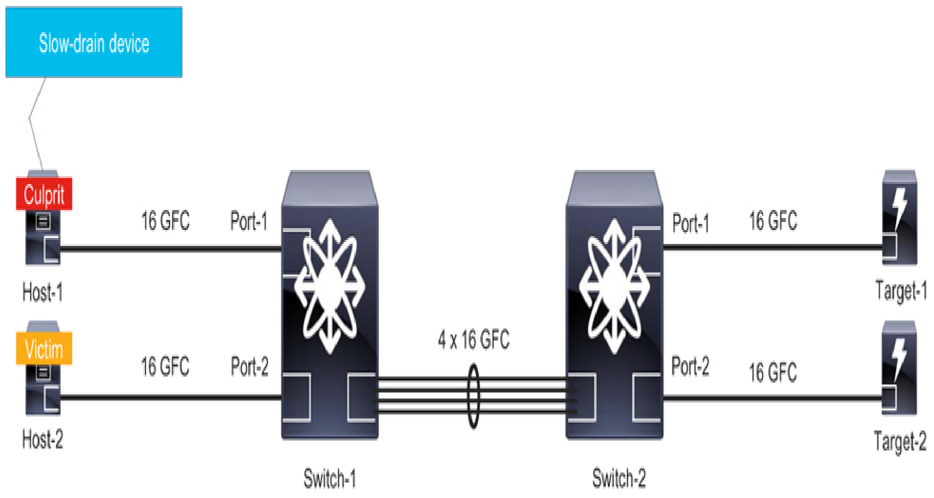


Figure 6-19 *Test setup for demonstrating no-credit-drop timeout and Congestion Isolation.*

Target-1 sends traffic to Host-1 at the data rate of 16GFC. Likewise, Target-2 sends traffic to Host-2 at the data rate of 16GFC.

The effect of slow-drain, and the results after enabling the features on Cisco MDS Switches are measured on the Linux VM on Host-2. The aim is to measure traffic recovery on the victim host (Host-2) while a slow drain device (Host-1) exists in the fabric. The effect on the culprit device itself is outside the scope of this testing.

Effect of Slow-drain on the Victim Host (Host-2)

Figure 6-20 shows the throughput seen by the FIO tool on the Linux VM on Host-2. Before Host-1 causes slow-drain, Host-1 receives 1600 MB/s, which is the maximum data rate of a 16GFC link. When Host-1 starts causing slow-drain by delaying R_RDYs by 80 ms, traffic on Host-2 drops from 1600 MB/s to less than 30 MB/s (98% drop).

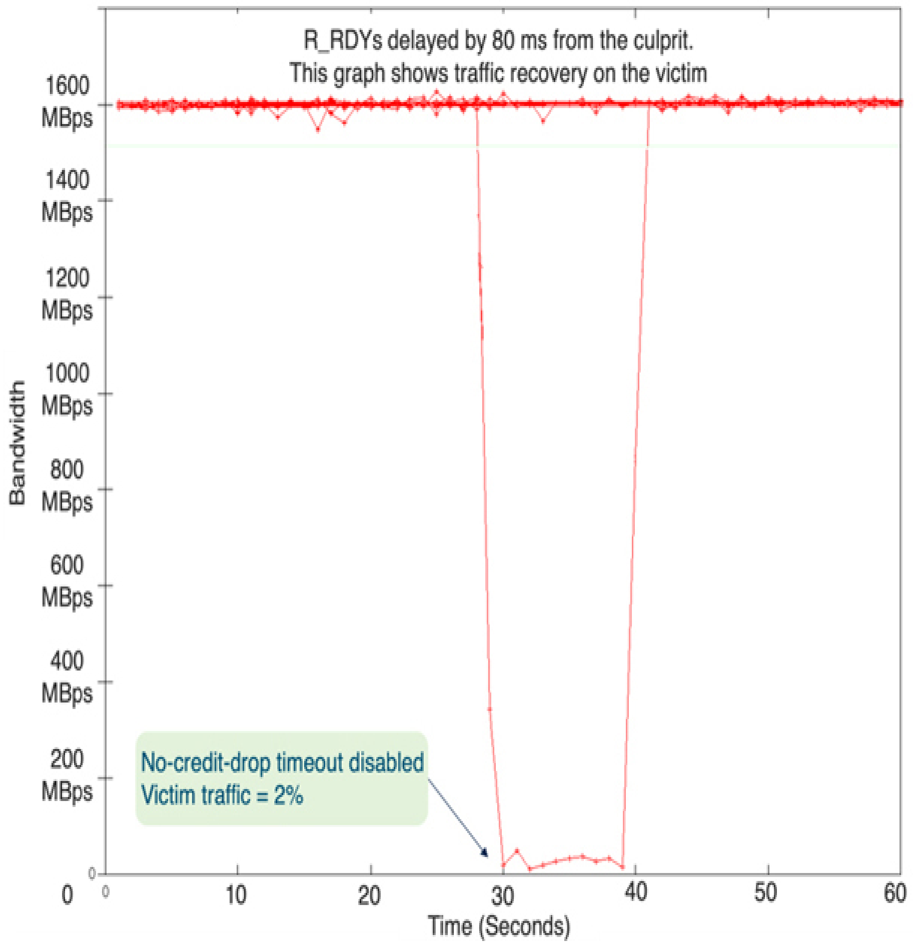


Figure 6-20 *Throughput drop on Host-2 because of slow-drain caused by Host-1*

Recovery by Enabling no-credit-drop timeout at 50 ms

Now, no-credit-drop timeout at 50 ms is enabled on Switch-1 in Figure 6-19. Note that the traffic recovers to 850 MB/s which is more than a 50% recovery, as shown in Figure 6-21. Note that if the R_RDY delay is less than the no-credit-drop threshold of 50 ms, no-credit-drop would not help at all.

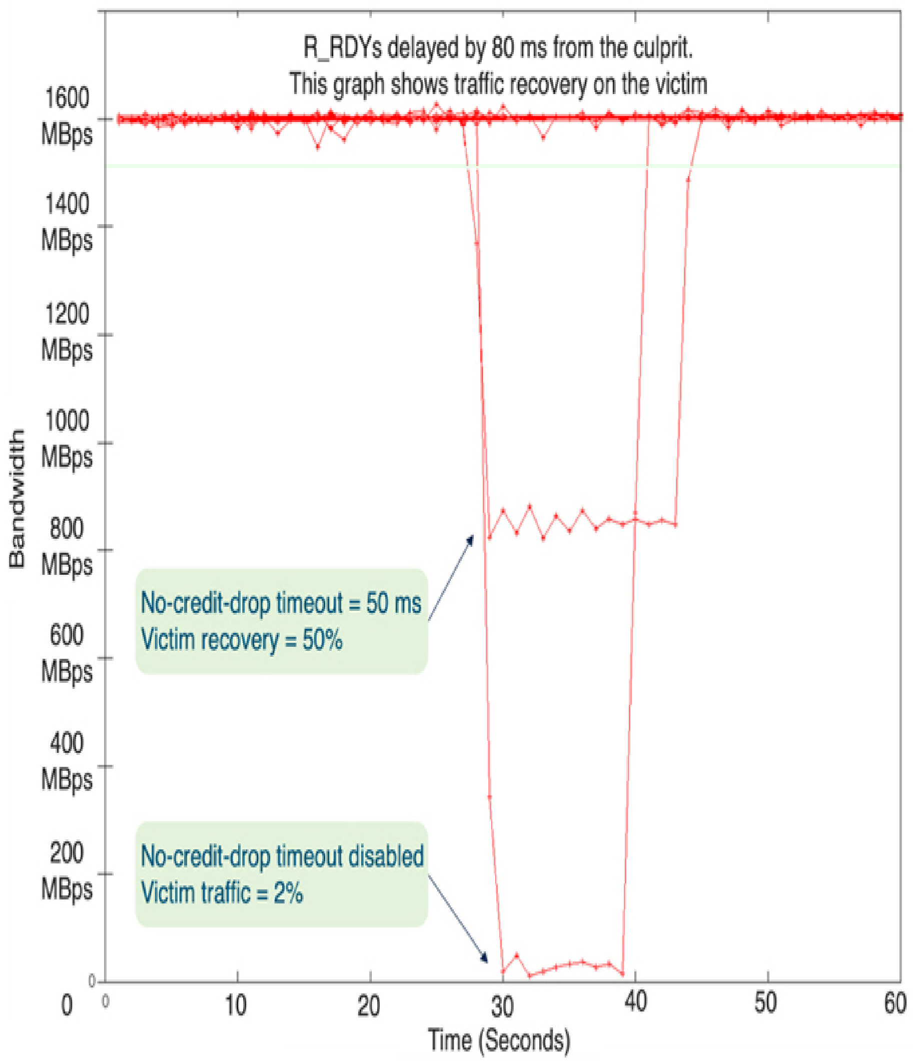


Figure 6-21 *Host-2 throughput recovers by 50% using no-credit-drop timeout of 50 ms*

Recovery by Enabling no-credit-drop timeout at 30 ms and 10 ms

Figure 6-22 shows that with no-credit-drop timeout of 30 ms, throughput recovers to 1250 MB/s which is 80% of the original throughout. With no-credit-drop timeout of 10 ms, throughput recovers to 1600 MB/s, which is 100% recovery.

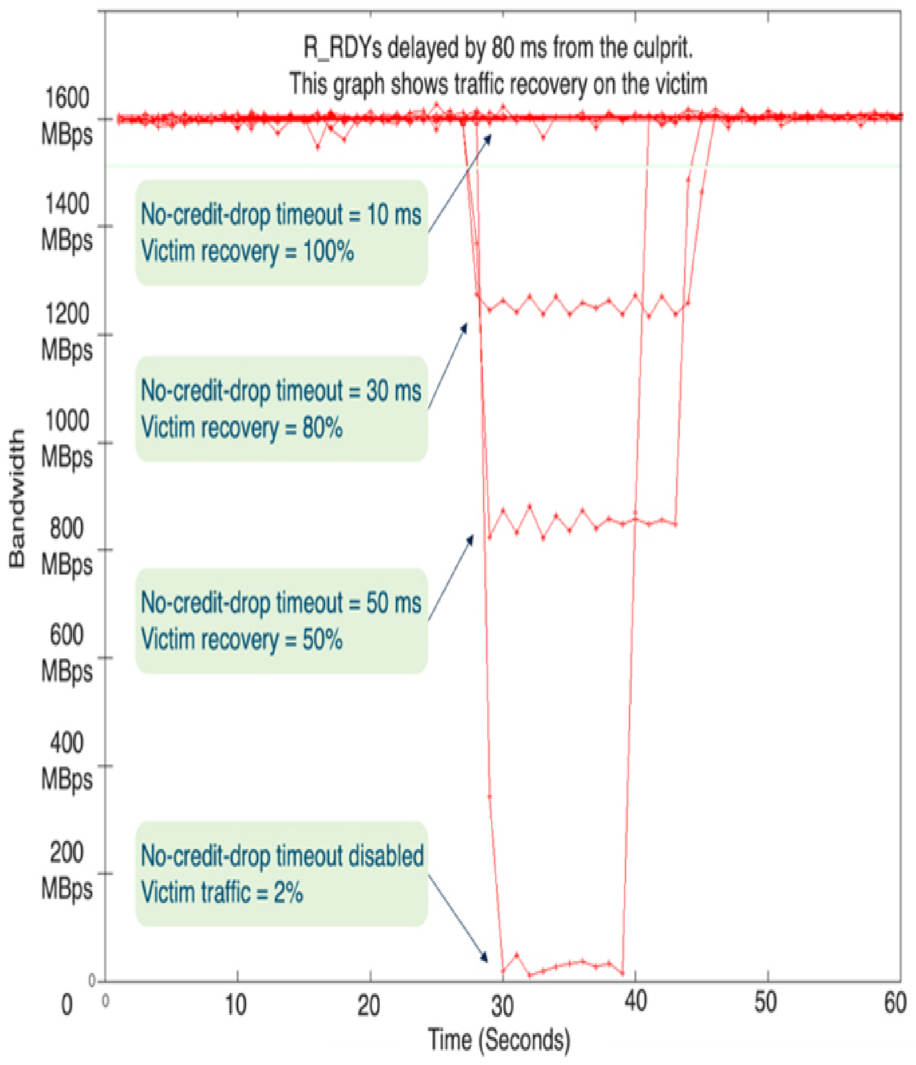


Figure 6-22 *Host-2 throughput recovers using smaller no-credit-drop timeout values*

As is evident, lowering the no-credit-drop timeout increases the traffic recovery on victim devices. This aligns with the conclusions of the earlier section on *No-credit-drop timeout in Action*.

Recovery by Congestion Isolation

Next, in the same setup, Congestion Isolation is enabled, and no-credit-drop timeout is disabled. Figure 6-23 shows the recovery of traffic on Host-2 after being affected by the slow drain device (Host-1).

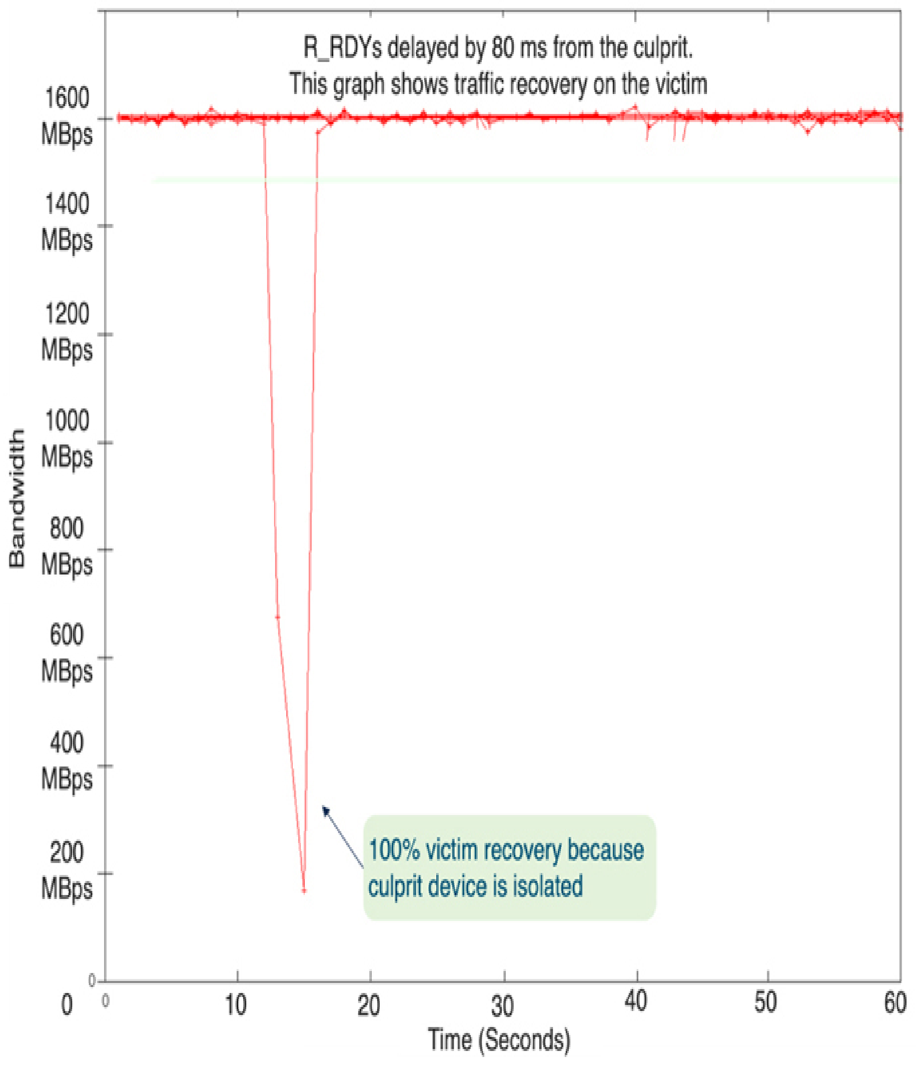


Figure 6-23 *Host-2 throughput recovers using Congestion Isolation*

The following are the key observations from this test:

- The traffic recovery takes 1-2 seconds.
- The traffic recovery is 100%

Recovery by Enabling no-credit-drop timeout and Congestion Isolation Together

Finally, no-credit-drop timeout at 30ms and Congestion Isolation are enabled together. Figure 6-24 shows the result.

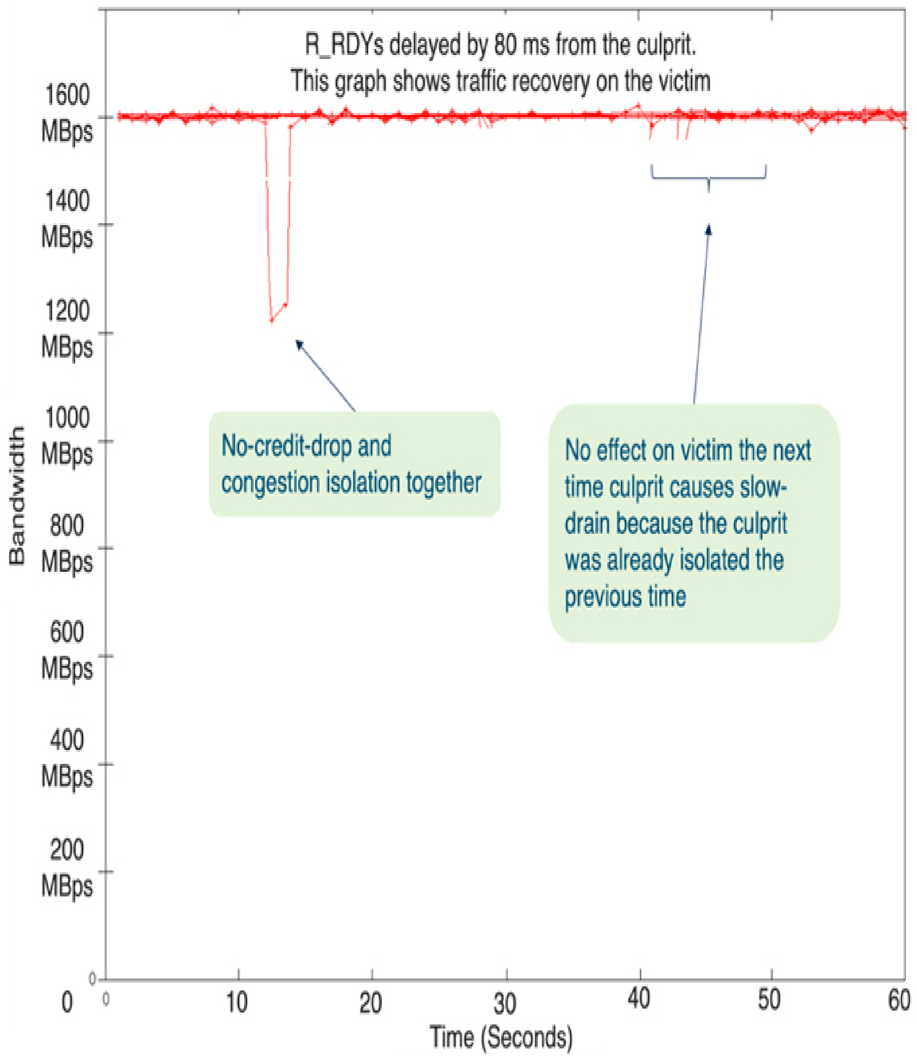


Figure 6-24 *Host-2 throughput recovers using Congestion Isolation and no-credit-drop timeout*

The following are the key observations from this test:

- The traffic recovery is 100% because of Congestion Isolation
- The dip in the traffic is not as deep as seen after enabling only the Congestion Isolation feature (Figure 6-23) because no-credit-drop timeout gets activated at a much lower granularity of 30 ms. Soon after, within 1-2 seconds, Congestion Isolation changes the VL assignment of the traffic to another VL, resulting in full recovery of the traffic.

In this test, Host-1 becomes a slow drain device again at 40th second in Figure 6-24. However, Host-2 does not get affected at all because traffic destined for Host-1 was already moved to a separate VL when it became a slow drain device earlier, hence, no change is visible in the graph.

Recommended Values for no-credit-drop timeout and Congestion Isolation

The following are the recommended values for using no-credit-timeout and Congestion Isolation.

1. Congestion Isolation and no-credit-drop timeout can be used together in the same fabric. Both features work independently and increase the overall effectiveness against congestion.
2. Enable Congestion Isolation for a longer duration of credit unavailability and no-credit-drop timeout for a relatively shorter duration. With Port-Monitor on Cisco MDS switches, using TxWait rising threshold of 40% (400 ms in 1 second) for Congestion Isolation and no-credit-drop timeout of 80 - 100 ms is a good start. These thresholds are just starting points. After the initial configuration, monitor your fabrics and adjust the thresholds.
3. When Congestion Isolation and no-credit-drop timeout are enabled on the same switch, if a device withholds credits for a continuous time equal to or longer than the no-credit-drop timeout, the switch will still drop frames for that device regardless of the VL that this device is assigned to. This means that a previously isolated device (being in VL2 (assigned for traffic destined for slow-drain devices)), can still be subjected to no-credit-drop timeout.

Too many VLs — The Hidden Side Effects

Dedicating B2B credits for each VL has pros and cons. As previously explained, its benefit is that congestion in one VL doesn't affect other VLs. However, the hidden side effect is that the B2B credits can't be shared with other VLs even if the other VLs need more B2B credits whereas the B2B credits in one VL may be under-utilized.

This condition is like a carpool or high-occupancy vehicle (HOV) lane on a highway. In some parts of the world, such lanes remain underutilized, while the other lanes remain congested.

Consider a fabric with 400 hosts. 100 hosts connect at 4GFC, 100 hosts connect at 8GFC, 100 hosts at 16GFC, and 100 hosts connect at 32GFC. The ISL in this fabric has 1000 B2B credits, which are divided equally among 4 VLs, each with 250 B2B credits. The 4 VLs are assigned to the 4 categories of hosts based on their speed. In other words, VL1 is dedicated to hosts connected at 4GFC, VL2 is dedicated to hosts connected at 8GFC, and so on. In this fabric, consider the following points:

1. **The complexity of traffic assignment and its implementation**: This example assumes traffic assignment based on the speed of the end devices. Many other schemes are possible, as explained in the earlier section in *Categorizing Traffic for Segregation*. This leads to two complexities. First, defining a scheme that works in an environment, and second, implementing the scheme. In other words, "what to classify" and "how to classify".

2. **Reduced buffers per flow**: The most important consideration is that dividing the 1000 B2B credits into 4 categories reduces the number of credits per flow. During traffic bursts (very common with NVMe and All-Flash arrays), the 1000 buffers on ISL can absorb the backpressure if the host causes congestion only momentarily. In contrast, the same burst in a VLs has only 250 buffers to absorb it. When the fabric can't absorb congestion, it spreads to the traffic source, which affects many more devices. In this example, the fabric with four VLs is four times less likely to absorb microburst, which is sub-optimal when 750 credits may be unused in other VLs. This concept is explained in detail in Chapter 2, the section on *Buffering and Ability to Absorb Congestion*.

To summarize, it may seem like increasing the number of VLs may increase the effectiveness of congestion segregation but too many VLs have hidden side effects.

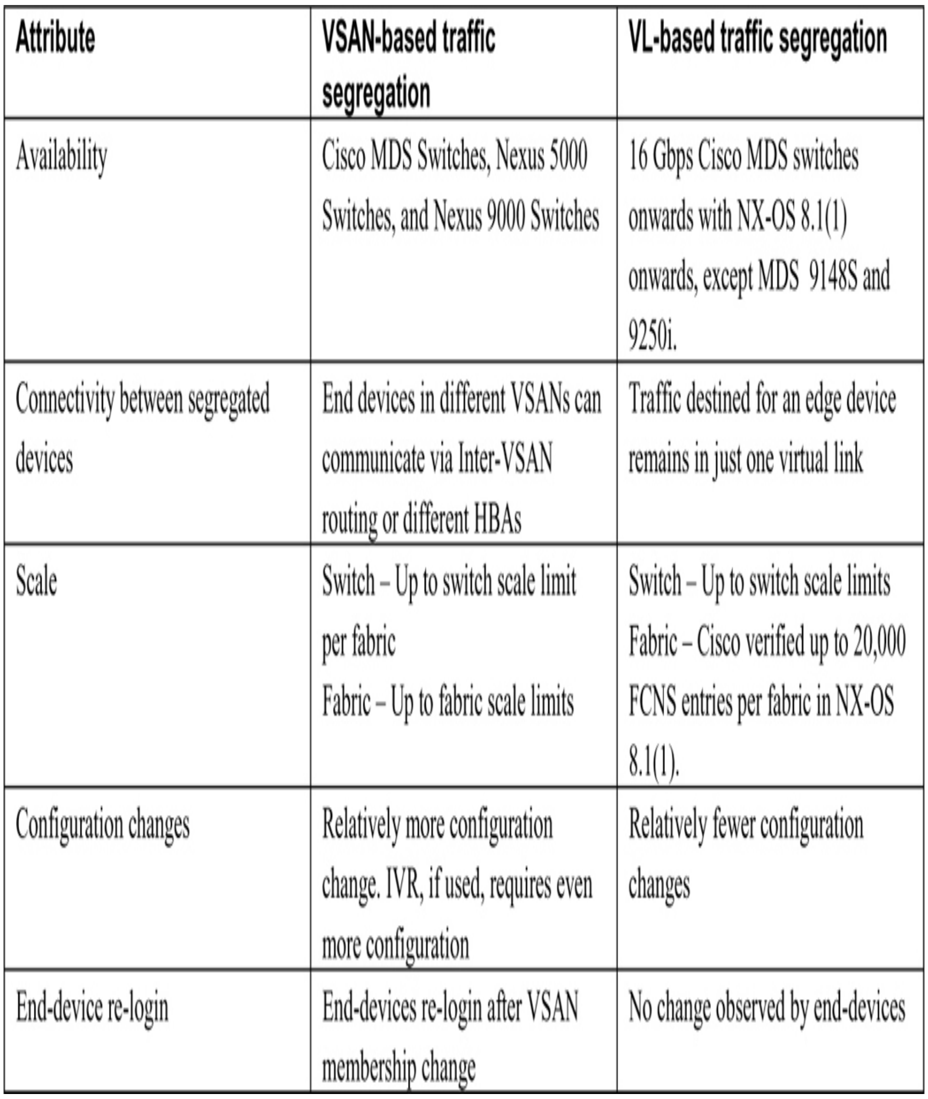
Traffic Segregation Considerations

This section explains the details of two approaches for segregating traffic, which can segregate the effect of congestion.

Comparing Traffic Segregation using VSAN and Virtual Links

Table 6-4 summarizes the comparison of the traffic segregation schemes using VSAN and virtual links.

Table 6-4 *Traffic segregation to dedicated ISLs versus virtual links*



Attribute	VSAN-based traffic segregation	VL-based traffic segregation
Availability	Cisco MDS Switches, Nexus 5000 Switches, and Nexus 9000 Switches	16 Gbps Cisco MDS switches onwards with NX-OS 8.1(1) onwards, except MDS 9148S and 9250i.
Connectivity between segregated devices	End devices in different VSANs can communicate via Inter-VSAN routing or different HBAs	Traffic destined for an edge device remains in just one virtual link
Scale	Switch – Up to switch scale limit per fabric Fabric – Up to fabric scale limits	Switch – Up to switch scale limits Fabric – Cisco verified up to 20,000 FCNS entries per fabric in NX-OS 8.1(1).
Configuration changes	Relatively more configuration change. IVR, if used, requires even more configuration	Relatively fewer configuration changes
End-device re-login	End-devices re-login after VSAN membership change	No change observed by end-devices

Remember that both these approaches can be used together. You can have dedicated ISLs per VSAN, create VLs on those ISLs, and assign a different traffic category to the VLs (automatically or manually). All of this can co-exist in the same fabric.

Congestion Segregation Using Virtual Links — Caution

This section explains two approaches for segregating traffic to VLs. The first approach is to permanently segregate the traffic by manually changing its VL assignment. The second approach is to temporarily segregate the traffic by automatically changing its VL based on congestion indications. This automatic approach is called congestion isolation on Cisco MDS switches. When using congestion isolation, be aware it can take a few seconds to detect the congestion and then assign the traffic to the segregated VL.

The following are considerations for using either form of traffic segregation using VLs:

- Creating VLs requires dedicated B2B credits for each VL. This lowers the fabric's ability to absorb congestion during traffic microbursts.
- Most importantly, recall that traffic segregation helps only the same-path victims, not the direct and indirect victims. In other words, congestion isolation does not isolate the congestion completely.

Congestion Prevention using Rate Limiters on Storage Arrays

Some storage arrays can limit the rate of traffic that they send to each host or initiator. When a congested host is sent less traffic, it can potentially process the lower traffic rate without applying backpressure, which leads to congestion prevention.

Consider the example in Figure 6-25. Host-1 connects to the fabric at 4GFC, whereas Host-2 connects at 8GFC. The storage array, however, connects at 16GFC on two interfaces. This test uses Dell EMC Unity as a storage array.

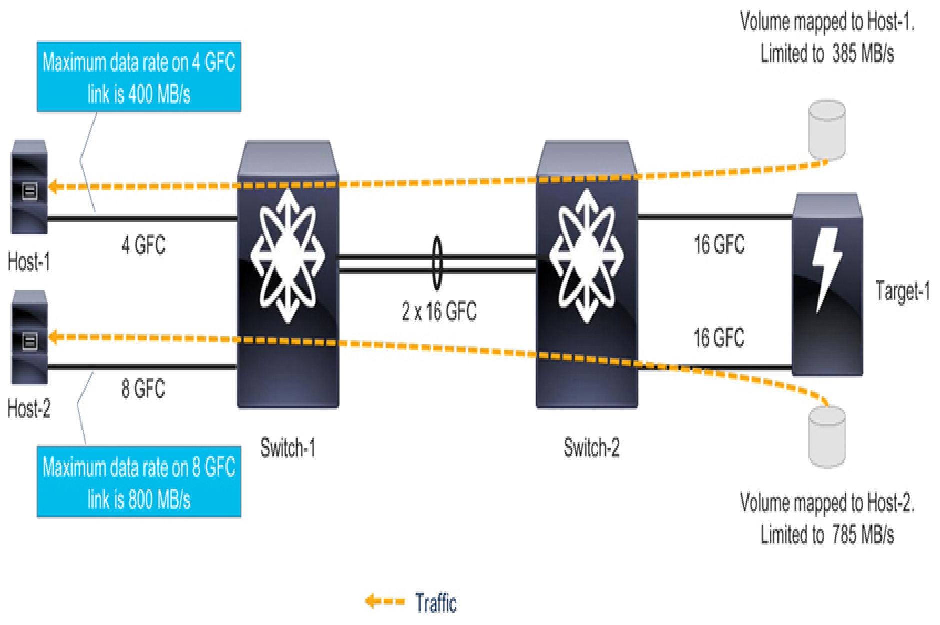

Maximum data rate on 4 GFC link is 400 MB/s
Volume mapped to Host-1. Limited to 385 MB/s
Host-1
4 GFC
2 x 16 GFC
16 GFC
Target-1
16 GFC
Host-2
8 GFC
Switch-1
Switch-2
Maximum data rate on 8 GFC link is 800 MB/s
Volume mapped to Host-2. Limited to 785 MB/s
Traffic

Figure 6-25 *Test setup demonstrating I/O rate limiters on storage arrays*

As Figure 6-26 shows, at 9:15, Host-2 starts receiving traffic from the storage array at 800 MB/s, which is the maximum data rate of its 8GFC link. At 11:00, Host-1 starts I/O operations, which results in congestion due to over-utilization of its 4GFC link. Host-1's throughput remains at the maximum data rate of 4GFC (400 MB/s), but Host-2's throughput is instantly affected due to congestion spreading on the ISL. Overall, Host-2 is the culprit and Host-1 is the victim.

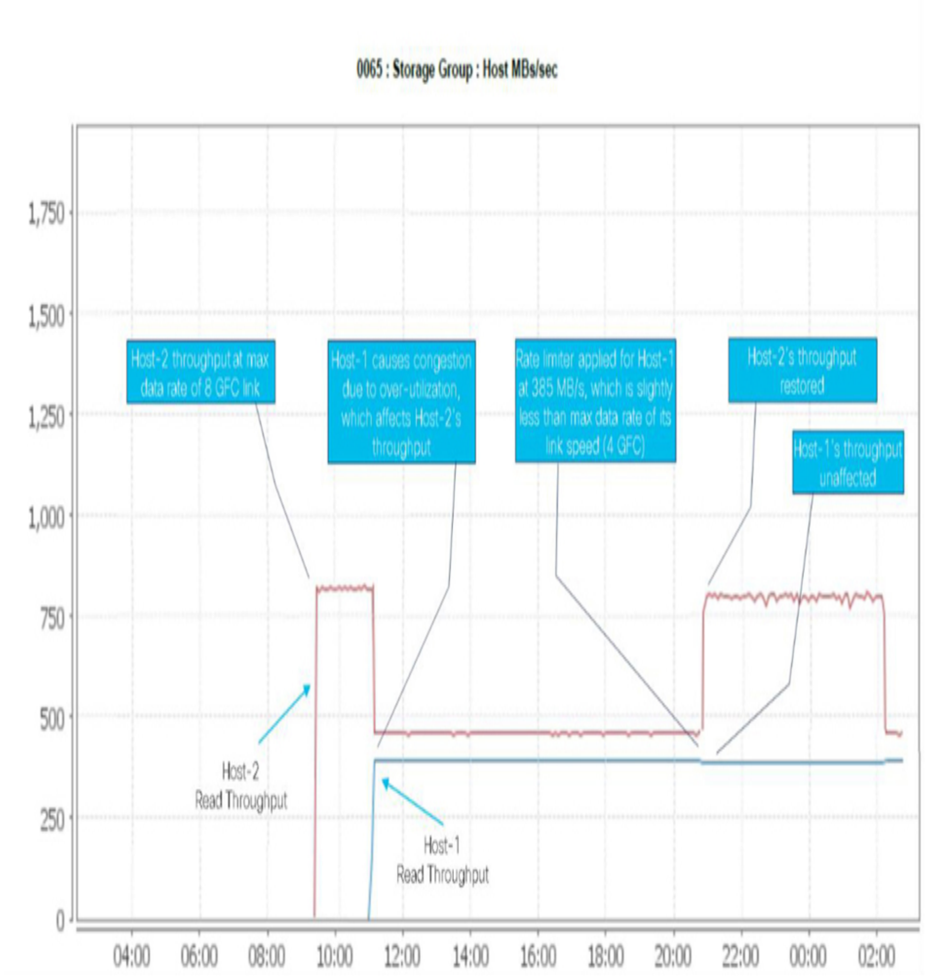


Figure 6-26 *Host-1 and Host-2 throughput demonstrating the benefit of I/O rate limiters on storage arrays*

To prevent the adverse effect of congestion on Host-2, at 20:30, a Host I/O limit is applied to the storage array for the traffic that is sent to Host-1. The limit is configured at 385 MB/s, which is slightly less than the maximum data rate of its

4GFC link. Instantly, the traffic to Host-2 restores to the original level. At the same time, the traffic to Host-1 remains unaffected. Overall, Host-1 is not a culprit anymore, and thus Host-2 is not a victim.

For the sake of simplicity, this section purposefully skips some details of the original testing.

Besides being an effective congestion prevention mechanism, rate limiters on storage arrays have the following additional benefits.

1. It does not disconnect the culprit end device (such as error-disabling a port on Cisco MDS switches) or drop any frames to the culprit devices (such as congestion-drop timeout and no-credit-drop timeout on Cisco MDS switches). In other words, it's not a big-hammer approach. Instead, it allows the culprits to do their I/O operations without affecting other devices in the fabric.
2. This approach limits traffic only to specific hosts or initiators. This is better than lowering the link speed of the storage ports which limits the entire traffic from the storage array regardless of the speed of the host port. Refer to the later section on *Lowering the Link Speed of Storage Ports* for more details.

Conversely, there are some considerations for the successful implementation of this approach in production environments.

1. **Detecting the source of congestion**: Fibre Channel B2B flow control mechanism has a link-level scope. While under egress congestion, a storage port only knows that the directly connected switchport is applying backpressure, but it does not know the ultimate source of congestion, which may be any host that is connected downstream in the fabric. Without knowing the source of congestion, a storage array can't rate limit the traffic going to it. FPIN is designed to address this limitation, although implementations may differ based on the type of storage array. Refer to the section on *Congestion Prevention by Notifying the End Devices* for details.

2. **Detecting the cause of congestion**: As Chapter 2 on *Understanding Congestion in Fibre Channel Fabrics* explains, the symptoms of congestion due to over-utilization and slow drain are different on the host ports. But their effect on the ISL and storage port is the same. Without knowing the cause of congestion, a storage array can't rate limit the traffic correctly. FPIN is designed to address this limitation, although implementations may differ based on the type of storage array. Refer to the section on *Congestion Prevention by Notifying the End Devices* for details.

3. **Knowing the correct traffic rate for a host**: Even if the storage arrays can rate-limit traffic per host, it is challenging to know the correct traffic rate that keeps congestion under control without excessively limiting the traffic. For example, how to find out if a host causes slow-drain when it is sent traffic faster than 600 MB/s? Knowing an appropriate traffic rate under speed-mismatch conditions is relatively easier, as demonstrated in Figure 6-26. For example, a host connected at 4GFC should be rate limited at the maximum data rate at that speed, which is 400 MB/s. Likewise, a host connected at 8GFC should be rate limited to 800 MB/s, and so on. But a storage array may not be able to detect this speed mismatch automatically.

4. **Fan-out ratio:** Consider the fan-out ratio when setting these rate limits. In Figure 6-25, there is a single target for each host but that is hardly a typical configuration. Many times hosts are zoned with 4, 8, or even more targets. So, if there are 8 targets zoned with the host and the host is running at 4GFC, should the individual targets set the host rate limit to 0.5 Gbps each? Or does the host just issue Read IOs to a single target at a time and the rate limit should be set to 4 Gbps? Or is it something in between? It's really very difficult to find the correct value. Too low and the host will underperform. Too high and the congestion will not be prevented.

Despite the challenges of this approach, some users use it because it works well in their environments. If you choose to use it, understand its tradeoffs and plan accordingly for a successful outcome.

Congestion Prevention using Dynamic Ingress Rate Limiting on Switches

Dynamic Ingress Rate Limiting (DIRL) is a congestion prevention mechanism available on Cisco MDS switches that limits the traffic “to” culprit devices so that congestion is eliminated or reduced. What’s interesting about DIRL is that for limiting traffic “to” a device, it rather limits the traffic “from” the device. As a result, it doesn’t act on the I/O operations that have already been initiated. Instead, it limits the culprit device from initiating new I/O operations, and when it initiates fewer I/O operations, it is delivered less traffic, resulting in eliminating or reducing congestion.

DIRL was invented by Edward Mazurek, who is a co-author of this book. He holds a patent on it. DIRL is an innovative approach for preventing congestion because, for the first time, the network switches account for the application traffic patterns (I/O operations) and then take a congestion prevention action.

Although DIRL can help in all kinds of storage networks (lossless Ethernet and TCP), this section focuses only on Fibre Channel because, at the time of this writing, DIRL is available only on Fibre Channel ports on Cisco MDS switches.

Before trying to understand the details of DIRL, refer to Chapter 5, the section on *I/O Operations and Network Traffic Patterns*. For the sake of brevity, those details are not repeated here.

How DIRL Prevents Congestion

Figure 6-27 shows the basic principle of DIRL.

How DIRL Prevents Congestion due to Over-utilization

Based on the direction and size of various frame types that carry I/O operations, it is evident that when a host HBA port has a high ingress utilization, which is the same as its connected-switchport's egress utilization, the real cause is the Read I/O data frames (Figure 6-27). Other frames such as Read I/O response frames, Write I/O transfer ready frames, Write I/O response frames, and other non-I/O frames, also flow in the same direction. Because they are small, their contribution to link utilization is negligible as compared to the Read I/O data frames. This implies that when a host becomes the cause of congestion due to link over-utilization, Read I/O data frames are most of the traffic.

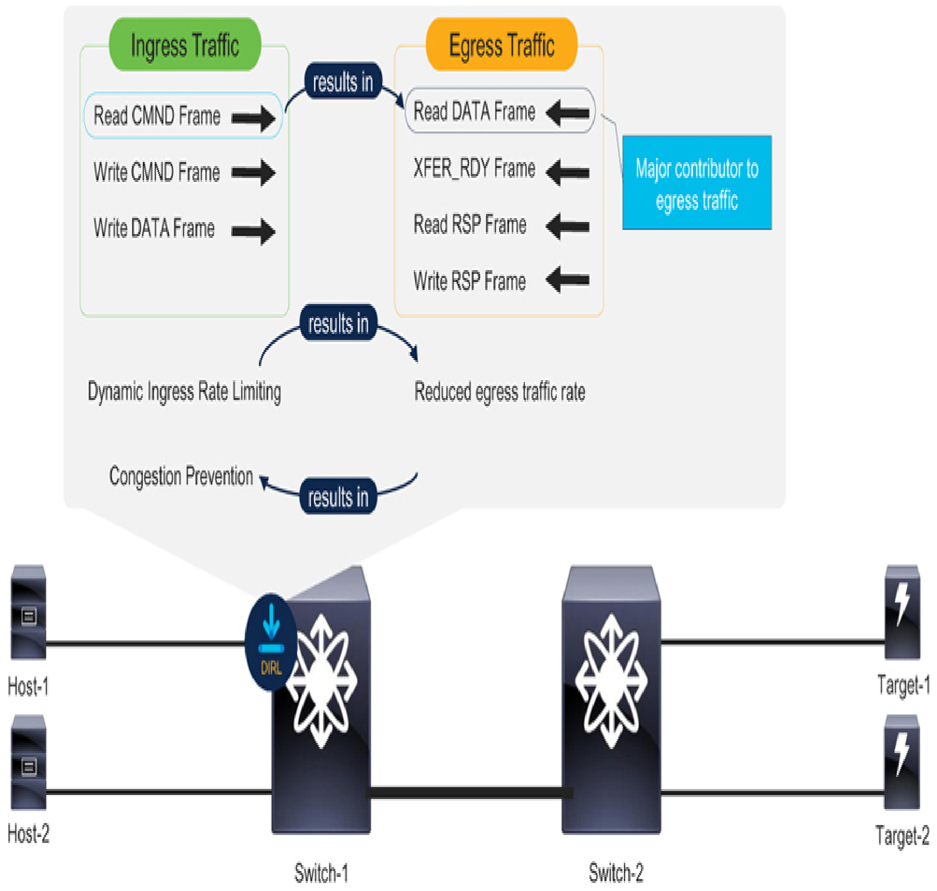


Figure 6-27 *Principle of DIRL for preventing congestion in storage networks*

These Read I/O data frames are just a response to the Read I/O command frames which were sent by the host port and are in the ingress direction on the connected switchport. Because controlling the rate of Read I/O command frames can bring down the rate of Read I/O data frames, controlling the rate of ingress traffic on the host-connected switchport can bring down the rate of egress traffic on this port. This logic makes up the foundation of DIRL and explains how DIRL prevents congestion due to over-utilization.

How DIRL Prevents Congestion due to Slow-drain

DIRL is equally effective in preventing congestion due to slow-drain, which happens when an end device has a slower processing rate as compared to the rate at which frames are being delivered to it. The ingress traffic to a host, which mostly contains Read I/O data frames, is just the response to the Read I/O command frames that the host sent and were in the ingress direction on the connected switchport. As previously explained, limiting the rate of ingress traffic on the connected switchport brings down the egress traffic rate on this switchport and thus lowers the traffic rate to the slow drain device, essentially sending only so much traffic to it that it can process without invoking B2B flow-control to slow down the ingress traffic.

Details of DIRL

The following are some important details about DIRL.

1. **Granularity of rate limiting**: Chapter 5 on *Storage I/O Performance Monitoring* section on *Correlating I/O Operations, Traffic Patterns, and Network Congestion* explains that in block-storage networks, less than 0.01% ingress utilization of a host-connected switchport can result in its 100% egress utilization. Thus, to lower the egress utilization, fine granularity of ingress rate limiting is needed. The Fibre Channel ASICs on the Cisco MDS switches can rate limit the ingress traffic at a granularity of 0.0001%.

2. **No frames are dropped:** DIRL does not drop any frames to rate limit the ingress traffic. It uses the B2B credit pacing mechanism that’s available on the Cisco Fibre Channel ASICs on MDS switches. This mechanism uses B2B flow control between the switchport and the host, and thus, it maintains the lossless nature of Fibre Channel.

3. **The effect on the culprit host:** When DIRL starts on the host-connected switchport, the host lowers its egress traffic rate because it receives R_RDYs at a reduced rate. This is the standard B2B flow control mechanism. The culprit host doesn't know that the connected switchport has started DIRL which enables the pacing of B2B credits for rate limiting. All the culprit device knows is that there appears to be some congestion in the storage network and that it needs to reduce its egress traffic. In most cases, this action should help the culprit device in lowering the severity of congestion.
4. **The effect on the Write I/O operations:** Rate limiting the traffic from the culprit host lowers the rate of Read I/O command frames. Additionally, it slows down the rate of Write I/O command frames and Write I/O data frames. This may be a concern with virtualized hosts, where one VM performs read-heavy I/Os, whereas other VMs perform write-heavy I/Os. Because of this reason, it can be argued that DIRL may affect Write I/O operations as well. However, DIRL activates only when the host causes congestion. When the rate of Read I/O operations reduces due to DIRL, if, hypothetically, Write I/O operations were allowed to continue at a normal rate this would not benefit the application. Although Read and Write I/O operations are different at the network layer, they are cohesive at the application level. Under congestion, when the culprit host is anyway affected, the hosted application can't benefit just by doing Write I/O operations to the same level as before activating DIRL.
5. **Use of DIRL on storage-connected ports:** The basic premise of DIRL is to limit the ingress traffic to lower the egress traffic on a switchport. This approach works on host-connected switchports because Read I/O command frames in the ingress direction result in Read data frames in the egress direction. Likewise, on the storage-connected switchports, when there is congestion (slow-drain or over-utilization), it is due to excessive Transfer-Ready frames the storage array is sending. Rate limiting the ingress traffic on storage-connected switchports can

be effective since the rate of Transfer-Ready frames will then be lowered. Ingress rate limiting will affect Read data frames, Write transfer ready frames, and response frames that are only the result of the command frames. All of this is very similar to the effect of DIRL on host-connected switchports. However, one difference is that since a typical storage array port is zoned to multiple hosts in the fabric, ingress rating limiting on the storage-connected switchport will slow down traffic to all the zoned hosts. Because of these reasons, Cisco MDS switches support DIRL on all F_Ports, but only the initiator ports are rate limited by default after enabling it. Rate limiting on the target ports must be explicitly configured using the NX-OS command **no member fc4-feature target**. For more details, refer to the Cisco MDS 9000 Series Interfaces Configuration Guide.

Benefits of DIRL

DIRL has the following benefits.

1. **DIRL is end-device independent**: DIRL is a function of the network switch (Cisco MDS). It has no dependency on the end devices. Hence, upgrading the end devices is not needed, although that may be the ultimate solution to the problem.
2. **Adaptive**: The ‘D’ in DIRL stands for ‘Dynamic’. DIRL dynamically adjusts as per the traffic profile of the culprit host. It monitors the congestion severity during the last few seconds and adjusts the extent of late limiting accordingly. If the existing rate limiting is not preventing congestion, DIRL increases the ingress rate limiting until the congestion is reduced to an acceptable level. In contrast, if the existing rate limiting is preventing congestion such that the end device is not showing any congestion indications, DIRL allows more traffic. It continues to adjust the traffic rate and eventually finds a stasis state. This dynamic behavior of DIRL makes it adaptive in production networks, which are diverse and behave differently during different times.

3. **Easy adoption**: DIRL is available on Cisco MDS Switches after a software-only non-disruptive upgrade. A hardware upgrade of the switches is not needed.
4. **Affordable**: DIRL is affordable because upgrading end devices (HBA, operating system, firmware, etc.) is not needed. Upgrading only the NX-OS software on the Cisco MDS switches is enough, which is at no charge. Also, DIRL is included in the base offering on Cisco MDS switches. An extra license is not needed.
5. **No side effects on other end devices:** DIRL applies the rate-limiting only to the culprit host. It does not rate limit other hosts and storage ports.
6. **Topology independent**: DIRL works in edge-core, edge-core-edge, or collapsed core (single-switch fabric) topologies. DIRL even works on MDS switches in NPV mode.
7. **Per-switch enablement:** DIRL is enabled on a per-switch basis and doesn't have to be enabled on the entire fabric at once. This gives the ability to gradually roll it out on the switches that are experiencing the most congestion.

Enabling and Using DIRL on Cisco MDS Switches

Cisco launched DIRL in February 2021 with MDS NX-OS 8.5(1). At the time of this writing, the support for DIRL is available on 32 Gbps and faster MDS switches as well as the MDS 9148S 16 Gbps fabric switch when it is running in NPV mode (NX-OS 9.3(1) onwards).

DIRL uses the following components on Cisco MDS switches (Figure 6-28).

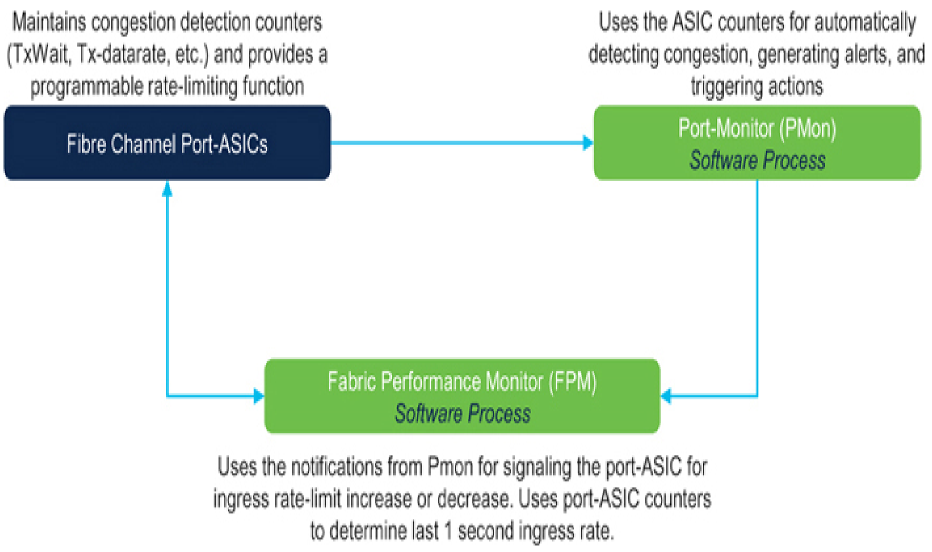

Maintains congestion detection counters (TxWait, Tx-datarate, etc.) and provides a programmable rate-limiting function
Fibre Channel Port-ASICs
Uses the ASIC counters for automatically detecting congestion, generating alerts, and triggering actions
Port-Monitor (PMon)
Software Process
Fabric Performance Monitor (FPM)
Software Process
Uses the notifications from Pmon for signaling the port-ASIC for ingress rate-limit increase or decrease. Uses port-ASIC counters to determine last 1 second ingress rate.

Figure 6-28 *DIRL components in a Cisco MDS switch*

1. **Fibre Channel Port-ASICs**: The Fibre Channel Port-ASICs that are available on MDS switches are custom-designed by Cisco. These port-ASICs collect and maintain congestion detection counters, such as TxWait and Tx-datarate in hardware in an always-on fashion. These counters are enabled by default and no user action is needed. These ASICs also have the programable ingress rate limiting function.
2. **Port-Monitor (PMon)**: Port-Monitor is a feature on Cisco MDS switches that uses various ASIC counters for detecting congestion at a low granularity, such as 1 second. Then, it takes automatic actions such as DIRL. For enabling DIRL, you must configure Port-Monitor with the automatic action of DIRL. Refer to Chapter 3 for more details on Port-Monitor.
3. **Fabric Performance Monitor (FPM)**: FPM is a feature on Cisco MDS switches that listens to notifications from Port-Monitor and if DIRL is configured as an action, FPM signals the port-ASICs for increasing or decreasing the ingress rate limit on the switchport. A user must enable FPM before using DIRL.

The following are the steps for enabling DIRL on a Cisco MDS switch.

Enable FPM

Use NX-OS command **feature fpm** for enabling FPM.

Configure Port-Monitor

Configure Port-Monitor policy for automatically detecting congestion and starting DIRL action. The txwait counter in Port-Monitor detects and prevents congestion due to slow-drain, whereas tx-datarate and tx-datarate-burst counters detect and prevent congestion due to over-utilization.

Refer to Example 6-8 for enabling Port-Monitor for preventing congestion using DIRL. If an edge port on this MDS switch exceeds 80% average egress utilization (measured using tx-datarate) for 10 seconds, DIRL gets activated which starts limiting the ingress traffic on that switchport. DIRL continues to decrease the ingress traffic as long as the egress utilization remains above the rising threshold of 80%. DIRL maintains the ingress rate limiting when the egress utilization is between the rising (80%) and falling thresholds (70%). When the egress utilization falls below the falling threshold (70%), DIRL eases up the ingress limiting to allow more traffic in the ingress direction which eventually results in more egress traffic. This explains the prevention of congestion due to over-utilization using DIRL.

Example 6-8 *Port-Monitor configuration on Cisco MDS for congestion prevention using DIRL*

```
MDS# show running-config | section port-monitor
port-monitor name fabricmon_edge_policy
  logical-type edge
<output truncated>
  counter tx-datarate poll-interval 10 delta rising-
threshold 80 event 4 falling-
threshold 70 event 4 alerts syslog rmon obfl
portguard DIRL
    counter txwait poll-interval 1 delta rising-
threshold 30 event 4 falling-
threshold 10 event 4 alerts syslog rmon portguard
DIRL
<output truncated>
```

For preventing congestion due to slow-drain, Example 6-8 uses the txwait counter. If an edge port on this MDS switch has zero remaining-Tx-B2B-credits for a cumulative duration of 300 ms in 1 second (30% in 1-second polling interval), DIRL gets activated and starts limiting the ingress traffic on that switchport. DIRL continues to decrease the ingress traffic as long as TxWait remains at or above the rising threshold of 30%. DIRL maintains the ingress rate limiting when TxWait is between the rising (30%) and falling thresholds (10%). When TxWait reaches or falls below the falling threshold (10%), DIRL eases up the ingress limiting to allow more traffic in the ingress direction which eventually results in more egress traffic.

Enable DIRL for handling congestion due to slow-drain (txwait) and congestion due to over-utilization of the host-links (tx-datarate and tx-datarate-burst) simultaneously so that it can automatically prevent unexpected congestion events.

DIRL in Action

This section shows DIRL in action by comparing the effect on applications that are hosted on culprit and victim hosts before and after enabling DIRL.

Test Setup

Figure 6-29 shows a setup for demonstrating DIRL. Three all-flash storage arrays—AFA-1, AFA-2, and AFA-3—connect to the fabric using multiple ports operating at 32GFC. The fabric has an edge-core design with Cisco MDS 9718 as the storage-edge switch (MDS 9718-A) and Cisco MDS 9706 as the host-edge switch (MDS 9706-C). The switches connect via two 32GFC links aggregated in a port-channel.

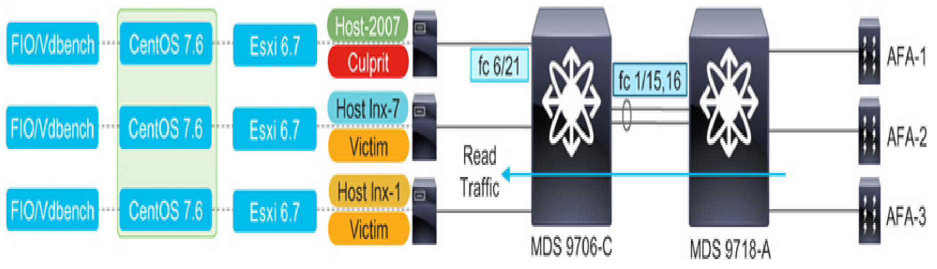


Figure 6-29 *Test setup for DIRL demonstration*

Three hosts connect to the fabric, and they have storage volumes assigned in all three storage arrays. The hosts are virtualized with VMware ESXi 6.7 as the hypervisor and CentOS 6.7 as the guest VM. FIO and Vdbench generate Read and Write I/O operations.

Host 2007 acts as the culprit device, which makes AFA-1, AFA-2, and AFA-3 its direct victims and Host-lnx-1 and Host-lnx-7 indirect victims.

This section demonstrates the following scenarios:

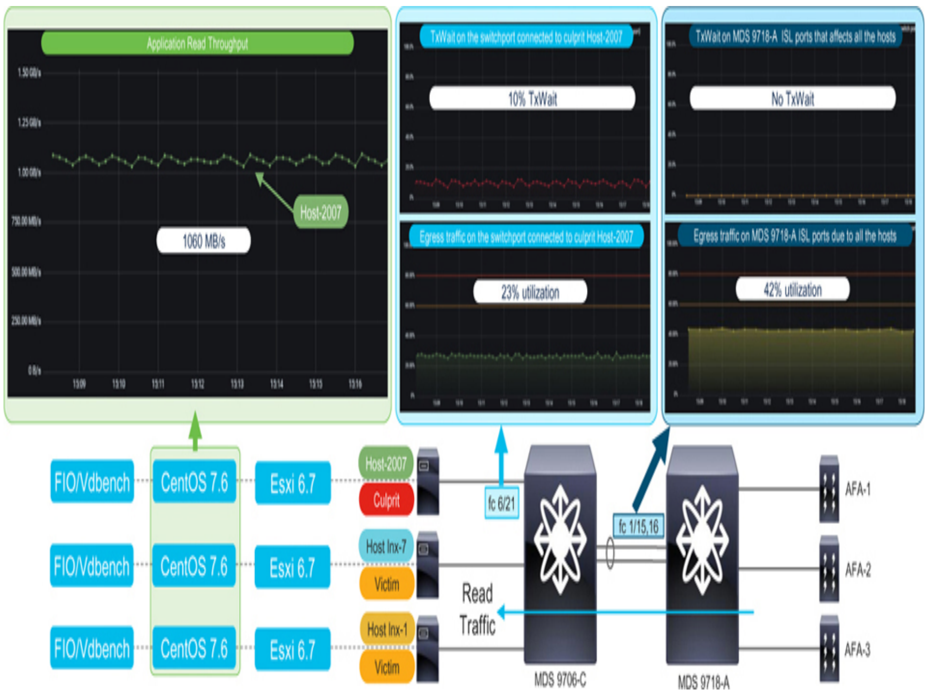
1. Slow-drain caused by Host-2007 (Culprit) when congestion is observed only on its host-edge link, but congestion does not spread to the core of the fabric (ISL).
2. Slow-drain caused by Host-2007 (Culprit) when congestion is observed on its host-edge link and congestion spreads to the core of the fabric (ISL).
3. Preventing congestion due to slow drain using DIRL.
4. Preventing congestion due to over-utilization using DIRL.

In all the scenarios, the results are verified by measuring I/O throughput at the application (FIO/Vdbench) and virtual machine (CentOS) layers. The switchports measure congestion and link utilization. These measurements are shown in a format similar to Figure 6-30.

- The left graph shows Read I/O throughput as seen by the virtual machines on the Hosts.
- The middle-top graph shows TxWait on the switchport that connects to the culprit Host-2007.
- The middle-bottom graph shows the egress utilization of the switchport that connects to the culprit Host 2007
- The right-top graph shows TxWait on the ISL ports of the storage-edge switch (MDS 9718-A).
- The right-bottom graph shows the egress utilization of the ISL ports of the storage-edge switch (MDS 9718-A).

1. Congestion due to Slow-drain Without Spreading

In this scenario, congestion due to slow-drain caused by Host-2007 remains confined to the local link only. Refer to Figure 6-30.



Application Read Throughput
1060 MB/s
Host-2007
TxWait on the switchport connected to culprit Host-2007
10% TxWait
TxWait on MDS 9718-A ISL ports that affects all the hosts
No TxWait
Egress traffic on the switchport connected to culprit Host-2007
23% utilization
Egress traffic on MDS 9718-A ISL ports due to all the hosts
42% utilization
FIO/Vdbench
CentOS 7.6
Esxi 6.7
Host-2007
Culprit
Host lnx-7
Victim
Host lnx-1
Victim
fc 6/21
fc 1/15,16
Read Traffic
MDS 9706-C
MDS 9718-A
AFA-1
AFA-2
AFA-3

Figure 6-30 *Congestion due to slow-drain without spreading*

Read I/O throughput for Host-2007 (measured by the virtual machine) is 1060 MB/s, which leads to 23% egress utilization of the connected switchport (measured by fc 6/21 on MDS 9706-C). This switchport is under 10% TxWait, which indicates minor congestion due to slow drain, but it does not affect any other host in the same fabric because no TxWait is seen on the ISL port of MDS 9718-A. Note that the ISL utilization is at 42% without congestion.

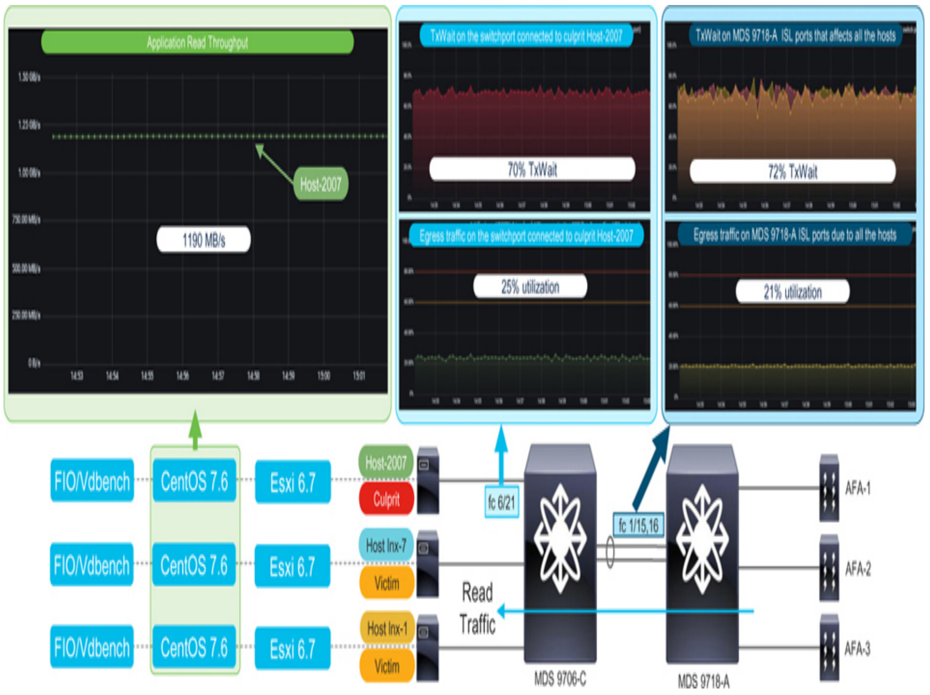
Be aware that the 10% TxWait not affecting other devices in this example shouldn't be generalized. It applies only to this environment under the specific traffic profile. It is possible that 10% TxWait under other conditions may spread the congestion to the core of the fabric. This difference in behavior may be seen because TxWait is a cumulative value at a granularity of 2.5 µs. For example, 10% TxWait means that the cumulative TxWait was 100 ms in 1-second duration but it doesn't convey the exact continuous duration of zero remaining-Tx-B2B-credits. To know the continuous duration of zero remaining-Tx-B2B-credits, use the slowport-monitor feature on Cisco MDS switches. Refer to Chapter 3 for more details on it.

The non-continuous behavior of zero remaining-Tx-B2B-credits helps congestion to be absorbed by the buffers within the switch. In this test, if it were 100 ms of continuous zero remaining-Tx-B2B-credits, although TxWait would still be 10%, the effect on congestion would have been different. Congestion might have spread across the fabric to the storage ports via ISL.

This explains that in production environments small variations may lead to different behaviors, although the measured metrics remain unchanged. The next demonstration is yet another example of behavior changes with minor variations.

2. Congestion due to Slow-drain With Spreading

In this scenario, congestion due to slow-drain caused by Host-2007 spreads throughout the fabric. This results in AFA-1, AFA-2 and AFA-3 being direct victims. Host-lnx-1 and Host-lnx-7 are indirect victims. Refer to Figure 6-31.



Application Read Throughput
Host-2007
1190 MB/s
TxWait on the switchport connected to culprit Host-2007
70% TxWait
Egress traffic on the switchport connected to culprit Host-2007
25% utilization
TxWait on MDS 9718-A ISL ports that affects all the hosts
72% TxWait
Egress traffic on MDS 9718-A ISL ports due to all the hosts
21% utilization
FIO/Vdbench
CentOS 7.6
Esxi 6.7
Host-2007
Culprit
Host lnx-7
Victim
Host lnx-1
Victim
fc 6/21
fc 1/15,16
Read Traffic
MDS 9706-C
MDS 9718-A
AFA-1
AFA-2
AFA-3

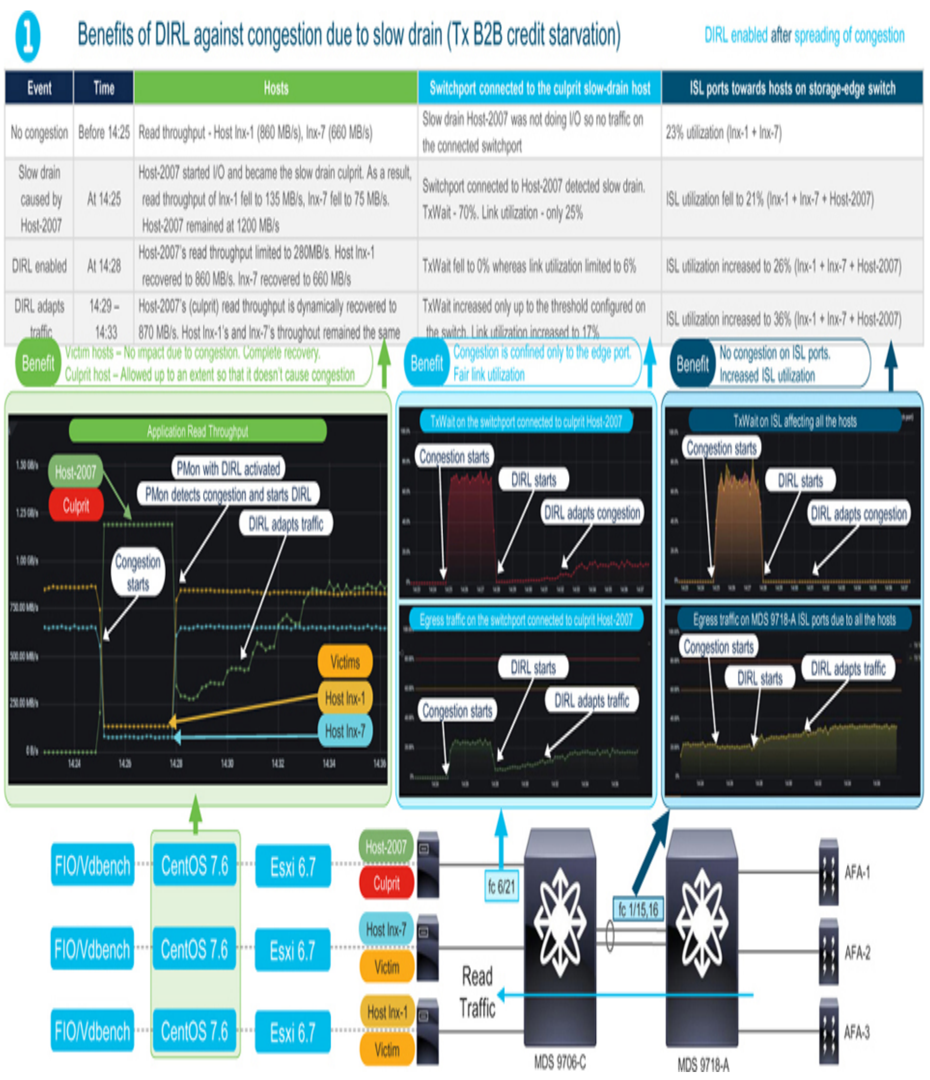
Figure 6-31 *Congestion due to slow-drain with spreading*

The setup of Figure 6-31 is the same as Figure 6-30. The only difference is that the Host-2007's Read throughput increased to 1190 MB/s. With only a 12% increase in Read I/O throughput and only a marginal increase in the egress utilization of the connected switchport, the TxWait increased to 70%. This congestion on the host-connected switchport spread to the core of the fabric. The ISL ports on the upstream switch, MDS 9718-A, show TxWait of 72%. Notice the drop in ISL utilization from 42% (Figure 6-30) to 21% (Figure 6-31) which is a direct result of the increase in TxWait. The ISL utilization drops approximately 50% because the slow-drain caused by Host-2007 affects the throughput of other devices that share the same ISLs.

3. Preventing Congestion due to Slow-drain using DIRL

Refer to Figure 6-32 to visualize the effect of congestion due to slow-drain before enabling DIRL and after enabling DIRL.

Note that the DIRL tests in Figure 6-32 through Figure 6-35 were conducted under different conditions than the earlier tests, so their metrics may not match with that shown in Figure 6-30 and Figure 6-31, although these differences are not relevant for understanding DIRL.



1 Benefits of DIRL against congestion due to slow drain (Tx B2B credit starvation)

DIRL enabled after spreading of congestion

Event	Time	Hosts	Switchport connected to the culprit slow-drain host	ISL ports towards hosts on storage-edge switch
No congestion	Before 14:25	Read throughput - Host lnx-1 (860 MB/s), lnx-7 (660 MB/s)	Slow drain Host-2007 was not doing I/O so no traffic on the connected switchport	23% utilization (lnx-1 + lnx-7)
Slow drain caused by Host-2007	At 14:25	Host-2007 started I/O and became the slow drain culprit. As a result, read throughput of lnx-1 fell to 135 MB/s, lnx-7 fell to 75 MB/s. Host-2007 remained at 1200 MB/s	Switchport connected to Host-2007 detected slow drain. TxWait - 70%. Link utilization - only 25%	ISL utilization fell to 21% (lnx-1 + lnx-7 + Host-2007)
DIRL enabled	At 14:28	Host-2007's read throughput limited to 280MB/s. Host lnx-1 recovered to 860 MB/s. lnx-7 recovered to 660 MB/s	TxWait fell to 0% whereas link utilization limited to 6%	ISL utilization increased to 26% (lnx-1 + lnx-7 + Host-2007)
DIRL adapts traffic	14:29 – 14:33	Host-2007's (culprit) read throughput is dynamically recovered to 870 MB/s. Host lnx-1's and lnx-7's throughout remained the same	TxWait increased only up to the threshold configured on the switch. Link utilization increased to 17%	ISL utilization increased to 36% (lnx-1 + lnx-7 + Host-2007)

Benefit: Victim hosts – No impact due to congestion. Complete recovery. Culprit host – Allowed up to an extent so that it doesn't cause congestion

Benefit: Congestion is confined only to the edge port. Fair link utilization

Benefit: No congestion on ISL ports. Increased ISL utilization

Figure 6-32 *Effect of slow-drain before and after starting DIRL.*

At the beginning (left graph), two hosts, lnx-1 and lnx-7, show read throughout of 860 MB/s and 660 MB/s respectively without congestion.

At 14:25 the culprit device, Host-2007, starts causing slow-drain. At the same time, the throughput of victim hosts, lnx-1 and lnx-7, fall to 135 MB/s (15%) and 75 MB/s (11%) respectively. The switchport that connects to Host-2007 shows egress utilization (middle-bottom graph), which is not high. It also shows TxWait of 70% (middle-top graph), which spreads to the ISL port of the upstream switch, MDS 9718-A (right-top graph). The ISL utilization falls marginally.

At 14:28 a Port-Monitor policy with a TxWait counter specifying DIRL is activated on MDS 9706-C. In 1 second (the minimum detection period for slow-drain) Port-Monitor detects the TxWait and starts DIRL. As a result, the victim devices (lnx-1 and lnx-7) instantaneously see 100% traffic recovery. The throughput of the victim device, Host-2007, is limited to 23% which results in 0% TxWait on the connected switchport and ISL ports.

Over the next few minutes, DIRL dynamically recovers the throughput of the culprit device (Host-2007) to 72% of its original value. As per the thresholds configured on the switch, 870 MB/s is the maximum throughput that the Host-2007 can achieve without causing congestion in the fabric.

The overall benefits are:

- **Storage arrays**: Because DIRL prevented congestion on ISLs, which is the core of the fabric, the storage arrays are able to transmit without any congestion. They are not direct victims anymore.
- **Hosts**: DIRL results in 100% recovery of indirect victim hosts. The culprit host is not completely blocked and still allowed up to 72% of the original throughout.
- **Host-connected switchport**: Congestion is limited only to the switchport that connects to the culprit host.
- **ISL**: No congestion is seen on the ISL. The utilization of ISL increases because of collective utilization by the three hosts.

Example 6-9 shows the Port-Monitor policy for TxWait counter on MDS 9706-C.

Example 6-9 *Port-Monitor policy for preventing congestion due to slow-drain using DIRL*

```
port-monitor name fabricmon_edge_policy
  logical-type edge
<output truncated>
  counter txwait poll-interval 1 delta rising-
threshold 30 event 4 falling-
threshold 10 event 4 alerts syslog rmon portguard
DIRL
<output truncated>
```

Example 6-10 shows utilization and congestion on fc6/21, on MDS 9706-C, which connects to Host-2007.

Example 6-10 *Verifying utilization and TxWait on culprit-connected switchport*

```
MDS9706-C# show interface fc6/21 counters brief

Interface          Input (rate is 5 min avg)
Output (rate is 5 min avg)
-----------------------------------------------------
--------------------------
                   Rate     Total
Rate     Total
                   MB/s     Frames
MB/s     Frames
fc6/21             13       588652759
118      939338076
MDS9706-C#
MDS9706-C# show interface fc6/21 counters
fc6/21
    5 minutes input rate 110042944 bits/sec,
13755368 bytes/sec, 8536 frames/sec
    5 minutes output rate 980491968 bits/sec,
122561496 bytes/sec, 60772
frames/sec
    588758076 frames input, 1074409886456 bytes
      0 discards, 0 errors, 0 CRC/FCS
      0 unknown class, 0 too long, 0 too short
    940078963 frames output, 1745743592096 bytes
      0 discards, 0 errors
    0 timeout discards, 0 credit loss
    0 input OLS, 0 LRR, 0 NOS, 0 loop inits
    0 output OLS, 0 LRR, 0 NOS, 0 loop inits
    0 link failures, 0 sync losses, 0 signal losses
    211827749 Transmit B2B credit transitions to
zero
    0 Receive B2B credit transitions to zero
    556528640 2.5us TxWait due to lack of transmit
credits
    Percentage TxWait for last 1s/1m/1h/72h:
66%/70%/9%/0%
```

```
    32 receive B2B credit remaining
    0 transmit B2B credit remaining
    0 low priority transmit B2B credit remaining
    Last clearing of "show interface" counters:
23:31:19
MDS9706-C#
```

Example 6-11 shows logs when Port-Monitor policy with DIRL action was activated.

Example 6-11 *Logs of DIRL action*

```
2021 Jun 16 14:27:48 MDS9706-C %PMON-SLOT6-2-
PMON_CRIT_INFO: Port Monitor
Critical Information: Policy (fabricmon_edge_policy)
activation is I .
2021 Jun 16 14:27:49 MDS9706-C %PMON-SLOT6-4-
RISING_THRESHOLD_REACHED_WARNING:
TXWait has reached the rising threshold (port=fc6/21
[0x1294000], value=70%) .
2021 Jun 16 14:27:49 MDS9706-C %PMON-SLOT6-2-
PMON_PORTGUARD_ACTION: PortGuard
Config (DIRL) rate-limit action sent to FPM for Port
fc6/21(1294000) Counter
TXWait .
2021 Jun 16 14:27:50 MDS9706-C %FPM-5-
DIRL_RED_EVENT: fpm: Dynamic ingress rate
limiting interface <fc6/21>
2021 Jun 16 14:27:50 MDS9706-C %PMON-SLOT6-4-
FALLING_THRESHOLD_REACHED_WARNING:
TXWait has reached the falling threshold
(port=fc6/21 [0x1294000], value=8%) .
2021 Jun 16 14:28:51 MDS9706-C %PMON-SLOT6-2-
PMON_PORTGUARD_ACTION: PortGuard
Config (DIRL) recovery action sent to FPM for Port
fc6/21(1294000) Counter TXWait
.
2021 Jun 16 14:29:51 MDS9706-C %PMON-SLOT6-2-
PMON_PORTGUARD_ACTION: PortGuard
Config (DIRL) recovery action sent to FPM for Port
fc6/21(1294000) Counter TXWait
.
```

```
2021 Jun 16 14:30:51 MDS9706-C %PMON-SLOT6-2-
PMON_PORTGUARD_ACTION: PortGuard
Config (DIRL) recovery action sent to FPM for Port
fc6/21(1294000) Counter TXWait
.
2021 Jun 16 14:31:52 MDS9706-C %PMON-SLOT6-2-
PMON_PORTGUARD_ACTION: PortGuard
Config (DIRL) recovery action sent to FPM for Port
fc6/21(1294000) Counter TXWait
.
2021 Jun 16 14:32:52 MDS9706-C %PMON-SLOT6-2-
PMON_PORTGUARD_ACTION: PortGuard
Config (DIRL) recovery action sent to FPM for Port
fc6/21(1294000) Counter TXWait
.
```

Finally, refer to Example 6-12, which shows DIRL actions. Read the entries in chronological order from bottom to up.

Example 6-12 *DIRL actions using show fpm ingress-rate-limit*

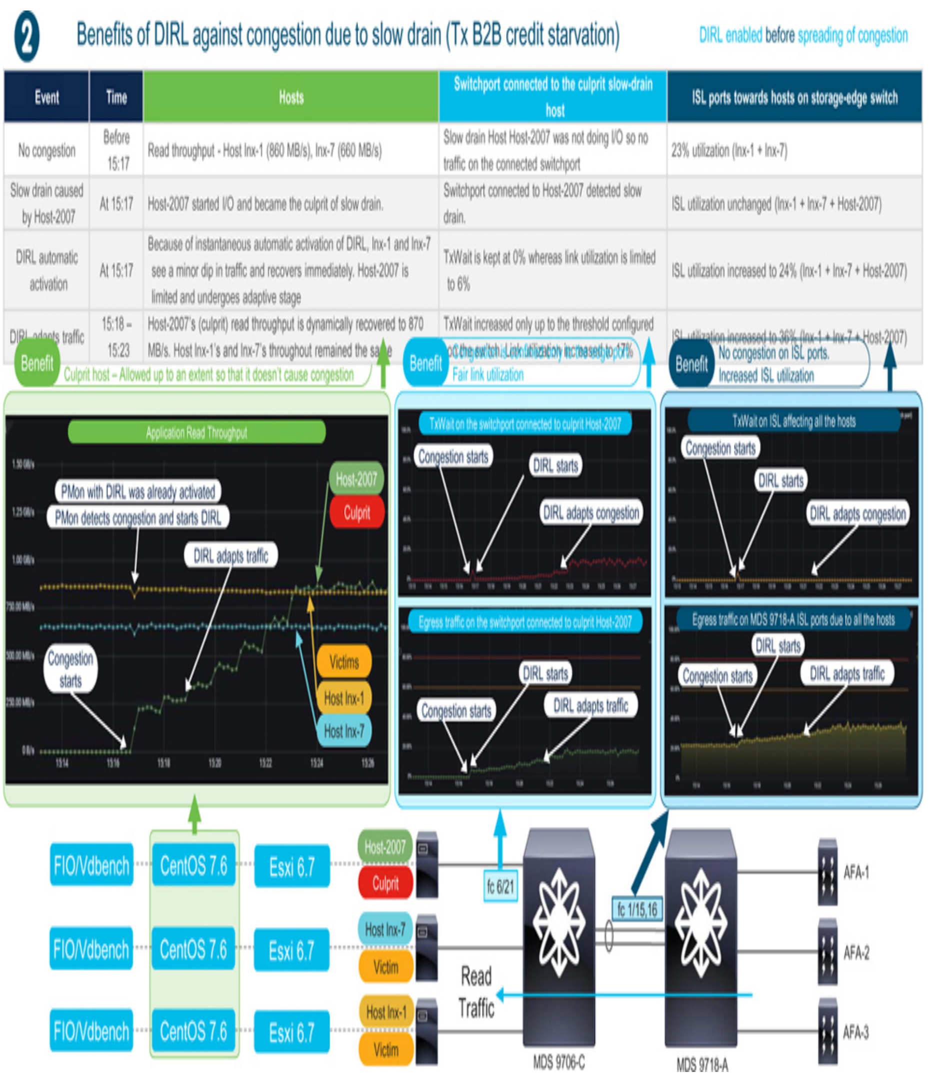
```
MDS9706-C# show fpm ingress-rate-limit events
interface fc 6/21
-----------------------------------------------------
---------------------------
Interface |  Counter   |  Event   |      Action
|Operating |  Input   | Output |
Current | Applied |       Time
          |           |         |
|port-speed|  rate    |  rate   |
RL%    |   RL%    |
          |           |         |                 |
Mbps    |  Mbps    |  Mbps   |
|         |
-----------------------------------------------------
---------------------------
fc6/21       txwait     recovery   rate-recovery
8000     114.46    1235.73
3.0648    3.8310    Wed Jun 16 14:32:53 2021
fc6/21       txwait     recovery   rate-recovery
8000      96.74     865.86
2.4519   3.0648    Wed Jun 16 14:31:52 2021
```

```
fc6/21        txwait     recovery   rate-recovery
8000       91.62      766.96
1.9615   2.4519    Wed Jun 16 14:30:52 2021
fc6/21        txwait     recovery   rate-recovery
8000       55.90      576.48
1.5692   1.9615    Wed Jun 16 14:29:52 2021
fc6/21        txwait     recovery   rate-recovery
8000       54.94      465.34
1.2554   1.5692    Wed Jun 16 14:28:51 2021
fc6/21        txwait     rising     rate-reduction
8000      213.41    2079.69
100.0000  1.2554    Wed Jun 16 14:27:50 2021
```

Match the timestamps of the logs (Example 6-11) and the command-output (Example 6-12) (taken from the MDS switch) with the Read I/O throughput, left-graph in Figure 6-32 (collected from the virtual machine on the hosts).

- 2021 Jun 16 14:27:48: Port-Monitor policy was activated.
- 2021 Jun 16 14:27:49: In the next second, Port-Monitor detected TxWait and informed FPM to take rate-reduction (limit) action.
- 2021 Jun 16 14:27:50: DIRL limits the ingress traffic
- 2021 Jun 16 14:27:51 - 2021 Jun 16 14:32:52: Every 60 seconds, DIRL adapts the traffic by rate recovery to keep TxWait between rising and falling threshold.

The test of Figure 6-32 enabled Port-Monitor after congestion starts simply to illustrate the dramatic effect that DIRL has on it. However, in production environments, Port-Monitor with DIRL should always be enabled. When congestion starts, it will automatically detect and start DIRL for the culprit switchport. The test in Figure 6-33 enabled Port-Monitor before congestion starts. DIRL is ready for immediate prevention.


2 Benefits of DIRL against congestion due to slow drain (Tx B2B credit starvation)

DIRL enabled before spreading of congestion

Event	Time	Hosts	Switchport connected to the culprit slow-drain host	ISL ports towards hosts on storage-edge switch
No congestion	Before 15:17	Read throughput - Host lnx-1 (860 MB/s), lnx-7 (660 MB/s)	Slow drain Host Host-2007 was not doing I/O so no traffic on the connected switchport	23% utilization (lnx-1 + lnx-7)
Slow drain caused by Host-2007	At 15:17	Host-2007 started I/O and became the culprit of slow drain.	Switchport connected to Host-2007 detected slow drain.	ISL utilization unchanged (lnx-1 + lnx-7 + Host-2007)
DIRL automatic activation	At 15:17	Because of instantaneous automatic activation of DIRL, lnx-1 and lnx-7 see a minor dip in traffic and recovers immediately. Host-2007 is limited and undergoes adaptive stage	TxWait is kept at 0% whereas link utilization is limited to 6%	ISL utilization increased to 24% (lnx-1 + lnx-7 + Host-2007)
DIRL adapts traffic	15:18 – 15:23	Host-2007's (culprit) read throughput is dynamically recovered to 870 MB/s. Host lnx-1's and lnx-7's throughout remained the same	TxWait increased only up to the threshold configured on the switch. Link utilization increased to 17%	ISL utilization increased to 36% (lnx-1 + lnx-7 + Host-2007)

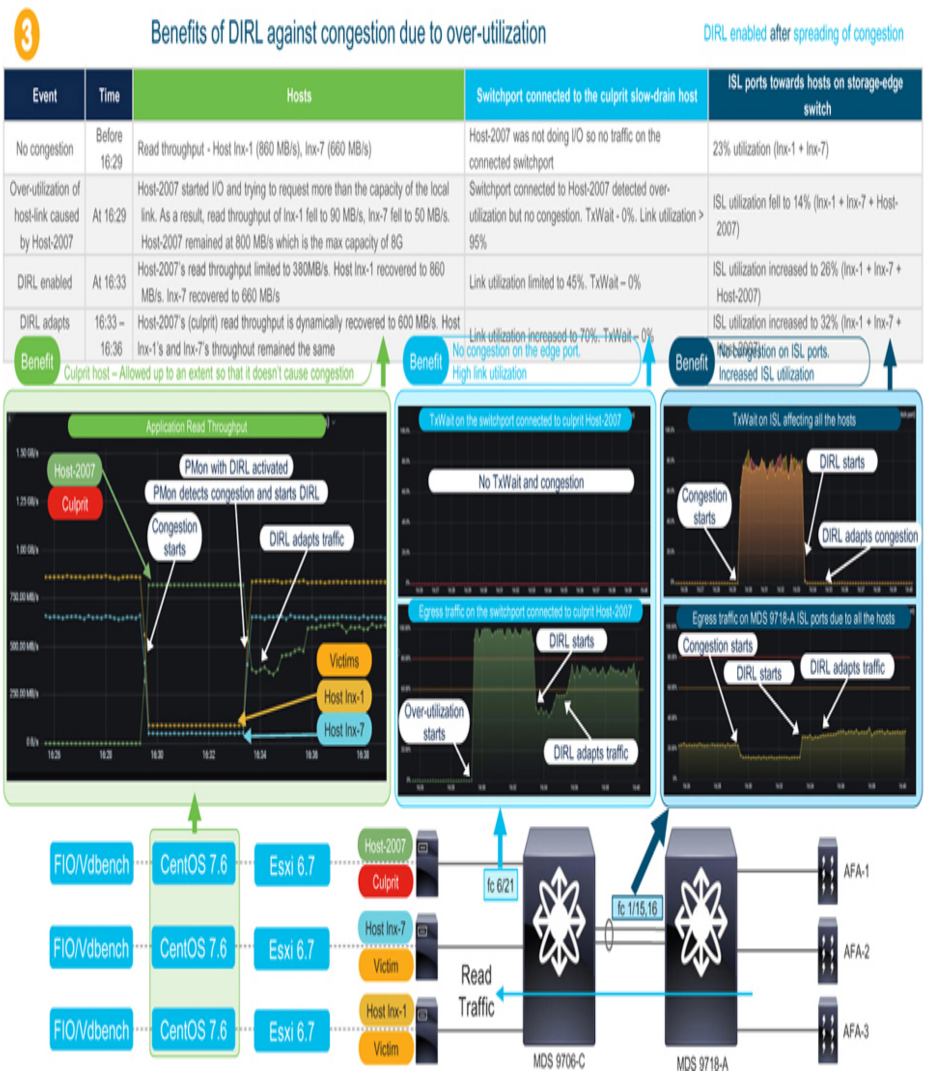
Benefit
Culprit host – Allowed up to an extent so that it doesn't cause congestion
Benefit
No congestion on the edge port.
Fair link utilization
Benefit
No congestion on ISL ports.
Increased ISL utilization
Application Read Throughput
PMon with DIRL was already activated
PMon detects congestion and starts DIRL
DIRL adapts traffic
Host-2007
Culprit
Congestion starts
Victims
Host lnx-1
Host lnx-7
TxWait on the switchport connected to culprit Host-2007
Congestion starts
DIRL starts
DIRL adapts congestion
TxWait on ISL affecting all the hosts
Congestion starts
DIRL starts
DIRL adapts congestion
Egress traffic on the switchport connected to culprit Host-2007
DIRL starts
Congestion starts
DIRL adapts traffic
Egress traffic on MDS 9718-A ISL ports due to all the hosts
DIRL starts
Congestion starts
DIRL adapts traffic
FIO/Vdbench
CentOS 7.6
Esxi 6.7
Host-2007
Culprit
Host lnx-7
Victim
Host lnx-1
Victim
fc 6/21
fc 1/15,16
Read Traffic
MDS 9706-C
MDS 9718-A
AFA-1
AFA-2
AFA-3

Figure 6-33 *Port-Monitor detects congestion and immediately starts DIRL*

When congestion starts because of Host-2007, a minor traffic dip for Host lnx-1 and lnx-7 is because of the 1-second TxWait detection granularity of Port-Monitor. But soon, these hosts are insulated from the effects of congestion as if the culprit (Host-2007) was not even present.

4. Preventing Congestion due to Over-utilization Using DIRL

Refer to Figure 6-34 to visualize the effect of congestion due to over-utilization before enabling DIRL and after enabling DIRL. It's important to note that the functioning of DIRL for over-utilization is exactly the same as with slow-drain. Only the triggering event is different.



Benefits of DIRL against congestion due to over-utilization

DIRL enabled after spreading of congestion

Event	Time	Hosts	Switchport connected to the culprit slow-drain host	ISL ports towards hosts on storage-edge switch
No congestion	Before 16:29	Read throughput - Host lnx-1 (860 MB/s), lnx-7 (660 MB/s)	Host-2007 was not doing I/O so no traffic on the connected switchport	23% utilization (lnx-1 + lnx-7)
Over-utilization of host-link caused by Host-2007	At 16:29	Host-2007 started I/O and trying to request more than the capacity of the local link. As a result, read throughput of lnx-1 fell to 90 MB/s, lnx-7 fell to 50 MB/s. Host-2007 remained at 800 MB/s which is the max capacity of 8G	Switchport connected to Host-2007 detected over-utilization but no congestion. TxWait - 0%. Link utilization > 95%	ISL utilization fell to 14% (lnx-1 + lnx-7 + Host-2007)
DIRL enabled	At 16:33	Host-2007's read throughput limited to 380MB/s. Host lnx-1 recovered to 860 MB/s. lnx-7 recovered to 660 MB/s	Link utilization limited to 45%. TxWait – 0%	ISL utilization increased to 26% (lnx-1 + lnx-7 + Host-2007)
DIRL adapts	16:33 – 16:36	Host-2007's (culprit) read throughput is dynamically recovered to 600 MB/s. Host lnx-1's and lnx-7's throughout remained the same	Link utilization increased to 70%. TxWait – 0%	ISL utilization increased to 32% (lnx-1 + lnx-7 + Host-2007)

Figure 6-34 *Effect of congestion due to over-utilization before and after starting DIRL*

At the beginning (left graph), two hosts, lnx-1 and lnx-7, show Read I/O throughput of 860 MB/s and 660 MB/s respectively.

At 16:29, the culprit host, Host-2007, starts the over-utilization of its directly connected link (middle-bottom graph), which causes congestion on the ISL port of the upstream switch (right-top graph). At the same time, the throughput of victim hosts, lnx-1 and lnx-7, fall to 10% and 7% respectively.

Note that fc6/21 on MDS 9706-C, which connects to Host-2007, shows high egress utilization but no TxWait (middle-top graph). This condition is different from Figure 6-32, which shows high TxWait and low utilization. In both cases, ISL shows high TxWait indicating congestion has spread across the ISL toward the source.

At 16:33 a Port-Monitor policy with a tx-datarate counter specifying DIRL is activated. In 10 seconds (the minimum detection period for over-utilization) Port-Monitor detects the high utilization and starts DIRL. This results in 100% recovery of the victim devices whereas the throughput of the culprit device (Host-2007) is limited to 44% of its original value. Within the next few minutes, DIRL dynamically adapts the throughput of the culprit device and recovers it to 75%, which is the maximum throughput allowed without spreading congestion in the fabric.

The overall benefits are:

- **Storage arrays**: Because DIRL prevented congestion on ISLs, which is the core of the fabric, the storage arrays are able to transmit without any congestion. They are not direct victims anymore.
- **Hosts**: DIRL results in 100% recovery of indirect victim hosts. The culprit host is not completely blocked and still allowed up to 75% of the original throughout.
- **Host-connected switchport**: The host link is 70% utilized and no congestion is seen on it.
- **ISL**: No congestion is seen on the ISL. As a result, the utilization of ISL increases collectively by all the hosts.

Example 6-13 shows the Port-Monitor policy on MDS 9706-C for preventing congestion due to over-utilization.

Example 6-13 *Port-Monitor configuration on a Cisco MDS switch for congestion prevention due to over-utilization using DIRL*

```
port-monitor name fabricmon_edge_policy
  logical-type edge
<output truncated>
  counter tx-datarate poll-interval 10 delta rising-
threshold 80 event 4 falling-
threshold 70 event 4 alerts syslog rmon obfl
portguard DIRL
<output truncated>
```

Example 6-14 shows utilization and congestion on fc6/21, on MDS 9706-C, which connects Host-2007.

Example 6-14 *Verifying utilization and congestion on culprit-connected switchport*

```
MDS9706-C# show interface fc6/21 counters brief

-----------------------------------------------------
---------------------------
Interface           Input (rate is 5 min avg)
Output (rate is 5 min avg)
                    -----------------------------  --
---------------------------
                    Rate     Total
Rate     Total
                    MB/s     Frames
MB/s     Frames
-----------------------------------------------------
---------------------------
fc6/21              61       637539668
560      1282725080
MDS9706-C# show interface fc6/21 counters
fc6/21
    5 minutes input rate 497836928 bits/sec,
62229616 bytes/sec, 39069 frames/sec
    5 minutes output rate 4517989792 bits/sec,
564748724 bytes/sec, 280361
frames/sec
    637656081 frames input, 1150757749936 bytes
      0 discards, 0 errors, 0 CRC/FCS
      0 unknown class, 0 too long, 0 too short
    1283592452 frames output, 2432654709140 bytes
      0 discards, 0 errors
    0 timeout discards, 0 credit loss
    0 input OLS, 0 LRR, 0 NOS, 0 loop inits
    0 output OLS, 0 LRR, 0 NOS, 0 loop inits
    0 link failures, 0 sync losses, 0 signal losses
    283446369 Transmit B2B credit transitions to
zero
    0 Receive B2B credit transitions to zero
    804058368 2.5us TxWait due to lack of transmit
```

```
credits
    Percentage TxWait for last 1s/1m/1h/72h:
0%/0%/0%/0%
    32 receive B2B credit remaining
    14 transmit B2B credit remaining
    14 low priority transmit B2B credit remaining
    Last clearing of "show interface" counters:
1d01h
MDS9706-C#
```

Example 6-15 shows logs when Port-Monitor policy with DIRL action was activated.

Example 6-15 *Logs of DIRL action*

```
2021 Jun 16 16:33:12 MDS9706-C %PMON-SLOT6-2-
PMON_CRIT_INFO: Port Monitor
Critical Information: Policy (fabricmon_edge_policy)
activation is successfull .
2021 Jun 16 16:33:22 MDS9706-C %PMON-SLOT6-4-
RISING_THRESHOLD_REACHED_WARNING: TX
Datarate has reached the rising threshold
(port=fc6/21 [0x1294000],
value=10443746680) .
2021 Jun 16 16:33:22 MDS9706-C %PMON-SLOT6-2-
PMON_PORTGUARD_ACTION: PortGuard
Config (DIRL) rate-limit action sent to FPM for Port
fc6/21(1294000) Counter Tx-
datarate .
2021 Jun 16 16:33:23 MDS9706-C %FPM-5-
DIRL_RED_EVENT: fpm: Dynamic ingress rate
limiting interface <fc6/21>
2021 Jun 16 16:33:41 MDS9706-C %PMON-SLOT6-4-
FALLING_THRESHOLD_REACHED_WARNING:
TX Datarate has reached the falling threshold
(port=fc6/21 [0x1294000],
value=4748056725) .
2021 Jun 16 16:34:42 MDS9706-C %PMON-SLOT6-2-
PMON_PORTGUARD_ACTION: PortGuard
Config (DIRL) recovery action sent to FPM for Port
fc6/21(1294000) Counter Tx-
datarate .
```

```
2021 Jun 16 16:35:42 MDS9706-C %PMON-SLOT6-2-
PMON_PORTGUARD_ACTION: PortGuard
Config (DIRL) recovery action sent to FPM for Port
fc6/21(1294000) Counter Tx-
datarate .
```

Finally, refer to Example 6-16, which shows DIRL adaptive actions. Read the entries in chronological order from bottom to up.

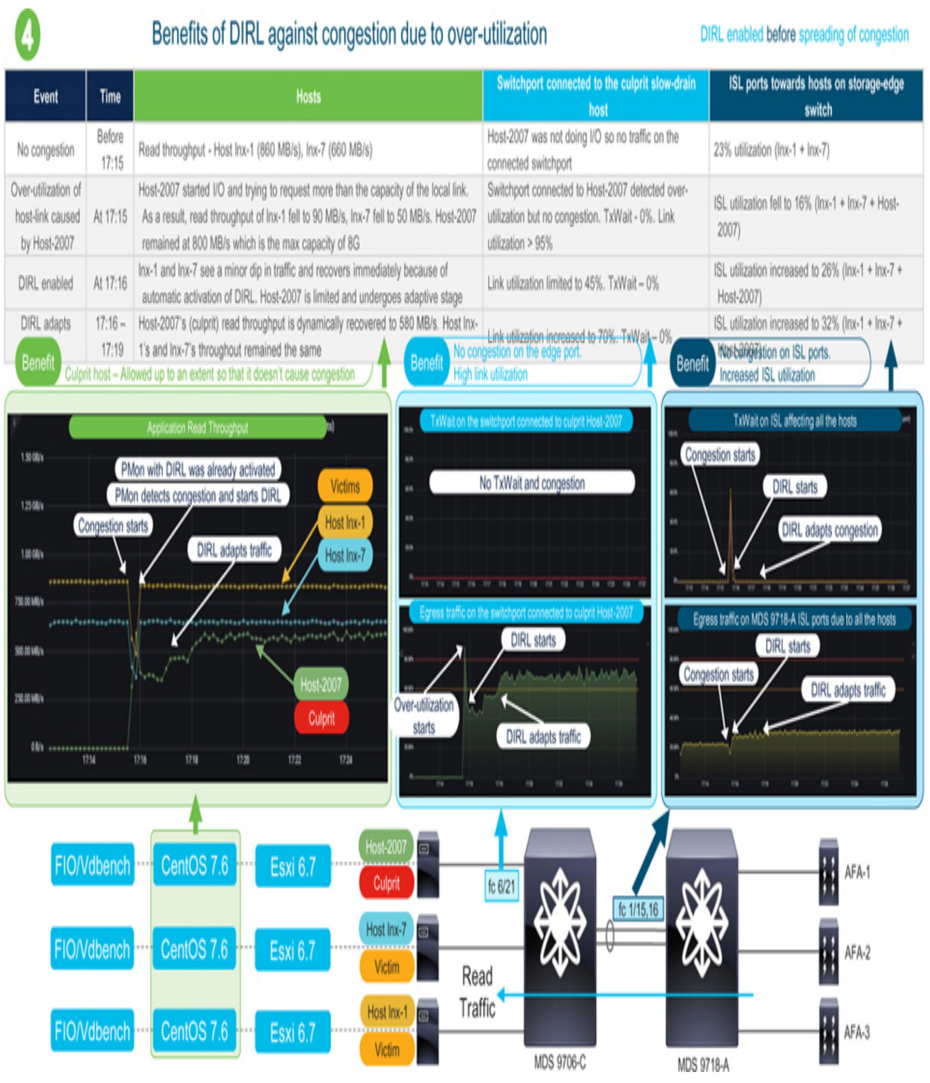
Example 6-16 *DIRL actions using show fpm ingress-rate-limit*

```
MDS9706-C# show fpm ingress-rate-limit events
interface fc 6/21
-----------------------------------------------------
------------------------------
------------------------------------------------
Interface |   Counter    |  Event     |    Action
|Operating |  Input   | Output
| Current | Applied |        Time
          |              |            |
|port-speed|  rate    |  rate
|   RL%    |   RL%    |
          |              |            |
|   Mbps    |  Mbps    |  Mbps
|          |          |
-----------------------------------------------------
------------------------------
------------------------------------------------
fc6/21      txdatarate   recovery    rate-recovery
8000       544.30
5056.45   6.6865    8.3582    Wed Jun 16 16:35:43
2021
fc6/21      txdatarate   recovery    rate-recovery
8000       435.20
3916.69   5.3492    6.6865    Wed Jun 16 16:34:43
2021
fc6/21      txdatarate   rising      rate-reduction
8000       909.37
8366.33   100.0000  5.3492    Wed Jun 16 16:33:23
2021
```

Match the timestamps of the logs (Example 6-15) and the command-output (Example 6-16) (taken from the MDS switch) with the Read I/O throughput, left-graph in Figure 6-34 (collected from the virtual machine on the hosts).

- 2021 Jun 16 16:33:12: Port-Monitor policy was activated.
- 2021 Jun 16 16:33:22: In the next ten seconds, Port-Monitor detected Tx-datarate and informed FPM to take rate-reduction (limit) action.
- 2021 Jun 16 16:33:23: DIRL limits the ingress traffic
- 2021 Jun 16 16:34:42 - 2021 Jun 16 16:35:42: Every 60 seconds, DIRL adapts the traffic by rate recovery to keep Tx-datarate between rising and falling threshold.

The test of Figure 6-34 enabled Port-Monitor after congestion starts simply to illustrate the dramatic effect that DIRL has on it. The test in Figure 6-35 enabled Port-Monitor before congestion starts. DIRL is ready for immediate prevention of congestion by the culprit device.


Benefits of DIRL against congestion due to over-utilization

DIRL enabled before spreading of congestion

Event	Time	Hosts	Switchport connected to the culprit slow-drain host	ISL ports towards hosts on storage-edge switch
No congestion	Before 17:15	Read throughput - Host lnx-1 (860 MB/s), lnx-7 (660 MB/s)	Host-2007 was not doing I/O so no traffic on the connected switchport	23% utilization (lnx-1 + lnx-7)
Over-utilization of host-link caused by Host-2007	At 17:15	Host-2007 started I/O and trying to request more than the capacity of the local link. As a result, read throughput of lnx-1 fell to 90 MB/s, lnx-7 fell to 50 MB/s. Host-2007 remained at 800 MB/s which is the max capacity of 8G	Switchport connected to Host-2007 detected over-utilization but no congestion. TxWait - 0%. Link utilization > 95%	ISL utilization fell to 16% (lnx-1 + lnx-7 + Host-2007)
DIRL enabled	At 17:16	lnx-1 and lnx-7 see a minor dip in traffic and recovers immediately because of automatic activation of DIRL. Host-2007 is limited and undergoes adaptive stage	Link utilization limited to 45%. TxWait – 0%	ISL utilization increased to 26% (lnx-1 + lnx-7 + Host-2007)
DIRL adapts	17:16 – 17:19	Host-2007's (culprit) read throughput is dynamically recovered to 580 MB/s. Host lnx-1's and lnx-7's throughout remained the same	Link utilization increased to 70%. TxWait – 0%	ISL utilization increased to 32% (lnx-1 + lnx-7 + Host-2007)

Benefit: Culprit host – Allowed up to an extent so that it doesn't cause congestion

Benefit: No congestion on the edge port. High link utilization

Benefit: No congestion on ISL ports. Increased ISL utilization

Figure 6-35 *Port-Monitor detects congestion and instantly starts DIRL*

When Host-2007 causes congestion, a traffic dip for Host lnx-1 and lnx-7 is because of the 10-second Tx-datarate detection granularity of Port-Monitor. Note that this traffic dip is larger than the traffic dip in Figure 6-33, because TxWait has a detection granularity of 1 second. Regardless, soon these hosts are prevented by the effect of congestion as if the culprit (Host-2007) was not even present.

Comparing DIRL with Other Approaches

DIRL has many advantages over the other approaches that are discussed in this chapter.

DIRL versus No-credit-drop Timeout

The earlier section on *No-credit-drop Timeout and Congestion Isolation in Action* demonstrates that no-credit-drop timeout works as expected. Despite its ability to alleviate the adverse effects of a slow drain device on victim devices, this feature has not seen wide adoption because many users do not want to configure a deliberate feature to drop frames in a lossless fabric. They are ok if a slow drain device becomes the cause of congestion, but the admins do not want to take ownership of intentionally dropping frames. Although not a technical reason, it explains the operational challenges of production deployments.

Unlike no-credit-drop timeout, DIRL doesn't drop any frames. DIRL uses the B2B flow control mechanism to limit the traffic rate, which is a basic function of the Fibre Channel fabrics. This benefit of DIRL alleviates the operational issue of the users, which are often more complex to solve than the technical issues.

The following are the other difference between DIRL and no-credit-drop timeout.

1. No-credit-drop timeout activates after a continuous duration of zero remaining-Tx-B2B-credits. DIRL can operate on TxWait which captures continuous and non-continuous duration of zero remaining-Tx-B2B-credits.
2. No-credit drop timeout doesn't help in congestion due to over-utilization. In contrast, DIRL helps in congestion due to over-utilization as well as congestion due to slow-drain.
3. No-credit-drop timeout is not adaptive. It drops all the frames going out of a port that's connected to a slow-drain device or sends all the frames. In contrast, DIRL adapts the traffic rate to keep congestion under control.
4. No-credit-drop timeout may be too harsh on some slow-drain devices. Some devices may not be permanent culprits. If a device reduces the rate of R_RDY during a small period (for example 10 mins), dropping its frames may severely affect it, like an OS crash. Because DIRL

doesn’t drop frames, the effect on a slow-drain device shouldn’t be that severe. For example, Figure 6-31 shows that Host-2007 shows 70% TxWait yet its Read I/O throughput is 1200 MB/s. Using no-credit-drop timeout on this server would bring down its throughput to zero, which is too harsh. DIRL only lowered the throughput to 870 MB/s.

The following are the similarities between DIRL and no-credit-drop timeouts.

1. DIRL and no-credit-drop timeout can work in single-switch fabrics.
2. They don’t require a topology or design change.
3. Their configuration is simple.
4. At the time of writing, both are available only on Cisco MDS switches.

DIRL versus Traffic Segregation Using Virtual Links

The result and operations of DIRL are different from Traffic Segregation using Virtual Link in the following ways.

1. **The effect on the culprit device**: Traffic Segregation using virtual links may reduce the scope of the congestion by lowering the number of victim devices, but it does not attempt to change the behavior of the culprit devices. In contrast, DIRL not just helps the victim devices, it eliminates or reduces the severity of congestion caused by the culprits by reducing the traffic destined for them.
2. **Spread of congestion**: Traffic segregation to Virtual Links limits the spread of congestion, but it doesn’t completely isolate the effects of the congestion to the culprit device only. Refer to the earlier section on *Scope of Congestion Segregation Using Virtual Links* for more details. How much the scope of the congestion is reduced (if any) is dependent on the topology and zoning configuration in place or traffic pattern. This makes it somewhat hard to predict the benefits depending on

which device is the culprit at the time. For example, in a fabric of 400 hosts, creating 4 virtual links and assigning 100 hosts to each, limits the effect of congestion to 100 devices as long as the hosts in each group are communicating with unique sets of targets. Affecting only 100 hosts is better than affecting all 400 hosts but this is not the complete isolation of congestion. In contrast, DIRL limits the effect of congestion only to the culprit device. In the same example, traffic segregation using Virtual Links limits the spread to 25% (100 out of 400 devices), whereas DIRL limits the spread to 0.25% (1 out of 400).

3. **Operational simplicity**: Traffic segregation has an operational challenge of creating the correct categories and assigning the devices in those categories. This task requires accurate and detailed monitoring of the environment. Only a few users may be able to segregate the traffic correctly. Others may keep struggling with the changes, keep moving the devices to different virtual links, and, as a result, may not find it simple. Traffic can be categorized based on many approaches, as the earlier section on *Categorizing Traffic for Segregation* explains. A categorization approach that works well for one user may not work well for others. For example, traffic categorization based on the speed of the host link speed is not fair when 80% of devices are connected at 16GFC, and only 20% of devices at 8GFC and 32GFC. Other approaches have similar challenges. Also, because of the dynamic nature of the production environment, an approach that works today may not be the best approach after a few months of changes. In contrast, DIRL doesn’t have any such challenges because it dynamically adapts to the congestion severity of a device in real-time.
4. **Limited use of B2B credits on ISLs**: The basic property of VLs is that they have dedicated B2B credits which can’t be shared among the VLs even if one VL needs more whereas another VL may have spare. Without VLs, all the B2B credits are available for all the flows. Imagine if a few hosts are getting microburst from All-Flash

arrays. With 1000 B2B credits on ISL ports, the burst may be absorbed within the fabric without spreading the backpressure to the storage ports. If the 1000 buffers are divided into 4 VLs, with 250 B2B credits per VL, the fabric is now four times less likely to absorb a similar traffic burst. In contrast, DIRL doesn't require any changes for reserving B2B credits.

5. **Topology**: VL-based traffic segregation only works with ISLs and at least two switches. Single-switch fabrics can't benefit from it. In contrast, DIRL works in all kinds of topologies.

To summarize the benefits of DIRL as compared to traffic segregation using virtual links:

1. DIRL actively limits the traffic going to a culprit device to eliminate or reduce congestion.
2. DIRL pinpoints the effect of congestion only on the culprit device. Storage ports and other hosts are not affected. There are no direct and indirect victims.
3. DIRL is simple to use. No categorization of devices, enabling VL, B2B credit allocation, etc. DIRL is adaptive which suits all kinds of traffic, speeds, SCSI or NVMe, and tiers of applications.
4. DIRL doesn't limit the use of B2B credits on ISL.
5. DIRL is topology independent. It works in edge-core, edge-core-edge, and single-switch fabrics.

Congestion Prevention by Notifying the End Devices

End devices and their applications may not be aware of congestion in the network. Culprits may not be aware that they cause congestion, and victims may not be aware that they continue to use a congested or degraded network path. To solve this problem, the network switches can 'explicitly' notify the end devices as soon as they detect congestion. In response, the culprit devices can take steps to prevent congestion and the victim devices can take steps to safeguard themselves from congestion such as moving their I/O traffic to a healthy path or reducing the rate of IO to a culprit device.

The idea of network devices (such as switches and routers) notifying the end devices is not new. The following are a few examples of this idea.

- Explicit Congestion Notification (ECN) for TCP/IP networks is described in RFC 3168, which was published in September 2001. An earlier proposal of ECN is described in RFC 2481, which was published in January 1999. Refer to Chapter 8 for more details on ECN.
- Quantized Congestion Notification (QCN) for Ethernet networks is described in the IEEE standard 802.1Qau, which was approved in September 2006.
- Forward Explicit Congestion Notification (FECN) and Backward Explicit Congestion Notification (BECN) in InfiniBand and Frame Relay go back to 1990s. Frame Relay is primarily a WAN technology. It’s not used in storage networks. Both Frame Relay and InfiniBand are outside the scope of this book.
- Fabric Performance Impact Notifications (FPIN) and Congestion Signals were drafted in the Fibre Channel standards in 2018 and initial implementations started emerging in 2020.

Although the exact implementation varies based on the transport type (such as TCP/IP, lossless Ethernet, or Fibre Channel), the basic idea to prevent congestion by notifying the end devices remains the same. This is because, in most production deployments, applications stay agnostic to the transport type. Typically, an application hosted on a virtualized server doesn’t even know if its block storage is physically inside the same enclosure or if it’s physically inside an array connected via a network. If connected via a network, the application remains agnostic if the transport type is FC, FCoE, RoCE, iSCSI, or NVMe/TCP.

When such a host learns about network congestion, their action(s) should remain similar regardless of the transport type. For example, if a culprit server causes congestion by asking for too much traffic, the real solution is for the server to lower its I/O rate irrespective of the transport type.

This section focuses only on Fibre Channel notifications and signals. Chapter 7 explains this concept in lossless Ethernet networks using RoCEv2 Congestion Management (RCM) and Chapter 8 explains this concept in TCP/IP networks. Knowing the similar mechanisms across different transport types helps in understanding them better, applying the same best practices, and avoiding common mistakes.

Readiness of Notifications and Signals in Fibre Channel

Before attempting to use Fibre Channel notifications and signals, set realistic expectations about the readiness of the products and their adoption in production environments.

The success of the approach of congestion prevention by notifying the end devices depends upon the following factors:

1. **Congestion detection**: At the time of writing, most Fibre Channel switches have the robust capability for detecting congestion events.
2. **Sending a notification**: At the time of writing, most Fibre Channel switches have added capabilities of sending notifications and signals after a software upgrade. Some advanced capabilities are available after a hardware upgrade.
3. **Receiving a notification**: At the time of writing, most HBAs have released newer software versions with the support of receiving notifications and signals.
4. **Congestion prevention action**: This factor is still emerging at the time of writing. Although some vendors have demonstrated initial implementations in a lab, adoption in production environments is yet to be seen.

The most important factor for the success of this approach is a precise action by the end device (the 4th factor), which isn’t ready at the time of writing. The other three factors have limited use in preventing congestion just by themselves because the same information is already available on the switches.

Overall, the approach of notifying end devices has the potential of being an effective congestion prevention mechanism. The real success, however, depends on how well the end devices take a cohesive action and the readiness of production environments.

Notifications and Signals in Fibre Channel Fabrics

In Fibre Channel fabrics, devices can be explicitly notified about an event (such as congestion) using Fabric Performance Impact Notifications (FPINs) and Congestion Signals.

FPINs and Congestion Signals are sent to only those devices that register to receive them. This registration takes place when a device connects to a Fibre Channel fabric and completes the following functions.

1. **Register Diagnostics Function (RDF):** RDF is the registration process for receiving FPINs.
2. **Exchange Diagnostics Capabilities (EDC)**: EDC is the registration process for receiving Congestion Signals.

Register Diagnostics Function

RDF is a type of ELS (Extended Link Service) frame that an end device sends to the Fabric to register itself to receive FPINs of various types identified in descriptors. Registration for RDF is successful when the Fabric returns an LS_ACC (Link Service Accept) with a list of accepted descriptors. This is explained in the FPIN section.

Note:

An FC port sends an ELS request to its neighbor switch to perform a function. The neighbor switch typically sends a Link Service Accept (LS_ACC) response. ELS requests and responses follow the rules of Exchange, Sequence, and frames. An ELS request and the corresponding response belong to a single FC Exchange. Refer to Chapter 2, section *Transforming an I/O Operation to FC frames*, for more details on Exchange, Sequence, and Frames.

Exchange Diagnostics Capability

EDC is also a type of ELS (Extended Link Service) that the two neighboring FC ports exchange before sending and/or receiving Congestion Signals indicating support for the following events.

- Link Fault: This capability indicates the degraded quality of a link using FEC counters, their thresholds, and monitoring duration.
- Congestion Signals: An FC port may agree on sending two types of Congestion Signals
  1. Warning Congestion Signal: This typically indicates a congestion event of lower severity.
  2. Alarm Congestion Signal: This typically indicates a congestion event of higher severity.

Using EDC, the neighboring FC ports agree on the following:

1. The FC ports may send and/or receive the Link Degrade and Congestion Signals.
2. The FC ports may send only warning Congestion Signals, or they may send warning and alarm Congestion Signals.
3. The neighboring FC ports agree on the frequency of sending and receiving the Congestion Signals.

Fabric Performance Impact Notification (FPIN)

FPIN is a Fibre Channel ELS (Extended Link Service) that can be sent by any of the devices in a fabric including both initiators, targets, and switches to the devices that register to receive them using RDF. FPINs from the end devices to the

Fabric can be forwarded to other devices that it is communicating with.

Instead of just calling them Notifications, the use of the prefix —Fabric Performance Impact—conveys that their scope is much wider than just congestion events. FPINs convey several other events that are related to the overall performance of the fabric, such as link integrity issues and frame delivery failures (i.e., frame drops).

FPIN can contain the following descriptor types:

1. **Link Integrity**: The Link Integrity FPINs notify the events that are related to the integrity of a link. They are of the following types:

   a. Link Failure.

   b. Loss of Synchronization.

   c. Loss of Signal.

   d. Primitive Sequence Protocol Error.

   e. Invalid Transmission Words.

   f. Invalid CRC.

   g. FEC uncorrected blocks.

   Refer to Chapter 2, section *Counters on Fibre Channel Ports*, for details on these counters. Additionally, link integrity events convey the following:

   1. Number or count of these events.
   2. Duration of detection of these events.
   3. The port in the fabric that is detecting these events.
   4. The port that's directly attached to the port where these events are detected.

2. **Frame Delivery**: The Delivery FPINs notify when a frame is discarded or dropped in the fabric. These FPIN describe when frames are not delivered, which makes these FPINs curiously misnamed. These are really a lack-of-delivery event. They convey the following:

   a. The port on which a frame discard is detected

b. The port that is directly attached to the port where the frame discard is detected.

c. The reason for frame discard, such as timeout, unable to route, or any other device-specific reasons. On Cisco MDS switches, frame dropped because of congestion-drop timeout and no-credit-drop timeout results in timeout discards.

3. **Congestion**: Congestion FPINs notify when a port detects congestion. They are sent by a detecting port to its directly attached port. They can be sent by both N_Ports and F_Ports. They contain the following information:

   a. The type of event.

   b. The event modifier

   c. The duration for which the event persists.

   d. The severity — warning or alarm.

4. **Peer congestion**: Peer Congestion FPINs notify when another port in the fabric detects congestion. Unlike Congestion FPINs that are sent to the directly attached device, peer congestion FPINs are sent to all other devices that communicate via the port that detects the congestion event. They convey the following information:

   a. The port that detects the congestion.

   b. The port that's directly attached to the port that detected the event.

   c. The type of event, which can be one of the following:

      i. Over-utilization of the link.

      ii. Slow-drain.

      iii. Lost Credit.

      iv. Any other device-specific event.

   d. Clear/None: This event is sent when the switchport stops detecting any of these conditions or if its connected device is disconnected from the fabric.

e. The duration for which the event persists.

The following are important details about FPIN:

1. A single FPIN ELS can convey one or more descriptors.
2. Switchports as well as end devices can send FPINs.
3. FPIN ELS don’t cross zone boundaries.

Congestion Signals

Congestion Signals are Fibre Channel primitives, which is a small bit sequence conveying a special function. Refer to Chapter 2, section *Special Functions — Delimiters, Primitive Signals, and Primitive Sequences*, for more details on FC primitive signals. Neighboring FC ports send and/or receive Congestion Signals only if they agreed to do so during the EDC exchange.

Like the other primitives, such as R_RDY and fill words, Congestion Signals remain local to a link. They indicate the congestion condition between two directly attached FC ports.

The benefit of using primitive signals is that they can be sent even if a link is severely congested since they are not a frame and don’t require any B2B credits. Under severe congestion, FPINs may not reach a slow-drain device. But Congestion Signals can still be sent and received.

Fibre Channel defines two types of Congestion Signals—Warning and Alarm. These signals are sent periodically until the condition resolves.

Examples of RDF, EDC, FPIN, and Congestion Signals

Refer to Figure 6-36, which shows an edge-core fabric with three initiators (Host-1, Host-2, and Host-3) and three targets (Target-1, Target-2, and Target-3). This fabric uses single-initiator-multi-target zoning scheme. Each initiator is zoned with all three targets.

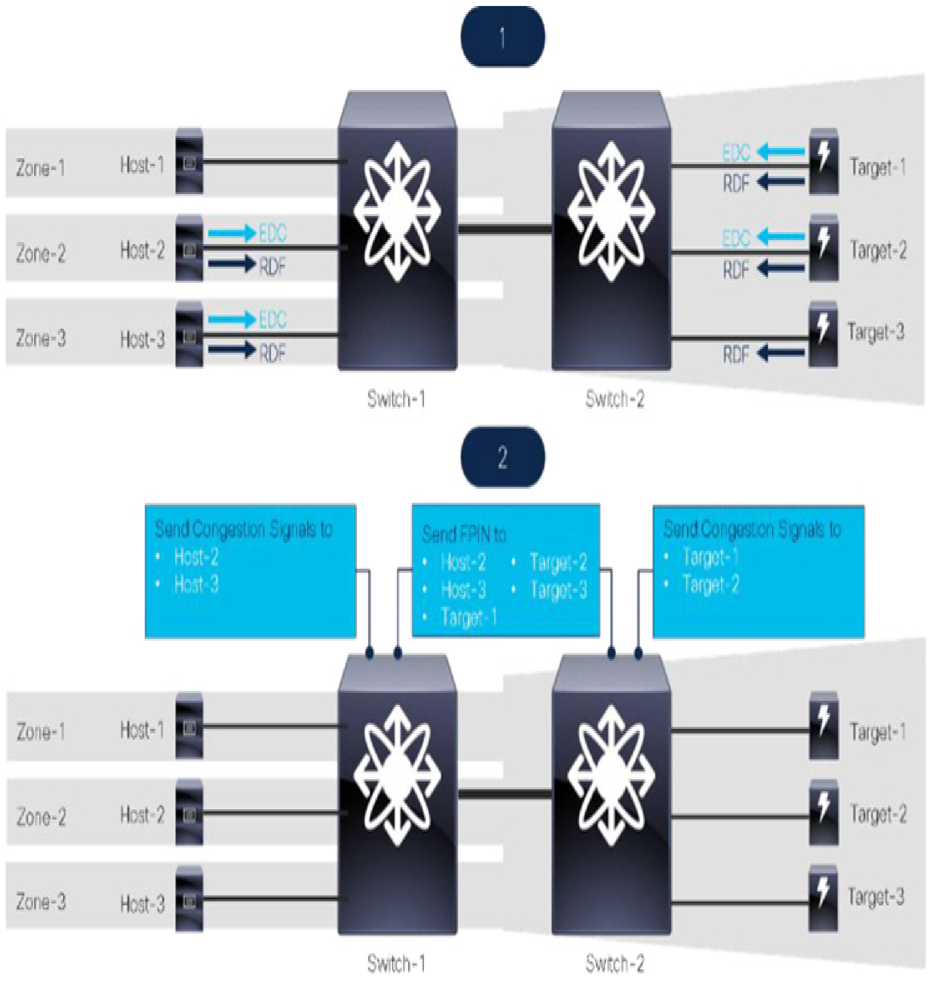


Figure 6-36 *EDC and RDF registrations for receiving Congestion Signals and FPIN*

As step-1 shows, Host-1 doesn't send EDC and RDF. Target-3 registers via RDF, but does not send EDC. The rest of the end devices send and complete EDC and RDF. The switches accept (send the proper LS_ACC) each of the RDF and EDCs. As a result, as step-2 shows, when needed, the switches will send FPINs, when necessary, to Host-2, Host-3, Target-1, Target-2, and Target-3. FPINs won't be sent to Host-1 because it didn't register via RDF. Switch-1 will send Congestion Signals to Host-2 and Host-3. Switch-1 won't send Congestion Signals to Host-1 because it didn't send EDC. Switch-2 will send Congestion Signals to Target-1 and Target-2, but not to Target-3 because it didn't send EDC.

Now, Host-1 becomes a slow-drain device (Figure 6-37). When the configured thresholds are exceeded, the switches send Peer Congestion FPINs to all three targets that are zoned with Host-1.

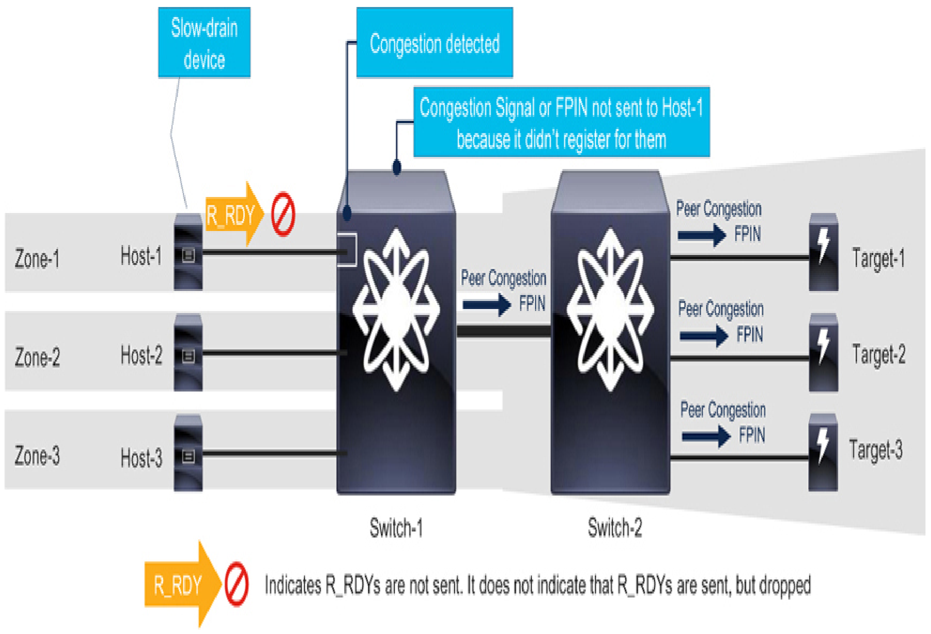


Figure 6-37 *Peer Congestion FPIN from switches to the end devices*

Later, Target-3 becomes a slow-drain device (Figure 6-38). Switch-2 sends Congestion FPINs to Target-3 but doesn't send Congestion Signals because Target-3 did not send EDC. The switches send Peer Congestion FPIN to Host-2, Host-3, Target-1, and Target-2. FPINs are not sent to Host-1 because it didn't register via RDF.

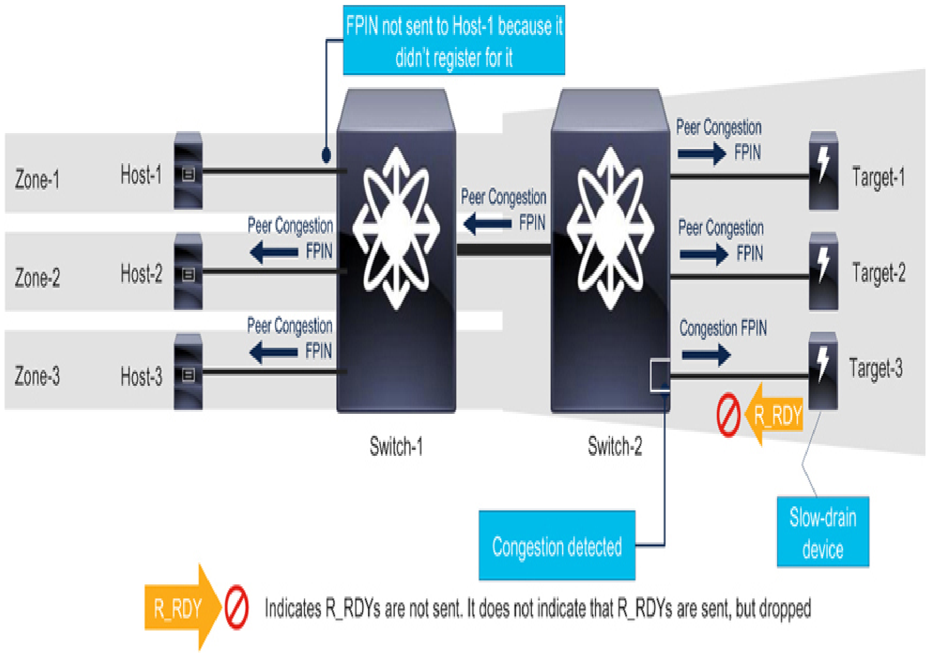


Figure 6-38 *Congestion FPIN to the slow-drain device with single-initiator-multi-target zoning scheme*

Figure 6-38 uses single-initiator-multi-target zoning scheme for explaining FPIN and signals. However, in production environments, a target wouldn't care about congestion on another target. So, sending an FPIN to it when another target is congested is of little use. Because of this reason, changing zoning to a single-initiator-single-target scheme would reduce the number of unnecessary FPINs.

Finally, refer to Figure 6-39, which shows that Host-3 becomes a slow-drain device. As a result, Switch-1 sends congestion FPINs and Congestion Signals to it because it registered via RDF and EDC. Other devices that are zoned with Host-3 (Target-1, Target-2, Target-3) are sent peer congestion FPIN but no Congestion Signals are sent to them because the event is not detected on their directly connected link.

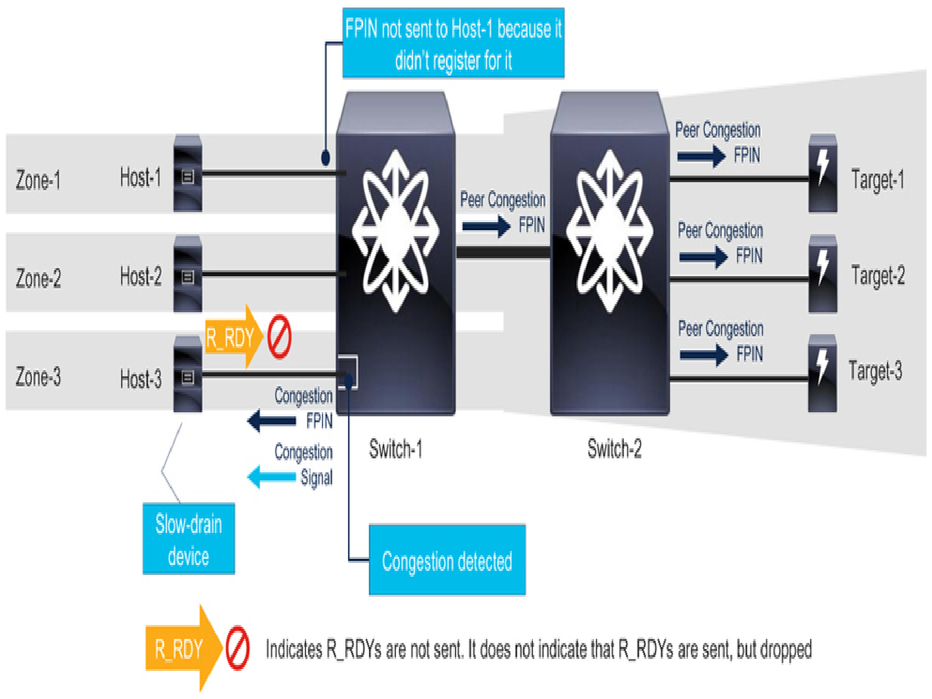


Figure 6-39 *Congestion FPIN and Congestion Signal to the slow-drain device*

In some cases, the use of Congestion Signals or FPINs to inform an end device about congestion may seem redundant. For example, in Figure 6-39, Host-3 becomes a slow-drain device and its connected switchport detects this condition and sends FPINs and Congestion Signals to it. Typically, the remaining-Rx-B2B-credits on Host-3 remain in sync with the remaining-Tx-B2B-credits on its connected switchport. Therefore, when Switch-1 detects Host-3 as a slow-drain device because of zero remaining-Tx-B2B-credits, at the same time Host-3 should be able to detect this condition itself because of zero remaining-Rx-B2B-credits. Therefore, the need to explicitly notify an end-device (using FPIN or Congestion Signals) may not be evident.

The real benefits of explicitly notifying end devices can be in the following conditions:

1. A sender's remaining-Tx-B2B-credits may not always be in sync with the receiver's remaining-Rx-B2B-credits. Such conditions may arise because of bit errors on a link. Refer to Chapter 2, section *Loss of Tx B2B Credits due to Bit Errors*, for more details. In this condition, a sender may believe there is congestion, although the receiver doesn't observe it, and thus an explicit notification in the form of FPIN or Congestion Signal is beneficial.
2. The scope of FPIN is larger than just the local link. For example, when Host-3 becomes the slow-drain device, the congestion soon spreads to the links between Switch-2 and Target-1, Target-2, and Target-3. Without peer congestion FPIN, these devices don't know the root cause and the source of congestion. With peer congestion FPIN, Target-1, Target-2, and Target-3 know that Host-3 is the culprit. This is important information if the targets decide to take preventive actions, such as applying an I/O rate limiter.
3. Finally, FPINs are not just about congestion. As previously mentioned, FPINs convey the overall performance of the fabric, such as frame drops and link integrity issues like CRC errors.

Comparing FPIN and Congestion Signals

FPIN is an FC ELS. It is sent using a normal FC frame. As a result, under congestion, FPINs are subjected to the same treatment as any other frame. If a port can't send frames due to zero remaining-Tx-B2B-credits, it can't send an FPIN either. This is where congestion Signals have merit because these are

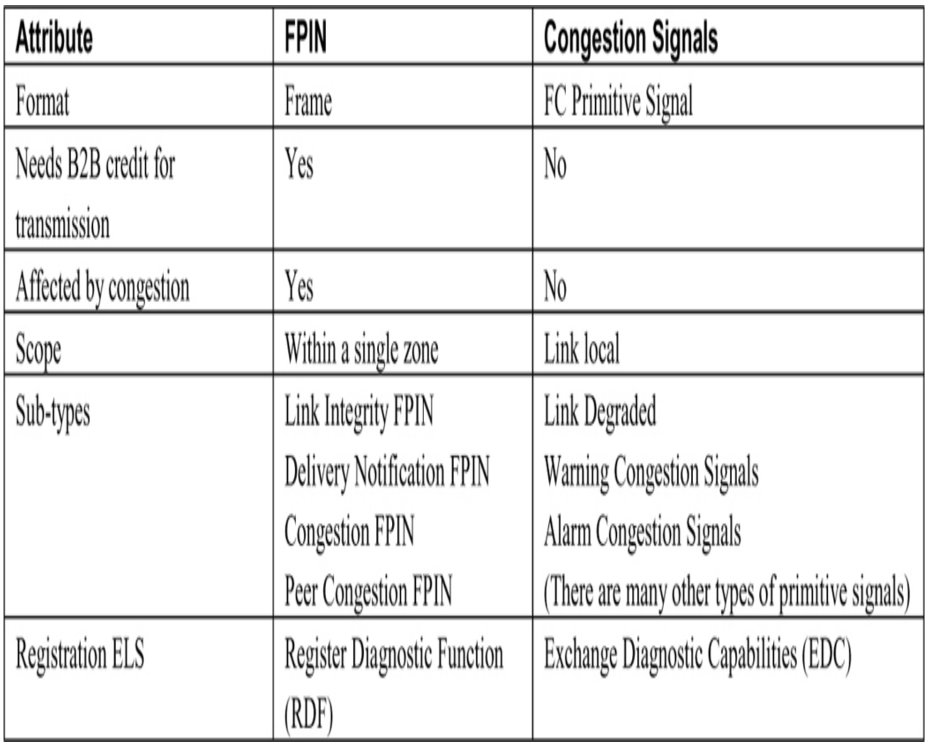
FC primitives. They can be sent even if the port has zero remaining-Tx-B2B-credits.

The other major difference is that congestion signals remain local to a link, whereas FPINs remain local to a zone.

Both Signals and FPIN can be sent for the same event types.

Table 6-5 compares FPIN and Congestion Signals.

Table 6-5 *Comparing FPIN and Congestion Signals*



Attribute	FPIN	Congestion Signals
Format	Frame	FC Primitive Signal
Needs B2B credit for transmission	Yes	No
Affected by congestion	Yes	No
Scope	Within a single zone	Link local
Sub-types	Link Integrity FPIN Delivery Notification FPIN Congestion FPIN Peer Congestion FPIN	Link Degraded Warning Congestion Signals Alarm Congestion Signals (There are many other types of primitive signals)
Registration ELS	Register Diagnostic Function (RDF)	Exchange Diagnostic Capabilities (EDC)

The Possible Results of FPIN and Signals

As the earlier section on *Readiness of Notifications and Signals in Fibre Channel* explains, the implementations are at an early stage for taking congestion prevention actions after receiving FPIN and signals. This section discusses some possible actions. Be aware that these results may not be available on the products in the market at the time of this writing. Refer to the documentation of the vendor for more details.

Logging

An end device can log when it sends or receives FPINs or Congestion Signals. This information can help in detailed troubleshooting to find the following.

- What is the problem, which can be extracted from the type of FPIN or Congestion Signal.
- Where is the problem, which can be extracted from the received FPIN that carries the source of the event (such as congestion) and the list of affected devices.
- When was the problem, which can be extracted from the timestamp of the received notifications and signals and when they stop.
- Severity of the problem, which can be extracted from severity levels such as Warning (low or moderate severity) and Alarm (high severity)

This information can be obtained directly from the switches because that's where most of the signals and FPINs originate. Most Fibre Channel switches already have robust detection and logging capabilities. So the benefit of logging already exists without FPIN and Congestion Signals, just not on the end devices. But in some cases, logging on the end devices is beneficial where different teams are involved and they may not have access to the switch logs.

Behavior Modification

An end device can change its behavior based on the type and severity of the notification.

- **Congestion notification**: A host can reduce its I/O rate. A storage port can lower the rate of sending data to the host or even reduce the burst size in the Transfer-Ready it is sending.
- **Link integrity notification**: A host can modify the behavior of the multipathing layer to stop using a degraded path. Alternatively, the host can abort the I/O instead of waiting for the timeout to act and recover faster from the situation. The host can also choose to increase the number of retries.
- **Frame drops**: A host can abort the I/O or initiate a retry instead of waiting for the timeout period, which can result

in faster recovery.

Configuring FPIN and Congestion Signals on Cisco MDS Switches

Cisco MDS switches can be configured to send FPINs and Congestion Signals using the Port-Monitor feature. As Chapter 3, section *Port-Monitor* explains, this feature monitors the switchports at a low granularity and can take automatic actions when the thresholds are exceeded on any port.

Refer to Example 6-17. The name of the Port-Monitor policy is fabricmon_edge_policy. It applies to all the ports that are of logical-type edge, which includes all the ports that connect to the end devices (operating as F_Ports) but not the ports that connect to an NPV switch. This configuration enables monitoring of TxWait on all edge ports at a poll interval of 1 second. The threshold values are specified in percent. Recall that TxWait is a counter on the port-ASICs that measures the duration for which a port has zero remaining-Tx-B2B-credits and a frame is waiting to be sent on this port. TxWait has a granularity of 2.5 microseconds. A 30% TxWait value over 1-second poll interval means that the port had zero remaining-Tx-B2B-credits for an aggregate 300 milliseconds.

As per this configuration,

- When TxWait reaches 200 milliseconds (warning-threshold) in 1-second interval, but less than 300 milliseconds (rising-threshold) the following actions are taken:

  1. A syslog and RMON alert (SNMP trap) with severity level 4 is generated. But no action is taken.

- When TxWait reaches 300 milliseconds or more (rising-threshold) in 1-second interval, the following actions are taken:

  1. A syslog and RMON alert (SNMP trap) with severity level 4 is generated.
  2. Congestion FPINs are sent on that port and Peer Congestion FPIN are sent to all the other devices zoned

to the devices logged in on the affected port. As previously mentioned, FPIN are sent only to those devices that registered to receive them via RDF.

Example 6-17 *Enabling FPIN and Congestion Signals on Cisco MDS switches*

```
MDS# configure
MDS (config)# port-monitor name
fabricmon_edge_policy
MDS (config-port-monitor)# logical-type edge
MDS (config-port-monitor)# counter txwait poll-
interval 1 delta rising-threshold
30 event 4 falling-threshold 10 alerts syslog rmon
warning-threshold 20 portguard
FPIN
MDS (config-port-monitor)#
MDS (config-port-monitor)# counter txwait warning-
signal-threshold 20 alarm-
signal-threshold 30 portguard congestion-signals
MDS (config-port-monitor)# exit
MDS (config)# port-monitor activate
fabricmon_edge_policy
MDS (config)# end
MDS#
```

DIRL versus Notifying the End Devices for Congestion Prevention

Users can start preventing congestion using DIRL after a software-only upgrade on Cisco MDS switches. In contrast, FPIN and signals require upgrading the end devices. This is not an easy task in most production environments. As previously explained, speed-mismatch and legacy/slower devices are the major reasons for congestion in storage networks. Upgrading these devices would anyway eliminate congestion to a large extent.

Even if upgrading end devices is considered, as the earlier section on *Readiness of Notifications and Signals in Fibre Channel* explains, initial implementations of the preventive action after receiving FPIN and signals are still evolving and their success in production environments is yet to be proven. In other words, FPIN and signals have a potential of an effective congestion prevention approach in the future, whereas DIRL works today on Cisco MDS switches.

Additionally, even a single culprit device can cause serious congestion in a Fibre Channel fabric affecting many other devices. Because of this, if 99% devices are able to receive FPINs and act on them to reduce congestion, but just 1% either do not support FPINs or support FPINs but do not actually take action, then those 1% of devices could still cause problems. For FPINs to really handle congestion from wherever congestion occurs 100% of devices should support it and must be able to act to reduce congestion when they receive a congestion FPIN.

The following points compare DIRL with Fibre Channel FPIN and Congestion Signals.

- **Basic premise**: DIRL rate limits the traffic coming from a culprit device thereby reducing the amount of traffic going to that device resulting in eliminating or reducing congestion. In contrast, FPIN and Congestion Signals rely on actions, such as rate limiting, by the end devices.
- **Scope**: DIRL primarily helps in preventing congestion. In contrast, FPIN may help with link integrity and frame drop conditions as well.
- **Dependency**: DIRL is fully integrated within Cisco MDS switches, and it has no dependency on the end devices. In contrast, FPIN and signals are dependent on the switches and end devices.
- **Upgrading the devices**: DIRL doesn’t need any end devices to be upgraded. DIRL is available on the existing Cisco MDS switches after a software-only upgrade. In contrast, for congestion prevention using FPIN and signals, end devices must be upgraded.
- **Availability**: DIRL is a Cisco-patented approach. At the time of writing, it’s available only on Cisco MDS

switches. In contrast, FPIN and signals are standard-based and thus, they work on non-Cisco switches as well.

Network Design Considerations

A good network design can eliminate congestion to a large extent. Some environments, despite starting with good design principles, could not follow the best practices as the networks matured. People are under time pressure for making changes and they end up making them quickly instead of correctly. Other times, people move on, and the new team members may not follow the same design principles.

This section explains design optimizations that some users employ in their storage networks for reducing the severity of congestion. As previously mentioned, most production networks have some extent of congestion, and eliminating it completely may not be possible. The aim, however, should be to reduce the severity to an extent that application performance is acceptable.

Lowering the Link Speed of Storage Ports

For avoiding congestion due to over-utilization of the host-edge links, some users lower the speed of the storage ports, especially in those environments where the cause of congestion is confirmed to be speed-mismatch and upgrading the host links is not immediately possible.

All-flash arrays can send frames at the maximum possible data rate of their link speed, such as 32GFC. But the hosts that connect at slower speeds, such as 8GFC, can't be sent these frames at the same rate. This speed mismatch results in congestion due to over-utilization of the host-edge links. Refer to the section on *Congestion due to Over-Utilization on Host-edge Links* in Chapter 2 for more details.

In such cases, lowering the speed of storage ports from 32GFC to 8GFC avoids the speed mismatch, which may avoid the over-utilization of the host-edge links, and therefore may avoid congestion caused by it. It is a ‘may’ and not a ‘will’ because the host’s I/O traffic pattern may still issue multiple large concurrent read I/O operations to multiple storage arrays. If this occurs, even if the storage arrays are at the same speed as the host, there may still be over-utilization.

The following are the reasons in favor of using this approach in production environments.

- The resources (such as the budget) for upgrading the host links are unavailable.
- Users are under a time constraint to reduce the severity of congestion and its adverse effect on the applications. Lowering the link speed of storage ports is a temporary step until they validate better approaches or upgrade the host link speeds.
- Some users believe that there is no point in making the storage arrays faster than the hosts can handle them.
- Some users believe that lowering the link speed is a tried-and-tested approach. It is not dependent on the software or hardware version of the devices, different vendors, availability of features, and potential bugs.

On the other hand, the following are the reasons against using this approach.

- Typically, newer storage arrays come up with faster link speeds. They also bring the latest benefits of All-Flash and NVMe technologies. Essentially, these are ultra-fast storage arrays, and forcing them to connect at a slower speed limits their potential.
- Some users consider this change as a waste of investments. Consider an owner of a high-performance automobile such as a Mustang GT500. The Mustang has a high-performance supercharged 5.8-liter V8 engine. Suppose the Mustang had a problem and the service department suggested that the problem would be alleviated by lowering the engine power or making it run on just four cylinders. The owner will find this as a waste of their investment in a high-performance automobile.

- Lowering the link speed of storage ports helps only when congestion is caused by the over-utilization of the host-edge links. It doesn't help when congestion is caused by a slow-drain host or storage array.
- Lowering the link speed of the storage ports increases the probability of overutilization of storage-edge links, especially if many hosts send traffic to the storage ports at the same time.
- Lowering the link speed of the storage ports does not help if the source of congestion is the targets and not the hosts.

Overall, lowering the link speed of storage ports for reducing congestion has its trade-offs. The decision of using it or not depends on the constraints of the environment. If deciding to use this approach, don't forget that it helps only with congestion due to over-utilization. Other approaches are still required to handle congestion due to slow-drain.

Edge-Core-Edge or Edge-Core to a Collapsed Core Design

Congestion in a lossless storage fabric, such as Fibre Channel, affects many end devices because they share the same switches and ISLs to send and receive traffic. For instance, Figure 6-40 shows an edge-core topology with three host-edge switches and one core switch. Each host-edge switch connects to 200 host ports and the core switch connects to 100 storage ports. In total, the fabric with 4 switches and it connects 700 end devices.

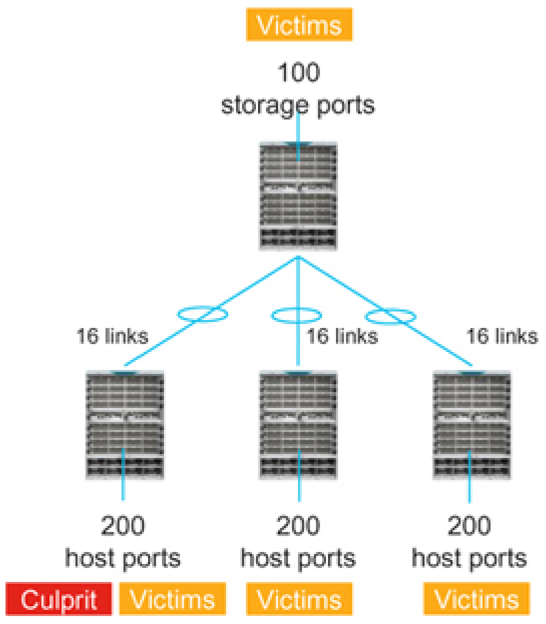


Figure 6-40 *A typical edge-core design with one culprit and 250 victims.*

Let’s assume that a host starts causing slow-drain. The congestion caused by this culprit spreads through the ISLs to the core switch and eventually to the storage ports. This situation can potentially affect the rest of the 199 hosts that connect to the same host-edge switch as the culprit host because they share the same ISLs to the core switch (same-path victims).

Additionally, some hosts that connect to other edge switches may also get affected as indirect victims because of receiving traffic from the storage ports that would be the direct victims of the culprit device. For example, let’s assume the count of such devices is 25 per host-edge switch, which leads to 50 more victims.

Overall, in this edge-core design, a single culprit device can affect up to 250 hosts.

Ensuring that the hosts on a single edge switch communicate exclusively with a set of targets on the core switch would have reduced the number of affected devices by 50 to just 200.

Next, compare the edge-core design of Figure 6-40 with the collapsed-core design of Figure 6-41.

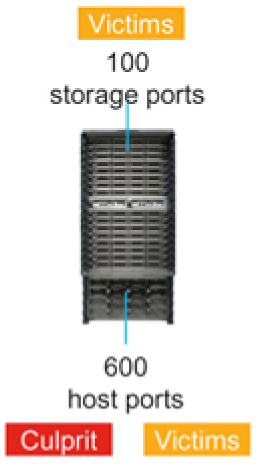


Figure 6-41 *Collapsed core design with one culprit and 50 victims*

The design of Figure 6-41 helps in limiting the spread of congestion caused by a single culprit device. All the traffic between the 600 host ports and 100 storage ports is switched by the backplane of the switch. There is no network and shared ISLs anymore.

The single culprit device may still affect the storage arrays it is communicating with as direct victims. Other hosts that receive traffic from these direct victims are also affected as indirect victims. However, the hosts that are not zoned with the direct victims storage arrays will not experience any congestion symptoms. Using the same numbers as Figure 6-40, the number of such hosts can be up to 75 because earlier 25 such hosts were connected to three host-edge switches.

Overall, this design helps in limiting the spread of congestion from up to 250 devices to only up to 75 devices. In other words, changing from a 4-switch edge-core design to a 1-switch collapsed-core design lowers the spread of congestion from 35% (250 out of 700) to only 10% (75 out of 700).

Although moving from an edge-core design to a collapsed core design reduces the spread of congestion, there are additional benefits:

- **Simple management**: Smaller fabrics are easier to manage and maintain because of fewer SAN switches to upgrade the firmware, and setup network services like AAA, SNMP, etc.

- **Increased stability**: Smaller fabrics are more stable. A single switch doesn't need distributed fabric services like FSPF, distributed zoning database, etc. It has fewer links and no ISLs which means a lesser probability of disruption due to the failure of those links.
- **Affordable**: Smaller fabrics are more affordable. It takes less rack space in data centers, consumes less power, and takes less cooling. Fewer switches also result in fewer licenses, which further lowers the cost.

The following may be the challenges in moving from an edge-core to a collapsed-core design.

- **Cable plant**: The data center cable plant may not be ready for replacing multiple switches with a single switch. For example, a 32G Fibre Channel shortwave (SW) SFP operating at 32G speed supports a maximum distance of 100 meters using OM4 cable. If the storage ports and host ports are not within this distance requirement, collapsing the design may not work.
- **Rack space**: The data center may not have enough space in a single rack to install a larger switch, although they may have space to install multiple smaller switches in different racks. The height of MDS 9718 is 26 RU, whereas the height of MDS 9710 is 14 RU.

Increased Traffic Localization to a Single Switch

Some large environments cannot be accommodated within a collapsed core design, as shown in Figure 6-41. This is because the densest Fibre Channel switch in the market can have up to 768 ports at the time of this writing.

In such cases, some users increase the traffic localization by connecting the hosts and the associated storage arrays to the same switch. Consider Figure 6-42 showing a typical edge-core design with one storage-edge switch that connects 200 storage ports and three host-edge switches, each connecting 300 hosts. In this fabric, most traffic flows across the two switches via ISLs. As previously explained, congestion caused by a culprit device affects other hosts that connect to the same host-edge switch as the culprit device (same-path victims) and hosts that connect to other host-edge switches (indirect victims) and receive traffic from the storage ports that are victimized by the culprit hosts (direct victims).

The spread of congestion can be reduced by localizing as much traffic as possible after making the change shown in Figure 6-42. In this design, storage ports and host ports are distributed across all the switches. Most hosts have their storage volumes available on the storage arrays that connect to the same switch. As a result, most traffic remains local to the backplane of the switch, and only a limited amount of traffic passes through the ISL. When a host becomes the culprit, the spread of congestion is much smaller.

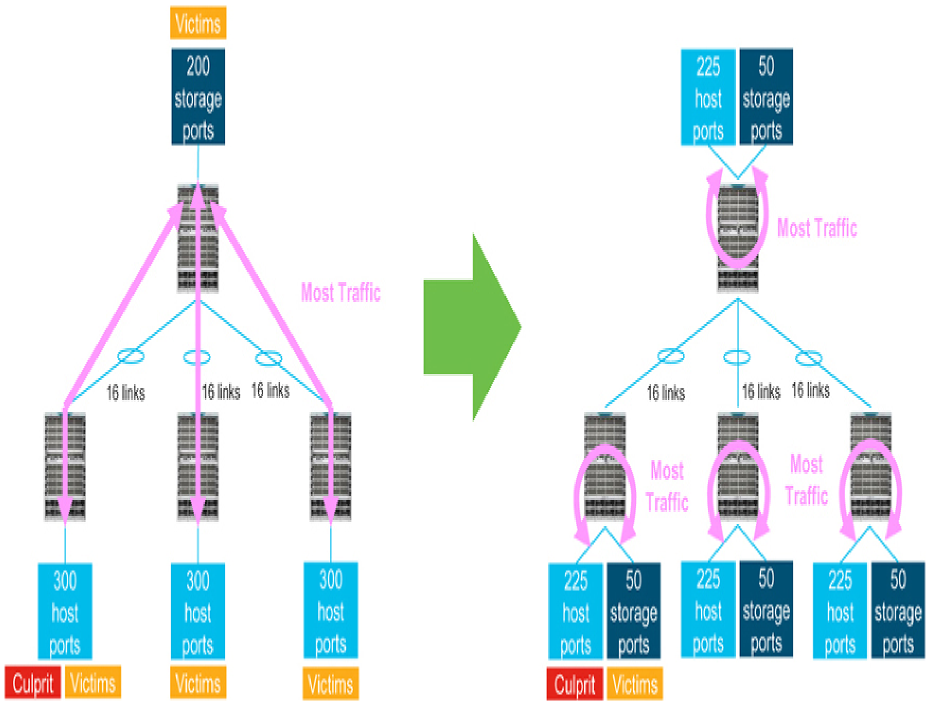


Figure 6-42 *Increasing traffic localization*

Splitting Large Fabrics into Smaller Islands

Some users take a drastically different approach to designing their storage networks. Instead of a large-shared storage fabric, they localize the network, compute, and storage resources within a single rack. Each rack contains 18-24 servers, 1-2 storage arrays, two SAN switches, and two LAN switches. With this design, the storage traffic never leaves the rack and shares only 96, 48, or fewer ports in a fabric.

Consider the example of Figure 6-43 which connects the same number of host and storage ports as the examples of Figure 6-40 and Figure 6-41. Now, only 30 host ports and only 5 storage ports connect to a 1-RU 48-port switch. This smaller design is replicated 20 times to connect 600 host ports and 100 storage ports. When a culprit device causes congestion, the spread of congestion remains limited only with the same fabric, which is up to 30 hosts. The actual number of victims will be much fewer because only those hosts are victimized that receive traffic from the storage ports which are victimized by the culprit.

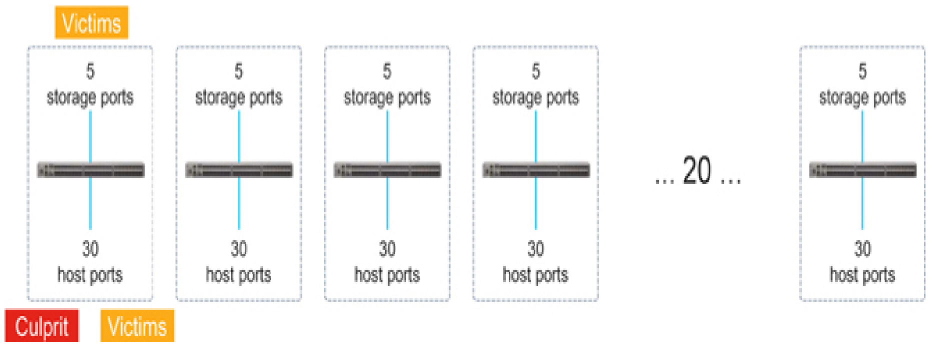


Figure 6-43 *Smaller fabrics for reducing the spread of congestion*

Although controlling the spread of congestion is a clear benefit of breaking larger fabrics into multiple smaller fabrics, this change means a total re-thinking of the storage design. You will have to analyze the operational practices of the application owner and the utilization of storage per rack. In the long term, make sure you continue to have a uniform utilization of storage. If not, the whole benefit of having a distributed storage network goes away. To manage a large number of switches and fabrics, you also need a automation framework to avoid the manual delay in provisioning new racks.

Overall, breaking large fabrics into smaller fabrics is a major design consideration, which should be adopted for the overall benefit of your organization. Controlling the spread of congestion should only be considered a pleasant side effect, and not a primary reason.

Summary

This chapter explains various approaches for reducing or eliminating congestion in a storage network.

- Congestion Recovery by dropping frames:
  - Congestion-drop timeout, which drops the frames based on their age in the switch.
  - No-credit-drop timeout, which drops the frames to a slow-drain device.
- Congestion recovery by disconnecting a culprit device.
- Congestion Segregation by traffic segregation
  - Traffic segregation to ISLs or Virtual Links: Permanent and long-term approach
  - Congestion Isolation: Automatically segregating the traffic going to a culprit device
- Congestion Prevention by
  - Dynamic Ingress Rate Limiting (DIRL) on switches
  - Host I/O limits on storage arrays
  - Notifying the end devices using Fibre Channel FPINs and Congestion Signals.

Congestion can also be eliminated or reduced by a network re-design such as lowering the link speed of the storage arrays, moving from large edge-core-edge or edge-core designs to a collapsed-core design or splitting a large fabric into smaller islands.

This chapter explains the pros and cons of these approaches, which helps in understanding which approaches may work the best in your environment. Often, using multiple approaches together makes an effective multi-tier strategy for handling congestion. For example, DIRL on TxWait, Congestion

Isolation or device disconnection on a credit-loss event, and device disconnection when too many link errors are detected.

The approaches that this chapter explains should be used only until you know the source and cause of congestion, and have enough time for finding a corrective action and implementing the change. These approaches shouldn't be used as a permanent solution.

Finally, this is the last chapter with a focus on Fibre Channel. The following chapters focus on lossless Ethernet and TCP/IP transports, but the underlying behavior of storage traffic remains the same. Because Fibre Channel continues to be the predominant transport for block storage traffic, it's important to learn from it, and apply the knowledge for preventing congestion in lossless Ethernet and TCP/IP storage networks.

References in This Chapter

- 802.1Qau — Congestion Notification — https://1.ieee802.org/dcb/802-1qau/
- RFC 2481 https://datatracker.ietf.org/doc/html/rfc2481
- RFC 3168 https://datatracker.ietf.org/doc/html/rfc3168
- Dell Technologies Whitepaper — Congestion Spreading and How to Avoid It (H17762.5)
- Dell Technologies Whitepaper — Getting the Most Out of your 32GFC Storage (H18140.1)
- FIO: https://github.com/axboe/fio
- Vdbench: https://www.oracle.com/technetwork/server-storage/vdbench-1901683.pdf
- Cisco MDS DIRL SAN Congestion Software Performance Validation: https://www.cisco.com/c/dam/en/us/products/collateral/storage-networking/dcn-miercom-report-for-san.pdf
- https://uspto.report/patent/grant/10,992,580
- VMware vSAN: https://www.vmware.com/products/vsan.html

- Cisco VSAN: https://www.cisco.com/c/en/us/td/docs/dcn/mds9000/sw/9x/configuration/fabric/cisco-mds-9000-nx-os-fabric-configuration-guide-9x/configuring_and_managing_vsans.html
- Cisco Unified Computing System Solution Overview: https://www.cisco.com/c/en/us/products/collateral/servers-unified-computing/solution-overview-c22-744677.html
- INCITS 488-2016, FC-FS-4, Fibre Channel Framing and Signaling - 4
- INCITS 545-2018, FC-FS-5, Fibre Channel Framing and Signaling - 5
- INCITS 512-2015, FC-PI-6, Fibre Channel Physical Interfaces - 6
- INCITS 533-2016, FC-PI-6P, Fibre Channel Physical Interfaces - 6P
- INCITS 479-2011, FC-PI-5, Fibre Channel Physical Interfaces - 5
- IEEE 803.2-2018, IEEE Standard for Ethernet
- Cisco Slow-Drain Device Detection, Troubleshooting, and Automatic Recovery White Paper
- Cisco MDS NX-OS Software Configuration Guides.

Chapter 7. Congestion Management in Ethernet Storage Networks

This chapter covers the following topics.

- Ethernet flow control.
- Priority-based Flow Control at layer 2 and layer 3 of the OSI model.
- Congestion spreading in lossless Ethernet networks.
- Detecting congestion in lossless Ethernet networks.
- Preventing congestion in lossless Ethernet networks.
- RoCEv2 Congestion Management (RCM).
- Congestion management of lossless traffic with VXLAN.

This chapter focuses on the storage protocols that use lossless Ethernet networks without TCP, such as FCoE, RoCE, and RoCEv2. Chapter 8, "Congestion Management in TCP Storage Networks," focuses on storage protocols that use TCP transport in a lossy Ethernet network, such as iSCSI and NVMe/TCP.

Ethernet Flow Control

Ethernet flow control comes in two flavors.

1. **Link-Level Flow Control (LLFC)**: LLFC enables flow control between directly connected devices for all the

traffic on a link. LLFC is an IEEE standard (IEEE 802.3x)

2. **Priority-based Flow Control (PFC)**: PFC enables flow control between directly connected devices for specific traffic classes, while other traffic classes continue to operate without flow control. PFC is also an IEEE standard (IEEE 802.1Qbb).

LLFC is similar to the Fibre Channel B2B flow-control mechanism, which flow controls all traffic on a link (Chapter 2, "Understanding Congestion in Fibre Channel Fabrics," section *Fibre Channel Flow Control*). In contrast, PFC is similar to the ER_RDY (or VC_RDY) flow-control mechanism when a Fibre Channel link operates with multiple virtual links or circuits (VLs or VCs) (Chapter 6, "Preventing Congestion in Fibre Channel Fabrics," section *Understanding Virtual Links*). In Fibre Channel fabrics, all traffic needs lossless behavior, hence all the VLs enable flow control, whereas in Ethernet networks only some traffic classes are flow-controlled, while traffic in other classes operates without flow control. This helps in achieving lossy and lossless behavior on the same Ethernet link.

As mentioned in Chapter 1, in the section on *Lossless Networks*, Ethernet flow control (or a lossless network in general) does not guarantee that frames are not dropped. Frames are still dropped when they are corrupted due to bit errors and when they are held up in the buffers for a long duration when congestion becomes severe. These details are explained later in the section on *Congestion Recovery by Dropping Frames*.

How Ethernet Flow Control Works?

Ethernet uses a special frame, called a Pause frame for flow control. The content of the Pause frame, when it is sent, and how often it is sent allows the directly connected sender to pace its transmission rate to avoid buffer overrun on the receiver. This avoids packet drops due to a lack of free buffers on the receiver and therefore, achieves a lossless network.

Pause Time

Pause frames have a field, called a quanta, to convey the duration for which the frame transmission must be stopped (called pause time). This quanta ranges between 0 and 65535 (16-bit maximum value 0xFFFF). To calculate the pause time, the quanta value is multiplied by the time taken to transmit 512 bits on the link. This scheme helps in a consistent implementation regardless of the operational speed of the link. It also ensures that the maximum pause time decreases as the link speed increases, which is needed to accommodate the faster transmission rates on high-speed links.

As the following equation shows, to convert pause quanta to seconds, multiply it by 512 and divide by the port speed in bits per second.

> Pause time in seconds = (Pause quanta x 512) / Port speed in bits per second

For example, on a 10 GbE link, the maximum pause time conveyed by a single Pause frame is 3.355 ms ((65535 x 512) / (10 x 10^9)). When a 10 GbE port receives a Pause frame with quanta 0xFFFF, it can stop transmission for 3.355 ms. Likewise, the maximum pause time on a 100 GbE link is 0.355 ms.

The traffic receiver, after sending a Pause frame with non-zero quanta, can request the traffic sender to restart the transmission in two ways:

1. Wait for the duration represented in the last Pause frame sent. For example, a 10 GbE port could simply wait 3.355 ms.

2. Send a new Pause frame with zero quanta, which requests the traffic sender to immediately resume transmission. Because of this reason, a Pause frame with zero quanta is also known as a Resume frame or an Un-Pause frame.

The traffic sender in the state of "pause" because of receiving a Pause frame with non-zero quanta immediately resumes transmission under the following two conditions:

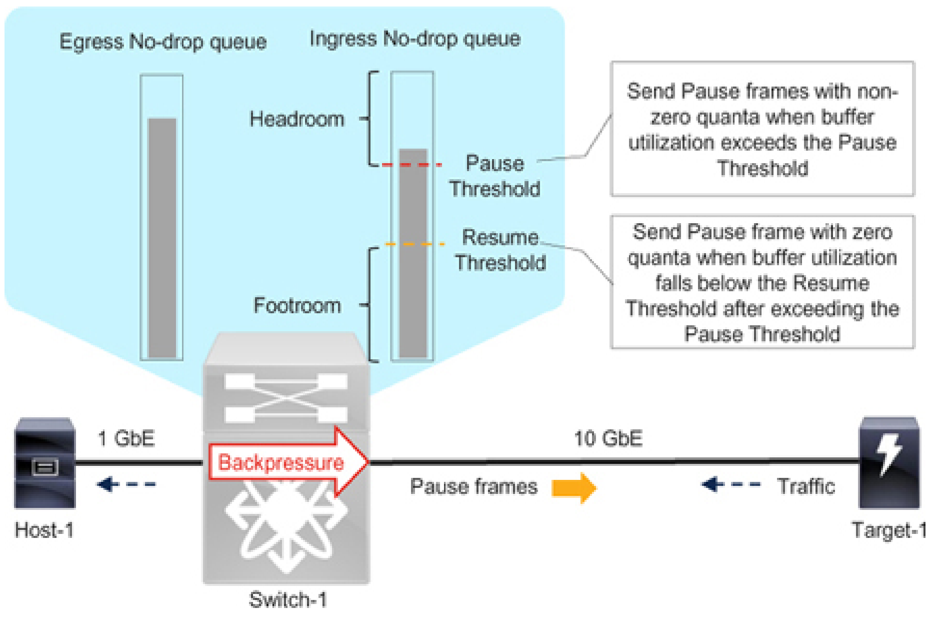
1. The duration specified in the Pause frame expires. For example, a 10 GbE port after 3.355 ms of receiving a Pause frame with a quanta of 65535.
2. It receives a Pause frame with a quanta set to zero. Again, this can be known as a Resume or Un-Pause frame.

This means that Pause frames have a dual purpose. A Pause frame with non-zero quanta stops or pauses the traffic, whereas a Pause frame with zero quanta starts or resumes the traffic.

Next, let's understand when these Pause frames are sent.

When are the Pause Frames Sent?

Figure 7-1 shows a speed-mismatch scenario between two ports of Switch-1 that connect to Target-1 at 10 GbE and Host-1 at 1 GbE. Switch-1 allocates buffers for lossless traffic and organizes these buffers into an ingress no-drop queue. For the sake of simplicity, only one no-drop queue is shown assuming that only one traffic class needs lossless behavior.



Egress No-drop queue
Ingress No-drop queue
Headroom
Pause Threshold
Resume Threshold
Footroom
Send Pause frames with non-zero quanta when buffer utilization exceeds the Pause Threshold
Send Pause frame with zero quanta when buffer utilization falls below the Resume Threshold after exceeding the Pause Threshold
1 GbE
Backpressure
10 GbE
Pause frames
Traffic
Host-1
Switch-1
Target-1

Figure 7-1 *Ethernet flow control with Pause Threshold and Resume Threshold*

When Target-1 starts transmission of lossless traffic at 10 Gbps, Switch-1 can transmit to Host-1 only at 1 Gbps. This difference in traffic rates results in excess frames, which start consuming the buffers of the no-drop queue on Switch-1.

When Switch-1’s buffer fills up more than a threshold (called Pause Threshold), it sends a Pause frame with non-zero quanta to Target-1. When Target-1 receives this Pause frame, it stops transmitting traffic for a specific time interval, called pause time as explained in the previous section. Consequently, Switch-1’s buffer utilization does not increase further, rather starts decreasing as transmission to Host-1 continues. Eventually, when Switch-1’s buffer utilization falls below the Resume Threshold, Switch-1 sends a Pause frame with zero quanta, which is a signal for Target-1 to resume traffic transmission. If Switch-1’s buffer utilization does not fall below the Resume Threshold within the pause time of sending the first Pause frame with non-zero quanta, it continues to send more Pause frames with non-zero quanta to stop or pause transmission from Target-1 until Switch-1’s buffer utilization falls below the Resume Threshold.

Target-1 eventually resumes transmission either because the duration specified in the last Pause frame with non-zero quanta expires, or it receives a Pause frame with zero quanta. But its faster transmission rate again leads to filling up the no-drop queue on Switch-1 and soon exceeds the Pause Threshold. This is when Switch-1 again sends a Pause frame with non-zero quanta to Target-1 and the same events repeat.

Finally, tight coordination between the Pause Threshold and the Resume Threshold of Switch-1 and sending of the Pause frames (non-zero quanta) and Un-Pause frames (zero quanta) achieves Ethernet flow control.

Note the following points:

1. Ethernet flow control in Figure 7-1 results in rate equalization between the ingress port (connected to Target-1) and the egress port (connected to Host-1). If the links are monitored in this state, no Pause frames flow on the Host-1 link, whereas many Pause frames flow from Switch-1 to Target-1.

2. If flow control is not enabled in Figure 7-1, Switch-1 would drop any excess frames from Target-1. This would make it a lossy network.

3. Although the value of quanta can range anywhere between 0 and 65535, most products (including Cisco MDS switches, Nexus switches, and UCS) set it to the maximum possible value of 65535. Setting the time quanta to a smaller value will require a receiver to predict the duration in which it will be ready to receive more frames. This kind of prediction is complex to implement. Most products keep the implementation simple by setting the value of the pause quanta to the maximum value of 65535, and then, sending an Un-Pause frame (quanta of 0) when they become ready to receive traffic. Because of these reasons, Cisco MDS switches, Nexus switches, and UCS do not allow configuring the pause quanta values.
4. It would be incorrect to assume that when a Pause frame is sent with a non-zero quanta the traffic is stopped for time represented by the quanta. Usually, a Pause frame with a non-zero quanta is followed quickly by an Un-Pause or Resume frame. The maximum time the traffic is paused is really the time between the Pause and Un-Pause frames. On a 10 GbE link, this may be 0.05 microseconds (time to receive 64-byte Pause frame), or it may be 3.355 ms or anything in between. If the buffer utilization is still above the Pause threshold, another Pause frame can be sent with a non-zero quanta after an implementation-dependent amount of time that is less than the time represented by the quanta. This Pause frame ‘extends’ the time the traffic is paused.
5. A Pause frame does not preempt the transmission of a frame that has already begun.
6. The Pause Threshold is also known as the XOFF Threshold, and the Resume Threshold is also known as the XON Threshold. The use of X in XOFF and XON is an acronym for Transmission. Because Pause frames flow in the opposite direction of the traffic, the XOFF Threshold is associated with stopping the traffic, whereas the XON Threshold is associated with restarting/resuming the traffic. Generally speaking, XOFF is the same as “pause the traffic” and XON is “un-pause the traffic”. Remember this detail while troubleshooting

congestion issues because the XOFF and XON terminology may also be used.

Ingress and Egress Queues

Cisco Nexus 9000 switches have egress buffering architecture. This means that queues are maintained on the egress ports.

Only for no-drop class, additional buffers are reserved for all the ingress ports that can receive traffic in the no-drop class. As Figure 7-1 shows, no-drop traffic is queued to ingress queues (with Pause Threshold and Resume Threshold) in addition to the egress queues.

During congestion, first, the egress queues start filling up. For lossy traffic, any new incoming frames are dropped if these egress queues are full.

However, for lossless traffic, before the no-drop egress queue is full, an internal backpressure is applied to the ingress queue (with Pause Threshold and Resume Threshold). This results in any new incoming traffic consuming the ingress queue instead of being dropped. The utilization of the ingress queue controls the ingress traffic rate by sending the Pause frames as explained earlier.

These ingress queues have Pause Threshold and Resume Threshold, which are different from the Active Queue Management (AQM) mechanisms, such as Weighted Random Early Detect (WRED) of the egress queues. The ingress and egress queues on a port of the Cisco Nexus 9000 switch can be displayed using the NX-OS command **show queuing interface**.

All traffic is subjected to the egress queuing on Cisco Nexus switches and hence, AQM is relevant for lossy and lossless traffic. The handling of the egress queues and AQM mechanisms are explained in Chapter 8, the section on *Switch Buffer Management*. This chapter primarily focuses on the handling of the ingress queues (with Pause Threshold and Resume Threshold). Understanding the handling of both queues (egress and ingress) is crucial because, as mentioned, the ingress queue is used only after the egress queue fills up.

Hence, monitoring the utilization of the egress queue also provides a key indication of congestion. Refer to Chapter 8, the section on *Queue Depth Monitoring and Microburst Detection* for further details.

Location of Ingress No-Drop Queues

Ingress queues for no-drop traffic are maintained by all the ports in a lossless Ethernet network. For the sake of simplicity, Figure 7-1 shows ingress no-drop queue(s) only at one location, but in reality, all ports have ingress no-drop queue(s).

1. Figure 7-1 already shows ingress no-drop queue(s) on Switch-1 for receiving traffic from Target-1. The utilization of this no-drop queue controls sending Pause frames to Target-1.
2. A similar ingress no-drop queue(s) exists on Switch-1 for receiving traffic from Host-1. The utilization of this no-drop queue controls sending Pause frames to Host-1.
3. Target-1 creates an ingress no-drop queue(s) for receiving traffic from Switch-1. The utilization of this no-drop queue controls sending Pause frames to Switch-1.
4. Host-1 creates an ingress no-drop queue(s) for receiving traffic from Switch-1. The utilization of this no-drop queue controls sending Pause frames to Switch-1.

Number of Ingress No-Drop Queues Per Port

Typically, a no-drop traffic class needs one no-drop queue per port. More one than one no-drop queue can also be created based on use cases, such as for carrying FCoE and RoCE traffic via the same link. Multiple no-drop queues have their own Pause Threshold and Resume Threshold. The maximum number of no-drop queues on a device depends on its capabilities. For example, Cisco Nexus 9000 switches support up to three no-drop queues. But there are more considerations based on the maximum frame size and length of a link. These limits apply because a no-drop queue requires buffer reservation, and every device has a finite buffer space. Refer to the documentation of the devices in your environment, but overall, be aware of these limits and plan accordingly.

Implementation Differences and The Scope of this Book

Some implementations, although less common, continuously send Pause frames with zero quanta when there is no congestion. In other words, they send Un-Pause frames even if their buffer utilization is less than the Resume Threshold. This is unnecessary because just one Un-Pause frame is enough to resume traffic and there is no need to send them continuously unless a Pause frame with non-zero quanta is sent in between. Although such implementations do not violate the standards, this unnecessary action makes congestion detection almost impossible when combined with the inability to report Pause and Un-Pause frames separately and the inability to report the duration of traffic pause (TxWait/RxWait). Refer to the later section on *Congestion Detection Metrics* for more details on these metrics. Also, Un-Pause frames that are sent continuously in large numbers may lead to a noticeable link utilization because these are actual frames that take bandwidth. This type of implementation is outside the scope of this book. Most congestion detection and troubleshooting techniques explained in this book do not apply to such implementations.

The scope of this book is on the implementations that send only one (or a few) Un-Pause frame (zero quanta) to resume traffic and do not continuously send Un-Pause frames when the buffer utilization is below the Resume Threshold. This is the most common type of implementation, including that of Cisco MDS switches, Nexus switches, and UCS. The scope of this book is only on this type of implementation.

The following are the details about the implementation of Cisco MDS, Nexus, and UCS:

1. Send one Pause frame with maximum pause quanta when the Pause Threshold is exceeded.
2. Send the next Pause frame with maximum pause quanta after 50% of maximum quanta expires if buffer utilization

is higher than the Resume Threshold.

3. If buffer utilization falls below the Resume Threshold, send one Pause frame with zero quanta (Un-Pause). There is no subsequent transmission of the Pause frame with zero quanta. If this Pause frame is corrupted and lost, traffic resumes after the expiry of the pause time conveyed by the earlier Pause frame.

Clearly, the Cisco implementation sends Pause frames only when necessary. This conservative approach of sending Pause frames makes the foundation of detecting congestion by counting the number of Pause frames, such as used by the UCS Traffic Monitoring (UTM) app, explained in detail in Chapter 9.

Pause Threshold and Resume Threshold

This section explains the significance of the Pause Threshold and Resume Threshold for the correct headroom and footroom buffer size. These are shown in Figure 7-1.

The buffer (queue) size depends on the capability of a device. Every PFC-enabled port must reserve the buffers. Therefore, the queue size is a function of the total buffer space on that device and the number of PFC-enabled ports.

Pause Threshold

As mentioned, a traffic receiver sends a Pause frame with non-zero quanta when its buffer (queue) utilization exceeds the Pause Threshold.

Resume Threshold

As mentioned, a traffic receiver sends a Pause frame with zero quanta (Un-Pause) when its buffer (queue) utilization falls below the Resume Threshold.

The Pause Threshold must be larger than the Resume Threshold. The overall buffer (queue) size must be large enough for the sufficient difference between the Pause Threshold and Resume Threshold.

Headroom

The space between the buffer (queue) size and the Pause Threshold is called the headroom.

The Pause Threshold should be sufficiently less than the maximum buffer size (or queue size) to maintain adequate headroom. Insufficient headroom may lead to packet drops. Pause Threshold should not be too low either. If this happens, the sender is paused too soon while headroom remains unused, which may result in lower link utilization than expected.

A traffic-receiver sends a Pause frame with non-zero quanta at the Pause Threshold instead of when its buffers are completely consumed (Figure 7-1) because it must maintain enough "headroom" to accommodate:

- The inflight frames that are already on the wire.
- The frame that is already being scheduled to be transmitted by the sender and can't be interrupted.
- The delay caused by the serialization and propagation of the Pause frame from the traffic-receiver to the traffic-sender.

As distance increases, the traffic-receiver must increase its headroom because more frames can be in flight, and it will take longer for the Pause frame to reach the traffic-sender. But buffers are a finite resource, and thus the maximum supported distance of a lossless Ethernet link depends on the capabilities of its endpoints. Ingress packets that can't be accommodated in the available headroom at that time are dropped. This is a deviation from the behavior of a lossless network. Adequate headroom should avoid this condition.

Footroom

Buffer space below the Resume Threshold is called the footroom.

Resume Threshold should be sufficiently larger than zero to maintain adequate footroom such that there are enough frames in the no-drop queue while the Un-Pause frame makes its way to the traffic sender, that device resumes transmission, and those frames arrive. If the Resume Threshold is too low, the no-drop queue will be empty for some time between sending of Un-Pause frame and receiving the traffic. This may lead to lower than expected link utilization.

Configuring Buffer Size, Pause Threshold, and Resume Threshold

For most practical purposes, there should not be a need to change the buffer (queue) size, Pause Threshold, and Resume Threshold for short-distance intra-data center links. Most products, including Cisco MDS switches, Nexus switches, and UCS, configure the tested values by default for the supported length (such as 100 meters) of a link. Refer to the product documentation for details.

The default buffer size and thresholds on Cisco Nexus 9000 switches are adequate for a cable length of 100 meters. If there is a need to change these values, such as for cables longer than 100 meters, as Example 7-1 shows, change the thresholds using the command **pause buffer-size** on Cisco Nexus switches. Different types of devices may use different values based on their buffer availability and architecture. Hence, use Example 7-1 only for generic understanding, but do not directly use the values without consulting the product documentation.

Example 7-1 *Configuring buffer size, Pause Threshold, and Resume Threshold on Cisco Nexus 9000 switches*.

```
policy-map type queuing INPUT_Q
  class type queuing RDMA
    pause buffer-size 120000 pause-threshold 46000
resume-threshold 32000
```

In Example 7-1:

1. The values in the command **pause buffer-size** are specified in bytes.
2. The buffer-size (120,000 bytes) is the queue size. Pause Threshold is 46,000 bytes, and Resume Threshold is 32,000 bytes.

3. The difference between buffer-size and pause-threshold is the headroom (120,000 – 46,000 = 74,000 bytes). As explained earlier, if the headroom is insufficient, ingress packets that can't be accommodated in it at that time are dropped.
4. The resume-threshold is the footroom (32,000 bytes). As explained earlier, insufficient footroom may lead to lower link utilization than should occur.
5. Based on the maximum frame size, the configuration of these thresholds should accommodate at least a minimum number of frames. The default Ethernet payload is 1500 bytes resulting in a frame size of 1522 bytes after accounting headers. But storage traffic typically requires a larger frame size. For example, FCoE frames can be up to approximately 2300 bytes. RoCE frames can be up to approximately 2 KB or 4 KB. Ethernet jumbo frames can be up to 9216 bytes. In Example 7-1, the headroom of 74, 000 bytes can accommodate approximately eight full-size jumbo frames of 9216 bytes, although not all the frames are of the same size.

As Example 7-2 shows, use the command **show queuing interface** for verifying the Pause Threshold, Resume Threshold, and headroom for a no-drop queue on Cisco Nexus 9000 switches.

Example 7-2 *Verifying buffer size, Pause Threshold, and Resume Threshold on Cisco Nexus 9000 switches*.

```
switch# show queuing interface ethernet 1/1
<snip>
Ingress Queuing for Ethernet1/1
----------------------------------------------------
-
QoS-Group#                       Pause
             Buff Size           Pause Th         Resume Th
----------------------------------------------------
-
       7              -               -               -
       6              -               -               -
       5              -               -               -
```

```
      4          -          -          -
      3     120000      46000      32000
      2          -          -          -
      1          -          -          -
      0          -          -          -
<snip>
```

Long Distance Links with PFC

The use of long-distance lossless Ethernet links is not as common as short-distance links. Most intra-datacenter links using short-range SFPs are under a few hundred meters. The devices that support lossless Ethernet should have enough buffers for these intra-datacenter links. However, if you come across a long-distance Ethernet link enabled with PFC, verify the support on the endpoints and increase the buffer size, Pause Threshold, and Resume Threshold by following the vendors' guidelines.

As mentioned earlier, the default buffer size, Pause Threshold, and Resume Threshold on Cisco Nexus 9000 switches are adequate for a cable length of 100 meters. Cisco Nexus 93180YC-FX supports FCoE via a 10 km link after making a configuration change. For any other use case and longer distances, follow the official documentation.

The general concept is that all lossless networks have distance limitations because they need additional buffers as the distance increases. This is true of Fibre Channel as well as lossless Ethernet. For Fibre Channel, there needs to be an increase in the number of B2B credits, which are directly associated with physical buffers. For PFC/LLFC on Ethernet, there must be to be adequate buffer space above the Pause Threshold (headroom) as well as below the Resume Threshold (footroom).

The effects of insufficient buffers in Fibre Channel and lossless Ethernet are different. In Fibre Channel, a lack of B2B credits leads to lower than expected link utilization because the traffic sender is waiting for credits much of the time. In lossless Ethernet with PFC, there are two effects. First, if the headroom is insufficient, then packets may be dropped in the no-drop traffic class. Second, if the footroom is insufficient, the result is similar to a lack of B2B credits in Fibre Channel, which results in underperformance because the link never reaches the expected traffic rate in situations where there are even small amounts of pauses.

This section provides an academic explanation of calculating the headroom size for a lossless Ethernet link. This explanation is based on IEEE 802.1Qbb standard for Priority-based Flow Control. It is good information for a curious reader, but practically, always follow the recommendation of the vendor of the devices in your environment because of two key reasons. First, multiple components are implementation dependent, and the vendors do not share such details publicly. Second, the practical solution must go through certification and should carry support from the vendors.

Pause Threshold for Long Distance Links

This section uses the following basic concepts:

- **Bit Time (BT)**: It is the time taken to transmit one bit. It is the reciprocal of the bit rate. For example, BT of a 10 GbE port is 1/10,000,000,000 seconds or 0.1 nanoseconds and that of 100 GbE port is 0.01 nanoseconds.
- **Pause Quanta**: It is the time taken to transmit 512 bits. In other words, it is 512 BT.
- **Inter-Frame Gap (IFG)**: Ethernet has an inter-frame gap of 12 bytes. It takes 96 BT for transmission.
- **Preamble and Start Frame Delimiter (SFD)**: An Ethernet frame starts with 7 bytes of preamble and 1 byte of SFD. These 8 bytes are typically not counted in an Ethernet frame size of 1522 bytes. Collectively, the preamble and SFD take 64 BT for transmission.

As Figure 7-2 explains, the headroom on a traffic receiver should be large enough to accommodate frames during the following delay factors.

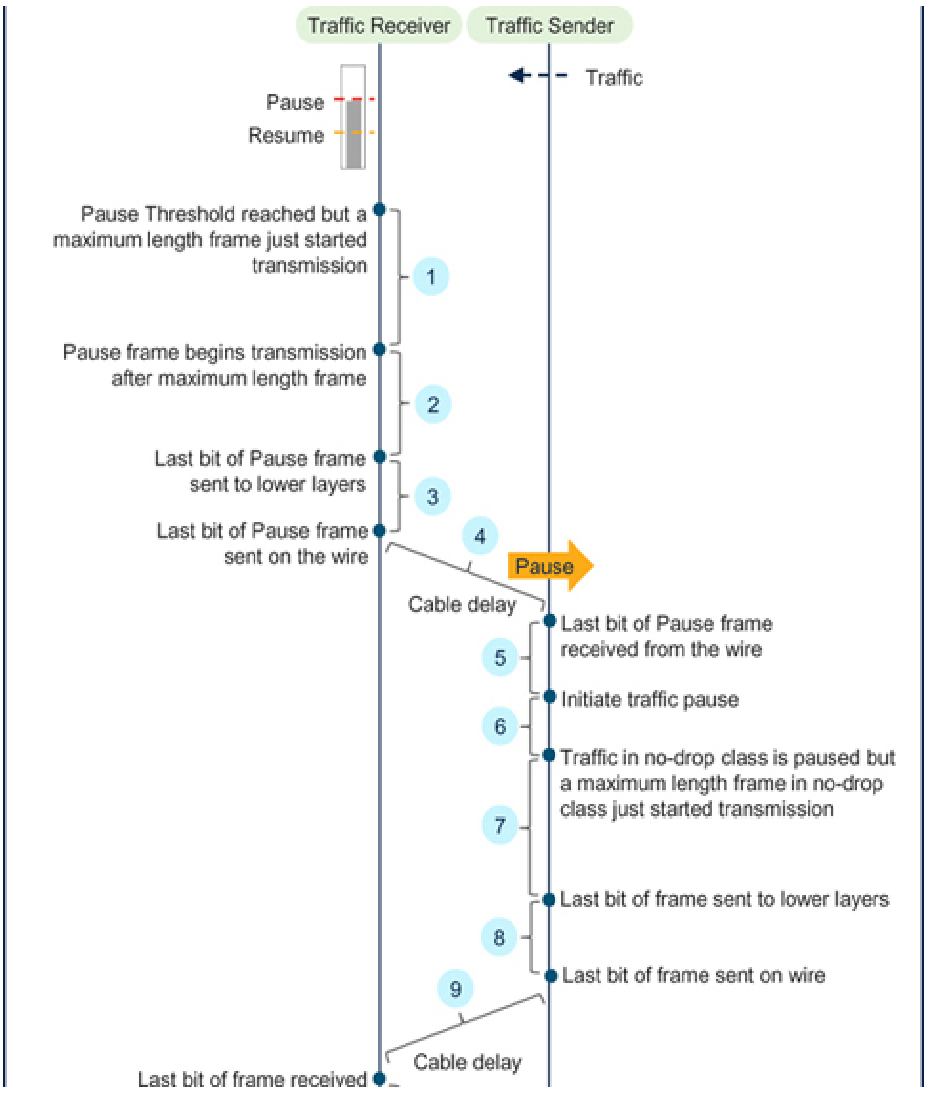

Traffic Receiver
Traffic Sender
Traffic
Pause
Resume
Pause Threshold reached but a maximum length frame just started transmission
1
Pause frame begins transmission after maximum length frame
2
Last bit of Pause frame sent to lower layers
3
Last bit of Pause frame sent on the wire
4
Pause
Cable delay
Last bit of Pause frame received from the wire
5
Initiate traffic pause
6
Traffic in no-drop class is paused but a maximum length frame in no-drop class just started transmission
7
Last bit of frame sent to lower layers
8
Last bit of frame sent on wire
9
Cable delay
Last bit of frame received

Figure 7-2 *Worst-case delay for calculating the headroom for PFC.*

1. **Delay due to transmission of a frame of maximum length (D-Max-Frame-Len)**: The queue on the traffic receiver reached the Pause Threshold. But, at the traffic receiver, a frame just started transmission. A Pause frame does not preempt the transmission of another frame. Hence, the transmission of the Pause frame is delayed until the other frame is transmitted. This delay accounts for the maximum frame length of all the traffic classes. Considering the maximum frame length of 9216 bytes, this delay can be up to 73728 BT (9216 x 8). Adding 96 BT for IFG and 64 BT for preamble and SFD, the total delay can be up to 73888 BT.

2. **Delay due to transmission of Pause frame (D-Pause):** The size of a Pause frame is 64 bytes, and therefore it takes 512 BT for transmission. Adding 96 BT for IFG and 64 BT for preamble and SFD, the total delay is 672 BT.
3. I**nterface delay for transmitting the Pause frame (D-Intf):** This is the delay caused by the lower layers while transmitting and receiving a frame. Ethernet standards define the upper value of this interface delay. For example, maximum delays of 8192 BT for 10 GbE, 6144 BT for 25 GbE, 24576 BT for 40 GbE, 122880 BT for 100 GbE. These values include transmit and receive delay. There are different implementations at the same Ethernet speed, and each may have a different interface delay. For further details, refer to IEEE Standard for Ethernet 802.3 and search for delay constraint.
4. **Cable delay (D-Cable)**: This is the delay caused by the propagation of bits of the Pause frame from the traffic receiver to the traffic sender. It is defined by the speed of light in the transmission media. In fiber cables, the speed of light is approximately 200,000 km/s. Hence, a 1-meter optical cable causes a delay of 5 nanoseconds. It can be converted to bit time by multiplying by the port speed. For example, for a 10 GbE port, a 1-meter cable causes a delay of 50 BT, and a 100-meter cable causes a delay of 5000 BT.
5. **Interface delay to receive the Pause frame (D-Intf)**: This is the same as explained in (3) on the receiver of the Pause frame (traffic-sender).
6. **Response delay of traffic sender (D-Resp):** The traffic sender (Pause receiver) takes some time to process the Pause frame and actually pause the traffic in the no-drop class after receiving the Pause frame. This response delay is implementation-dependent, but Ethernet standards place an upper limit on it. For 10 GbE, this delay can be up to 60 pause quanta (30720 BT). For 25 GbE, it is 80 pause quanta (40960 BT). For 40 GbE, it is 118 pause quanta (60416 BT). For 100 GbE, it is 394 pause quanta

(201728 BT). For 400 GbE, it is 905 pause quanta (463360 BT). For further details, refer to IEEE Standard for Ethernet 802.3, the section on MAC Control PAUSE operation (Annex 31B), and the subsection on Timing considerations for PAUSE operation.

7. **Delay due to the transmission of a frame of maximum length in no-drop class (D-Max-No-Drop-Frame-Len)**: Just before the traffic sender pauses traffic in the no-drop class, a frame transmission may start. Transmission of this frame is not interrupted and hence it leads to additional delay. Unlike D-Max-Frame-Len on the traffic receiver explained in (1), which accounts for the maximum frame length in any traffic class, this delay on the traffic sender needs to account for the maximum frame length only in the no-drop class. This is because the traffic receiver needs to reserve headroom only for the frames in the no-drop class for which it had sent the Pause frame. For FCoE traffic, the maximum frame length in no-drop class is approximately 2300 bytes. For RoCE, it can be approximately 2 KB or 4 KB. These values can even be smaller as defined by the application. Here, let's use 2300 bytes, which results in a delay of 18400 BT. Adding 96 BT for IFG and 64 BT for preamble and SFD, the total delay can be up to 18560 BT.
8. **Interface delay to transmit the frame in the no-drop class (D-Intf)**: This is the same as explained in (3).
9. **Cable delay (D-Cable):** This is the same as explained in (4).
10. **Interface delay to receive the frame in the no-drop class (D-Intf)**: This is the same as explained in (3).

Besides the delays shown in Figure 7-2, there may be more delays because of MACsec (IEEE 802.1AE) and other implementation delays. These are collectively called Higher Layer Delay and are ignored in this section. But this explains why additional considerations are needed when using MACsec and why Cisco Nexus switches do not claim support of PFC with MACsec as of this writing. This does not mean that PFC does not work on Cisco Nexus switches when MACsec is enabled. It just means that Cisco has not officially certified it yet.

The total delay (D-Total) is the sum of all the delay values shown in Figure 7-2.

D-Total = D-Max-Frame-Len + D-Pause + D-Intf + D-Cable + D-Intf + D-Resp + D-Max-No-Drop-Frame-Len + D-Cable

Here, the interface delay (D-Intf) is counted only twice instead of four times because their values account for transmit and receive delay.

Considering the maximum frame length of 9216 bytes in any traffic class and maximum frame length of 2300 bytes in no-drop traffic class, cable length of 100 meters, and 10 GbE port, the following is the total delay:

D-Total = 73888 + 672 + 8192 + 5000 + 8192 + 30720 + 18560 + 5000 = 150224 BT

This means the traffic receiver should have a headroom to absorb frames for up to 150224 BT. At 10 GbE, this translates to 18.778 KB of headroom.

But this calculation is not over yet because of the arrangement of buffers in cells on the traffic receiver.

Buffers and Cells

Cisco Nexus switches organizes buffers into cells. Each cell can store up to a fixed number of bytes. The most common cell sizes are 208 bytes, 416 bytes, and 624 bytes. Its exact value depends on the type of the switch, and it cannot be changed by the users. The cell size on a Cisco Nexus switch can be verified using the command **show hardware internal buffer info pkt-stats input**. Refer to Example 7-3 for the output of this command on Cisco Nexus 93180YC-FX.

Example 7-3 *Finding the cell size on Cisco Nexus switches*.

```
switch# show hardware internal buffer info pkt-stats
input
<snip>
Instance    0
```

```
======================================================
==================
Ingress Queue Info:  1 cell = 416 bytes,   Total
cells:  25976,
Instant cell usage:      0,   Remaining cell:  25976
<snip>
```

A cell is used only by one frame. If a frame is larger, it consumes as many cells as it needs. But any leftover space in a cell remains unused and becomes an overhead. For example, a 64-byte frame consumes one cell. Assuming a cell size of 416 bytes, the leftover 352 bytes become an overhead. Likewise, a 2300-byte frame completely consumes five 416-byte cells, and only 220 bytes of the sixth cell. The leftover 196 bytes in this sixth cell becomes overhead.

This means that the 18.778 KB of headroom calculated earlier must account for the cell size and overhead. Assuming a minimum frame size of 64 bytes and cell size of 416 bytes, because each frame consumes 416 bytes of buffers, it consumes 6.5 (416 ÷ 64) times buffer space. Therefore, 18.778 KB translates to 122 KB (18.778 x 6.5) of headroom on the traffic receiver.

This multiplication factor reduces as frame size increases. For example, a frame size of 512 bytes consumes two cells of 416 bytes, resulting in a multiplication factor of 1.62 ((2 x 416) ÷ 512). Likewise, a frame size of 1024 bytes consumes three cells of 416 bytes, resulting in a multiplication factor of 1.21 ((3 x 416) ÷ 1024), and so on.

Note the following:

1. The calculations in this section consider the worst-case values just to explain the concept theoretically. In reality, not all frames are of large size. Also, the interface delay (D-Intf) and response time (D-Resp) values considered in this section are the upper limits as per the standards, but the actual values are much lower.

2. Another way to look at these calculations is that the responsibilities are shared among the manufacturers and users. Manufacturers have better knowledge of interface

delay (D-Intf) and response delay (D-Resp), whereas users have a better knowledge of cable length and maximum frame length as per the traffic profile.

Because of these reasons, the academic explanation in this section should not be directly used in production environments without consulting with the device manufacturer.

The next section provides a simpler and more practical approach for calculating the PFC headroom.

A Practical Approach

As previously mentioned, Cisco Nexus 9000 by default configures the Pause Threshold and Resume Threshold for 100-meter cable length. Regardless of the length of the cable, first enable PFC to find the default values for a 100-meter cable. Next, to change the thresholds for a long-distance link, the only factor is to account for the cable delay (D-Cable). All the other factors explained in Figure 7-2 remain the same as accounted for calculating the default values of the Pause Threshold and Resume Threshold for 100-meter cable.

Consider the example of FCoE ports on Cisco Nexus 93180YC-FX. The default FCoE policy for a 10 GbE port configures the following values:

- Buffer-size — 104000 bytes.
- Pause-threshold — 20800 bytes.
- Resume-threshold — 19136 bytes.

This results in a headroom of 83200 bytes (104000 — 20800). This is for a short distance link.

As per Cisco documentation, for a 10 km link, the FCoE policy for a 10 GbE port should be modified to use the following values:

- Buffer-size — 166400 bytes.
- Pause-threshold — 20800 bytes.
- Resume-threshold — 19136 bytes.

This results in a Headroom of 145600 bytes (166400 – 20800). This is an increase of 62400 bytes as compared to the headroom for 100-meter cable (145600 – 83200). This increase in headroom from 100-meter to 10-km link cannot be explained just by the cable delay because of two key reasons. First, the default thresholds for 100 meters are over-provisioned, as in, they can work for a much longer link. Second, the internal details of the switch architecture are not known.

If Cisco ever decides to support FCoE on even longer distance links, such as 15 km, the value of buffer-size can be extrapolated. As explained earlier, 1-meter cable adds a delay of 5 ns. Therefore, a 5-km cable adds a delay of 25000 ns. In this time, a 10 GbE port can transmit 250000 bits or 31,250 bytes. Accounting for a round-trip of cable delay, this value must be doubled resulting in 62,500 bytes of additional headroom. Therefore, buffer-size can be configured to 166400 + 62500 = 228900 bytes for a 15 km FCoE link. Be aware that this is not supported by Cisco as of this writing, and cell size consideration is ignored. The intention here is to only explain the concept and how these thresholds can be adjusted in a more practical way.

The other consideration is that the headroom and footroom requirements increase as port speed increases. Also, if many long-distance lossless Ethernet links are configured on a switch, reserving the buffers for all of them may exceed the finite buffer capacity of the switch. If sufficient buffers are not available to bring up a long-distance link, Cisco Nexus switches generate a buffer allocation failure message as shown in Example 7-4.

Example 7-4 *Ingress buffer allocation failure on Cisco Nexus switches*

```
switch(config-if)# interface ethernet1/8
switch(config-if)# service-policy type queuing input
ld_10G_fcoe_in_que_policy
switch(config-if)# no shutdown
2022 Oct 31 07:39:21 HW1 %$ VDC-1 %$ %ACLQOS-SLOT1-
2-ACLQOS_FAILED: ACLQOS
```

```
failure: Ingress buffer allocation failed for
interface Ethernet1/8
```

To resolve this issue, reduce the number of ports on this switch that are configured for PFC, reduce their buffer-size, or a combination of both, which essentially reduces the reserved buffers.

To find if a port has insufficient headroom and footroom:

1. A port with insufficient headroom drops frames on the ingress in the no-drop traffic and sends Pause frames. In other words, simultaneous increase in Tx Pause and Rx packet drops is a symptom of insufficient headroom.
2. A port with insufficient footroom may become a source of congestion, reports lower ingress utilization, and sends Pause frames. All these conditions simultaneously are a symptom of insufficient footroom.

If the buffer size and thresholds are configured correctly, the PFC mechanism should prevent packet drops and achieves an expected link utilization.

Ethernet Pause Compared with Fibre Channel B2B Credits

Although Ethernet Pause frames and Fibre Channel B2B credits operate differently, they achieve hop-by-hop flow control by notifying the directly connected sender to slow down so that the receiver does not run out of its buffers.

The following points compare Ethernet Pause to Fibre Channel B2B credits:

1. **Initial exchange**: Fibre Channel B2B credit number is communicated to the directly connected neighbor during link initialization. In contrast, the Ethernet flow control doesn’t exchange the number of buffers with the directly connected neighbor, although DCBX may exchange other information such as the type of traffic that needs lossless behavior.

2. **Link utilization**: Fibre Channel R_RDYs do not contribute to the link utilization because they are primitives and use the place of a fill word between two frames. On the contrary, Pause is a proper Ethernet frame with a header, padding, and CRC. The size of a Pause frame is 64 bytes. Hence, when Pause frames are sent in large number, they contribute to the link utilization. The later section on PFC storm explains a scenario when one million Pause frames may be sent on a link every second. This would lead to a throughput of 512 Mbps (64 bytes x 8 x 1000,000). Although less common, be aware of this aspect when many Pause frames flow on a link.
3. **Duration exchange**: Ethernet Pause frame conveys the duration for which the transmitter must stop transmission, although this duration rarely conveys the actual time the traffic was paused. Fibre Channel R_RDYs do not convey a duration.
4. **When they are sent**: Fibre Channel R_RDYs convey the readiness of a traffic-receiver to receive one frame. In contrast, Ethernet Pause frames convey the duration for which the transmitter should stop transmission or start transmission immediately.
5. **How many**: Fibre Channel R_RDYs are important for the healthy operation of a link. In other words, a large number of R_RDYs on an FC link is a good sign. In contrast, Ethernet Pause frames are sent to stop the traffic. Although a (Un-)Pause frame also resumes the traffic, it only follows a Pause frame. In other words, the fewer Pause frames on a link, the healthier it is.
6. **Direction**: The starvation of Tx B2B credits on a Fibre Channel port indicates egress congestion. In contrast, Tx Pause on an Ethernet port indicates ingress congestion. Likewise, the starvation of Rx B2B credits on a Fibre Channel port indicates ingress congestion, whereas Rx Pause on an Ethernet link indicates egress congestion.
7. **Configuration**: Fibre Channel B2B credits are configured in numbers, while each credit is for one frame regardless of the frame size. In contrast, Ethernet Pause

Threshold and Resume Threshold are configured in bytes, so the frame size should be accounted for making a configuration change. Most intra-data center links with short distances need not require changing the configuration. But for long-distance links, B2B credits or Pause Threshold must be configured correctly. Here, the Pause Threshold and Resume Threshold should not be confused with pause quanta, which most implementations do not allow to change.

8. **Troubleshooting**: When a Fibre Channel port sends R_RDY, it is a good sign, whereas when an Ethernet port sends a Pause frame, it indicates congestion. This basic difference between the two mechanisms makes the foundation of congestion detection and troubleshooting in lossless Ethernet networks.
9. **Scope**: Both are hop-by-hop flow control mechanisms, that is, both operate between the directly connected devices.
10. **Distance:** In Fibre Channel, if there are insufficient B2B credits for the distance, speed, and average frame size, then the link simply operates below its maximum capacity without any errors. In lossless Ethernet, if there is insufficient headroom for the distance, speed, and average frame size, there will be frame drops on the connecting ports, which results in I/O errors on the end devices. If there is insufficient footroom then the link will likely underperform in the presence of congestion. The important point is that both Fibre Channel and lossless Ethernet must take distance into account.

Priority Flow Control

The original Pause frames as per IEEE 802.3x enables flow control on the entire link (LLFC), which does not satisfy the requirement of carrying lossless and lossy traffic on the same link.

PFC solves this problem by enhancing the Pause frame format to carry quanta values for eight classes of traffic. As Figure 7-

3 shows, the PFC Pause frame also carries a Class Enable Vector, which conveys if the quanta is valid for a specific traffic class by turning on the bit for that class. Other classes that are not enabled by the Class Enable Vector ignore the Pause frame. Consequently, PFC is also known as Class-based Flow Control (CBFC) or Per Priority Pause (PPP).

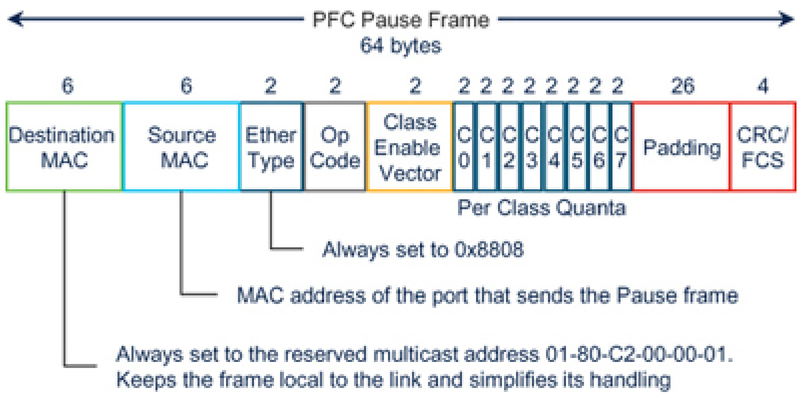


Figure 7-3 *PFC Pause frame format.*

Mapping Traffic Classes to Pause Frame Class Enable Vector

Without mapping the Class Enable Vector to a traffic class, a Pause frame receiver has no way to know which traffic to stop. Imagine the situation when a traffic-receiver sends a PFC Pause frame to stop class-A traffic, but the traffic-sender doesn't know the mapping between the Class Enable Vector in the ingress PFC Pause frame and class-A traffic or has a wrong mapping. This would lead to dropping the lossless traffic.

For PFC to work successfully, the following are important factors:

1. Defining a scheme to map a traffic class to the Class Enable Vector in the PFC Pause frame.
2. Applying this scheme consistently on the end devices and the switches. Typically, DCBX or a software-defined networking solution can simplify this step.

This section focuses on the first factor of defining a mapping scheme. Applying the configuration is outside the scope of this writing. Refer to the documentation of the products in your environment and resources in the references section.

Conceptually, a lossless traffic class can contain any type of traffic as long as the sender and the receiver agree on the same classification, and they have a way to classify it from the rest of the traffic. But practically, two types of mappings are common at the time of this writing.

1. **Layer 2 PFC**: The original use-case of PFC was to allow lossless FCoE traffic to be converged with lossy traffic on the same link. But the traffic must be virtually separated for classification. For this virtual separation, FCoE traffic is assigned to a dedicated VLAN, which is called an FCoE VLAN. The VLAN header contains three bits (Priority Code Point) for classifying traffic, which is uniquely mapped to the Class Enable Vector in the PFC Pause frame. This Layer 2 PFC can also be used for RoCE traffic when it needs a lossless layer 2 network.
2. **Layer 3 PFC**: With the evolution of data center architectures, routed IP networks are becoming more common, primarily because of the improved scale. But for PFC to work in IP-routed networks, traffic mapping based on the Ethernet VLAN header is not sufficient because the VLAN header changes at every layer 3 hop, and in some cases, the VLAN header may not even be present. The solution is to map the Class Enable Vector (in the PFC Pause frame) to a traffic classification scheme that works at layer 3 of the OSI model. IPv4 and IPv6 headers contain a DSCP field, which is widely used for Quality of Service (QoS) and can be used for PFC as well. At the time of this writing, RoCEv2 is the primary use-case of Layer 3 PFC, but the same implementation can also be used for other protocols that need a lossless layer 3 network.

Layer 2 PFC enables lossless traffic delivery within an OSI layer 2 domain, whereas Layer 3 PFC enables lossless traffic delivery via IP-routed networks. The following sections explain these mapping schemes in detail.

Layer 2 Priority Flow Control

As Figure 7-4 shows, at layer 2 of the OSI model, traffic is assigned to different VLANs by adding an IEEE 802.1Q VLAN header. This VLAN header contains a 3-bit Priority Code Point (PCP) field, which allows 8 unique traffic classes. PCP is commonly known as the Class of Service (CoS).

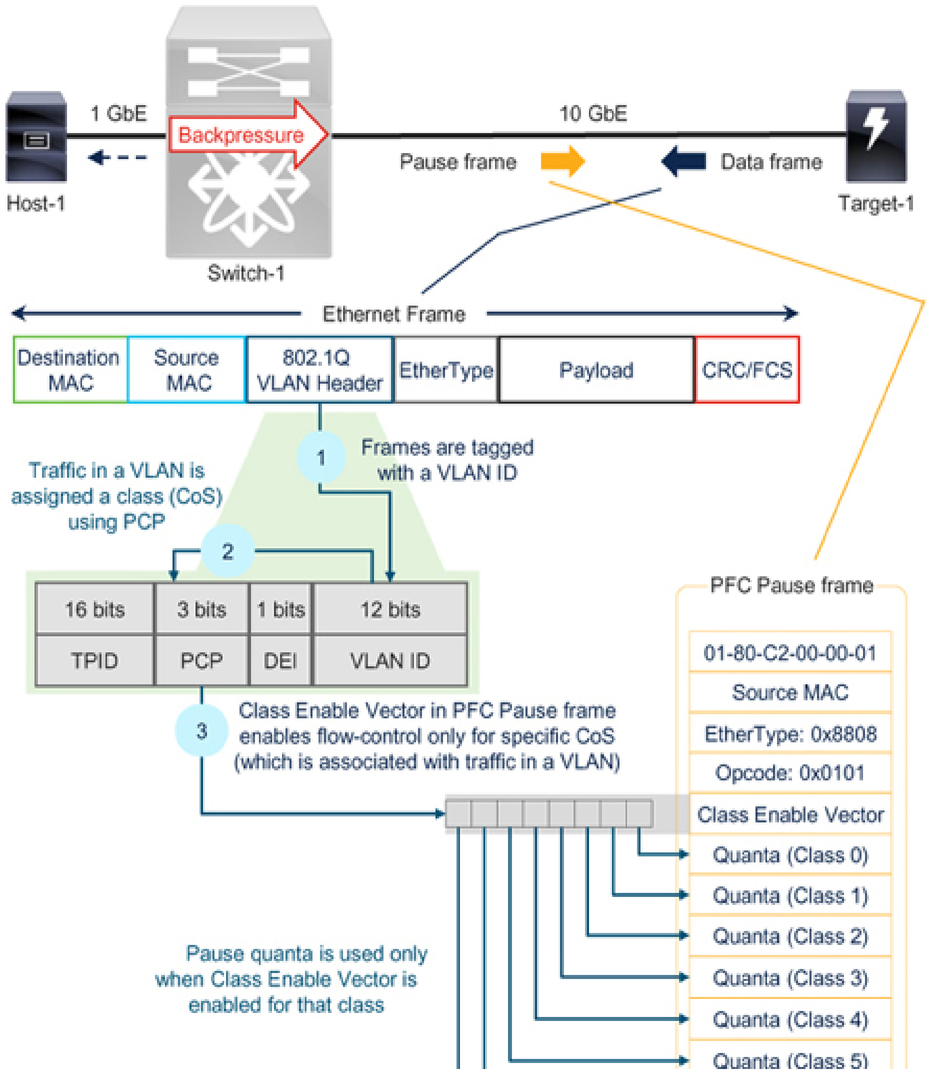


Figure 7-4 *Relationship between Ethernet VLAN CoS of a data frame and Class Enable Vector in Pause frame*

The PFC Pause frame contains the following values:

- **Class Enable Vector**: This is an eight-bit field indicating which class(es) the Pause frame applies to.
- **Quanta values**: These are eight 16-bit values corresponding to each possible traffic class. When the bit

is turned on in the Class Enable Vector, then the respective quanta is valid in that Pause frame.

The mapping between the Ethernet VLAN CoS field and the Class Enable Vector of the PFC Pause frame enables Layer 2 PFC.

Note the following points:

1. A VLAN is uniquely identified by the 12-bit VLAN ID field in the IEEE 802.1Q header. This allows up to 4096 VLANs. But only 8 traffic classes (CoS) are possible. Therefore, it is technically possible to assign the same CoS to multiple VLANs but this approach increases complexity and should be avoided in storage networks. In other words, assign all FCoE or RoCE traffic to a VLAN (let's say VLAN 100), mark all traffic in this VLAN as a unique CoS (let's say 3), keep other traffic in the rest of the VLANs, and do not assign CoS 3 to any other VLAN. Also, avoid using multiple VLANs for FCoE or RoCE traffic to keep it simple.
2. The Class Enable Vector in the PFC Pause frame has eight bits. To indicate that the quanta is valid for CoS N, the Class Enable Vector enables the Nth bit from the right (least significant bit). For example, a Class Enable Vector of 00001000 indicates that this Pause frame is for CoS 3, and a Class Enable Vector of 00011000 indicates that this Pause frame is for CoS 3 and CoS 4.
3. A successful Layer 2 PFC requires two mappings.
   a. VLAN ID and CoS: These values are carried in the Ethernet header of the data frames. Although not a real technical requirement, as explained, this mapping simplifies the deployments and helps in keeping the configuration consistent across the end devices and the switches.
   b. CoS and Class Enable Vector: CoS is carried in the Ethernet VLAN header of the data frames, whereas Class Enable Vector is carried in the PFC Pause frame. Data frames and Pause frame flow in opposite directions. This explains why the configuration must

be in sync between the sender and the receiver and that you should avoid changing the default CoS values that are provided by vendors, such as CoS 3 for FCoE, for a consistent implementation.

Layer 3 Priority Flow Control

At layer 3 of the OSI model, traffic is identified by IPv4 or IPv6 source and destination addresses. As Figure 7-5 shows, the IP header (v4 and v6) contains a 6-bit DSCP field, which allows up to 64 classifications, although not all are used.

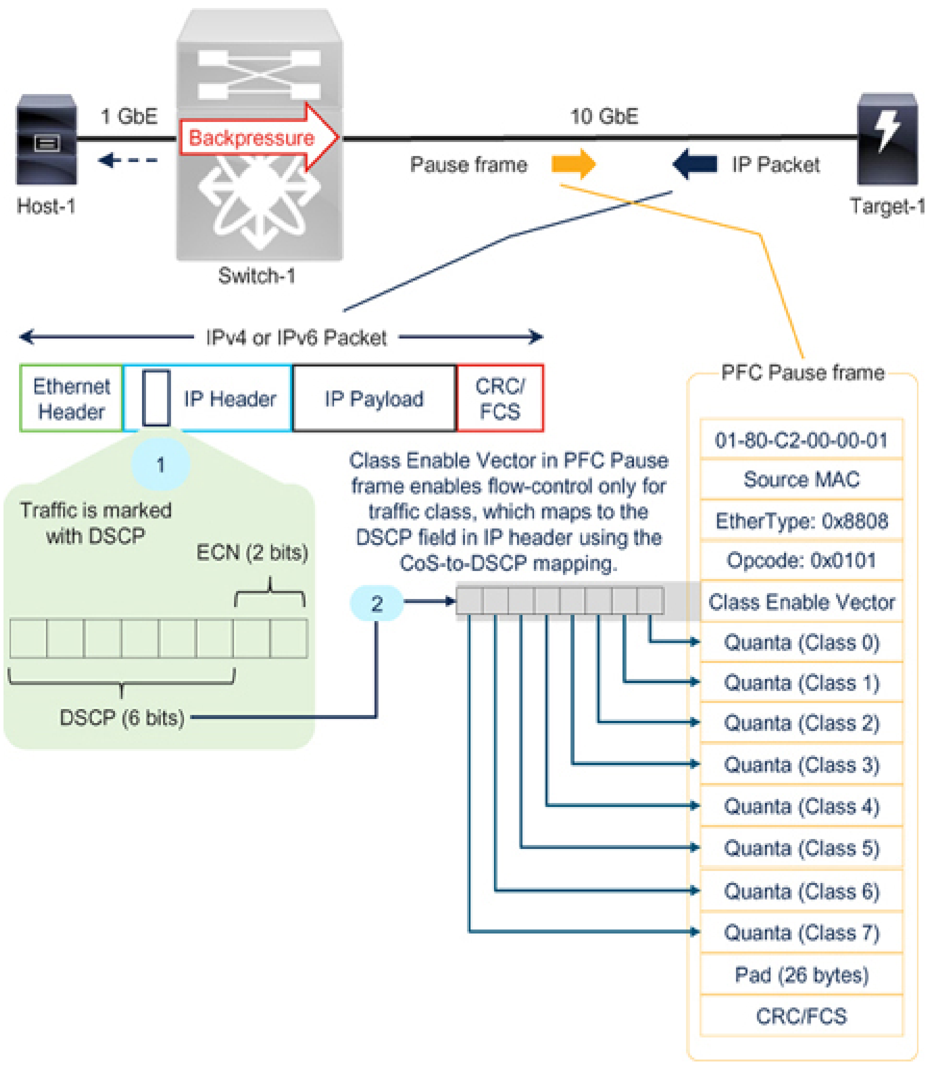


Figure 7-5 *Relationship between DSCP field of an IP packet and Class Enable Vector in PFC Pause frame*

But PFC Pause frames carry quanta values only for eight traffic classes and hence, a mapping (Table 7-1) is required for successful Layer 3 PFC. This is known as CoS-to-DSCP or DSCP-to-CoS mapping.

In Figure 7-5, Host-1, Switch-1, and Target-1 agree on using CS3 for lossless traffic. Target-1 marks DSCP value 24 (CS3 or 011000 in binary) in the IP header. Switch-1 assigns CS3-marked IP packets (refer to the mapping in Table 7-1) to a no-drop queue. When this queue exceeds the Pause Threshold, Switch-1 sends a PFC Pause frame with a Class Enable Vector of 00001000. As a result, Target-1 stops the transmission of CS3-marked IP packets without affecting traffic in other classes. To enable lossless behavior also for CS4 traffic, the Class Enable Vector would be 00011000.

Table 7-1 *Ethernet VLAN CoS and IP DSCP Mapping*

DSCP Name	DSCP Decimal	DSCP Binary	CoS Binary	CoS Decimal
CS0	000000	0	000	0
CS1	001000	8	001	1
AF11	001010	10		
AF12	001100	12		
AF13	001110	14		
CS2	010000	16	010	2
AF21	010010	18		
AF22	010100	20		
AF23	010110	22		
CS3	011000	24	011	3
AF31	011010	26		
AF32	011100	28		
AF33	011110	30		
CS4	100000	32	100	4
AF41	100010	34		
AF42	100100	36		
AF43	100110	38		
CS5	101000	40	101	5
	101010	42		
	101100	44		
EF	101110	46		
CS6	110000	48	110	6
	110010	50		
	110100	52		
	110110	54		
CS7	111000	56	111	7
	111010	58		
	111100	60		
	111110	62		

To know the default CoS-to-DSCP and DSCP-to-CoS mapping on Cisco Nexus 9000 switches, use the NX-OS command **show system internal ipqos global-defaults**.

Converged Ethernet Networks

As explained in Chapter 1, an Ethernet network that allows lossy and lossless traffic on the same network is called converged Ethernet network in this book. In addition to PFC, a converged Ethernet network needs the following functions:

- **Bandwidth guarantee**: When sharing an Ethernet link between lossless and lossy traffic, bandwidth must be appropriately allocated so that one type of traffic does not consume the full capacity of the link leading to starvation for the other traffic. Bandwidth guarantee is enabled by Enhanced Transmission Selection (ETS), which is an IEEE standard (IEEE 802.1Qaz).
- **Consistent configuration**: For successful operation of PFC, the directly connected devices must have a consistent understanding of what they define as lossless and lossy traffic and how much bandwidth is guaranteed. Making these changes consistently on all the network devices is slow and prone to errors when done manually. A better approach is to use automatic discovery among the directly connected devices. This discovery and advertisement are provided by Data Center Bridging Exchange (DCBX), which is an IEEE standard (IEEE 802.1Qaz). DCBX is an extension of Link Layer Discovery Protocol (LLDP) which is another IEEE standard (IEEE 802.1AB-2005).

PFC, ETS, and DCBX belong to an IEEE standard category, called Data Center Bridging (DCB). It has many other names, such as Data Center Ethernet (DCE), Converged Ethernet (CE), Converged Enhanced Ethernet (CEE), etc.

Note that the terms—Converged Networks or Converged Ethernet Networks—have been used in some literature to refer to the networks that carry storage and non-storage traffic regardless of lossless behavior. This book, however, refers to such networks as Shared Storage Networks. A sub-category of a Shared Storage Network is a Converged Network, when it is configured to carry lossy and lossless traffic simultaneously. For example, when a network carries lossless RoCE and lossy HTTP/Web traffic, it called Shared Storage Network and Converged Network. When a network carries lossy iSCSI and lossy HTTP/Web traffic, it is called a Shared Storage Network but not a Converged Network. If your understanding is different, this book does not intend to change that. However, this book uses Converged Network to convey that the network carries lossy and lossless traffic, and hence has PFC, ETS, and (mostly) DCBX enabled.

Configuring Lossless Ethernet

Unlike Fibre Channel, where B2B flow-control is enabled by default, configuring Ethernet flow control requires extra steps and understanding of Quality of Service (QoS) and Modular QoS CLI (MQC). These configuration details are outside the scope of this writing. In-depth educational content is already written on these topics, which is listed in the references section. For the sake of understanding, this chapter uses only a simplified explanation of the QoS concepts and purposefully leaves out implementation details.

Enabling PFC requires the following steps:

1. **Classifying and marking the traffic**: Classifying and marking the traffic is the first step because various types of traffic, such as storage, voice, video, web, and FTP can be on the same port. Classification is achieved by the Ethernet VLAN CoS field at layer 2. Beyond a layer 2 boundary or when the frames are not VLAN tagged, the DSCP field in the IP header is used. End devices can mark the frames before sending them on the network and the network can trust these markings. Alternatively, the edge switchports can classify and mark the packets themselves.

2. **Flow-control and bandwidth allocation**: After classification, it must be decided what traffic needs lossless behavior and what traffic needs lossy behavior.

Lossless traffic is enabled for flow control by PFC. As mentioned, the traffic class that is flow-controlled by PFC is known as the no-drop class. The switchports must also provide a bandwidth guarantee to the no-drop class, such as 50% of the link capacity. Traffic in other classes is not flow-controlled, although bandwidth may still be guaranteed to them. Traffic in no-drop classes becomes lossless and traffic in all other classes remains unchanged (lossy).

3. **Consistent implementation**: Finally, QoS configuration must be consistently applied on all the end devices and switches. For example, if a switch enables no-drop behavior for CoS 3, whereas another switch in the same network enables no-drop behavior for CoS 4, the result will be lossless traffic becoming lossy. Also, QoS misconfiguration may cause congestion and many more issues. As mentioned earlier, DCBX or a software-defined networking technology can simplify the implementation.

Configuring these steps depend on the switch type and its architecture, and often involves a learning curve. Although configuration can be simplified using automation or a GUI, troubleshooting a congestion issue requires an understanding of the commands that achieve each of these functions.

Note the following points about configuring Lossless Ethernet:

1. We recommend using the vendor-provided QoS configuration instead of customizing it. If the vendor documentation uses CoS 3 for no-drop class, prefer using the same in your environment. Although changing the classification is technically possible, aligning it with vendor documentation keeps your environment consistent with other deployments across the world. Users can copy/paste the configuration commands directly, and during troubleshooting, they don’t have to remember the different CoS values for the no-drop class.

2. As explained earlier in the section on *Pause Threshold and Resume Threshold*, we recommend avoiding changing the default or vendor-suggested PFC

configuration for pause-threshold and resume-threshold for short distance intra data center links. Customizing these values requires an understanding of the buffer allocation and queuing architecture of the devices. If not changed correctly, the result may be delayed sending of Pause frames or sending them earlier then needed, which will result in performance degradation. However, long distance lossless Ethernet links require changing these thresholds.

3. Multiple no-drop classes may be possible based on the use-cases and the capability of the devices. For example, one no-drop class for FCoE, whereas another no-drop class for RoCEv2. In such cases, flow-control in one no-drop class does not interfere with the flow control in other no-drop class. The Class Enable Vector in the PFC Pause frame enables bits for the respective traffic classes and their quanta values.
4. After the configuration assigns a traffic class (DSCP) to a no-drop queue, the switches use hop-by-hop flow control on all the packets that are marked with the DSCP value. If the end devices wrongly mark the packets, the switches will not know this and assign the lossy traffic to the no-drop queue or lossless traffic to the drop queue.

This book refers to traffic in no-drop classes as lossless and other traffic as lossy traffic regardless of the approach of classification, marking, bandwidth allocation, and device-specific implementation details.

Dedicated and Converged Ethernet Network

Just because a network is configured as a converged Ethernet network (lossy and lossless traffic), doesn’t necessarily mean that lossy and lossless traffic runs on it simultaneously. For example, assume you configured a network with 50% bandwidth allocated to lossless traffic and 50% bandwidth for other traffic. If there is no lossless traffic at that time, at the data layer this network is no different than any other lossy

Ethernet network. On the other hand, if there is no traffic in the lossy classes, this network operates like a dedicated lossless network. In other words, it is “configured” as a converged network but “operating” as a dedicated network.

Consider the following examples.

1. **Cisco UCS Servers**: Cisco UCS servers use converged Ethernet for carrying lossless (Fibre Channel and RoCE) and lossy (TCP/IP) traffic on the same links. But if no server uses storage via Fibre Channel or RoCE, internal links will never report Pause frames, although the configuration for converged Ethernet remains applied on them. Refer to Chapter 9, “Congestion Management in Cisco UCS Servers,” for more details on Cisco UCS servers.
2. **FCoE on Cisco MDS Switches**: Although the FCoE ports on Cisco MDS switches are configured for converged Ethernet, they only handle lossless traffic, and do not send/receive traffic in lossy classes.

A key point to understand is that even the dedicated lossless networks are configured the same as the converged networks. The configuration of traffic classification, bandwidth guarantee, flow control, etc. as explained in the previous section is applied in both types of networks. However, traffic pattern makes one different from the other. For congestion detection and troubleshooting, verifying configuration is the first step, which remains the same for both types of networks. The next step is to focus on the traffic pattern which depends on if the network carries only lossless traffic (dedicated) or if the network carries lossless and lossy traffic simultaneously (converged).

Understanding Congestion in Lossless Ethernet Networks

Lossless Ethernet networks are prone to similar types of congestion as Fibre Channel fabrics because both use flow-control between directly connected devices. Also, this congestion spreads toward the source of the traffic victimizing many other devices that share the same network paths and traffic class.

Chapter 1, the section on *Congestion in Storage Networks — An Overview* explains the basics of congestion spreading and various causes and sources of congestion. This section expands on those basics in lossless Ethernet networks.

Slow-Drain

An end device that has a slower processing rate as compared to the rate at which traffic is being delivered to it is called a slow-drain device and the congestion caused by it is called slow-drain. This end device uses Pause frames to pace the ingress traffic rate. Typically, this results in sending of excessive Pause frames from this device. These are reported as ingress Pause frames on its connected switchport.

Over-utilization of a Link

Congestion due to over-utilization happens when a switchport is transmitting at the maximum speed yet it has more frames than can be transmitted on that link. This switch uses Pause frames to pace the traffic rate from the upstream devices that are sending traffic via the over-utilized link. This results in high or full utilization of the link that is the source of congestion and sending of excessive Pause frames to the upstream devices.

Bit Errors

Bit errors may interfere with LLFC/PFC resulting in congestion or worsening existing congestion. When a traffic-sender detects CRC error in the received Pause frame, it discards the corrupted Pause frame, and hence, does not stop the traffic. The next valid Pause frame should stop the traffic however the buffer headroom may become full by the time the next Pause frame stops the traffic. This may result in frame drops in the no-drop class.

Another significant effect of bit errors is on performance in FCoE fabrics because an entire I/O operation is reinitiated even if just one packet is dropped from that operation. This concept is explained in Chapter 1, the section on *Bit Errors on a Link*.

Because of these reasons, detect bit-errors as soon as possible, find their root cause, and make the corrective change.

Congestion Spreading in a Single-Switch Lossless Ethernet Network

Refer to the single-switch lossless Ethernet network in Figure 7-6, which connects 40 hosts and 8 targets. Each host accesses storage from all 8 storage ports in the no-drop class. Hosts do not communicate among themselves in the no-drop class. Likewise, targets do not communicate among themselves within the no-drop class. All devices communicate with other devices outside the no-drop class.

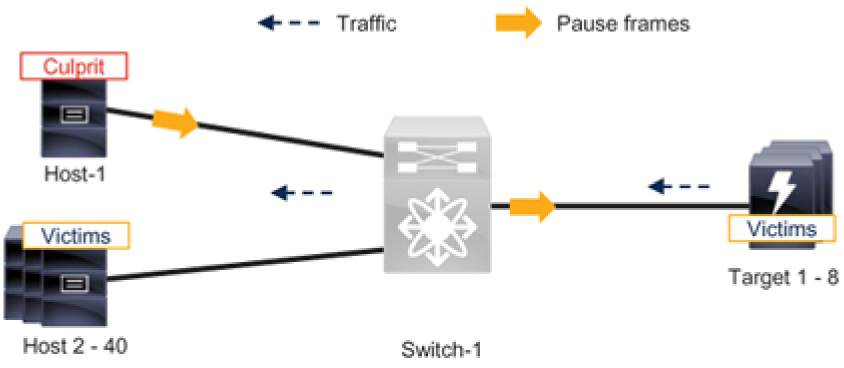


Figure 7-6 *Congestion in a single-switch lossless Ethernet network*

When a host (Host-1) becomes a slow-drain device, it slows down the rate of ingress traffic by sending Pause frames. The switch can buffer some frames, but eventually, its queues exceed the Pause Threshold, and therefore sends Pause frames to the eight targets to slow them down.

This is an expected behavior. But it has a side-effect. PFC slows down all the traffic within the no-drop class regardless of traffic destination. As a result, the other 39 servers are also affected even though they are not invoking PFC.

Finally, in this single-switch network with 40 hosts and 8 targets, one culprit host can victimize all the other devices. This effect of congestion spreading in this lossless Ethernet network is no different from a similar Fibre Channel fabric because both use hop-by-hop flow control.

As Chapter 4, "Troubleshooting Congestion in Fibre Channel Fabrics," the section on *Identify the Affected Devices (Victims)* explains, the targets are the direct victims because they are in direct communication with Host-1. Hosts 2 – 40 are indirect victims because they are in communication with the direct victims (targets).

Congestion Spreading in an Edge-Core Lossless Ethernet Network

Next, consider Figure 7-7, which shows an edge-core network with three host-edge switches and one core switch. Each host-edge switch connects to 200 hosts and the storage-edge switch connects to 100 targets. Storage traffic remains in the no-drop class. Hosts do not communicate among themselves in the no-drop class. Likewise, targets do not communicate among themselves within the no-drop class. All devices communicate with other devices outside the no-drop class.

When a host becomes a slow-drain device, it slows down the rate of ingress traffic by sending Pause frames. The host-edge switch can buffer some frames, but eventually, its queues exceed the Pause Threshold, and therefore it sends Pause frames to the core switch. This slows down all the traffic in the no-drop class between that edge switch and the core switch regardless of traffic destination, which affects up to 199 additional hosts that connect to the same edge switch.

Further, the core switch can buffer some frames, but eventually, its queues also exceed the Pause Threshold, and hence it sends Pause frames to all the targets that are sending traffic on the congested ISL. These targets slow down the traffic in the no-drop class regardless of traffic destination, which may be any host connected to any host-edge switch and even a host connected to the storage-edge switch (not shown in Figure 7-7).

As a result, one culprit device victimizes many devices across the entire network. The effect of congestion spreading in this lossless Ethernet network is the same as observed in a similar Fibre Channel fabric. Compare Figure 7-7 to the example in Chapter 6, the section on *Network Design Considerations*.

As Chapter 4, the section on *Identify the Affected Devices (Victims)* explains, the victims in Figure 7-7 can be further classified as direct victims, indirect victims, and same-path victims.

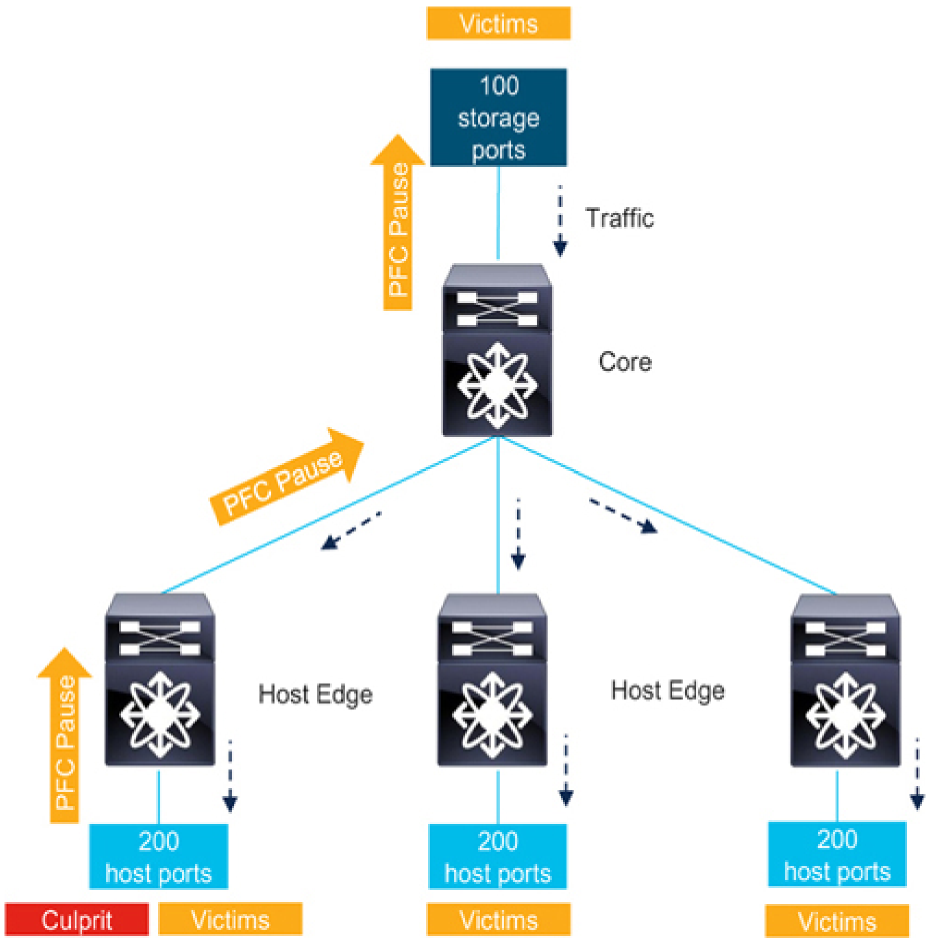


Figure 7-7 *Congestion in an edge-core lossless Ethernet network*

For reducing the spread of congestion in Figure 7-7, as Chapter 6 the section on *Increased Traffic Localization to a Single Switch* explains, had the targets and their hosts been connected to the same switch, then the end devices on the other switches would not have been victims.

Congestion Spreading in a Lossless Spine-Leaf Network

Consider Figure 7-8 to understand congestion spreading in a lossless spine-leaf network. Hosts and targets are connected to the leaf switches in no specific order. Storage traffic remains in the no-drop class. Hosts do not communicate among themselves in the no-drop class. Likewise, targets do not communicate among themselves within the no-drop class. All devices communicate with other devices outside the no-drop class. All the links between spine and leaf switches are uniformly utilized because of equal cost multi-path (ECMP). For example, traffic from Target-1 to Host-1 goes to Leaf-1, then to all four spines, and finally to Leaf-6 via all its uplinks.

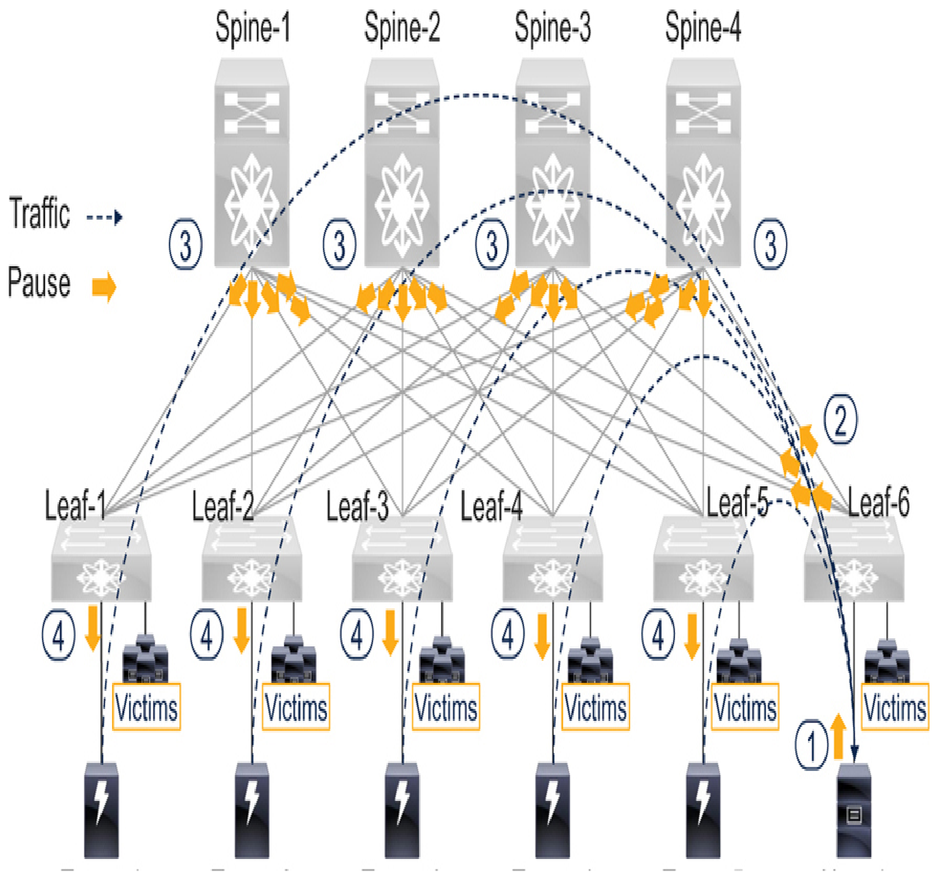


Figure 7-8 *Congestion in lossless Ethernet spine-leaf network*

Host-1, which connects to Leaf-6, receives traffic from five targets that connect to Leaf-1 through Leaf-5 switches. When Host-1 becomes a slow-drain device, it sends Pause frames to Leaf-6 to slow down the ingress traffic. Eventually, Leaf-6's

Pause Threshold exceeds, and therefore it sends Pause frames to its upstream neighbors. This slows down the traffic from the four spine switches to Leaf-6, which victimizes other hosts that connect to Leaf-6 and receive traffic from any other target behind any other leaf switch, although those hosts do not invoke PFC.

Further, the spine switches exceed their Pause Thresholds and therefore they send Pause frames to all the leaf switches (Leaf-1 – Leaf-5) that are sending traffic to Leaf-6. This slows down all the traffic from the leaf switches regardless of its destination, and hence, many unrelated devices are also victimized.

Finally, the target-connected leaf switches (such as Leaf-1) exceed the Pause Threshold and therefore they send Pause frames to the targets (such as Target-1) to slow them down. On Leaf-1, because Target-1 slows down all the traffic in the no-drop class regardless of its destination, even the hosts that connect to Leaf-6 and receive traffic from Target-1 are affected. These victims are often overlooked because their traffic remains local to the leaf switch while the culprit connects to a different leaf switch. But because the culprit adversely affects the targets, all the hosts that receive traffic from these targets are victimized regardless of where they reside.

As Chapter 4, the section on *Identify the Affected Devices (Victims)* explains, the victims in Figure 7-8 can be further classified as direct victims, indirect victims, and same-path victims.

Slow-drain

Figure 7-8 explains when Host-1 has a slower processing rate as compared to the rate at which frames are delivered to it (slow-drain). But its link is not fully utilized. For example, Host-1 connects to Leaf-6 at 10 GbE but it can receive traffic only at 5 Gbps because of some other problem within Host-1, which is preventing it from processing more than 5 Gbps of ingress traffic and hence, it invokes PFC.

Over-utilization of host-edge link

Congestion due to over-utilization happens when Host-1 can process the ingress traffic at the full capacity of its link (10 Gbps). It is not invoking PFC. However, Leaf-6 is being delivered more traffic (for example, 11 Gbps) than can be sent to Host-1, and therefore Leaf-6 invokes PFC to slow down the traffic from the spine switches.

Comparing Congestion due to Slow-drain and Over-utilization

The effect on the fabric is the same when congestion is caused by a slow-drain device or over-utilization of a host-edge link.

The difference lies in the edge switchports. When congestion is caused by slow-drain, the culprit-connected edge switchport receives many Pause frames and its egress utilization is not high. In contrast, when congestion is caused by over-utilization, the edge switchport does not receive Pause frames and the egress utilization is high. This difference makes the foundation for detecting the cause of congestion.

Detecting Congestion in Lossless Ethernet Networks

The following are the key factors for congestion detection workflow in lossless Ethernet networks.

- What to detect.
  - What is the effect of congestion? In other words, how severe is it?
  - What is the cause of congestion?
  - Where is the source of congestion (culprits)?
  - Where is the spread of congestion (victims)?
  - When did congestion happen?
- How to detect:

    - Reactive approach: Detect and troubleshoot congestion events after they happen.
    - Proactive: Detect congestion events in real time.
    - Predictive: Predict congestion events before they happen.
- Where to Detect
    - Natively on the devices, such as switches, hosts/servers, or storage arrays.
    - Using remote monitoring platforms, such as the UCS Traffic Monitoring App, as explained in Chapter 9.

Refer to Chapter 3, "Detecting Congestion in Fibre Channel Fabrics,", the section on "Congestion Detection Workflow" for a detailed explanation of these topics. The same details also apply to lossless Ethernet networks; hence they are not repeated here. This section provides only a brief overview as it applies to lossless Ethernet networks.

Congestion Direction — Ingress or Egress

Congestion is directional. Congestion usually occurs in one direction, while the reverse direction may be uncongested. Attempting to follow the congestion in the wrong direction does not lead to finding the source and the cause of it.

For example, in the spine-leaf network shown in Figure 7-8, on the spine switches, only the traffic in the direction of Leaf-6 is affected by congestion. Traffic coming from Leaf-6 to the spine switches is not affected by congestion.

Congestion on an egress switchport results in congestion on an ingress port, which causes egress congestion on at least some upstream ports along the traffic path toward the source. In other words, egress congestion causes ingress congestion. Ingress congestion does not cause egress congestion. Because of this reason, congestion detection workflow and metrics in the egress direction are more important for identifying the source of congestion.

Figure 7-9 shows a subset of the spine-leaf network shown in Figure 7-8. The source of congestion is Host-1, which causes egress congestion on Leaf-6. This egress congestion results in ingress congestion on the ports of Leaf-6 that are connected to the leaf switches. The same sequence of ingress and egress congestion continues to the source of the traffic (Target-1). Depending upon the location of a port, investigate congestion in the ingress or egress direction.

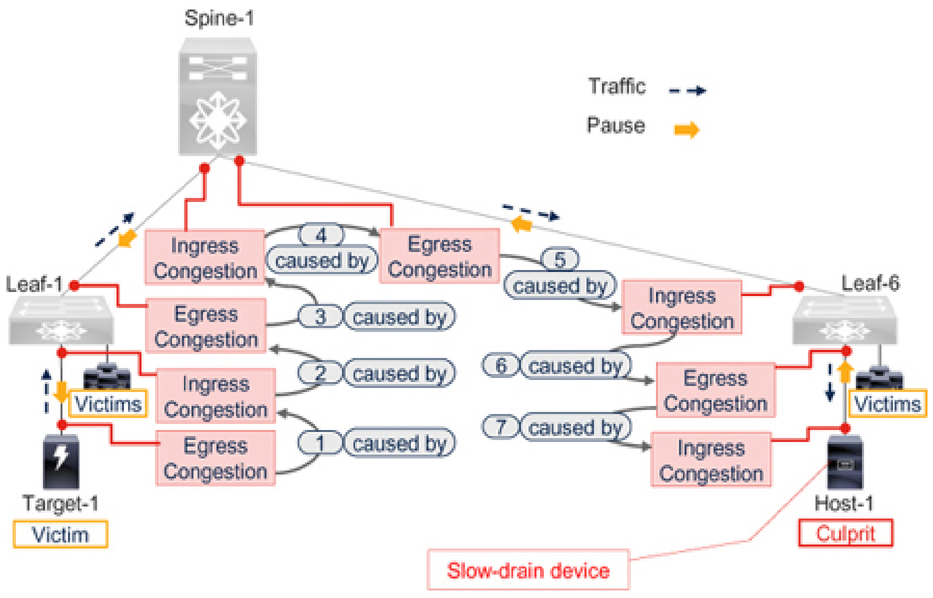


Figure 7-9 *Congestion direction and traffic direction in a lossless spine-leaf network*

Congestion Detection Metrics

The following are the metrics that can help in detecting congestion in lossless Ethernet networks.

- Metrics that detect traffic pause, also known as, Pause frame monitoring.
  - Duration of traffic pause: How long a port could not transmit because of receiving Pause frames from the neighbor.
  - Number of times when the traffic was paused: The number of Pause frames received or sent.

- The instantaneous value of available buffers: On a traffic receiver, an instantaneous buffer utilization shows how far is it from the Pause Threshold, which triggers a Pause frame.
- Metrics that detect frame drops.
- Metrics that detect bit errors, such as CRC-corrupted frames.
- Metrics that detect link utilization, such as the number and size of frames received and sent on a port.
- Metrics that detect the application I/O profile, such as the timing, size, type, and rate of I/O operations that are carried within the frames.

These are the same types of metrics as explained in Chapter 3, section *Congestion Detection Metrics* on Fibre Channel ports. Lossless Ethernet ports do not have metrics that are specific only to Fibre Channel B2B flow control, such as for Link Reset Protocol or the B2B State Change mechanism. Fibre Channel ports initiate a credit loss recovery (via Link Reset Protocol) when it has zero remaining-Tx-B2B-credits for an extended duration. Lossless Ethernet does not have a similar concept regardless of how long a port is continuously paused by its neighbor.

If an Ethernet port does not report the metrics or does not make accessing them easy enough, a workaround is to use the metrics from the directly connected neighboring port in the reverse direction. For example,

- The ingress utilization of a port is the same as the egress utilization of its neighbor and vice-versa.

- In most cases, Pause frames sent by an Ethernet port are the same as the Pause frames received by its neighbor. It is possible that the Pause frames get corrupted in-between and may not be recognized by the peer. In such cases, the port will continue to send Pause frames until its queue utilization falls below the Resume Threshold. For this reason, and for all practical purposes, it is ok to monitor the Pause frames on just one port on a link. For example, on a link between host port and its connected switchport, it is ok to monitor TX and RX Pause on just one of these ports.

Such correlations across neighboring devices are better done on a remote monitoring platform.

Duration of Traffic Pause — TxWait and RxWait

TxWait is the duration for which a port could not transmit because of receiving Pause frames from its neighbor. It is also known as Pause duration.

RxWait is similar to TxWait in the reverse direction. It is the duration for which a port could not receive traffic because of sending Pause frames.

TxWait and RxWait can be converted to something more meaningful, called percentage TxWait, which is the percentage of time when a port could not transmit in a duration. For example, 50% TxWait in 20 seconds indicates that the port could not transmit for 10 seconds.

Note the following points about TxWait and RxWait:

1. TxWait is because of Rx Pause and RxWait is because of Tx Pause.
2. If a port does not report TxWait, using RxWait on its neighbor is an alternative.
3. Prefer using TxWait and RxWait (or similar metrics that detect the duration of traffic pause) for detecting congestion in lossless Ethernet networks. Use other metrics, such as the number of Pause frames, when TxWait and RxWait are unavailable.

4. At the time of writing, Cisco MDS and Nexus 7000 switches collect TxWait and RxWait on FCoE ports. Cisco Nexus 9000 switches and Cisco UCS servers do not collect TxWait and RxWait.

TxWait and RxWait on Ethernet ports are similar to TxWait and RxWait on Fibre Channel ports. This section briefly explains the commands that show TxWait and RxWait on FcoE ports on Cisco MDS switches. More details are available in Chapter 3, the section on *Tx Credit Unavailability in Microseconds — TxWait* and *Rx Credit Unavailability in Microseconds — RxWait*.

Raw and Percentage TxWait and RxWait

Example 7-5 shows raw and percentage TxWait and RxWait on the FCoE port on Cisco MDS switches. MDS switches report TxWait and RxWait in 2.5 microseconds (µs) increments so a value of 4 equals 10 µs. Example 7-5 shows RxWait of 49% in the last 1 second, which indicates that this port sent Pause frames to the neighbor for stopping the traffic for 490 milliseconds (ms) in the last 1 second.

Notice that TxWait is measured for VL3 only. This is the FCoE class used for CoS3 traffic, and it is flow-controlled using PFC.

Example 7-5 *TxWait and RxWait in show interface on Cisco MDS switches*

TxWait and RxWait History Graphs

Cisco MDS switches show three TxWait and RxWait History graphs.

- Last one minute — Each column shows the cumulative TxWait or RxWait in each second.
- Last one hour — Each column shows the cumulative TxWait or RxWait in each minute.
- Last 72 hours — Each column shows the cumulative TxWait or RxWait in each hour.

Refer to Chapter 3, the section on *TxWait History Graphs* for sample outputs, which are the same for FC and FCoE interfaces.

TxWait and RxWait History in OBFL

As Example 7-6 shows, Cisco MDS switches make an entry for TxWait and RxWait in the Onboard Failure Logging (OBFL) buffer when their values increase by 100 ms or more in a 20-second interval.

Example 7-6 *TxWait and RxWait history in show logging onboard txwait or rxwait on Cisco MDS switches*

Refer to Chapter 3, the section on *TxWait History Graphs* for a detailed explanation of this output, which is the same for FC and FCoE interfaces. Refer to Chapter 4, the section on *The OBFL Commands — show logging onboard* for more details on OBFL.

Number of Pause Frames

When TxWait and RxWait are unavailable, the next option is to know the number of Pause frames sent and received by an Ethernet port.

These are similar to the “credit transition to zero” counters on Fibre Channel ports except not as specific because only some Pause frames that have non-zero quanta actually stop traffic.

The following are some important points while using the number of Pause frames to detect congestion:

1. At the time of this writing, most implementations do not count Pause (non-zero quanta) and Un-Pause (zero quanta) separately. The Pause frame counts are the total of both Pause frame types.
2. Pause frame counter may increment under normal conditions minimally without any effect on the application performance. When the Pause frame count increments on an edge port by a small value but not on the upstream ports, this indicates that the congestion is absorbed by the buffers on this switch. This condition

does not victimize other devices, and hence, it is not as big as a concern as congestion spreading. It just means that LLFC/PFC is working well.

3. The Pause frame count may be reported only as an accumulated value since the counters were last cleared. A large Pause count does not convey if the congestion occurred yesterday, last week, or last month. At the time of this writing, on Cisco devices, only the accumulated Pause frame count is available, except for Cisco MDS and Nexus 7000 switches, which maintain their time and date-stamped history.

Example 7-7 shows ingress and egress PFC Pause frames on Cisco UCS Fabric Interconnect. NX-OS command **show interface priority-flow-control** was run twice separated by 10 seconds. This output indicates the following:

1. Ethernet 1/1 and Ethernet 1/8 use LLFC (Oper Off), whereas Ethernet 1/1/7 uses PFC (Oper On).
2. Ethernet 1/1/7 uses PFC only in CoS 3. The VL bmap shows the Class Enable Vector in the PFC Pause frame (Figure 7-4). It is in hexadecimal format. The value of 0x8 is 1000 in binary. While reading it from the right and starting with 0, 3rd bit is enabled. This indicates CoS 3 traffic is assigned to a no-drop class. Likewise, VL vmap of 0x28 would mean that PFC is enabled for CoS 3 and CoS 5 because 0x28 is 101000 in binary.
3. The RxPPP and TxPPP show the total count of PFC Pause frames in all the classes. PPP stands for Per-Priority Pause.
4. Ethernet 1/1 did not have ingress or egress congestion during those 10 seconds because the TxPPP and RxPPP counters did not change, although the Pause frames were sent and received at earlier unknown times.
5. Ethernet 1/8 had egress congestion because the RxPPP counter incremented by 1456 (846831773 – 846830317) in 10 seconds. It did not have ingress congestion because TxPPP counter was unchanged.

6. Ethernet 1/1/7 had ingress congestion because the TxPPP counter incremented by 38 (1406361419 – 1406361381) in 10 seconds. It did not have egress congestion because RxPPP counter was unchanged.
7. A comparison of the number of Pause frames in the 10-second interval on Ethernet 1/8 (1456) and Ethernet 1/1/7 (38) indicates that congestion was more severe on Ethernet 1/8, although in the reverse direction.

Example 7-7 *Pause frame count in show interface priority-flow-control on Cisco UCS*

The **show interface priority-flow-control** has been the primary command to detect and troubleshoot congestion on Cisco Nexus 9000 switches and Cisco UCS servers because of its simplicity. But it shows the combined count of Pause frames for all the classes, which is fine because the most environment may use PFC for only a single traffic class.

For finding the Pause frame count per class on Cisco Nexus 9000 switches, use the NX-OS command **show queuing interface**. Example 7-8 shows congestion indications in ingress and egress directions on Ethernet 1/1. At a class level, only CoS 1 is congested, whereas CoS 3 does not have any indication of congestion. Also, note that CoS 1 is under an Active state of Pause when this command was executed. One must be very lucky to find RxPause or TxPause in the Active state. If RxPause (ox TxPause) stays Active when this command is executed multiple times quickly, it indicates that the directly attached neighbor is a slow-drain device causing severe congestion.

Example 7-8 *Per-priority Pause frame count and current state in show queuing interface*

The **show interface** command on Cisco Nexus 9000 switches and Cisco UCS servers also show the pause counter (Example 7-9). But these are LLFC counters, and they do not increment when PFC is enabled.

Example 7-9 *LLFC Pause counter in show interface*

Frame Drops or Discards

When using LLFC or PFC, frames should not be dropped. However, if a port continues to receive traffic even after it exceeds the Pause Threshold and its buffer headroom is fully consumed, it drops the frames. Another scenario of frame drop is during severe congestion when the frames remain within the queue for a duration longer than the pause timeout or PFC watchdog interval (explained later), and the queue flushes all the packets.

Most Ethernet ports report dropped or discarded frames because of various reasons. Refer to Example 7-9 which shows input and output discards in the NX-OS command **show interface**. This is a collective count in no-drop and other classes.

For finding the packet drops per traffic class, use the NX-OS command **show queuing interface**. Refer to Example 7-10. It shows that PFC is enabled for CoS 3 and CoS 4. The switch from which this output is taken supports two no-drop queues. Packet drops in the no-drop queue that is assigned to the CoS with the smaller value (3) are shown under “Low Pause Drop Pkts”. Likewise, packet drops in the no-drop queue that is assigned to the CoS with the greater value (4) are shown under “High Pause Drop Pkts”. The Low and High refer to the smaller and the greater CoS values. No-drop class with CoS 3 has dropped 255 packets, whereas no packets were dropped in the no-drop class assigned to CoS 4 traffic.

Example 7-10 *Packet drops per no-drop class*

Bit Errors

The primary approaches for detecting bit errors on Ethernet ports are CRC, stomped CRC, and FEC.

CRC Counters

A sender calculates the Frame Check Sequence (FCS) polynomial on the frame content and places the calculated output in the CRC field of the frame. The receiver, after receiving the frame, calculates the same FCS polynomial on the frame content. If the calculated output does not match with the content of the CRC field, the frame fails the CRC verification, and it is called a CRC corrupted frame. Consequently, the receiver increments the input CRC counter.

A key point to remember is that CRC counter increments only when bit errors are within the frames. But if the bit errors are outside the frame boundary or if the traffic rate is low resulting in fewer frames on a link, CRC counter may not increment even if bit errors exist on a link. Chapter 2 *Case Study — An Online Retailer* demonstrates this scenario.

Example 7-9 shows CRC errors on Cisco Nexus 9000 switches and Cisco UCS servers using the NX-OS command **show interface**.

For detecting bit errors using CRC counters, an important consideration is the cut-through or storage-and-forward architecture of the switches. This point is explained in detail in Chapter 2, the section on *Ability to Detect and Drop CRC Corrupted Frames*. In short, cut-through switches reduce port-to-port switching latency. But these switches can't drop a CRC-corrupted frame because they start transmitting the frames before receiving them in their entirety and because CRC field resides at the end of a frame. When this happens, the same CRC corrupted frame results in incrementing the CRC counters on multiple ports on different switches in its path, therefore making it complex to detect the source of bit errors. This complexity can be reduced to an extent if the switches support stomped CRC, as explained in the following section.

Stomped CRC Counters

Stomped CRC counters help in finding the location of bit errors in a network that uses cut-through switches. More precisely, these counters help in finding where bit errors do not exist.

When a cut-through switch support stomped CRC feature, it encodes a special value in the CRC/FCS field of the corrupt frame. This is called frame stomping. As mentioned, the switch can't drop the corrupt frame because it already starts transmitting it. But it can stomp the frame because it knows the frame is corrupted and it is yet to transmit the CRC field at the end of the frame. When the next switch detects a stomped frame, it increments only the stomped CRC counter and does not increment the CRC counter. When all these ports are compared, ports with stomped CRC error can be excluded from the investigation and the port/link with CRC counters can be investigated. The corrupt frame is eventually dropped at its destination or on a store-and-forward switch.

Example 7-9 shows Stomped CRC errors on Cisco Nexus 9000 switches and Cisco UCS servers using the NX-OS command **show interface**.

Forward Error Correction

When FEC is enabled, a sender adds a few additional parity bits into the bitstream. The receiver can use these parity bits to detect and recover a limited number of bit errors.

When FEC is able to recover the corrupted bits:

- FEC corrected counter increments, and
- CRC counter does not increment because FEC already recovered the bit errors at a lower layer and bits are handed off to the framing layer as sent by the sender.

When FEC is unable to recover the corrupted bits:

- FEC uncorrected blocks counter increments, and
- CRC counter may increment if the bit error is within a frame.

On Cisco Nexus 9000 switches, use the command **show hardware internal tah mac hwlib show mac_errors fp-port** to display FEC Correctable and FEC UnCorrectable counters. As Example 7-11 shows, this is a module level command so before using it, you must use NX-OS command **attach module**.

Example 7-11 *FEC Counters on Cisco Nexus 9000 switches*

```
switch# attach module 1
module-1# show hardware internal tah mac hwlib show
mac_errors fp-port 15
```

```
<snip>
MAC: HSMCPCS Err per channel:     0        1       2
3
----------------------------------------------------
--------
BER Count                        .....   .....
.....   .....
Err Blocks Count                 .....   .....
.....   .....
Sync Loss Count                  .....   .....
.....   .....
Block Loss Count                 .....   .....
.....   .....
High BER Count                   .....   .....
.....   .....
Valid Err Count                  .....   .....
.....   .....
Unknown Err Count                .....   .....
.....   .....
Invalid Err Count                .....   .....
.....   .....

MAC: BIP Err per channel:         0        1       2
3       4
----------------------------------------------------
----------------
BIP Lanes 0-9                    .....   .....
.....   .....   .....   .....
BIP Lanes 10-19                  .....   .....
.....   .....   .....   .....
MAC: FEC Err per ch:               0        1
2       3
----------------------------------------------------
--------
RS FEC Err per lane            0   .....   .....
.....   .....
----------------------------------------------------
--------
RS FEC Correctable               .....
RS FEC UnCorrectable             .....
FEC Not enabled for  lane      1.
```

```
-----------------------------------------------------
--------
FEC Not enabled for  lane      2.
-----------------------------------------------------
--------
FEC Not enabled for  lane      3.
-----------------------------------------------------
--------
<snip>
```

FEC counters not only detect bit errors, but they also provide predictive insights into the health of a network. This is demonstrated in Chapter 2, the section on *Case Study — An Online Retailer*.

Refer to Chapter 2, the section on Forward Error Correction for a detailed explanation on FEC. The same details apply to Ethernet networks because Fibre Channel reuse the FEC code from IEEE 803.2 standard for Ethernet. The terminology, however, may be different. Ethernet refers as "blocks" what Fibre Channel refers as "Transmission Words". Refer to Example 7-11 that shows Err Blocks Count. This counter increments when FEC is enabled, and it is unable to recover a bit error. Likewise, FEC block, FEC frame, and FEC codeword refer to the same entity carrying the FEC payload and parity bits. These details are explained in Chapter 2 and not repeated here.

Link Utilization

Example 7-9 shows multiple counters for calculating link utilization on Cisco Nexus 9000 switches and Cisco UCS servers.

- Interface speed
- Accumulated input and output bytes
- 30-seconds and 5-minute average input and output rate. These values are calculated by the difference (delta) of the accumulated values of input and output bytes and then dividing by the polling interval.

- Txload and Rxload are like percentage utilization. The higher the load, the more the link utilization. Load of 255/255 is 100% utilization. Both the load and percentage utilization are calculated by dividing throughput by the link speed.

Most Ethernet ports report similar counters.

Microburst Detection

Cisco Nexus 9000 switches can detect traffic burst within a small period, such as microseconds. This allows capturing the events that may cause congestion at a lower granularity but are unnoticed by other means due to long polling intervals.

Cisco Nexus 9000 switches can detect microburst when egress queue utilization exceeds a rise-threshold. Microburst ends when the queue utilization falls below a fall-threshold. The minimum microburst granularity is 0.64 microseconds and the duration is 73 microseconds at the time of this writing depending on the switch model.

The key point to understand about microburst detection is that it is reported on the egress queue on Cisco Nexus 9000 switches (Figure 7-10). As explained earlier in the section on *Ingress and Egress Queues*, sending of Pause frames depends on the utilization of the ingress queue. But this ingress queue fills up only after the egress queue fill up to an extent (not full). Therefore, microburst detection is an indication of egress congestion. Because egress congestion leads to ingress congestion, microburst detection is also an early indication of ingress congestion. In other words, it is an early indication of sending of Pause frames by the ingress ports on that switch.

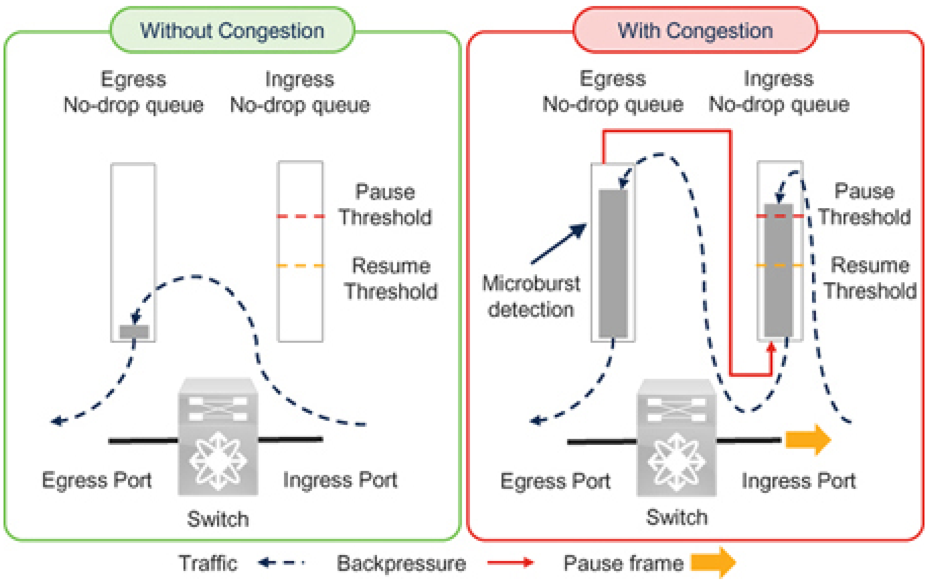


Figure 7-10 *Microburst detection on egress queues for lossless traffic*

For additional details, refer to Chapter 8, the section on Queue Depth Monitoring and Microburst Detection.

PFC Storm

PFC Storm has emerged as a term to refer to congestion in the lossless Ethernet networks (shared or dedicated) that carry RoCE and RoCEv2 traffic. This is probably because a port sends many Pause frames to slow down or stop the traffic, and it looks like a storm of PFC Pause frames on its neighbor.

But using this term may be misleading because “Storm” conveys a huge number of Pause frames, and it may be inaccurately related to congestion severity. A higher number of Pause frames may not necessarily indicate higher severity of congestion. This is because the result of Pause frames on the traffic depends on link speed, their type (zero or non-zero quanta), and their pattern.

The earlier sections on *Ethernet Flow Control* and *Pause Time* explain these details. After understanding the congestion detection metrics and troubleshooting, let’s revisit their practical effects.

Link Speed and PFC Storm

As the earlier section on *Pause Time* explains, the actual duration of traffic pause depends on the speed of a link. For example, a 10 GbE port could completely stop transmission if it receives just 3000 Pause frames per second, each with quanta of 65535 (pause time of 3.355 ms). Another port in the same network receives 6000 Pause frames per second, each with quanta of 65535, but still, it does not completely stop transmission because this is a 100 GbE port and it would require at least 30,000 Pause frames per second to completely stop the transmission.

In this case, the storm is more severe on the 100 GbE port, but actually, the 10 GbE port has a higher severity of congestion. Hence, only those links that operate at the same speed should be compared for detecting congestion based on the number of Pause frames.

Pause Time and PFC Storm

A port with fewer Pause frames with higher quanta is more severely affected as compared to a port that receives more Pause frames with smaller quanta values. As the earlier section on *When are the Pause Frames Sent?* explains, most implementations may use the max quanta value of 65535, and hence, this point may have lower practical significance. But you should verify the implementations of the products that are used in your environments.

Reason for Pause Frames and PFC Storm

Using the term—PFC Storm—does not convey the reason for congestion. As the earlier section on *Congestion in a Lossless Spine-Leaf Network* explains, both slow-drain and over-utilization result in sending Pause frames to the upstream directly connected devices.

The Pattern and Content of Pause Frames and PFC Storm

Among all other factors, the pattern of the received Pause frames is the most significant factor that practically affects data transmission. A secondary consideration is the state of the transmitter when it receives a Pause frame. Let's analyze both of these.

1. **The Pattern and content of Pause frames**: It would be incorrect to assume that when a Pause frame with a non-zero quanta is received, the traffic is stopped for the amount of time represented by the quanta. Practically, a Pause frame with a non-zero quanta may be followed quickly by an Un-Pause/Resume frame (Pause frame with zero quanta). If there were 3000 Pause frames in a second, did they all have max quanta? Did half of them have max quanta and the rest have zero quanta? If there were 3000 Pause frames all with max quanta, then that could completely stop transmission on a 10 GbE link for 1 full second. But if half of those Pause frames had zero quanta and were sent just 1 microsecond after the initial Pause frame with maximum quanta, then that may be just 1500 microseconds when traffic was paused in the same 1 second. In other words, just counting the Pause frames doesn't completely convey what is going on.
2. **The State of the Transmitter**: The time the traffic "may be" paused is the time between the Pause and Un-Pause frames. This is a "may be", and not "will be", because of a potential slight delay after receiving a Pause frame. Stopping the transmission is slightly delayed (not to be confused with pause time) after receiving a Pause frame because a port does not interrupt a frame that is already being transmitted at that moment. This delay depends on link speed, frame size, and when the Pause frame is received. If a 10 GbE port decides to stop transmission because of receiving a Pause frame when it is transmitting the last bits of a 1500-byte frame, the link transmission stops immediately. However, if the same 10 GbE port decides to stop transmission because of receiving a Pause frame when it starts transmitting a 1500-byte frame, the transmission does not stop for the next ((1500 x 8) bits / 10 Gbps) 1.2 microseconds. During this time, if an Un-Pause frame is received, then the port does not stop transmission despite receiving two Pause frames. If this sequence repeats every 1.2 microseconds

for 1 second, the port will report approximately ((1 second / 1.2 microsecond) x 2) 1.6 million Pause frames in that second while it continues to transmit at full capacity. Just by counting the number of Pause frames, this link may be categorized as under PFC Storm, but its behavior is significantly different from another 10 GbE link where just 3000 Pause frames can stop the transmission for 1 full second. Note the following additional points:

A. This example uses 1500-byte Frame size for the sake of simplicity. However, storage traffic may have a larger frame size of approximately 2300 bytes (FCoE) or even 4 KB (RoCEv2). As frame size increases, it may delay the action of the received Pause frame even longer, for example, 2.3 microseconds for a 2300-byte frame and 4.2 microseconds for a 4-KB frame on a 10 GbE link.

b. The Pause frame size is 64 bytes. Many pause frames may be received while one data frame is being transmitted.

c. As evident from this explanation, it would be inaccurate to calculate TxWait purely based on the pause quanta value or even the time difference between the received Pause and Un-Pause frame. An accurate TxWait value must calculate how long transmission was actually stopped.

This aspect of the “number of Pause frames” in Ethernet is similar to the “B2B credit transitions to zero” in Fibre Channel. A Fibre Channel port with one remaining-Tx-B2B-credit decrements this counter to zero and starts transmitting a frame. However, it may receive a credit while the frame is being transmitted, and hence, the next frame is not delayed at all. This condition results in incrementing the “B2B credit transitions to zero” counter, while the transmission does not actually stop.

Although such conditions are rarely reported and are even harder to detect, the key point to understand is that neither the “B2B credit transitions to zero” counter in Fibre Channel nor the “number of Pause frames” in Ethernet is a strong mechanism to detect congestion. Hence, using a term like PFC Strom, which is based on this counter, should be avoided because it may mislead some people into believing that the number of Pause frames is the real measurement of congestion.

This book uses TxWait and RxWait to indicate the actual amount of time transmission is stopped in each direction in a lossless network (Fibre Channel or lossless Ethernet). PFC Storm, as a term, may seem to fit well in environments where the cumulative pause time (TxWait and RxWait) are unavailable, and the number of Pause frames is the primary metric for detecting congestion. However, in the future, when the devices improve pause time detection (TxWait and RxWait), using the term—PFC Storm—will become even more misleading. For example, a 10 GbE port could show 100% TxWait with just 3000 Pause frames per second, whereas a 100 GbE port could show 10% TxWait even with 6000 Pause frames per second. The “Storm” severity may seem higher on the 100 GbE port, but the congestion severity is higher on the 10 GbE port.

One of the points that we wish to convey by this book is to use what has been learned on one transport type (such as Fibre Channel) to another (such as lossless Ethernet). Using “PFC Storm’ in Ethernet is like using “Credit Transition Storm” in Fibre Channel for referring to congestion. Many years ago, which now seems like a lifetime, when Fibre Channel switches did not provide TxWait, the “B2B credit transitions to zero” was the only metric for congestion detection. However, since the availability of TxWait, it has become the primary metric for detecting congestion, and “B2B credit transitions to zero” is only used as a last resort. Calling congestion “Credit Transition Storm” is inappropriate today, and so does calling it a “PFC Storm”. Because PFC Strom conveys the number of Pause frames, using it now means that lossless Ethernet is not really learning from Fibre Channel.

Of course, never use the term “Credit Transition Storm” in Fibre Channel fabric. For lossless Ethernet networks, despite this section arguing against using the term PFC Storm, some vendors, including Cisco, have used it. We will let the reader decide if and when the term—PFC Storm—feels misleading and then what you want to do after that.

Storage I/O Performance Monitoring

Traffic in a storage network is the direct result of an application initiating a read or write I/O operation. Because of this reason, network traffic patterns can be better understood by analyzing the application I/O profile, such as the timing, size, type, and rate of I/O operations. Essentially, the application I/O profile helps in understanding why the network has traffic or congestion.

Chapter 5 explains solving congestion by storage I/O performance monitoring. The same details, at least conceptually, also apply to FCoE and RoCE (except the transport protocol differences) because the upper layers (SCSI and NVMe) are the same. Before proceeding, please refer to the following sections in Chapter 5, which are not repeated here for the sake of brevity.

- Chapter 5, the section on *Why Monitor Storage I/O Performance?* explains the basic value of monitoring storage I/O performance.
- Chapter 5, the section on *How and Where to Monitor Storage I/O Performance* explains three locations for storage I/O performance monitoring—hosts, storage arrays, or the network.
- Chapter 5, the section on *Cisco SAN Analytics* explains how Cisco MDS switches monitors storage I/O performance natively within the switches. This capability is called SAN Analytics and it is available only on the Fibre Channel ports. This section also helps in understanding why a similar capability does not exist on Ethernet switches.
- Chapter 5, the section on *Understanding I/O Flows in a Storage Network* helps in understanding the difference between I/O flows in Fibre Channel fabrics versus

lossless Ethernet networks. Pay special attention to the sub-section on *I/O Flows versus I/O Operations*.

- Chapter 5, the section on *I/O Flow Metrics* helps in understanding how the performance of I/O flows can be monitored using various metrics, such as I/O completion time (called Exchange Completion Time in Fibre Channel), IOPS, Throughput, and I/O size.

After understanding these sections from Chapter 5, next, be aware that the performance of lossless Ethernet networks can be monitored at the following levels:

1. **Port or Traffic Class**: Most end devices and switches report counters such as packets sent/received and bytes sent/received on a network port or interface. Traffic for no-drop class can be monitored separately than other classes.
2. **UDP Flows**: Flows at layer 4 of the OSI model are identified by 5-tuples—Source IP, Destination IP, Source port, Destination port, and layer-4 protocol (TCP, UDP, etc.). In other words, RoCEv2 traffic can be classified in UDP flows. Counters such as packets sent/received, bytes sent/received, and dropped packets for each UDP flow can be monitored separately depending upon the capability of the devices in a network.
3. **I/O Flow**: An I/O flow is aware of SCSI or NVMe I/O operations to a storage volume (logical unit for SCSI and Namespace for NVMe). Monitoring performance at an I/O flow level allows calculating how long I/O Operations are taking to complete, their throughput, IOPS, type (read or write), I/O size, etc.

UDP Flow Monitoring versus I/O Flow Monitoring

UDP flow monitoring shouldn’t be confused with I/O flow monitoring because of the following reasons:

1. UDP belongs to the transport layer (layer 4) of the OSI model. It is not aware of the functions of the upper layer protocols such as RDMA, SCSI and NVMe.

2. Many I/O operations may travel in a single UDP flow. These I/O operations may belong to different I/O flows. Refer to Chapter 5, the section on *I/O Flows versus I/O Operations*.
3. As mentioned, a UDP flow has its own performance monitoring metrics, such as packets transferred per second, throughput, etc. This is different from the performance monitoring of I/O flows, such as IOPS, I/O throughput, time taken to complete I/O operations, I/O size, etc.

Unavailability of I/O Flow Monitoring in Lossless Ethernet Networks

At the time of writing, performance monitoring of I/O flows is not available in lossless Ethernet networks. Ethernet switches may report network latency, which is typically the time spent by a packet in the network. This is not the I/O completion time. Likewise, Ethernet switches may report the throughput of a UDP flow, but this is not read or write I/O throughput or IOPS.

Although the IP addresses of the initiator and the target can be considered to make an IT flow, this flow definition is of little value without knowing the I/O flow metrics.

Even in the future, it is unlikely that Ethernet switches will be able to monitor I/O performance similar to SAN Analytics on Fibre Channel ports on Cisco MDS switches. This is because Ethernet networks carry thousands of upper layer protocols each with different TCP and UDP port numbers. Enhancing the Ethernet switches to decode just the storage protocols (FCoE, RoCE, etc.) and measure their performance may not justify the additional cost for making these enhancements. This was not a challenge for Cisco MDS switches because Fibre Channel is purposefully built for storage traffic. In contrast, Ethernet networks carry all kinds of traffic, as explained by Figure 1-11 in Chapter 1.

Alternative Approaches

An alternative approach is to monitor storage I/O performance within the hosts or the storage arrays. Chapter 5 explains these details and how to use the I/O performance metrics for solving congestion problems. Although Chapter 5 focuses on Fibre Channel, the concepts apply to lossless Ethernet networks as well.

Some Ethernet switches (such as Cisco Nexus 9000 switches) monitor the performance of UDP flows. But, as mentioned, this is not the performance monitoring of the I/O flows, which are carried within those UDP flows. Hence, use Ethernet switches for UDP flow monitoring together with I/O performance monitoring on hosts and storage arrays. A RoCEv2 target/controller uses UDP port 4791, which means packets destined for targets/controller has destination port 4791, whereas packet destined for the host has source port 4791. UDP Port 4420 is assigned for NVMe/RoCE. Information from various sources can be correlated to build your own solution.

FCoE I/O Operations

SCSI and NVMe I/O Operations in FCoE networks are the same as Fibre Channel fabrics. Refer to Chapter 5 for more details.

But SAN Analytics is available only on Fibre Channel ports on Cisco MDS switches. If traffic passes through the Fibre Channel ports with analytics capability at least once in the end-to-end data path, it can still be inspected to collect I/O flow metrics. This approach works even if traffic passes through FCoE ports at other locations in a network. A typical example is Cisco UCS servers, which use FCoE internally. When the same traffic reaches Fibre Channel ports on MDS, it can be inspected to collect I/O flow metrics using SAN Analytics.

RoCE I/O Operations

Figure 7-11 shows NVMe over RoCE read I/O operation. A host initiates a read I/O operation by sending RDMA_SEND in a command capsule to the target. The target sends the data to the host via RDMA_WRITE in one or more packets based on the amount of data requested by the I/O Operation and the Maximum Transmission Unit (MTU) of the network (more details in Chapter 8, the section on *IP MTU and TCP MSS Considerations*). Finally, the I/O operation completes when the target sends a Response capsule.

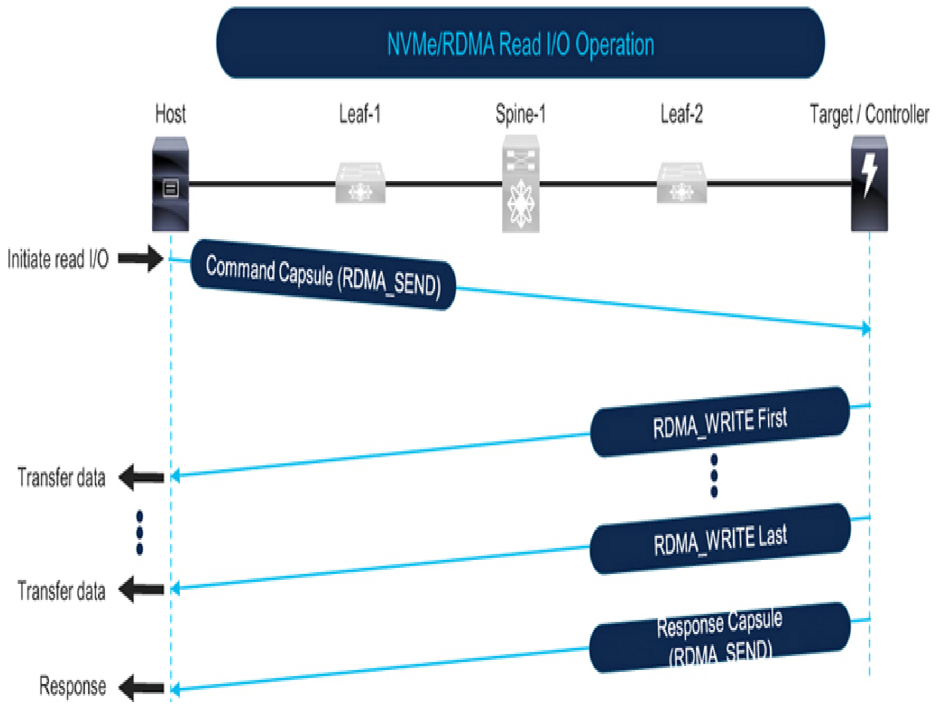


Figure 7-11 *NVMe over RoCE read I/O operation*

Figure 7-12 shows NVMe over RoCE write I/O operation. A host initiates a write I/O operation by sending RDMA_SEND in a command capsule to the target. The target then sends RDMA_READ request to the host. Next, the host sends data to the target via RDMA_READ response in one or more packets based on the amount of data requested by the I/O Operation and the MTU of the network. Finally, the I/O operation completes when the target sends a Response capsule.

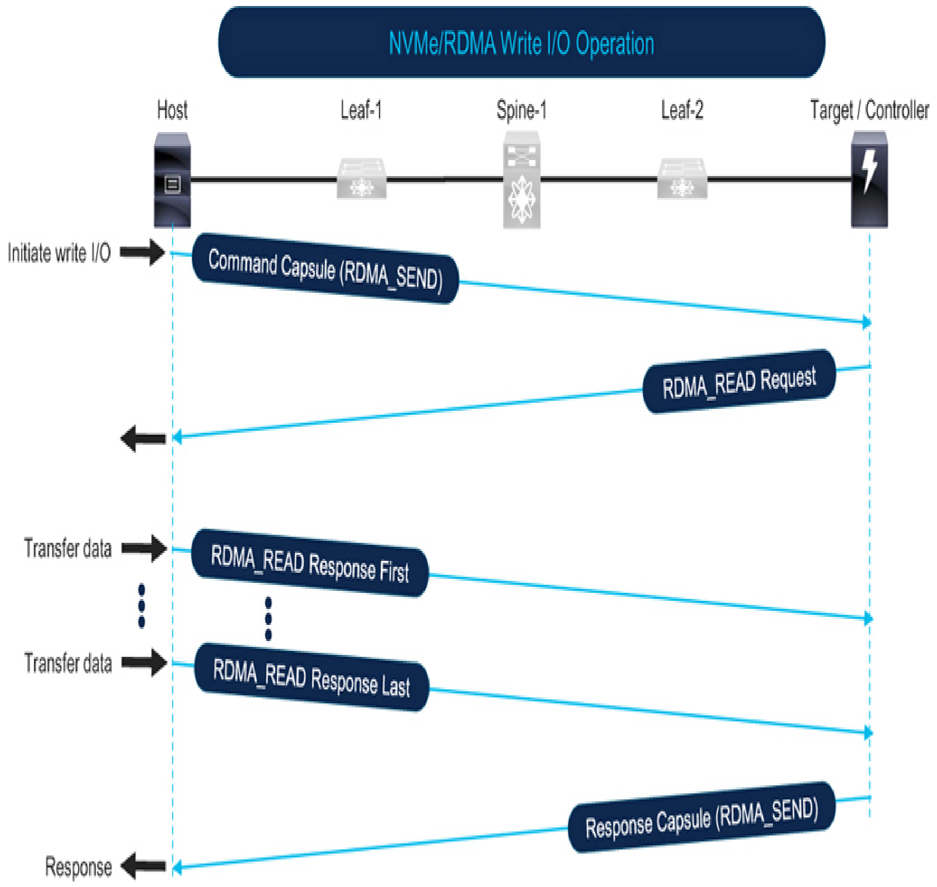


Figure 7-12 *NVMe over RoCE write I/O operation*

Table 7-2 shows the Ethernet frame sizes and directions based on the type of RDMA operation.

Table 7-2 *Typical size of Ethernet frames for NVMe/RoCE I/O operations*

NVMe/RoCEv2 Packet	Ethernet Frame Size
RDMA_READ Request	78-byte Ethernet frame
RDMA_READ Response or RDMA_WRITE	Approximately 700-byte Ethernet frame for 512-byte InfiniBand MTU 1100-byte Ethernet frame for 1024-byte InfiniBand MTU 2200-byte Ethernet frame for 2048-byte InfiniBand MTU 4200-byte Ethernet frame for 4096-byte InfiniBand MTU
RDMA_SEND	126-byte Ethernet frame

Correlating I/O Operations, Traffic Patterns, and Network Congestion

Please refer to the following sections in Chapter 5, which are not repeated here for the sake of brevity.

- Compare the section on *I/O Operations and Network Traffic Patterns* in Chapter 5 with the previous section to note striking similarities between traffic patterns. Therefore, the correlation with network congestion is also similar. For reading data, SCSI and NVMe use Read CMD, whereas RDMA uses the RDMA_WRITE verb in the reverse direction. Likewise, for writing data, SCSI and NVMe use the Write CMD, whereas RDMA uses the RDMA_READ verb in the reverse direction.
- Chapter 5, the section on *Network Traffic Direction* explains traffic on various port types because of read and write I/O operations.
- Chapter 5, the section on *Correlating I/O Operations, Traffic Patterns, and Network Congestion* explains that the key reason for congestion on a host link is the multiple concurrent large-size read I/O operations from that host. Likewise, the key reason for congestion on a storage link is the total amount of data being requested by that storage array.

Detecting Congestion on a Remote Monitoring Platform

Remote monitoring platforms can monitor all the ports in a network simultaneously to provide network-wide single-pane-of-glass visibility.

Refer to Chapter 3, the section on *Detecting Congestion on a Remote Monitoring Platform*, which explains using the following types of monitoring applications:

- An application developed by the device manufacturer/vendor, such as Cisco Nexus Dashboard Fabric Controller (NDFC) and Nexus Dashboard Insights.

- A 3rd party or a custom-developed application, such as the MDS Traffic Monitoring (MTM) App.

This section provides a high-level overview of Cisco Nexus Dashboard Insights for detecting congestion in Ethernet networks.

To learn more about a custom-developed application for detecting and troubleshooting congestion in lossless Ethernet networks, refer to Chapter 9, which explains using the UCS Traffic Monitoring App (UTM).

Chapter 3 also explains *The Pitfalls of Monitoring Network Traffic*. Its sub-section on *Average and Peak Utilization* apply to lossless Ethernet networks as well.

Congestion Detection using Cisco Nexus Dashboard Insights

Cisco Nexus Dashboard Insights receives metrics from switches and compute nodes at a low granularity of 1 second. It then analyzes the raw metrics using baselining, correlation, and forecasting algorithms to provide deep insights into traffic patterns. For congestion detection, Nexus Dashboard Insights detects data plane anomalies such as packet drops, latency, microbursts, and so on. It shows the end-to-end packet path of a flow with drops and drops reasons using an intuitive GUI.

Figure 7-13 shows end-to-end path of a RoCEv2 flow, average network latency, and burst on Nexus Dashboard Insights.

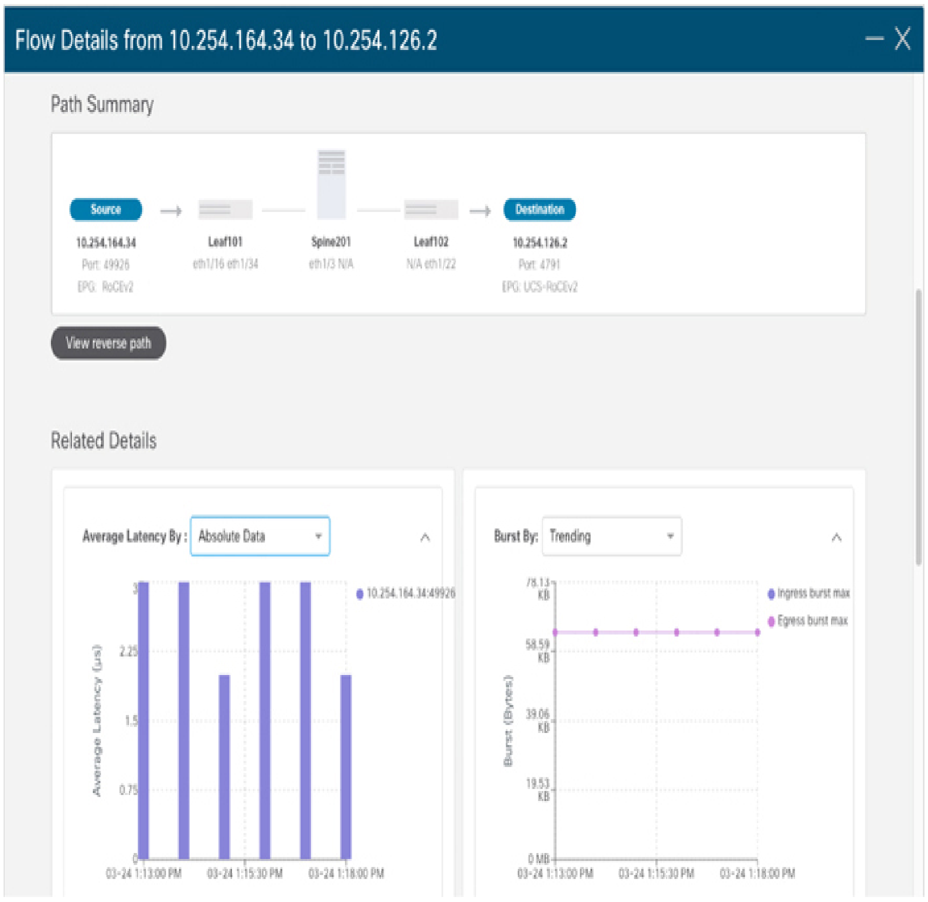


Figure 7-13 *Monitoring RoCEv2 in Cisco Nexus Dashboard Insights*

Metric Export Mechanisms

The mechanism for exporting metrics is a major consideration for a custom-developed application or a script. Most details that are explained in Chapter 3, the section on *Metric Export Mechanisms* apply to lossless Ethernet networks as well.

Pay special attention to the *Recommendations* for metric exports as explained in Chapter 3. Using the command-line outputs and SNMP has been historically common but using the APIs has become the norm now. For low-granularity metric export at scale, streaming telemetry is the best option, and it is seeing rapid adoption.

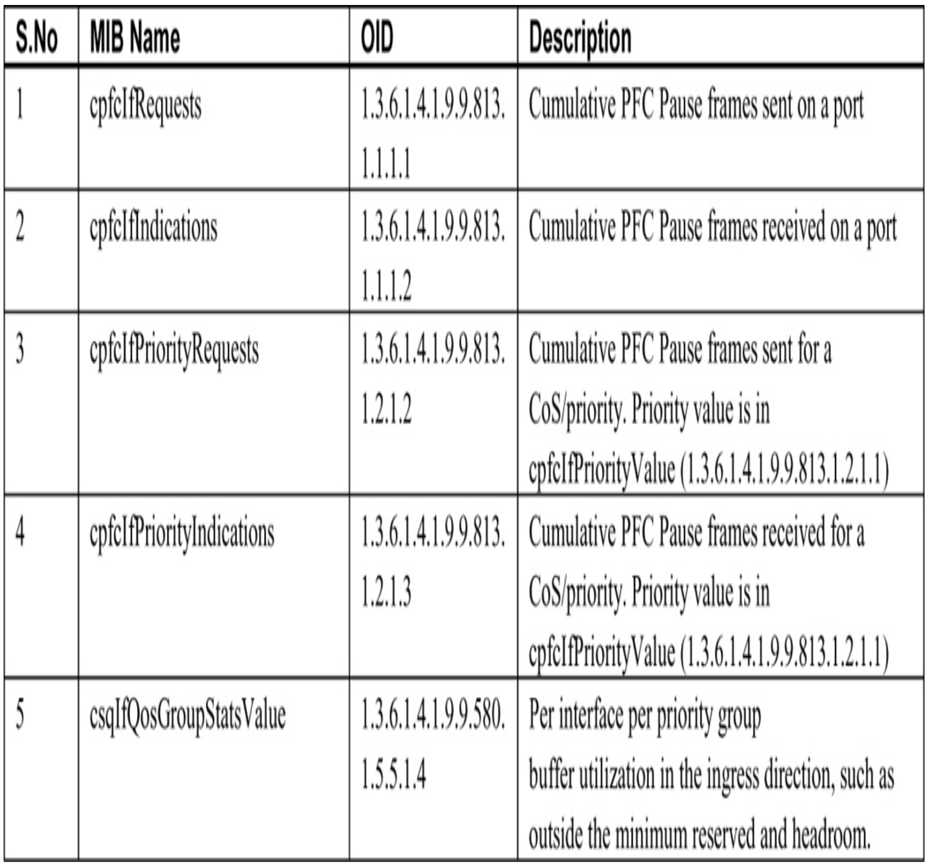
Cisco Nexus Dashboard Fabric Controller and Nexus Dashboard Insights select the best export mechanism by default.

The following sub-sections provide a brief overview of metrics export mechanisms on Cisco Nexus 9000 switches without repeating what was already explained in Chapter 3.

SNMP

Table 7-3 provides SNMP MIBs of the metrics that can detect congestion on LLFC and PFC-enabled ports.

Table 7-3 *SNMP MIBs for detecting congestion on Cisco Nexus switches*



S.No	MIB Name	OID	Description
1	cpfcIfRequests	1.3.6.1.4.1.9.9.813.1.1.1.1	Cumulative PFC Pause frames sent on a port
2	cpfcIfIndications	1.3.6.1.4.1.9.9.813.1.1.1.2	Cumulative PFC Pause frames received on a port
3	cpfcIfPriorityRequests	1.3.6.1.4.1.9.9.813.1.2.1.2	Cumulative PFC Pause frames sent for a CoS/priority. Priority value is in cpfcIfPriorityValue (1.3.6.1.4.1.9.9.813.1.2.1.1)
4	cpfcIfPriorityIndications	1.3.6.1.4.1.9.9.813.1.2.1.3	Cumulative PFC Pause frames received for a CoS/priority. Priority value is in cpfcIfPriorityValue (1.3.6.1.4.1.9.9.813.1.2.1.1)
5	csqIfQosGroupStatsValue	1.3.6.1.4.1.9.9.580.1.5.5.1.4	Per interface per priority group buffer utilization in the ingress direction, such as outside the minimum reserved and headroom.

Note the following points:

1. PFC counters on Cisco Nexus 9000 switches can be monitored by CISCO-PFC-EXT-MIB, which contains many more counters, such as for TxWait and RxWait, but Table 7-3 does not list all of them because Nexus switches do not support TxWait and RxWait at the time of this writing. Although the switches may respond to the MIB requests, their values do not change over time. CISCO-PFC-EXT-MIB can also be used to monitor the PFC watchdog if supported by the switch type.

2. As a caution, over the years, LLFC and PFC MIB counters on Cisco devices have been affected by lack of implementation on certain firmware versions and switch models. Before relying on the returned values, verify that they match the command-line output on the switch. An initial verification based on the returned value of zero is not enough because although zero is a valid value, it may not increment. This would give a false indication that there is no congestion.
3. IF-MIB contains interface speed, bytes in, and bytes out, which can be used to calculate percentage utilization.
4. Because PFC is implemented using QoS on the Ethernet switches, monitoring CISCO-SWITCH-QOS-MIB provides per-queue metrics.

Streaming Telemetry

Refer to Chapter 3, the section on *Streaming Telemetry* for details on streaming telemetry. Cisco Nexus 9000 switches have the following additional considerations:

1. Nexus switches can export metrics from front-panel data ports, which allows low-granularity metric export. MDS switches, at the time of this writing. are limited to exporting the metrics only from the management ports on the switches.
2. Nexus switches support NetFlow and sFlow. However, streaming telemetry can export the metrics at a much lower granularity.
3. Software telemetry on Cisco Nexus switches can export control-plane information and interface metrics.
4. Additionally, Nexus switches support hardware telemetry for exporting metrics at a granularity (as low as 1 second based on the switch type):
   a. Streaming Statistics Export (SSX) for raw ASIC statistics.
   b. Flow Table (FT) for exporting flow-level information.

c. Flow Table Events (FTE) for triggering notifications when configured conditions are met.

5. Nexus switches support In-band Network Telemetry (INT) for monitoring dropped packets and congested queues.

The field of network telemetry and analytics is evolving fast. We recommend referring to the documentation and release notes for knowing the capabilities of products in your environment and how to use them.

Troubleshooting Congestion in Lossless Ethernet Networks

Troubleshooting congestion in lossless Ethernet networks is the same as the Fibre Channel fabrics. Both use hop-by-hop flow control mechanisms, just that their implementations are different. When a switchport shows egress congestion, the source of congestion lies in the downstream traffic path. When a switchport shows ingress congestion, it must be because one or more ports on that switch are congested in the egress direction.

Goals

As Chapter 4 explains in detail, the goals of troubleshooting are twofold:

1. **Identify the source (Culprits) and cause of congestion**: The primary goal of the investigation and troubleshooting is to determine if there is congestion, the source of congestion, and the reason for congestion, which could be slow-drain (detected by high TxWait, if available, or fast incrementing Pause count) or over-utilization (detected by high egress utilization or microburst events). Once the source and cause of congestion are known, the culprit end device(s) can be investigated in detail.
2. **Identify the affected devices (Victims)**: The secondary goal is to identify the devices that were adversely affected

by the culprit device(s). These are the victims. Until the investigation completes, you may not know if a device is a culprit or a victim. Remember, sometimes it is helpful to identify different victim types—Direct victims, Indirect victims, and Same-path victims. Refer to Chapter 4, the section on *Identify the Affected Devices (Victims)* for more details.

It is important to understand that the victim devices may report performance degradation as the culprit devices. But the problem can only be solved by investigating the culprit(s), not the victims.

Congestion Severities and Levels

Chapter 4 defines Congestion Severities and Levels for Fibre Channel fabric. But they can't be directly used in lossless Ethernet networks because, unlike Fibre Channel, Lossless Ethernet does not have a concept of link (credit) reset, which is a symptom of severe (Level 3) congestion in Fibre Channel. Another reason is that if the devices in an environment do not have metrics to detect these conditions, categorizing them in various levels has limited practical significance. Notable examples are the TxWait and RxWait metrics, which are unavailable on most Ethernet ports at the time of this writing. Hence, unlike Fibre Channel, Level 1 congestion can't be detected using TxWait.

Overall, for Lossless Ethernet, the symptoms of congestion at various severities can be slightly modified as follows:

1. **Mild Congestion (Level 1)**: Increased Latency but no frame drops.

   a. Detect with excessive Pause frame count or TxWait/RxWait, if available, and high link utilization.

2. **Moderate Congestion (Level 2)**: Increased Latency and Frame drops.

   a. Detect with packet drops in no-drop class

3. **Severe Congestion (Level 3)**: Increased Latency, Frame drops, and a continuous duration of traffic pause.

a. Detect with pause timeout or PFC watchdog, explained later

Methodology

We recommend troubleshooting congestion in a decreasing level of severity. Recall that these congestion severities are not standardized. Their sole purpose is to help a user in developing a practical workflow, which helps in detecting and troubleshooting congestion quickly and accurately. You can customize it for your environment. For example, when no easy way of detecting Level 3 congestion is available, investigate Level 2 congestion (packet drops) followed by Level 1 congestion (Pause frame count or TxWait, if available).

Refer to Chapter 4 section on *Troubleshooting Methodology - Decreasing Level* for more details. While applying the information from Chapter 4 to lossless Ethernet, remember that Rx credit unavailability is the same as ingress congestion in Fibre Channel, which is detected by Tx Pause in Ethernet networks. Likewise, Tx credit unavailability is egress congestion in Fibre Channel, which is detected by Rx Pause in Ethernet networks.

We recommend studying the Chapter 4 case studies as well. Although they are described for Fibre Channel, learn from them how the troubleshooting methodology would be similar in lossless Ethernet networks.

Troubleshooting Congestion in Spine-Leaf Topology

Refer to Figure 7-14, which is a subset of the spine-leaf network of Figure 7-8. As the earlier section on *Congestion in a Lossless Spine-Leaf Network* explains, one culprit may victimize many devices, and users of these victim devices may report performance degradation as well. Because of this reason, it is natural to start troubleshooting where a problem was reported first, which can be victim devices or their connected switchports. Regardless of where to start, it is important to follow congestion to its source while looking for higher severity events followed by lower severity events (Level 3 ➔ Level 2 ➔ Level 1).

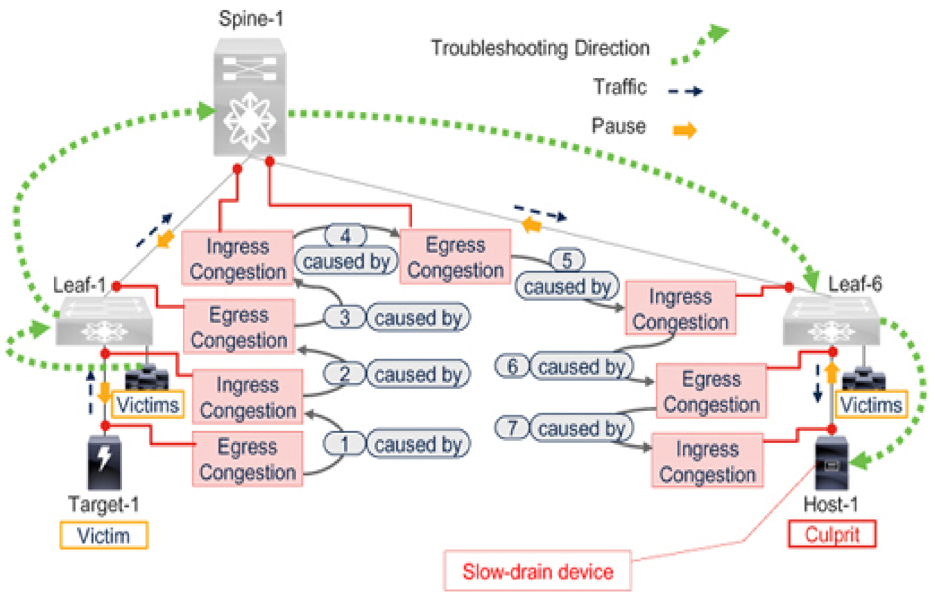


Figure 7-14 *Congestion troubleshooting direction in a lossless spine-leaf network*

Assume that the applications on victim hosts connected to Leaf-1 report performance degradation. These are indirect victims because they receive traffic from Target-1, which is a direct victim of the culprit Host-1. But this is not known until troubleshooting concludes. The troubleshooting workflow would be as follows:

1. Go to the directly connected switchport of the victim host. If symptoms of egress congestion (Rx Pause or egress packet drops) are found, then this host is the culprit.
2. If not, find symptoms of ingress congestion (Tx Pause) on any other edge port on the same switch. If such a device is found (Target-1), it must be sending traffic to the culprit (Host-1) (if there is only one congestion event or culprit).
3. Then, look for symptoms of egress congestion on the upstream ports on Leaf-1.
4. Go to one of the upstream devices (Spine-1, for example). The Tx Pause sent by Spine-1 should match with Rx

Pause on the upstream port on Leaf-1. If their values differ, investigate bit errors or firmware bugs.

5. On Spine-1, find the ports with symptoms of egress congestion (Rx Pause). These ports will typically be transmitting the traffic that is received from the ports that show ingress congestion.
6. Go to the next device in the traffic path (Leaf-6). The Tx Pause sent by Leaf-6 should match with the Rx Pause on Spine-1. As mentioned, if their values differ, investigate bit errors or firmware bugs.
7. On Leaf-6, look for edge ports with symptoms of egress congestion (Rx Pause or egress packet drops). This should be the switchport that connects to the culprit host (Host-1). If Rx Pause or egress packet drops are not found, then investigate ports for over-utilization.

On any device, if more ports show symptoms of egress congestion, follow the higher severe symptoms first. For example, follow packet drops before Rx pause. Likewise, follow fast-incrementing Pause counts before slow-incrementing Pause counts if TxWait is unavailable.

Reality Check

The troubleshooting workflow shown in Figure 7-14 requires a reality check because of the following points.

1. This troubleshooting workflow can only be used during an ongoing congestion condition because, unlike Cisco MDS switches and Nexus 7000/7700 switches, most Ethernet switches (including Cisco Nexus 9000 switches) do not maintain a history of congestion events with time and date stamps.
2. Even troubleshooting in real time is tedious because most Ethernet switches do not report percentage TxWait and RxWait. On Cisco Nexus 9000 switches and Cisco UCS servers, only the cumulative Pause count is available in the command outputs. A user must execute these commands multiple times and manually calculate the

difference as shown in Example 7-7. Doing these steps for hundreds of ports is difficult, time-consuming, and error prone. Imagine calculating this manually on the spine switches in Figure 7-8 that may have hundreds of ports.

3. As the earlier section on the *Number of Pause Frames* explains, a nominal increase in the Pause frame count does not necessarily indicate congestion.
4. Be aware that there could be more than one culprit causing congestion at a time. If the culprits are on different switches, then there might be more than one path when following the congestion.
5. Finally, because the events are not time and date-stamped if the congestion condition ceases while interpreting the number of Pause frames manually, there is no way to know when their count increased.

Because of these reasons, troubleshooting congestion on lossless Ethernet networks is difficult using the command-line interface.

Troubleshooting Congestion using a Remote Monitoring Platform

The unpleasant reality of troubleshooting congestion in lossless Ethernet networks can be changed by using a remote monitoring platform, which continuously polls the number of Pause frames to maintain a time and date-stamped history.

The UCS Traffic Monitoring (UTM) app is an example of such an application. Refer to Chapter 9 for more details on it. The UTM app can detect and troubleshoot congestion in near real-time using comparative analysis, treading, and seasonality.

Comparative Analysis

Compare the rate of Pause frames on the network ports (host ports and switch ports) and detect if a few ports have an excessively higher count than others.

In Figure 7-8, with thousands of hosts,

1. Poll the Tx and Rx Pause from the edge switchports or hosts every 60 seconds.
2. Calculate the delta of the accumulated number of Pause frames to know the change over the 60-second duration.
3. Sort the hosts in the descending order of the Tx Pause or edge switchports in the descending order of Rx Pause.
4. Investigate the top 10 hosts in this list. Typically, these hosts have higher severity of slow-drain.

The same comparative analysis should be used across all similar entities. For example, compare all the spine ports with each other and detect if a few ports report an excessive number of Pause frames.

Trends and Seasonality

The Pause frames are important for the operation of a lossless Ethernet network, and hence, their nominal activity is fine. But analyze any spikes and dips carefully. Also, find if the Pause frame count has been on the rise over the last few days or weeks, although there may not be any sudden spikes. Additionally, find any seasonality, that is, if the spikes in the number of Pause frames are observed during specific hours in a day or days in a week, or even months in a year.

In graphical terms, a straight line with low counts is fine. Pay attention to spikes, especially big spikes that sustain longer.

Monitoring a Slow-drain Suspect

A suspect is an end device that sends Pause frames. But it may or may not be a culprit. To be sure, more information is needed because, as mentioned earlier, just counting the number of Pause frames does not convey how long transmission was really stopped.

Based on our experience, the following are some approaches that help in deciding if a suspect is also a culprit.

1. An end device that does not send Pause frames is not the source of slow-drain. These devices can be excluded from the suspect list, allowing to focus only on those devices that send Pause frames.
2. Up to a few hundred Pause frames per second (such as less than 300) from an end device does not make it a suspect. However, the end device becomes a suspect as the Pause frame rate increases to a large multiple of hundreds or thousands per second. Millions of Pause frames per second definitely make an end device very close to being a culprit and should be monitored actively.
3. Check the symptoms of congestion spreading on the upstream ports. For example, in a spine-leaf topology, does the spike in the Pause frame rate from a host matches with the spike in the Pause frame rate seen by the ports on the spine switches that are in the traffic path to the host? If yes, this indicates congestion spreading, hence making the host a culprit. If not, it indicates the host paused the traffic only momentarily, hence does not make it a culprit.
4. Compare the rate of Pause frames sent on two (or more) ports by an end device. Typically, their rates should be uniform. But an investigation is needed if one port sends hundreds, whereas the other port sends millions of Pause frames per second or a similar non-uniform Pause frame rate.
5. A high Pause frame rate or an increase in the rate of Pause frames must be correlated with events at the higher layers in the end devices. For example, I/O errors on the end device, application performance degradation, increased I/O completion time, and reduced IOPS. This correlation must be performed on the suspect end device (that sends Pause frames) and other end devices (that do not send Pause frames) in the network. This is because, if the suspect is really a culprit, congestion caused by it would spread and victimize many other end devices.
6. Excessive Tx Pause frames from a network port should result in a reduced Rx traffic rate on that link. If your

results are different, it probably indicates that the Pause frames are either not reaching the traffic sender, not being handled correctly, or you are witnessing the limitation of detecting congestion only using the number of Pause frames in the absence of TxWait/RxWait, explained earlier in the section on *PFC Storm*. The important point to remember is to compare Pause frames and traffic in opposite directions.

Monitoring an Over-utilization Suspect

As explained in Chapter 1, the sections on *Defining Full Utilization vs High Utilization vs Over-Utilization* and *Monitoring Full Utilization vs High Utilization vs Over-Utilization*, we recommend treating any occurrence of high utilization (such as more than 80%) the same as over-utilization. This is because, in our experience, most occurrences of high utilization also have some duration of over-utilization. To be conclusive if the high utilization is causing congestion, monitor the pause frames on the upstream ports. But start the investigation by treating the high utilization occurrences at the same severity as over-utilization unless proven otherwise. This approach saves time in detecting the congestion and finding a solution.

FC and FCoE in the Same Network

Some switches, such as Cisco Nexus 9000 switches, Nexus 5000/6000 switches, and Cisco UCS Fabric Interconnects can have FC, FCoE (lossless), and Ethernet (lossy) ports. These switches spread congestion from a flow-controlled egress port to a flow-controlled ingress port in a similar way as explained earlier. The only difference is in the mechanism of flow control and hence, the metrics to detect congestion are also different.

Recall that FC ports detect ingress congestion using Rx B2B credit unavailability and egress congestion using Tx B2B credit unavailability. Refer to Chapter 3 for a detailed explanation of the Fibre Channel metrics.

FCoE ports (lossless Ethernet) detect ingress congestion using Tx Pause and egress congestion using Rx Pause. The earlier section on *Congestion Detection Metrics* explains these metrics.

Consider Figure 7-15 with the following components.

- **Lossy Ethernet**: A spine-leaf network with no hop-by-hop flow control, such as PFC or LLFC. Traffic in this part of the network relies on other mechanisms, such as TCP at layer 4 of the OSI model, for congestion control.
- **Lossless Fibre Channel**: An edge-core Fibre Channel fabric with B2B flow control. MDS-1 connects the storage arrays (Target-1 and Target-2).
- **Converged (Lossy + Lossless) Ethernet**: The leaf switches (L-4 and L-6) are shared between the lossy Ethernet network and Fibre Channel fabric. They have dedicated Ethernet uplinks (without flow control) to connect to the spine switches, and Fibre Channel uplinks to connect to MDS-1. The downlinks connect to one or more FEXes (FEX-4 and FEX-6), which ultimately connect to the hosts. The leaf to FEX links and FEX to host links use PFC. Lossless FC/FCoE traffic remains in the no-drop class, whereas lossy Ethernet traffic remains in other classes that are not flow-controlled.

FEX represents a Cisco Fabric Extender. Physically, it looks like a switch but is really like a remote module. It needs a parent switch for control functions (such as routing protocol) and management functions (such as configuring a port). The links between the parent switch and FEX uses SFPs and cables and run hop-by-hop flow control, such as PFC. This is similar to an ISL. Hence, for handling congestion, consider a FEX as a switch, although in a generic way, a FEX may not qualify as a switch by itself.

Figure 7-15 is a close representation of Cisco UCS architecture. Leaf switches represent Fabric Interconnects, FEX represents I/O Modules (IOM), and hosts represent the servers. Refer to Chapter 9 for more details on Cisco UCS servers.

Let’s analyze congestion events caused by Host-1 and Host-2 and follow the troubleshooting methodology.

Congestion Spreading due to Slow-Drain

In Figure 7-15, Host-1 is a slow-drain device because it has a slower processing rate of lossless traffic as compared to the rate at which the frames are being delivered to it. So, it sends PFC Pause frames to FEX-4 to reduce traffic rate in the no-drop class. Traffic outside the no-drop class is not flow-controlled and hence, can be dropped when queues are full.

When FEX-4’s Pause Threshold is exceeded, it sends PFC Pause frames to L-4 to reduce the traffic rate in the no-drop class. But L-4’s uplink (ingress port) runs Fibre Channel. When its buffers are consumed, it reduces the rate of sending R_RDYs to its upstream neighbor (MDS-1). Finally, MDS-1 reduces the rate of sending the R_RDYs to the targets for reducing traffic rate from the targets.

Congestion Spreading due to Over-Utilization

Consider Host-2 in Figure 7-15. It is not causing slow-drain, however, it is causing 100% utilization in the egress direction of its switchport, which may cause congestion due to over-utilization. While chasing the source of congestion from MDS-1 in a similar way as explained earlier, after reaching FEX-6, you may not find any egress port with Rx Pause increments. In such cases, look for ports that are running at high egress utilization, which is an indication of congestion due to over-utilization. This would result in finding the culprit Host-2.

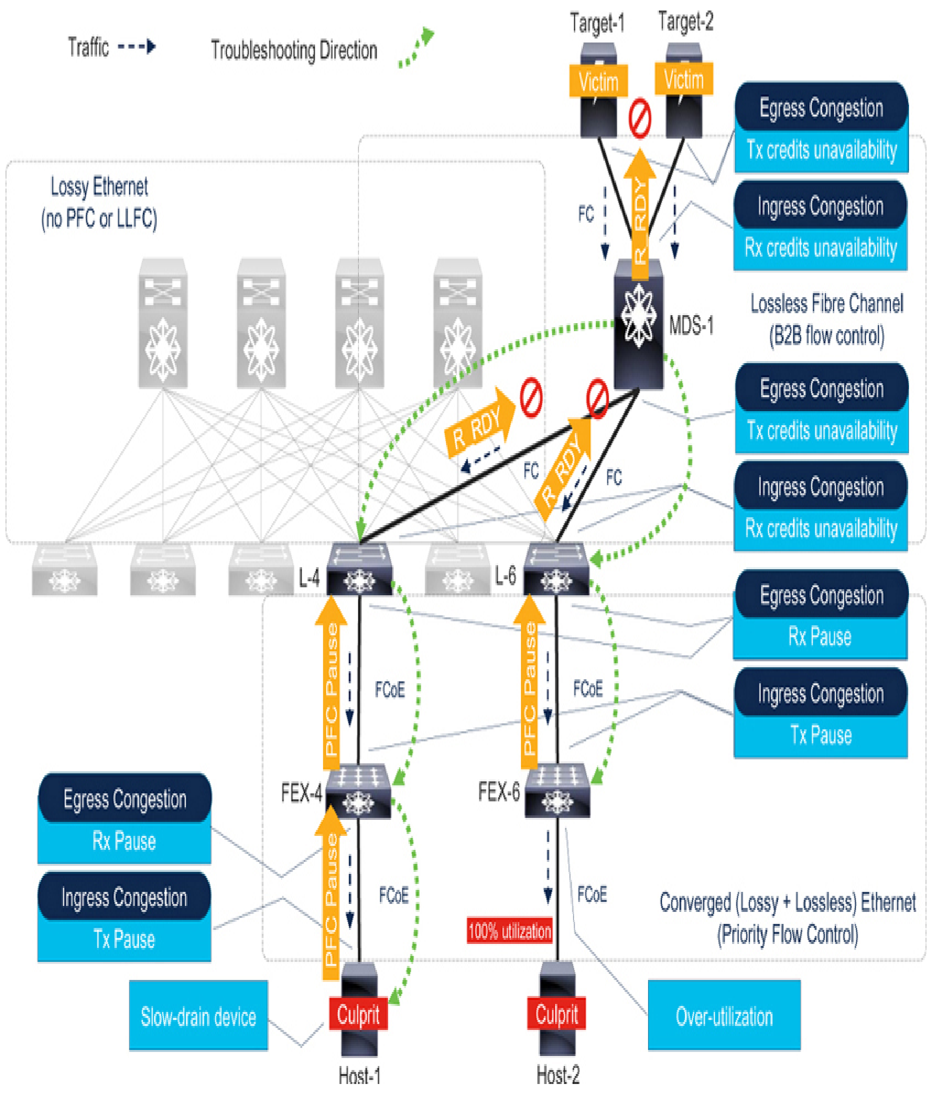


Figure 7-15 *Congestion troubleshooting workflow with FC and FCoE in the same network*

Note the following points:

1. Congestion in mixed FC and FCoE networks spreads similarly to only-FC or only-FCoE networks.
2. Slow-drain and over-utilization lead to similar symptoms on the uplinks of FEX-4 and FEX-6. Finding the cause of congestion is not possible even on L-4, L-6, and MDS-1. Only the edge links on FEX-4 and FEX-6 can clearly differentiate the cause of congestion because they directly connect to the source of congestion (culprit end device).
3. Congestion troubleshooting on lossless Ethernet links (L-4 and L-6 downlink) and Fibre Channel links (L-4 and L-6 uplinks) remains the same as explained earlier. The

difference is on the L-4 and L-6 switches themselves for chasing the source of congestion. On these switches, use Fibre Channel metrics on the ingress (FC) port and the PFC metrics on the egress (Ethernet) port. Their commands are different. Execute both types of commands to check egress congestion on the switches for FC and FCoE traffic.

4. In Figure 7-15, if L-4 does not provide ingress congestion detection metrics on uplink FC ports (for example Cisco UCS Fabric Interconnects), an alternative is to use the egress congestion detection metrics on MDS-1. Likewise, if Host-1 does not monitor Tx Pause frames (or does not make them easy enough), an alternative is to monitor Rx Pause on FEX-4.

Bit Rate Differences between FC and FCoE

Differences in the bit rates must be carefully accounted for when a switch transfers traffic between FC and FCoE ports. As explained in Chapter 2, the sections on *Fibre Channel Bit Rate, Difference Between Fibre Channel Speed and Bit Rate*, and *Table 2-4*, some Fibre Channel bit rates have considerable differences from the advertised speeds. For example, 8GFC port has a bit rate of 8.5 Gbps, but can transfer data only at 6.8 Gbps after accounting for the 8b/10b encoding. This makes 10 GbE almost 50% “faster” than 8GFC. Because of this reason, congestion due to speed mismatch may be seen when FCoE traffic is switched from the 10 Gbps Ethernet ports to 8GFC ports. Another common condition of congestion happens when traffic from 16GFC ports (bit rate of 14.025 Gbps) is switched to FCoE ports operating at 10 Gbps. As this chapter explains, the higher the speed or capacity mismatch the more likely congestion will occur and spread.

These differences in bit rates of Ethernet and Fibre Channel can’t be completely eliminated, but they must be minimized by keeping their speeds as close as possible, for example, 32GFC and 25 GbE. Besides the speed of the ports, the minimum bandwidth guarantee on the shared FCoE links should also be accounted for because the FC/FCoE traffic may not be allowed to consume the entire capacity of the link.

This should be a design-level decision on the switches with FC and FCoE ports, such as Cisco UCS Fabric Interconnect (Refer to Chapter 9). After an initial design, monitor the traffic patterns continuously, and add the additional capacity as needed.

Multiple no-drop Classes on the Same Link

When multiple no-drop classes are enabled in lossless Ethernet networks, follow the congestion troubleshooting methodology in one class (CoS) at a time.

The converged topology in Figure 7-15 has only one no-drop class for FCoE traffic. In the same topology, another no-drop class can be enabled for RoCEv2 traffic.

Refer to Figure 7-16. A storage array connects to a leaf switch (L-1) and it is configured for RoCEv2. Host-1 and Host-2 connect to leaf L-4 via FEX-4. These hosts access the RoCEv2 storage in a no-drop class assigned to CoS 5 traffic. The spine-leaf topology now offers PFC for CoS 5 traffic. The RoCEv2 traffic is classified using the DSCP field in the IP header leading to Layer 3 PFC explained earlier. However, for the sake of simplicity, this section assumes that RoCEv2 is CoS 5 traffic as per the CoS-to-DSCP mapping in Table 7-1.

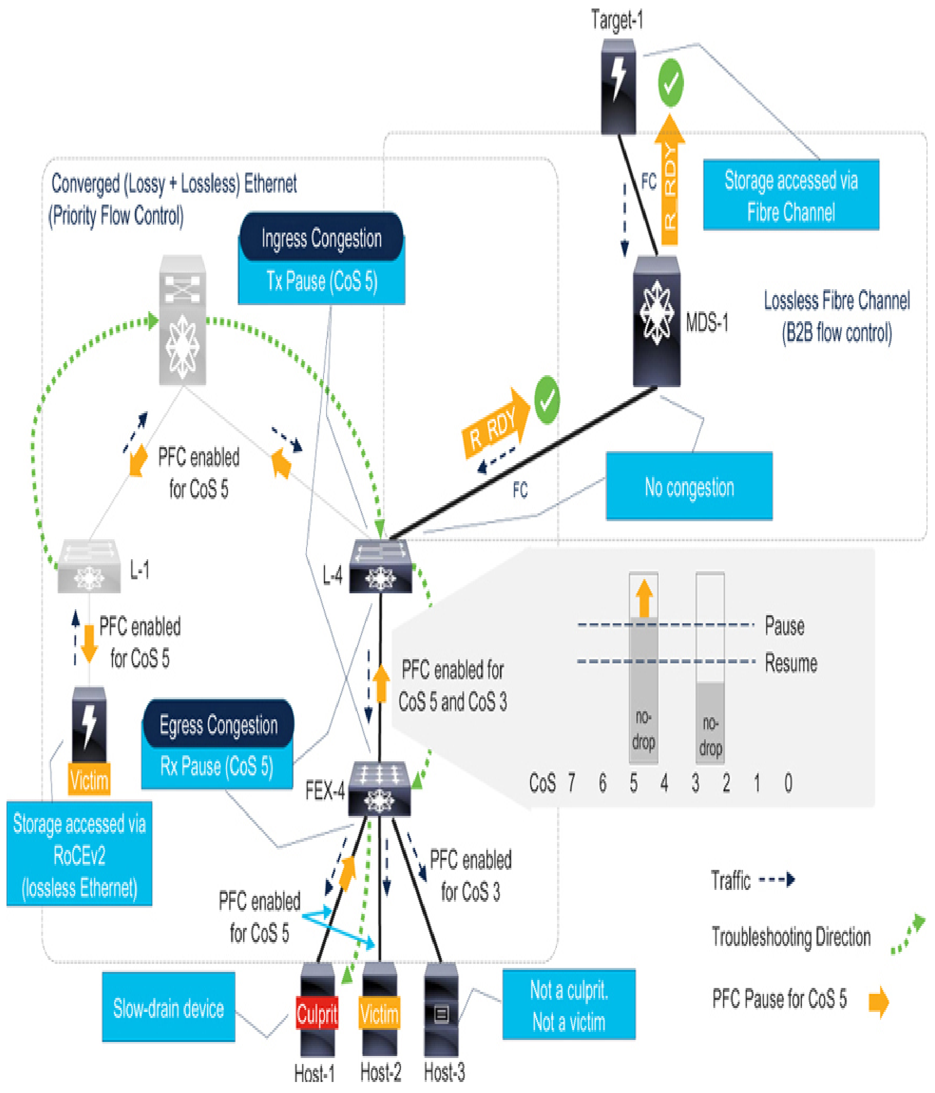


Figure 7-16 *Congestion troubleshooting with multiple no-drop classes*

An FCoE host (Host-3) connects to FEX-4 and it assigns FCoE traffic to CoS 3. FEX-4 and L-4 assign CoS 3 traffic to a dedicated no-drop class. On L-4, CoS 3 traffic is sent on the Fibre Channel uplinks.

Overall, in this topology, the link between L-4 and FEX-4 has PFC configured for two no-drop classes that are assigned to CoS 5 and CoS 3. As a result, the switchports create two separate no-drop queues with their own Pause Threshold and Resume Threshold.

When Host-1 (RoCEv2 host) becomes a slow-drain device, it sends PFC Pause frames for CoS 5. When FEX-4's Pause Thresholds exceed for CoS 5 queue, it sends PFC Pause frames to L-4 only for CoS 5. As a result, Host-2 (another

RoCEv2 host using CoS 5) is victimized because the shared link between L-4 and FEX-4 is congested.

L-4 does not receive PFC Pause frames for CoS 3 and hence, it does not slow down CoS 3 traffic. As a result, the FCoE host (Host-3) is not victimized.

On L-4, congestion spreads only to the CoS 5 uplink (spine switches), not to the Fibre Channel uplink.

Overall, when multiple no-drop classes are enabled, pay attention to congestion symptoms for each class separately. Analyzing the per-port Pause counters on L-4 may lead in the wrong direction. Troubleshooting only in CoS 5 allows for finding the source of congestion (Host-1).

Bandwidth Allocation Between Lossless and Lossy Traffic

Recall that Enhanced Transmission Selection (ETS, in addition to PFC) is a significant function for making converged (lossy + lossless) Ethernet successful because it provides a bandwidth guarantee to no-drop class while the link is shared with traffic in other classes.

Note the following points about bandwidth allocation using ETS:

1. A minimum number of buffers are reserved for each no-drop class.
2. The no-drop class is assigned a minimum bandwidth, such as 50% of the link capacity. But this is not the maximum bandwidth that the no-drop class can achieve. A traffic class can consume up to 100% of the link capacity when other classes do not have traffic.
3. However, when other classes have traffic, the no-drop class is limited to its bandwidth allocation (assuming 50%). This may cause congestion due to over-utilization.
4. A port can transmit in a no-drop class at 50% capacity and in the other classes at 50%.

5. When a port is already transmitting at full capacity at the bandwidth allocation of each traffic class, such as 50% traffic in no-drop class and 50% traffic in the other classes,
   a. If more traffic is received in the lossy classes by other ports on the switch to be sent of out this port, this excess traffic is dropped just like a lossy network.
   b. If more traffic is received in the no-drop class by other ports on the switch to be sent out of this port, those ports that receive the excess traffic send a Pause frame at the Pause Threshold. Each of these ingress ports maintains no-drop queue(s) with its own Pause Threshold, Resume Threshold, and headroom. This condition is like congestion due to over-utilization and results in congestion spreading.

Effect of Lossy Traffic on no-drop Class

It is a common misconception that traffic in lossy classes does not affect no-drop classes. It depends on how a problem is approached.

Traffic in no-drop class can consume 100% of capacity. However, it gets limited to the guaranteed bandwidth when other classes have traffic. In other words, a link that was not causing congestion until now may start causing congestion (due to over-utilization) after traffic in other (lossy) classes increase. For example, consider a 10 GbE link with 5 Gbps allocated for the no-drop class and 5 Gbps for other classes. Lossless traffic in no-drop class can consume the full capacity of the link if other traffic doesn’t exist. However, if other traffic needs 2 Gbps, the no-drop traffic will be limited to 8 Gbps. This switch now must equalize the ingress rate on other ports to 8 Gbps, which was 10 Gbps earlier. For achieving this rate equalization, this switch invokes PFC leading to congestion spreading.

This situation is not as easy to understand when monitoring only the I/O throughput. Earlier, 10 Gbps of I/O throughout did not cause congestion. Later, just 8 Gbps causes congestion.

Because of these reasons, determining over-utilization in converged and even in shared storage networks is much more difficult than in dedicated storage networks. No longer can the egress percentage utilization be looked at to determine over-utilization. Monitoring per-class traffic and per-class congestion separately as well as combined at a link level can help in better understanding and detecting such problems.

Case Study 1 — An Online Gaming Company

An online gaming company uses a converged Ethernet network for lossless storage I/O traffic and lossy TCP/IP traffic. Their servers connect to the network at 10 GbE, with 50% bandwidth allocated to the no-drop class. They rarely see more than 90% link utilization.

They reported the following observations:

- Many applications reported performance degradation.
- The problem happened only during business hours.
- They found a server with a high CPU during the same time frame, but this server's link utilization never exceeded 70%. The owner of the application that runs on this server suspected storage access issues.

High CPU was not strong evidence, but the application owners suspected storage issues. They did not see any I/O errors on the host. To verify network congestion, they checked the Pause frame count on the links to this server. They found that the server sends many Pause frames during the problem duration. Also, the switch that connects to this server was further sending Pause frames on its uplinks. This was congestion spreading, which might have victimized many other servers.

Next, they wanted to find the cause of the high Tx Pause from this server while the link utilization stayed below 70%. They started monitoring traffic utilization per class and found the following during the problem duration:

1. Server ingress link utilization increased from 5 Gbps to 7 Gbps.
2. Per-class traffic monitoring revealed that

a. Traffic in the no-drop class was reduced from 2 Gbps to 1.5 Gbps.

b. Traffic in other (lossy) classes increased from 3 Gbps to 5.5 Gbps.

Based on these observations, it was suspected that an increase in other (lossy) traffic might have caused high CPU utilization, which did not leave enough resources for processing the storage I/O. As a result, the server tried to slow down the ingress I/O traffic, which is evident from the number of Pause frames and reduced I/O throughout.

They validated this theory by moving the application to a powerful server with a higher number of CPUs. After this change, the application did not run into performance issues anymore. Traffic in lossy classes stayed between 3 Gbps and 5.5 Gbps. Traffic in the no-drop class averaged at 2 Gbps. The Pause frames count stayed minimal without any spikes and dips. The server did not report high CPU utilization.

Besides explaining the importance of monitoring per-port and per-class traffic utilization, this case study also demonstrates how full-stack observability can help in detecting congestion issues faster and more accurately. But full stack observability is not the agenda for this case study. The real lesson is that when traffic increases in a lossy class, it may cause congestion in the no-drop class as well. In this case, the real issue was outside the network, which is the CPU capacity of the server, but the effect is seen on the network which victimizes many other servers.

It should not be generalized that high CPU causes I/O performance issues. Only in this case study, high CPU caused a problem. I/O performance may get affected because of many other reasons. The key take away from this case study should be to understand the effect of traffic in lossy class on no-drop class.

Case Study 2 — Converged Versus Dedicated Storage Network

This case study is similar to Case Study 1. The difference is that the server is not running into a high CPU utilization anymore. Also, the average traffic in the no-drop class is 6 Gbps (60%), and the average traffic in lossy classes is 2 Gbps (20%). When application performance degradation is reported, the duration also coincides with 100% utilization on the converged link. After doing an investigation of the per-class traffic utilization, it was found that the traffic in the lossy class spiked from 2 Gbps to 5 Gbps. At the same time, the traffic in the no-drop class dipped from 6 Gbps to 5 Gbps. This is because on this 10 GbE link, the no-drop class was allocated a bandwidth guarantee of 5 Gbps (50%). But the collective ingress rate of the lossless traffic on other ports on this switch is still 6 Gbps. To equalize this traffic to 5 Gbps, this switch invokes PFC, which results in congestion spreading. This was confirmed with the spike in Tx Pause frames on the other ports on the switch that were receiving traffic to the sent out on the edge port.

In this case study, the clear problem is the lack of capacity on the converged link and traffic contention on that link between no-drop and lossy class. This issue was resolved by adding another 10 GbE link.

In this case study, it was chosen to use both links for both types of traffic (lossy and lossless), which is the approach of the shared storage network. A valid alternative approach would have been to dedicate one link to lossy (normal class) and the other link to lossless (no-drop class) traffic, which is the approach of the dedicated storage network.

The correct answer lies in the capacity of the link and the throughput that is expected on those links. A dedicated storage network is a different architecture and will require you to operate it differently. The pros are the independence of the fabrics, scalability, fault isolation, and easier troubleshooting. On the contrary, a dedicated storage network is more expensive to deploy and needs more resources to manage and operate.

Preventing Congestion in Lossless Ethernet Networks

The high-level approaches to eliminating or reducing congestion in lossless Ethernet networks are the same as the Fibre Channel fabrics. Over the decades, different transport types have implemented similar approaches with minor variations. Chapter 6 already explains the details of Preventing Congestion in Fibre Channel Fabrics. Because the hop-by-hop flow control leads to congestion spreading in both networks, the same concepts apply to lossless Ethernet networks as well, although there are implementation differences.

Eliminating or Reducing Congestion — An Overview

Recall that a culprit is any device that causes congestion in a storage network. A victim is any device that is adversely affected by network congestion.

To allow a storage network to automatically eliminate or reduce congestion as it occurs, the following are the high-level approaches:

- **Disconnect a culprit device**: Disconnecting a slow-drain device eliminates the source of congestion and therefore results in recovering the network from congestion. This would typically require monitoring the ingress Pause frames from the end devices and disabling (or shutdown) an edge switchport if it is unable to transmit for an extended duration, such as a few hundred milliseconds. Many production networks use this approach today, but this is a big-hammer approach because a disconnected device can't serve its purpose and it cannot be monitored to know if the problem continues or not. Refer to Chapter 6, the section on *Congestion Recovery by Disconnecting the Culprit Device* for additional considerations and how to disconnect, if needed.
- **Early dropping of frames**: Dropping frames frees up the buffers and makes them available for re-use leading to

recovering from congestion. The later sections explain this approach using the Pause Frame Timeout and PFC Watchdog features on Cisco Nexus 9000 switches.

- **Traffic Segregation**: Segregating the traffic going to a culprit device from the other traffic can avoid the effect of congestion on the other devices. Traffic can be segregated by creating multiple VLANs and dedicating the ISLs to the VLANs. Another approach is to create multiple no-drop classes and assign traffic to different classes. At the time of this writing, the use of these approaches in lossless Ethernet networks is unknown. Refer to Chapter 6, the section on *Traffic Segregation* for more details and if you like to experiment with similar approaches in lossless Ethernet networks.
- **Notifying the end devices about congestion**: Notifying the end devices about congestion allows them to take preventive actions such as lowering their traffic rate. This chapter explains the use of Explicit Congestion Notification (ECN) over a routed lossless Ethernet network for RoCEv2 Congestion Management (RCM). A similar approach within a layer-2 domain was standardized by IEEE 802.1Qau, and later incorporated into IEEE 802.1Q. It could have been useful in RoCEv1 (non-routed) and FCoE networks, but at the time of this writing, its implementations are less common and highly unlikely to evolve in the future. Because of this reason, this chapter does not explain IEEE 802.1Qau congestion notification within a layer-2 domain.
- **Limiting the traffic going to a congested device**: Limiting the traffic going to a slow-drain device or an over-utilized link can eliminate congestion.
    - Traffic rate limiters can be configured on the end devices. The use of this approach in lossless Ethernet networks is unknown at the time of this writing. When such implementations are available, the details in Chapter 6, the section on *Congestion Prevention using Rate Limiters on Storage Arrays* will apply to lossless Ethernet networks as well.

    - Alternatively, the network switches can detect congestion and dynamically adapt the traffic rate to the culprit devices. Cisco MDS switches use this approach for Dynamic Ingress Rate Limiting (DIRL) to prevent congestion in Fibre Channel fabrics. Although, in theory, DIRL could work in lossless Ethernet networks as well, its implementations are not available at the time of this writing. Refer to Chapter 6, the section on *Congestion Prevention using Dynamic Ingress Rate Limiting* for more details.

- **Re-designing the network**: Re-designing a network can eliminate or reduce the severity or spread of congestion. For example, converting large networks having thousands of devices to smaller islands limits the effect of culprit devices only within the island. Refer to Chapter 6, the section on *Network Design Considerations* for more details.

- **Upgrade**: Many times, upgrading a culprit device is the ultimate solution to congestion. For example, when a link is highly utilized (which may be causing congestion due to over-utilization), the ultimate solution is to increase the link speed or add additional links. Refer to Chapter 6, the section on *Link Capacity* for more details.

Congestion Recovery by Dropping Frames

During congestion, the time spent by a frame within a switch is much longer than the typical port-to-port switching latency. When a culprit device is not ready to receive frames for an extended duration, instead of allowing the frames to remain within a switch forever, the frames can be dropped after a timeout duration. Dropping these frames make the buffers available for re-use, and thus, helps in recovering from congestion. There are at least two ways for these frames to be dropped.

Dropping Frames Based on their Age in the Switch

Using this approach for congestion recovery, a switch drops a frame if it does not go out of the egress port within a timeout duration.

This approach is available on Cisco MDS switches for FCoE ports. It can be configured using the MDS NX-OS command **system timeout fcoe pause-drop**. This is similar to the congestion-drop timeout for Fibre Channel ports. Refer to Chapter 6, the section on *Dropping Frames Based on their Age in the Switch* for its pros and cons.

Dropping the Frames based on Slow-Drain on an Edge Port

Using this approach for congestion recovery, a switch drops frames that are destined for a slow-drain device. Typically, this approach requires monitoring the continuous duration when an edge switchport is unable to transmit because of receiving Pause frames from the connected device. If this duration exceeds a timeout, the frames going to that device are dropped. When frames are dropped, buffer utilization eventually falls below the Resume Threshold on the ports that receive traffic to be sent out of the port that is connected to the slow-drain device. As a result, this switch stops sending Pause frames to the upstream directly connected device allowing it to start the transmission. Traffic for the culprit device may still be dropped but victims may benefit.

In theory, this approach is similar to the no-credit-drop timeout feature for Fibre Channel ports on Cisco MDS switches. Refer to Chapter 6, the section on *Dropping the Frames based on Slow-Drain on an Edge Port* for a detailed explanation of how the victim devices benefit from it. This section does not repeat all those details.

The implementation of dropping frames (when to start dropping, when to stop dropping, and granularity) on lossless Ethernet switches are different. This section explains two types of implementations that are based on this approach.

1. **Pause Timeout**: Aimed for FCoE traffic.

2. **PFC Watchdog:** Primarily aimed at RoCEv2 traffic, although any lossless Ethernet traffic (including FCoE) may benefit from it.

Pause Timeout

Using the Pause timeout feature, a port drops all the egress traffic if it is unable to transmit for a timeout duration because of receiving Pause frames. Also, as long as this port remains in an Rx Pause state, all newly arriving frames on other ports on this switch that are destined to go out of this port are immediately dropped. As previously mentioned, dropping the culprit traffic on the edge port allows the upstream device to start transmission, which results in recovering the victim devices from the effect of congestion.

Note the following points:

1. At the time of this writing, Pause timeout is available only for FCoE ports on Cisco switches, although implementation differs on different platforms, as explained shortly.
2. Despite using PFC, the Pause timeout feature drops all the traffic on a port, not just in a no-drop class that is used by FCoE. This behavior is acceptable if the port is dedicated for traffic in no-drop class, such as FCoE ports on Cisco MDS switches. But this behavior may not be acceptable when a port carries traffic in no-drop and other classes.

The exact implementation of this feature may differ based on the type of switch. For example:

1. **Cisco MDS Switches**: For FCoE ports on Cisco MDS switches, this feature is called Pause-drop Timeout, and it is enabled by default at 500 ms. If required, change it using MDS NX-OS command **system timeout fcoe pause-drop**.
2. **Cisco Nexus Switches**: For FCoE ports on Cisco Nexus switches, this feature is called Pause Timeout, and it is disabled by default. When enabled using the NX-OS command **system default interface pause mode edge**,

the timeout value is 500ms. The timeout can be changed using the command **system default interface pause timeout <ms> mode edge** in 100ms increments between 100ms to 500ms.

3. **Cisco UCS**: In Cisco UCS servers, this feature can be enabled by Slow-drain Timers. The default timeout is 500 ms, which can be customized in 100 ms increments between 100 ms to 1000 ms. In the earlier releases of Cisco UCS Manager, this feature was enabled by default with a 500ms timeout value. Later releases, by default, disabled this feature and enabled PFC watchdog. Refer to the Cisco UCS Manager Release Notes for more details.

PFC Watchdog

PFC watchdog works similarly to Pause timeout, but it only drops the traffic in the queue that is unable to transmit continuously for a timeout duration because of receiving PFC Pause frames.

Depending upon the implementation, the PFC watchdog can trigger the following actions:

1. **Port flap or shutdown**: When the PFC watchdog is detected on a queue, the port is flapped or shut down, which affects all the traffic classes on this port. This is the same action as the pause timeout feature. Some people call it a disruptive watchdog.
2. **Alert only**: When the PFC watchdog is detected, it generates an alert, such as Syslog, but does not take any further action of dropping the traffic. Some people call it a log-only watchdog.
3. **Shutdown the queue**: This is the most common implementation at the time of this writing. When the PFC watchdog is detected in a no-drop queue, the following actions are taken:

   a. All frames in that queue are dropped (discarded).

   b. As long as this queue remains in an Rx Pause state, all newly arriving frames on other ports on this switch that are destined to go out of this queue are immediately dropped.

c. All ingress traffic on this port that belongs to the same traffic class (as the egress no-drop class) is also dropped. All ingress Pause frames are dropped as well. This is a unique differentiator of the PFC watchdog. Recall that Pause timeout (available on FCoE ports on Cisco MDS, Nexus, and UCS) and no-credit-drop timeout (available on FC ports on Cisco MDS switches) do not affect ingress traffic on a port. The argument for not dropping the ingress traffic by these features is that congestion is directional, hence the reverse traffic should not be affected. In contrast, the argument for dropping the ingress traffic by PFC watchdog is that applications running on a slow-drain device would not benefit just by unidirectional traffic (egress traffic for slow-drain device which is the same as the ingress traffic for switchport where PFC watchdog is enabled). In fact, egress traffic from a slow drain device may increase the duration and severity of congestion, as explained earlier in the section on *Correlating I/O Operations, Traffic Patterns, and Network Congestion*.

The exact implementation of PFC watchdog feature may differ based on the type of switch. For example:

1. **Cisco MDS Switches**: PFC watchdog is not available on Cisco MDS switches at the time of this writing. Using the Pause timeout feature is a valid alternative because the MDS switches use only a single no-drop class for only FCoE traffic.
2. **Cisco Nexus Switches**: PFC watchdog is available on Cisco Nexus 9000 and Nexus 3000 switches. The support and action (port flap, alert only, queue shutdown) depend on the model type and NX-OS version. Refer to the release notes for the latest information.
3. **Cisco UCS**: Cisco UCS servers support PFC watchdog in UCS Manager release 4.2 onwards. Pause timeout and

PFC watchdog are mutually exclusive. In later releases, the PFC watchdog was enabled by default.

The following are the configuration details of PFC watchdog on Cisco Nexus 9000 switches as of NX-OS 10.3(1).

1. Use the NX-OS command **priority-flow-control watch-dog-interval <on | off >** for enabling PFC watchdog.
2. This command enables polling the no-drop queue by a software process at every interval. By default, this polling interval is 100 ms. To change the polling interval, use the NX-OS command **priority-flow-control watch-dog interval <100 - 1000>** in 100 ms increments between 100 ms and 1000 ms.
3. How soon a queue is shut down depends on the shutdown-multiplier. By default, its value is 1, and it can be changed using NX-OS command **priority-flow-control watch-dog shutdown-multiplier <1 - 10>**. The no-drop queue is shut down at (watchdog interval x shutdown multiplier). Therefore, by default, the queue is shut down when it is a continuous state of Rx pause for 100ms. When the shutdown-multiplier is 2, the queue would shut down at 200 ms, and so on.
4. After a queue is in a shutdown state, it can be manually restored using the NX-OS command **priority-flow-control recover interface <shut-interface> qos-group <qos-group>**. The qos-group value identifies the traffic class that is associated with the no-drop queue that was shutdown by PFC watchdog.
5. Alternatively, restoring the queue can be automated using two schemes.
   a. The fixed-restore multiplier restores the queue after (watchdog interval x fixed-restore multiplier) regardless of the Rx Pause. In other words, the queue is restored even if the device continues to be a slow-drain device. The fixed-restore multiplier is disabled by default (value of 0). It can be enabled using the NX-OS command **priority-flow-control fixed-restore multiplier <0 - 100>**.

b. The auto-restore multiplier restores the queue only after the Rx Pause are not received for this queue for (watchdog interval x auto-restore multiplier) continuously. In other words, the queue is restored only “after” the device stops causing slow-drain. The auto-restore is enabled by default at 10 and it can be changed using NX-OS command **priority-flow-control auto-restore multiplier <0 – 100>**.

Consider a device, which starts causing slow-drain at time = T1. Its connected switchport has PFC watchdog configured with a 100ms interval, shutdown-multiplier of 1, and auto-restore multiplier of 10. The device sends Pause frames continuously, which stops the transmission on the connected switchport. When the switchport is unable to transmit continuously for 100ms, it shuts down the queue, which results in dropping all the packets in that queue and other actions described earlier. This happens at T1 + 100ms. At time = T2, the device stops sending PFC Pause frame. If the switchport does not receive a PFC Pause frame for this no-drop class in the last 1 second, the queue is auto-restored.

As Example 7-12 shows, to verify the PFC watchdog configuration on Cisco Nexus 9000 switches and the current state of the queues, use the NX-OS command **show queuing pfc-queue**. As shown, PFC watchdog is enabled on Ethernet 1/3 and Ethernet 1/5. As the VL bmap shows, PFC is enabled for CoS 1 traffic on both interfaces. Only on Ethernet 1/5, the no-drop queue assigned to CoS 1 traffic is in a shutdown state.

Example 7-12 *Verifying PFC watchdog on Cisco Nexus 9000 switches*

```
switch# show queuing pfc-queue

+-----------------------------------------------------------
-+
Global watch-dog interval [Enabled]
+-----------------------------------------------------------
-+
+-----------------------------------------------------------
-+
Global PFC watchdog configuration details
PFC watch-dog poll interval                : 100 ms
PFC watch-dog shutdown multiplier          : 1
PFC watch-dog auto-restore multiplier      : 10
PFC watch-dog fixed-restore multiplier     : 0
PFC watchdog internal-interface multiplier : 2
+----------------------------------------------------
-+
+----------------------------------------------------
----------+
| Port              PFC Watchdog  (VL bmap)  State
(Shutdown) |
+----------------------------------------------------
----------+
  Ethernet1/1       Disabled      ( 0x0 )    - - - -
- - - -
  Ethernet1/2       Disabled      ( 0x0 )    - - - -
- - - -
  Ethernet1/3       Enabled       ( 0x2 )    - - - -
- - N -
  Ethernet1/4       Disabled      ( 0x0 )    - - - -
- - - -
  Ethernet1/5       Enabled       ( 0x2 )    - - - -
- - Y -
  Ethernet1/6       Disabled      ( 0x0 )    - - - -
- - - -
```

Use NX-OS command **show queuing pfc-queue interface** to find the packet drop statistics due to PFC watchdog. Example 7-13 shows the output of this command. Interpret the Stats column in the following ways:

1. **Shutdown**: Number of times the queue assigned to QoS Group 1 (CoS 1) traffic was shut down.
2. **Restored**: Number of times the queue assigned to QoS Group 1 (CoS 1) traffic was restored.
3. **Total pkts drained**: Number of packets that were already in the queue and dropped when the queue was shut down last time.

4. **Total pkts dropped**: Number of packets that arrived on other ports on the switch trying to exit via this queue on Eth1/5 after the queue was shut down last time and were dropped.
5. **Total pkts drained + dropped**: Total pkts drained (3) + Total pkts dropped (4).
6. **Aggregate pkts dropped**: This is same as (4), except that it shows aggregate count from multiple previous shut/no-shutdown instances.
7. **Total Ingress pkts dropped**: Packets dropped that arrived ingress on Eth 1/5 belonging to the same QoS Group 1 (CoS 1)
8. **Aggregate Ingress pkts dropped**: This is same as (7), except that it shows aggregate count from multiple previous shut/no-shutdown instances.

Example 7-13 *PFC watchdog counters on Cisco Nexus 9000 switches*

```
switch# show queuing pfc-queue interface e1/5 detail
+----------------------------------------------------
-+
Ethernet1/5 Interface PFC watchdog: [Enabled]
+----------------------------------------------------
-+
+----------------------------------------------------
-+
| QOS GROUP 1 [Active] PFC [YES] PFC-COS [1]
+----------------------------------------------------
-+
|                                |  Stats
|
+----------------------------------------------------
-+
|                        Shutdown|
4|
|                        Restored|
4|
|              Total pkts drained|
752|
|             Total pkts dropped|
2197357321|
|   Total pkts drained + dropped|
2197358073|
|         Aggregate pkts dropped|
53487546587|
|     Total Ingress pkts dropped|
66649|
| Aggregate Ingress pkts dropped|
34987434|
+----------------------------------------------------
-+
```

To clear these counters, use the NX-OS command **clear queuing pfc-queue**.

Cisco Nexus 9000 switches notify the PFC watchdog queue Shutdown/ Restore actions by Syslog messages. Example 7-14 shows the queue shutdown Syslog message.

Example 7-14 *Syslog message when a queue is shut down due to PFC watchdog*

```
2021 Aug 18 10:18:51 N9K %$ VDC-1 %$ %TAHUSD-SLOT1-
2-
TAHUSD_SYSLOG_PFCWD_QUEUE_SHUTDOWN: Queue 1 of
Ethernet1/5 is shutdown due to PFC
watchdog timer expiring
```

Example 7-15 shows the queue restore Syslog message. Notice the number of ingress and egress dropped packets.

Example 7-15 *Syslog message when a queue is restored after being shut down by PFC watchdog*

```
2021 Aug 18 10:19:58 N9K %$ VDC-1 %$ %TAHUSD-SLOT1-
2-
TAHUSD_SYSLOG_PFCWD_QUEUE_RESTORED: Queue 1 of
Ethernet1/5 is restored due to PFC
watchdog timer expiring; 2197358073 egress
packets/66649 ingress packets dropped
during the event
```

The Granularity of Pause Timeout and PFC Watchdog

The configuration granularity of Pause timeout is 100ms, which is the same as the PFC watchdog interval. Both features rely on the software polling every 100ms. As a result, both can delay the action and recovery by up to 99ms. In other words, the PFC watchdog shutdown action may take up to 199ms instead of being applied exactly at 100ms.

Refer to Chapter 6, the section on *No-credit-drop timeout in Action* and Chapter 3 section *Differences Between TxWait, Slowport-monitor, and Tx-credit-not-available* to understand the results of using software polling and the added benefits of an ASIC-based implementation. On Cisco MDS switches, the Fibre Channel no-credit-drop timeout feature moved to an ASIC-based implementation in 2015. At the time of this writing, ASIC-based implementation for Pause timeout and PFC watchdog are unavailable.

The Benefits and the Limitations

Pause timeout and PFC watchdog drop frames destined for a slow-drain device, which frees up the buffers resulting in recovering the victim devices from the effect of congestion. Dropping the frames deviates from the no-drop behavior of lossless networks, but when a culprit device is unable to receive frames, instead of waiting forever and allowing other devices to be victimized, it is better to drop the frames for the culprit device. Because of these reasons, we recommend enabling these features depending upon the availability in your environment and as per the recommended thresholds by the device vendors.

While using these approaches, be aware of the following limitations:

1. At the time of this writing, most implementations of Pause timeout and PFC watchdog are based on software polling which may delay the actions and reduce the effectiveness of the recovery. Refer to the earlier section

on *The Granularity of Pause Timeout and PFC Watchdog*.

2. Pause timeout and PFC watchdog help to recover from congestion due to slow-drain. These approaches do not help when congestion is caused by the over-utilization of edge links.
3. The Pause timeout and PFC watchdog have a minimum granularity of 100 ms. Hence, these approaches do not help when transmission stops for a shorter period, such as 50 ms.
4. Pause timeout and PFC watchdog timeout act only on a continuous period of stopped transmission. A slow-drain device can cause severe congestion even with non-continuous durations of Pause frames.

These limitations should not deter from using Pause timeout and PFC watchdog. However, be aware of what they can and cannot achieve.

Congestion Notification in Routed Lossless Ethernet Networks

End devices and their applications may not be aware of congestion in the network. A culprit device may continue to send (or solicit) more traffic on the network making the severity of congestion worse or increasing its duration. To solve this problem, the network switches can 'explicitly' notify the end devices as soon as they detect congestion. In response, the culprit devices can take preventive actions.

This approach of preventing congestion by notifying the end devices is available in all kinds of networks with different degrees of implementation and adoption. Chapter 6, the section on *Congestion Prevention by Notifying the End Devices*, provides details of implementation in Fibre Channel fabrics using Fabric Performance Impact Notifications (FPIN) and Congestion Signals. Chapter 8, the section on *Congestion Notification in TCP Storage Networks* explains this approach in TCP/IP storage networks.

As previously mentioned, the implementations and adoption of this approach in FCoE and RoCE environments are uncommon. Hence, this section does not explain them. The focus of this section is on congestion notification in routed lossless Ethernet networks for RoCEv2 Congestion Management (RCM).

Solution Components

The success of the approach to congestion prevention by notifying the end devices depends upon the following factors:

1. **Congestion detection**: A switch (known as a Congestion Point) must be able to detect the symptoms of congestion.
2. **Sending a notification**: After detecting congestion, the switch must be able to inform the end devices about it. For this purpose, Fibre Channel switches send special frames (called FPINs) and Congestion Signals. In contrast, RoCEv2 and TCP/IP networks encode this information in the data packets by marking special bits in the headers.
3. **Receiving a notification**: The end devices must be able to understand the notification from the switches. This depends on the hardware and/or software capability of the end devices.
4. **Congestion prevention action**: After being notified about congestion in the network, the end devices must take preventive actions, such as reducing their rate. This is the most important component.

The following sections explain these components using RoCEv2 Congestion Management (RCM).

RoCEv2 Transport Overview

RoCEv2 packet has an IP header and hence, it can be classified using the DSCP field in the IPv4 or IPv6 header. Refer to the earlier section on *Priority Flow Control* for understanding how traffic can be classified and assigned to a no-drop class in routed layer 3 networks. Chapter 1, Figure 1-10 shows RoCEv2 packet format.

For notifying the end devices about congestion, the Explicit Congestion Notification (ECN) field in the IP header is used.

Note the following points:

1. Typically, the Ethernet VLAN CoS value is mapped to the DSCP field in the IP header. Table 7-1 provides this mapping.
2. In routed networks, the Ethernet header changes at every hop, so the CoS value is not retained unless configured otherwise. But the DSCP field in the IP header remains unchanged between the source and the destination.
3. The networks that do not use VLANs, using the DSCP field in the IP header is the only option for classifying the traffic because CoS is part of the VLAN header.
4. Overlay networks, such as Virtual Extensible LAN (VXLAN) typically copy the DSCP and ECN values to the outer IP headers before encapsulating the original packet and copy these values from the outer header to the decapsulated packet, and hence, no-drop behavior and ECN are retained when classifying the traffic using the IP header. Refer to the section on *Lossless Ethernet with VXLAN* for more details.

RoCEv2 Congestion Management

When a switch detects congestion, it flags it in the ECN field in the IP header. The destination receives the ECN-flagged packet and reflects this information to the source, which reacts by rate limiting the traffic. RFC 3168 explains Explicit Congestion Notification (ECN) using TCP as the layer 4 protocol. RCM relies on the same mechanism, but because RoCEv2 uses UDP (not TCP), it has some differences as explained next.

Refer to Figure 7-17, which is a subset of the spine-leaf network explained in Figure 7-8. The following are the high-level steps:

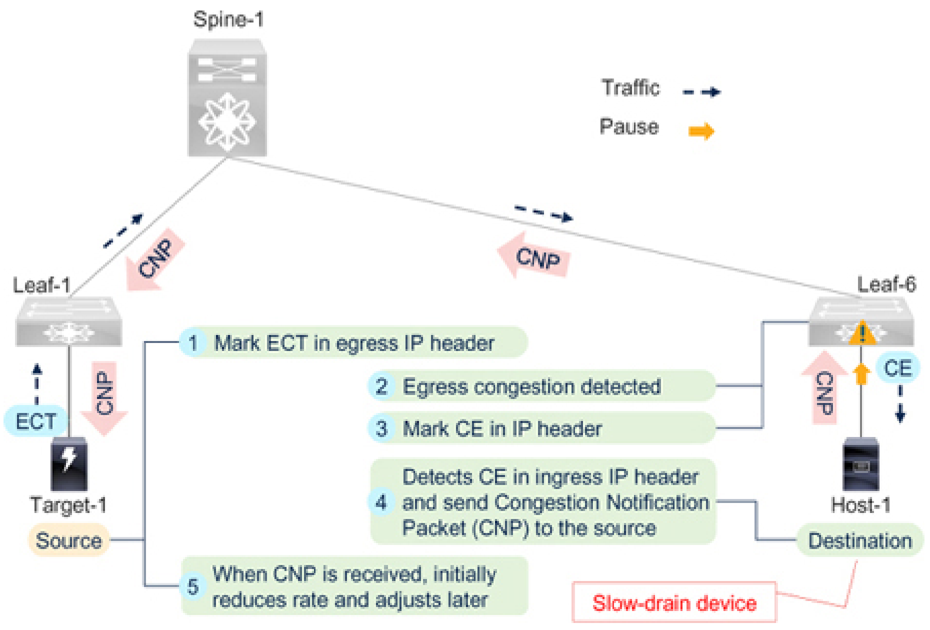

Spine-1
Traffic
Pause
Leaf-1
CNP
CNP
Leaf-6
ECT
CNP
1 Mark ECT in egress IP header
2 Egress congestion detected
3 Mark CE in IP header
4 Detects CE in ingress IP header and send Congestion Notification Packet (CNP) to the source
5 When CNP is received, initially reduces rate and adjusts later
CNP
CE
Target-1
Source
Host-1
Destination
Slow-drain device

Figure 7-17 *RoCEv2 Congestion Management (RCM)*

1. **Willingness to use or not use RCM**: When the end devices (Target-1 and Host-1) support RCM, they set the ECN-capable Transport (ECT) flag in the IPv4 or IPv6 header. The ECT flag is set with b‘01’ or b‘10’ in the ECN field in the IP header. A value of b‘00’ in the ECN field indicates that the endpoints are not ECN-capable, and they do not support RCM.

2. **Congestion detection**: A switch (Leaf-6) in the end-to-end data path detects congestion when the queue utilization exceeds a configured threshold. The queue utilization increases because of slow-drain (Rx Pause on egress switchport) or over-utilization of the switchport.

3. **Sending a notification**: When the switch detects congestion in a queue, it sets the Congestion Experience (CE) flag in the IPv4 or IPv6 header. The CE flag is set with b‘11’ in the ECN field. The congested switchport sets CE flag only for those packets that have the ECT flag (b‘01’ or b‘10’) enabled. Non-ECN-capable packets (ECN value b‘00’) are forwarded unchanged. Unlike Fibre Channel, in RoCEv2 networks, switches do not send any special packet (or frame or signal) to notify the end devices.

4. **Receiving a notification**: The destination (Host-1) detects network congestion when it receives a packet with the CE flag. Host-1 need not have any special mechanism to interpret a new kind of packet because the CE flag is part of the standard IP header. However, it must be able to detect and act on the CE flag. When Host-1 receives a CE-flagged packet, it reflects congestion to the source by sending a Congestion Notification Packet (CNP) to the source (Target-1) of the ingress packet that had the CE flag enabled. Because CNP is a special packet, the destination and the source must be capable of sending and receiving it and other details such as how often to send CNP, when to stop sending it, and its contents. Also, configure network QoS policies to classify and forward CNP with high priority.
5. **Congestion prevention action**: When the sender (Target-1) receives a CNP, it reduces the traffic rate to the destination that sent the CNP. As mentioned, because CNP is a special packet, the sender must be able to listen and act on it. Also, after an initial rate reduction action, the sender must be able to adjust its rate for an optimum balance between under-utilization and over-utilization.

RoCEv2 Congestion Management Considerations

Figure 7-17 provides an oversimplified explanation of RCM. In production environments, however, hundreds or thousands of devices may connect to the same fabric with many-to-one traffic patterns.

This section explains some important considerations for RoCEv2 networks. These may not be visible in greenfield environments, but as networks grow or mature, be aware of the limitations and take proactive actions.

Refer to Figure 7-8 and think about how RCM explained in Figure 7-17 would operate. Then, consider the following points:

1. **Mixed environments**: As per RoCEv2 standard, RCM is optional. Mixed environments with RCM-supported and RCM-unsupported end devices need special attention.

When many devices send traffic to the same device, but if a few senders do not support RCM, their traffic is not CE-flagged, whereas traffic from other RCM-supported devices is CE-flagged. Therefore, the slow-drain device (destination) sends CNP to the RCM-supported devices only. Consequently, the RCM-supported devices reduce their traffic rates, whereas the RCM-unsupported devices do not. This scenario may result in traffic starvation for the RCM-supported devices because they may keep reducing their rates and the RCM-unsupported device may consume all the link capacity. In IP networks with mixed TCP and UDP traffic, this condition is similar to TCP starvation by UDP traffic. Typically, newer devices are more likely to support RCM. If the existing devices do not support RCM, the newer devices may be penalized more.

2. **Rate reduction algorithm**: The RoCEv2 standard mentions that a sender shall reduce its traffic rate after receiving a CNP but does not explain an algorithm for a rate reduction or recovery. The lack of a standard approach leaves vendors to develop different implementations, which react differently in the same environment. A user is left to suffer the results of non-uniform implementations or forced into a vendor lock-in for bringing uniformity.
3. **Synchronization**: A traffic source that receives CNPs is unaware of other sources that send traffic to the same destination. The collective action of multiple such sources may lead to overreaction or under-reaction. For example, in Figure 7-8, if each target reduces its rate by 5%, the collective rate-reduction action by the five targets results in a 25% rate reduction (overreaction). Later, when the targets adjust/increase their rates, five such devices working independently may again lead to the original problem that would have resulted in the initial congestion event. In TCP/IP networks, this condition is similar to TCP global synchronization.
4. **Delayed action**: RCM (and other approaches for preventing congestion by notifying the end devices) may

not be effective in some storage environments with burst traffic patterns because of the delay between congestion detection and preventive action. In Figure 7-17, Leaf-6 detects congestion, Host-1 receives the CE-flagged packet, sends CNP to the source, and finally, the source reduces its rate. These steps involve a delay by the time Leaf-6 observes a change. By this time the original congestion symptoms may cease by themselves. To quote, RFC 3168, which is the same mechanism that RCM uses—“CE packets indicate persistent rather than transient congestion, and hence reactions to the receipt of CE packets should be those appropriate for persistent congestion”.

5. **Configuration complexity**: The success of RCM depends on how well it is configured in a network. Marking the CE flag on the switches typically uses a detection scheme such as Weighted Random Early Detection (WRED). Refer to Chapter 8, the section on *Active Queue Management* for more details. The thresholds for these mechanisms should be configured so that the rate-reduction action is observed on the switch before its queues are full. These values also depend on the type of switch because different switches have different architectures and buffer capacities. On the end devices, the frequency of the CNP is an important consideration. Too many CNP may increase the load on the end devices, whereas too few CNP may delay their actions. Also, the rate-reduction algorithm may have its variables.

These points should not deter RCM from being enabled. Follow the recommendations of the vendors and refine the configuration for your environments. The important point to remember is that just because something works well in newer (greenfield environments), that does not eliminate the need for monitoring congestion symptoms.

Consider an analogy of new cars with nitrogen-filled tires. Initially, they may not need air refills for many months or even a year. These new cars work so well that there have been some drivers who almost forgot that the tires need an air pressure check, only to be left with a flat tire on their not-so-new car. They could have prevented this problem by a proactive monthly check. Do not be that driver. Problems become visible as the environment grows or matures. Being already aware of the potential problems can help you in faster resolution when and if the problem surfaces.

PFC and ECN

PFC is a hop-by-hop flow control mechanism. In contrast, ECN notifies the destination, which in turn informs the sender to reduce its traffic rate for congestion prevention. As previously explained, there is a delay between congestion detection (when the CE flag is marked) and when the reduced rate is observed by the congested switchport. This is approximately twice the round-trip time and the processing delay of the end devices. During this time, the queue on the congested switchport may fill up. Instead of dropping the packets, the hop-by-hop PFC may get activated, resulting in congestion spreading in the no-drop class.

Using ECN and PFC together bring out the best of both. ECN is flow-based but its effect may be delayed. In contrast, PFC is priority- (class, or type) based but its prompt action avoids packet drops. Using them together makes the action prompt and flow based.

Configuring PFC and ECN Parameters

The rate-reduction action after notifying the end devices (by ECN and CNP) should eliminate the cause of congestion. When there is no congestion, there is no need to invoke PFC. Therefore, a properly working ECN should soon reduce the PFC Pauses. But configuring ECN thresholds on the switches require special consideration. The exact values depend on the switch type, so refer to the vendor documentation. This section explains only a conceptual overview.

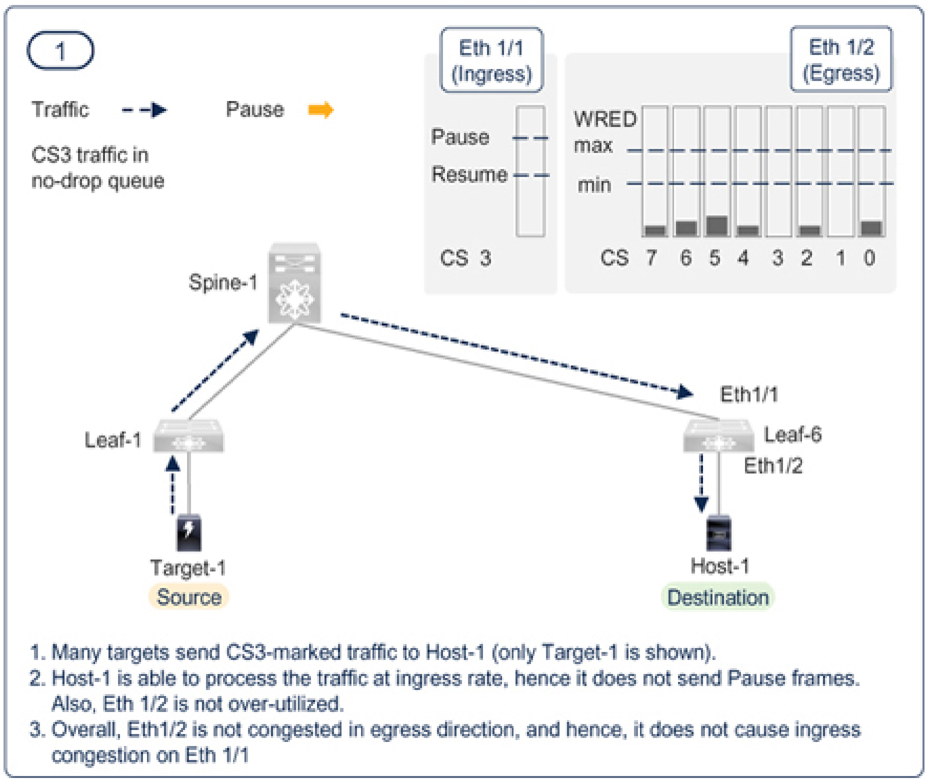
Note the following points.

1. Ingress queues/buffers start filling up only after egress queues are highly utilized. This is explained earlier in the

section on *Ingress and Egress Queues* and *Microburst Detection*.

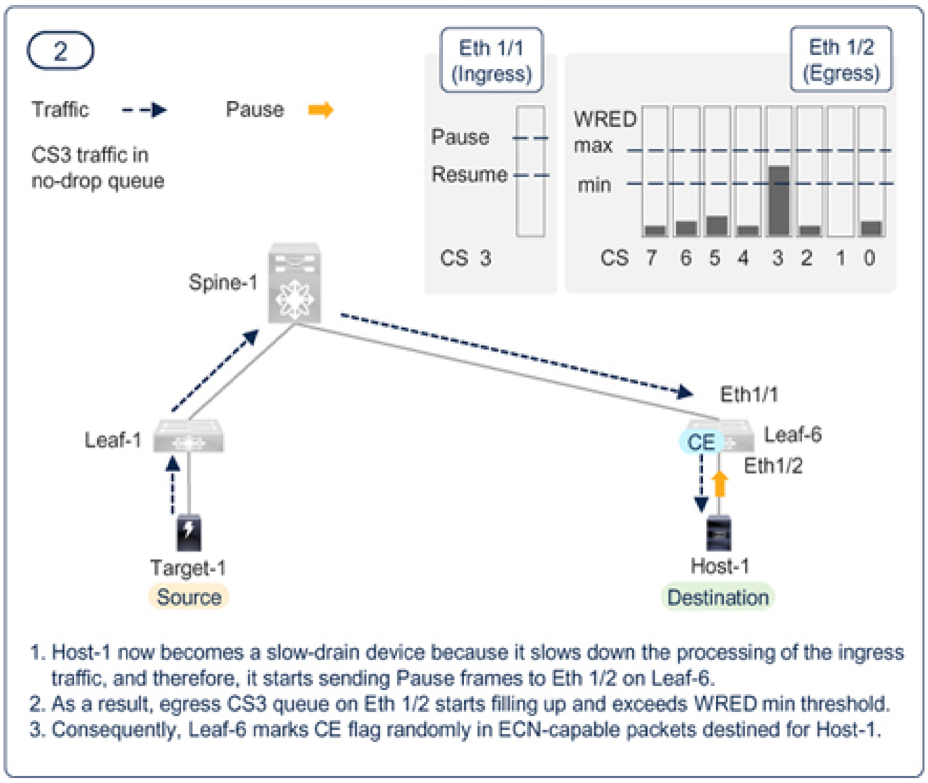
2. The thresholds for ECN marking (such as by WRED) are applied on the egress queues, whereas the PFC Pause Threshold and Resume Threshold are applied on the ingress queues/buffers.
3. Pause Threshold and Resume Threshold should be configured as per the details explained earlier in the section on *Pause Threshold and Resume Threshold*. Typically, for intra-datacenter links that are of short distance, changing the default Pause Threshold and Resume Threshold is not needed.
4. The ECN thresholds should be low enough to mark the CE flag much earlier to give enough time for the rate reduction action to be observed on the congested port.
5. The thresholds should be large enough to accommodate at least a few packets. For example, while enabling jumbo frames, a min size of 9000 bytes can't even keep one full-size jumbo packet in the queue.

Figure 7-18 to Figure 7-23 demonstrates the functioning of PFC and ECN together. A key point to understand is that even after adjusting the thresholds, hop-by-hop congestion spreading is not entirely eliminated. It is acceptable if the PFC is invoked momentarily. PFC is beneficial by itself because it avoids packet drops, but it has the side-effect of slowing down all the traffic with the same priority. If ECN can soon lower the traffic rate of only some flows, it can limit the spread and the duration of the hop-by-hop congestion spreading caused by PFC. As previously mentioned, the configuration should aim for the best of both approaches instead of aiming to eliminate PFC.



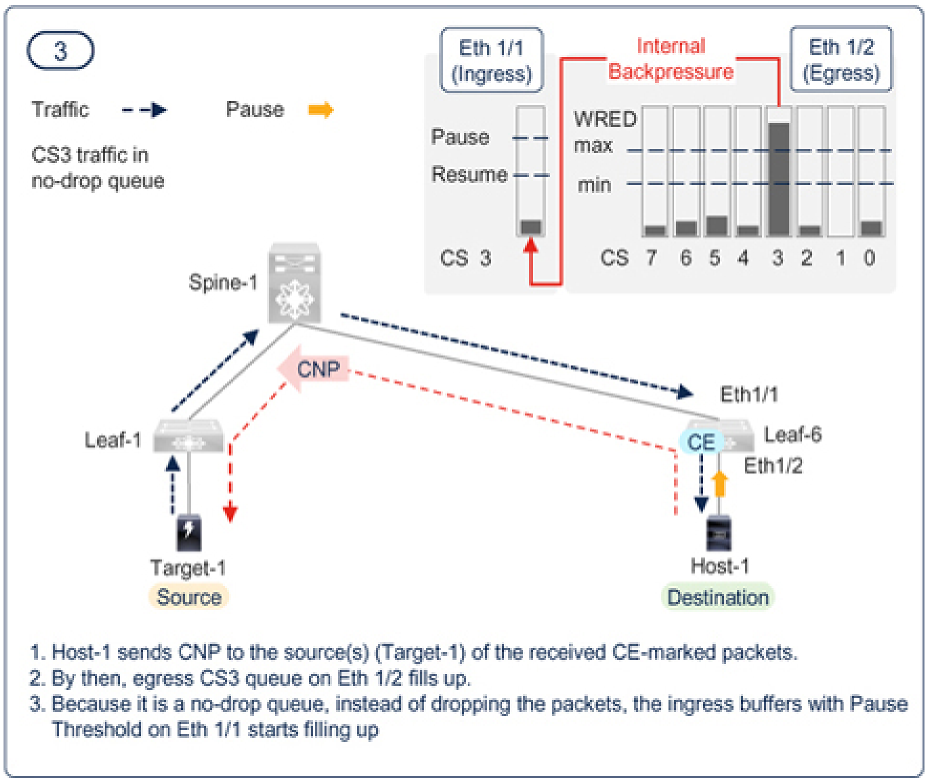
1. Many targets send CS3-marked traffic to Host-1 (only Target-1 is shown).
2. Host-1 is able to process the traffic at ingress rate, hence it does not send Pause frames. Also, Eth 1/2 is not over-utilized.
3. Overall, Eth1/2 is not congested in egress direction, and hence, it does not cause ingress congestion on Eth 1/1

Figure 7-18 *PFC and ECN working together in a RoCEv2 network — Step — 1*



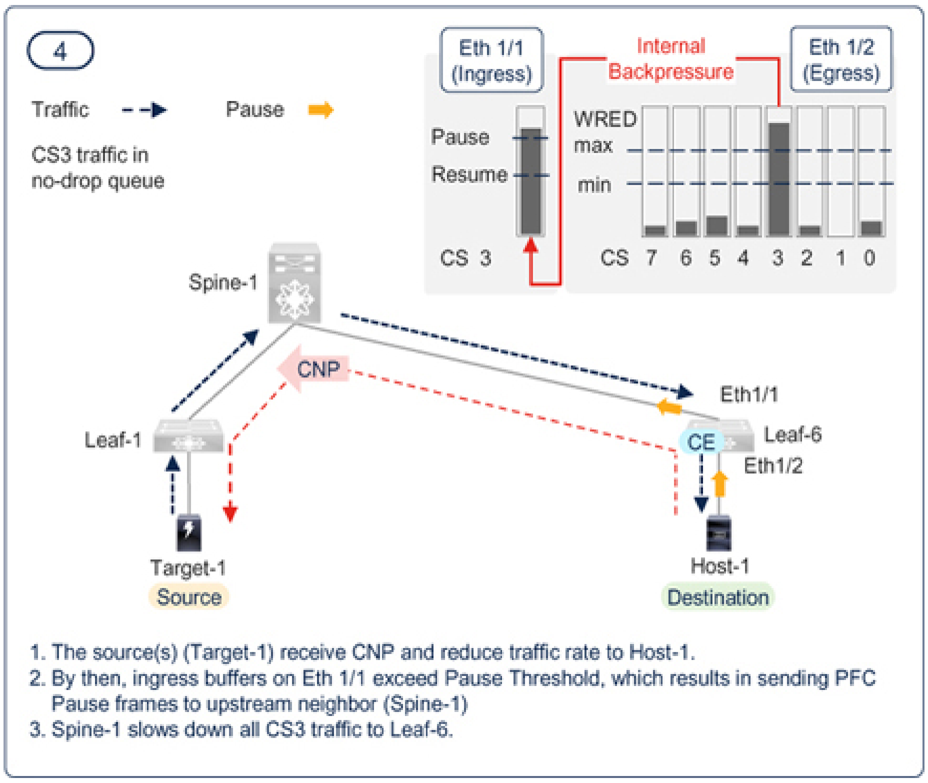
1. Host-1 now becomes a slow-drain device because it slows down the processing of the ingress traffic, and therefore, it starts sending Pause frames to Eth 1/2 on Leaf-6.
2. As a result, egress CS3 queue on Eth 1/2 starts filling up and exceeds WRED min threshold.
3. Consequently, Leaf-6 marks CE flag randomly in ECN-capable packets destined for Host-1.

Figure 7-19 *PFC and ECN working together in a RoCEv2 network — Step — 2*



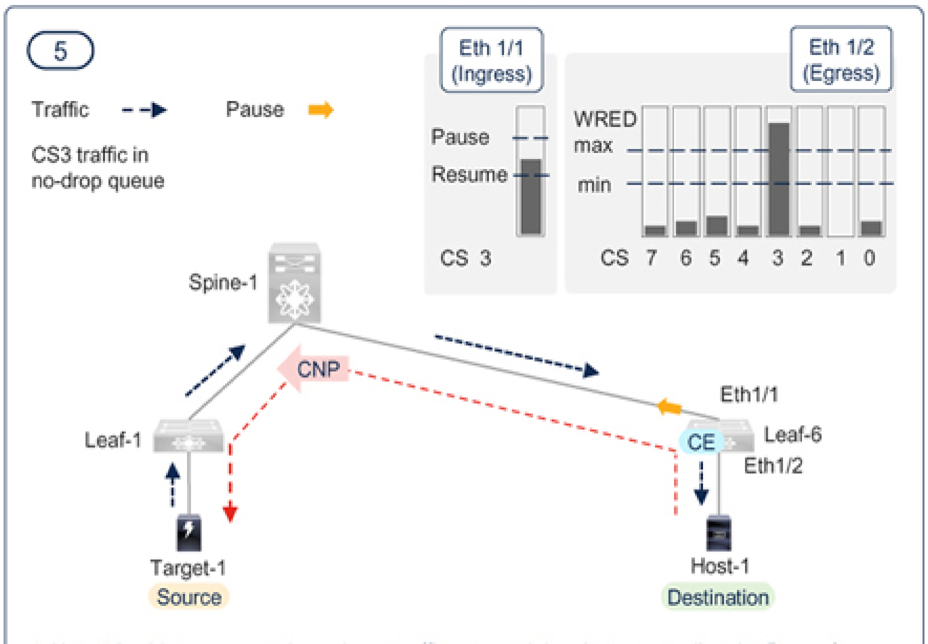
1. Host-1 sends CNP to the source(s) (Target-1) of the received CE-marked packets.
2. By then, egress CS3 queue on Eth 1/2 fills up.
3. Because it is a no-drop queue, instead of dropping the packets, the ingress buffers with Pause Threshold on Eth 1/1 starts filling up

Figure 7-20 *PFC and ECN working together in a RoCEv2 network — Step — 3*



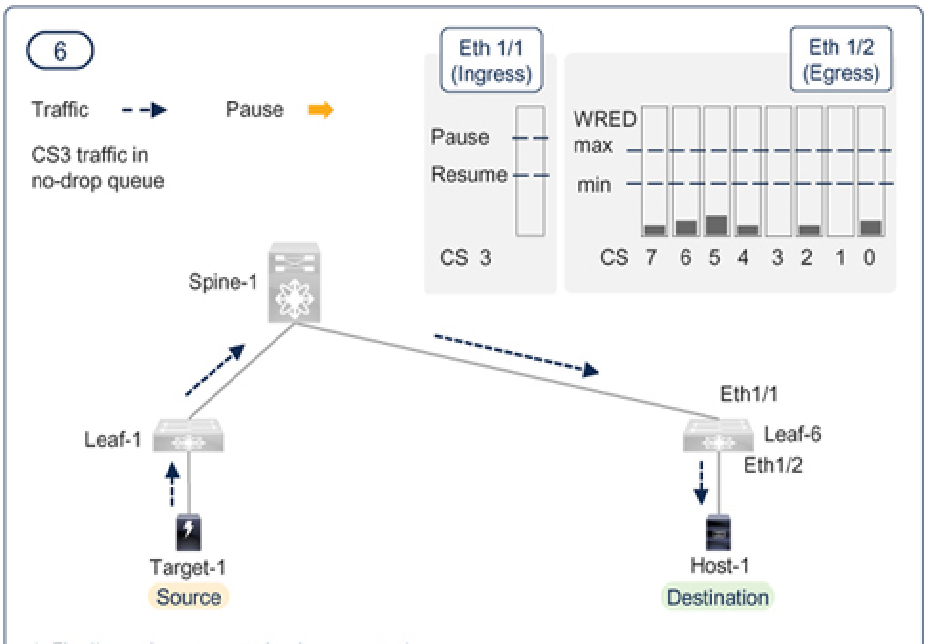
1. The source(s) (Target-1) receive CNP and reduce traffic rate to Host-1.
2. By then, ingress buffers on Eth 1/1 exceed Pause Threshold, which results in sending PFC Pause frames to upstream neighbor (Spine-1)
3. Spine-1 slows down all CS3 traffic to Leaf-6.

Figure 7-21 *PFC and ECN working together in a RoCEv2 network — Step — 4*



1. Host-1 is able to process the reduces traffic rate, and thus it stops sending the Pause frames.
2. This drains the egress CS3 queue on Eth1/2, which results in reduced ingress buffer utilization on Eth 1/1.
3. Pause frames are still sent to Spine-1 until the ingress buffer utilization falls below the Resume Threshold.

Figure 7-22 *PFC and ECN working together in a RoCEv2 network — Step — 5*



1. Finally, serious congestion is prevented.
2. Ingress buffers on Eth 1/1 are empty. Egress CS3 queue on Eth1/2 is empty or minimally used.
3. CNP may not avoid Pause frames. The real benefit is that they notify the source(s) to reduce rate, which helps the destination to stop sending the Pause frames soon. This limits the spread and duration of hop-by-hop congestion.

Figure 7-23 *PFC and ECN working together in a RoCEv2 network — Step — 6*

Lossless Traffic with VXLAN

Lossless traffic can be carried over Virtual Extensible LAN (VXLAN) similar to any other routed IP/Ethernet network. As a result, congestion detection, troubleshooting, and prevention remain similar.

VXLAN Overview

Virtual Extensible LAN (VXLAN) extends Layer 2 domains over a Layer 3 data center network. This allows flexible deployment of the workloads that need layer 2 adjacency across layer 3 boundaries. Another benefit of VXLAN is its higher scale. The Ethernet VLAN ID is a 12-bit field, which limits the number of VLANs to 4096. In contrast, the VXLAN network identifier (VNI) is a 24-bit field, which allows up to 16 million layer-2 domains. VXLAN also benefits from the equal-cost multipath (ECMP) routing of the layer 3 underlay network leading to high-speed non-blocking connections among the layer 2 domains.

VXLAN Transport

VXLAN extends Layer 2 domains over a Layer 3 network using a MAC-in-UDP encapsulation. Refer to Figure 7-24 for the VXLAN frame format. Two separate layer-2 networks connect to the layer-3 network via VXLAN tunnel endpoints (VTEPs). As their name suggests, the VTEPs connect via a VXLAN tunnel. They are aware of the MAC addresses reachable in the local layer 2 domain and the MAC addresses reachable via the remote VTEPs. When a VTEP receives a layer 2 frame destined for a device in the remote layer 2 domain, it encapsulates the layer 2 frame in an IP/UDP packet and sends it to the remote VTEP over the routed network. The remote VTEP decapsulates the packet and sends the underlying layer 2 frame to its destination.

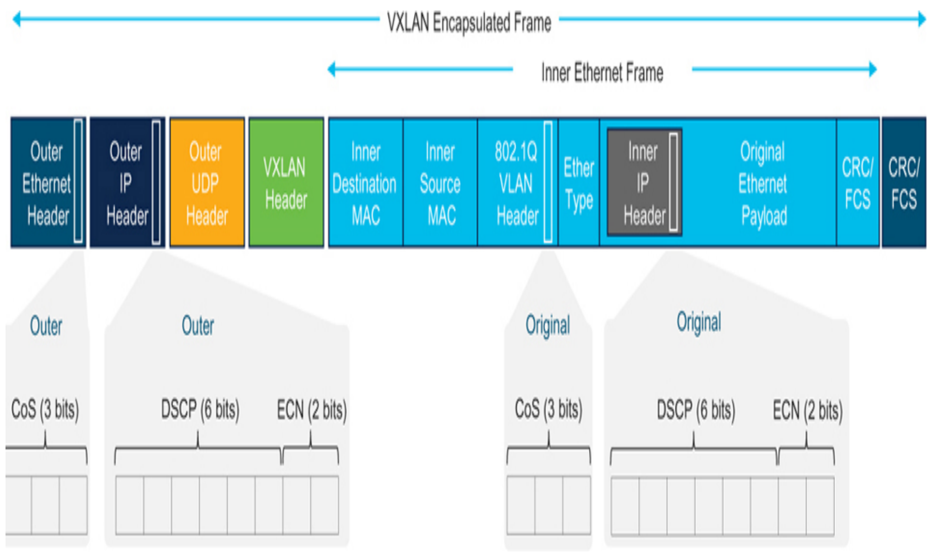


Figure 7-24 *VXLAN frame format*

Physical Topology

The VXLAN layer 3 networks are typically deployed in a spine-leaf topology (similar to Figure 7-8). This is called the underlay network. Packet forwarding within this underlay network is enabled by one of the IP routing protocols, such as IS-IS and OSPF.

The leaf switches, such as Cisco Nexus 9000, can act as hardware-based VTEPs. Each VTEP has two interfaces. One is a Layer 2 interface that connects the local layer 2 domains for local endpoint communication. The other is a Layer 3 interface on the routed underlay network.

Traffic between two VTEPs uses ECMP via multiple spine switches. An end device may not be aware that the packet to the destination will be encapsulated in the VXLAN packet by the VTEP. Likewise, the spine switches, which are in between the ingress and the egress VTEP, may not be aware that the packet belongs to a VXLAN tunnel. It just looks up the outer IP header to send to the destination VTEP.

MAC Address Learning

There are two common approaches for learning the MAC addresses of the devices that connect to the remote VTEP. The first approach uses flood-and-learn multicast-based mechanism. The second approach uses Multiprotocol Border

Gateway Protocol (MPBGP) Ethernet VPN (EVPN). Regardless of how the VTEPs learn the MAC addresses, the data path remains the same and hence, the congestion management remains the same as well.

Lossless Traffic over VXLAN

VXLAN can transport lossless traffic by classifying traffic based on the DSCP field in the IP header and assigning it to the no-drop queue. This scheme is explained in detail in the earlier section on *Layer 3 PFC*.

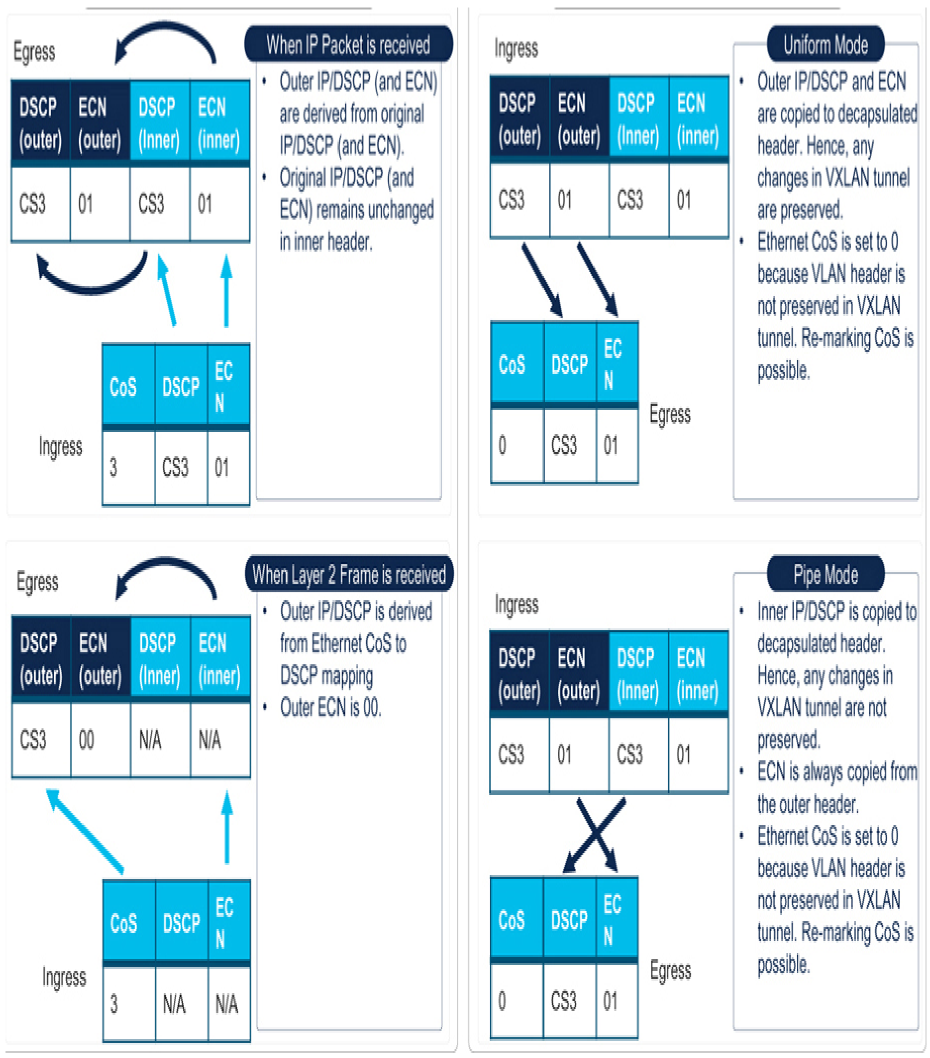
A classification scheme that enables *Layer 2 PFC* by classifying traffic based on the Ethernet CoS field is not enough for transporting lossless traffic over VXLAN because the IEEE 802.1Q VLAN header is not preserved in VXLAN tunnel, and hence, the CoS value is lost.

VXLAN Encapsulation

As Figure 7-25 explains, the ingress VTEP copies the DSCP value from the original IP header to the outer header of the VXLAN encapsulated packet. For a layer 2 frame, without an IP header, the DSCP field of the outer packet is derived from the CoS-to-DSCP mapping as explained in Table 7-1.

VXLAN Decapsulation

As Figure 7-25 explains, the egress VTEP copies the DSCP value from the outer VXLAN packet to the IP header of the decapsulated header. This is called uniform mode and it is the default behavior on Cisco Nexus 9000 switches. If required, the DSCP field value can be copied from the inner IP header to the decapsulated packet. This is called pipe mode.



When IP Packet is received

- Outer IP/DSCP (and ECN) are derived from original IP/DSCP (and ECN).
- Original IP/DSCP (and ECN) remains unchanged in inner header.

Uniform Mode

- Outer IP/DSCP and ECN are copied to decapsulated header. Hence, any changes in VXLAN tunnel are preserved.
- Ethernet CoS is set to 0 because VLAN header is not preserved in VXLAN tunnel. Re-marking CoS is possible.

When Layer 2 Frame is received

- Outer IP/DSCP is derived from Ethernet CoS to DSCP mapping
- Outer ECN is 00.

Pipe Mode

- Inner IP/DSCP is copied to decapsulated header. Hence, any changes in VXLAN tunnel are not preserved.
- ECN is always copied from the outer header.
- Ethernet CoS is set to 0 because VLAN header is not preserved in VXLAN tunnel. Re-marking CoS is possible.

Figure 7-25 *DSCP and ECN values during VXLAN encapsulation and decapsulation*

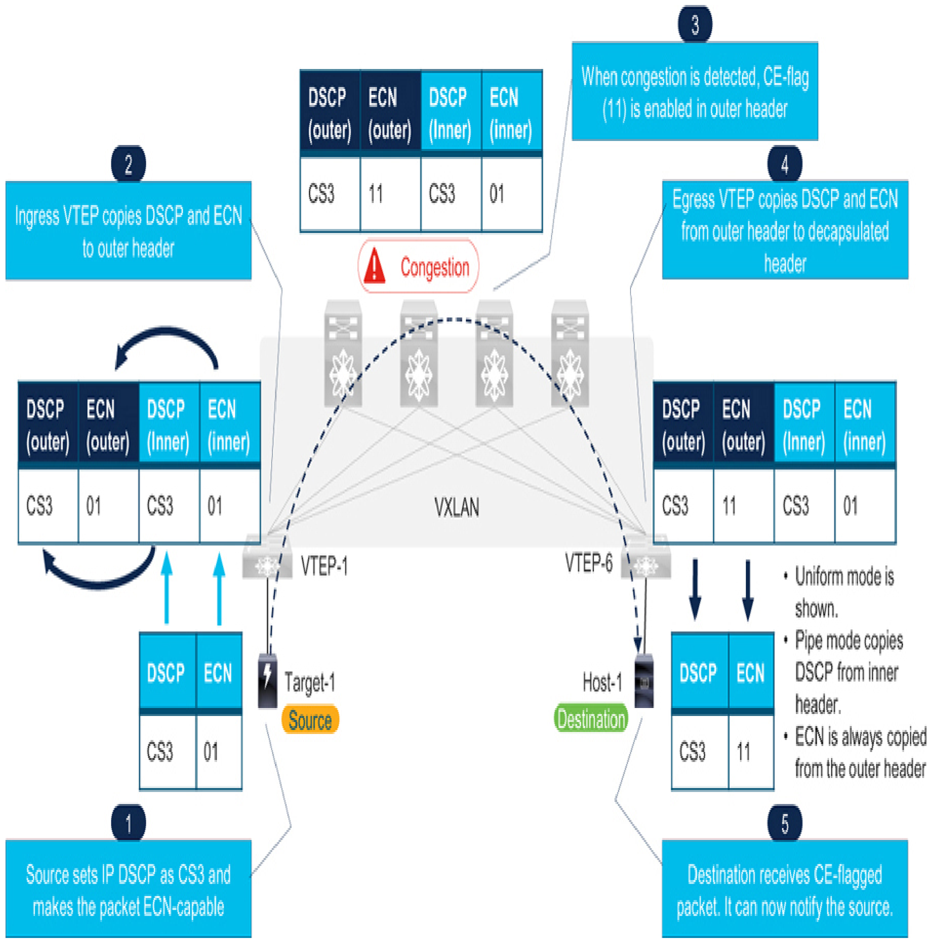
Congestion Notification over VXLAN

At the ingress VTEP, the ECN value of the ingress packet is copied to the outer header of the VXLAN encapsulated packet. At the egress VTEP, the ECN value is always copied from the outer VXLAN packet to the IP header of the decapsulated header, regardless of uniform or pipe mode.

Flow Control and Congestion Notification with VXLAN

Lossless traffic has two considerations with VXLAN. The first is the hop-by-hop flow control (PFC) for achieving the function of a lossless network. This is mandatory. The second optional consideration is to notify the end devices (ECN) when congestion is detected between the ingress and the egress VTEP.

Refer to Figure 7-26. Target-1 (source) sends traffic to Host-1 (destination). Hop-by-hop flow control is enabled using PFC by classifying the lossless traffic to the DSCP value of CS3 and assigning it to the no-drop queue. Because the ingress VTEP-1 copies the DSCP value from the original packet to the outer header, assigning the CS3-marked traffic to the no-drop queue on the spine switches achieves lossless behavior when the traffic is encapsulated in the VXLAN tunnel. At the egress VTEP-6, the DSCP value from the outer header is copied to the decapsulated packet. So, assigning the CS3-marked traffic to the no-drop queue on all the devices achieves lossless behavior. This is the same behavior as a non-VXLAN environment or a non-routed layer 2 network. The only difference with VXLAN is how the traffic is classified to be assigned to the no-drop queue.



3
When congestion is detected, CE-flag (11) is enabled in outer header
DSCP (outer) | ECN (outer) | DSCP (Inner) | ECN (inner)
CS3 | 11 | CS3 | 01
Congestion
2
Ingress VTEP copies DSCP and ECN to outer header
4
Egress VTEP copies DSCP and ECN from outer header to decapsulated header
DSCP (outer) | ECN (outer) | DSCP (Inner) | ECN (inner)
CS3 | 01 | CS3 | 01
VXLAN
DSCP (outer) | ECN (outer) | DSCP (Inner) | ECN (inner)
CS3 | 11 | CS3 | 01
VTEP-1
VTEP-6
DSCP | ECN
CS3 | 01
Target-1
Source
Host-1
Destination
DSCP | ECN
CS3 | 11
• Uniform mode is shown.
• Pipe mode copies DSCP from inner header.
• ECN is always copied from the outer header
1
Source sets IP DSCP as CS3 and makes the packet ECN-capable
5
Destination receives CE-flagged packet. It can now notify the source.

Figure 7-26 *PFC and ECN with VXLAN*

For congestion notification, the ingress VTEP-1 preserved the ECN values from the original header to the encapsulated packet. If a spine switch (or any switch in the VXLAN tunnel path) is congested, it flags the ECN-capable packets (b‘01’ or b‘10’) with the CE flag (b‘11’) in the outer header. The spine switches may not be aware that the IP packet belongs to a VXLAN tunnel or that there is another IP header within the packet. So they only mark the outer header. The egress VTEP copies the ECN value from the outer header to the decapsulated packet. When the destination receives this CE-marked packet, it reacts as per the capability of the upper layer protocols, such as RCM.

Congestion Management in VXLAN

As explained in the previous section, classifying and assigning the traffic to a no-drop queue preserves the lossless behavior of the traffic. This configuration must be consistent on all devices to preserve the end-to-end lossless behavior.

Note the following points:

1. **Understanding Congestion**: When PFC is enabled, congestion spreads in VXLAN as explained in the earlier sections. When an egress VTEP’s (or a leaf switch) queue starts filling up, it slows down the ingress traffic in the no-drop class by sending Pause frames. Consequently, the spine switch slows down all the traffic in that traffic class regardless of VXLAN encapsulation. Recall that for PFC, it does not matter how many headers are added to the packet. It just classifies and flow-controls the traffic using the DSCP field. The final state of congestion spreading is similar to as explained earlier in Figure 7-8.

2. **Detecting Congestion**: The congestion detection approach remains similar to what’s explained in the earlier section. The detection commands should account for DSCP-to-CoS mapping on the VTEPs.

3. **Congestion Troubleshooting**: The congestion troubleshooting methodology remains similar to what’s

explained in the earlier section. While chasing the source of congestion, follow the traffic class on switchports or interfaces. Especially, on the spine switches do not be misled by the IP addresses because the VXLAN encapsulated packets have the IP addresses of the ingress and the egress VTEP in the outer header. Multiple flows (source-IP and destination-IP) are transported within the same VXLAN tunnel. Because of this reason, focus on monitoring traffic and Pause frames for no-drop traffic class instead of flows.

4. **Congestion Prevention**: The congestion prevention features explained earlier apply to VXLAN as well. If the end devices support an action based on ECN values, it works the same regardless of the VXLAN underlay network. For example, RoCEv2 traffic can be transported over VXLAN, and if the end device supports RCM, it works the same with VXLAN as well.

Summary

By default, Ethernet handles congestion by dropping frames (called Lossy Ethernet) and relies on an upper-layer protocol, such as TCP, for retransmitting the lost packets. In contrast, lossless Ethernet uses a hop-by-hop flow control mechanism to slow down or stop the transmission by sending Pause frames. All the traffic on an Ethernet link can be flow-controlled using LLFC. Alternatively, PFC can selectively flow control only specific traffic classes. PFC makes the foundation of converged Ethernet networks by allowing lossless and lossy traffic on the same links. Additionally, ETS provides a minimum bandwidth guarantee for different traffic classes and DCBX simplifies the configuration on the end devices and the switches. PFC can be enabled by classifying the traffic using the PCP/CoS field in the Ethernet VLAN header at layer 2 of the OSI model. This Layer 2 PFC works for FCoE and RoCE. Alternatively, for RoCEv2 (Routable RoCE), PFC can be enabled by classifying the traffic using the DSCP field in the IP header at layer 3.

Lossless Ethernet networks are prone to similar types of congestion as Fiber Channel fabrics because both use hop-by-hop flow control. Congestion spreads similarly within the no-drop class because of slow-drain, over-utilization of the link, bit errors, or lack of enough buffers. The same congestion detection, troubleshooting, and prevention approaches apply to lossless Ethernet networks as well. But a lossless Ethernet switch may not report all the relevant metrics. Notable examples are the TxWait and RxWait metrics, which are not available on Cisco Nexus 9000 switches and UCS servers at the time of this writing. The next resort is to use the number of Pause frames for detecting congestion. But because these metrics are not stored on the switch with time and date stamps, using an external monitoring platform simplifies congestion detection and troubleshooting.

When using converged Ethernet Networks, traffic in lossy classes may affect traffic in no-drop classes based on how the problem surfaces. Regardless of using a dedicated or shared storage network, monitor the traffic utilization and congestion metrics at a per-port level and per-class level.

Recovering from congestion in lossless Ethernet networks can be achieved using Pause timeout and PFC Watchdog. These features drop frames when they can't be sent to their destination after a timeout interval, which helps in freeing up the buffers resulting in recovering the victim devices from the effect of congestion.

RoCEv2 networks can also benefit from notifying the end devices about network congestion if the end devices support RoCEv2 Congestion Management, resulting in a reduced traffic rate by the sender. Regardless of the type of prevention mechanism, none of these mechanisms should be used as a long-term solution. Monitor the networks, find the root cause, and make a corrective change as soon as possible.

Finally, note that many lossless Ethernet networks are relatively new. Congestion becomes more severe as networks grow and/or mature. Because Fibre Channel fabrics have been under use for many decades at a large scale, it is important to learn from them and apply the knowledge to lossless Ethernet networks to prevent congestion issues proactively.

References

- FC-BB-5: http://fcoe.com/09-056v5.pdf
- 802.1 Data Center Bridging Task Group: http://www.ieee802.org/1/pages/dcbridges.html
- 801.Qaz Enhanced Transmission Selection and DCBX: http://www.ieee802.org/1/pages/802.1az.html
- 801.Qbb Priority-based Flow Control: http://www.ieee802.org/1/pages/802.1bb.html
- End-to-End QoS Network Design: Quality of Service for Rich-Media & Cloud Networks (Networking Technology) 2nd Edition. Cisco Press. ISBN-13: 978-1587143694
- Cisco Nexus 9000 Series NX-OS Interfaces Configuration Guides
- I/O Consolidation in the Data Center Cisco Press (ISBN-13: 978-1587058882)
- Cisco Nexus 9000 Series NX-OS Programmability Guide
- NVMe over Fabrics (NVMe-oF) End-to-End Configuration with RoCEv2 and Fibre Channel - BRKDCN-3282 — Cisco Live 2020 Barcelona
- Cisco White Paper: Priority Flow Control: Build Reliable Layer 2 Infrastructure
- Supplement to InfiniBandTM Architecture Specification Volume 1 Release 1.2.1 Annex A17: RoCEv2
- SONiC PFC Watchdog Design: https://github.com/sonic-net/SONiC/wiki/PFC-Watchdog-Design
- The Addition of Explicit Congestion Notification (ECN) to IP — RFC 3168

- Building Data Centers with VXLAN BGP EVPN: A Cisco NX-OS Perspective — Cisco Press - ISBN-10: 1-58714-467-0, ISBN-13: 978-1-58714-467-7
- VXLAN BGP EVPN Multi-Site — BRKDCN-2913 — Cisco Live 2022, Las Vegas
- A Day in the Life of a VXLAN EVPN Multi-Site Packet - BRKDCN-2345 — Cisco Live 2022 Las Vegas
- Implementation of PFC and RCM for RoCEv2 Simulation in OMNeT++
- Revisiting Network Support for RDMA Extended version of the SIGCOMM 2018 paper
- Cisco Nexus 9000 Series NX-OS VXLAN Configuration Guide
- Cisco Press: Cisco IP Telephony Flash Cards: Weighted Random Early Detection (WRED): https://www.ciscopress.com/articles/article.asp?p=352991&seqNum=8
- Internet Assigned Numbers Authority Service Name and Transport Protocol Port Number Registry: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt
- Cisco White Paper — RoCE Storage Implementation over NX-OS VXLAN Fabrics: https://www.cisco.com/c/en/us/td/docs/dcn/whitepapers/roce-storage-implementation-over-nxos-vxlan-fabrics.html
- Cisco White Paper — Understanding FEC and Its Implementation in Cisco Optics
- Cisco Troubleshooting Technote — Nexus 9000 Cloud Scale ASIC CRC Identification & Tracing Procedure
- Cisco Troubleshooting Technote — Understand Cyclic Redundancy Check Errors on Nexus Switches

Chapter 8. Congestion Management in TCP Storage Networks

This chapter covers the following topics.

- Understanding congestion in TCP storage networks.
- Detecting congestion in TCP storage networks.
- Traffic patterns with iSCSI and NVMe/TCP and correlation with network congestion.
- Preventing congestion in TCP storage networks.

At the time of this writing, iSCSI is the most used protocol for carrying block storage traffic via a TCP/IP network. NVMe/TCP is emerging and expected to be in the mainstream in the coming years. As Chapter 1, "Introduction to Congestion in Storage Networks," explains, these "storage protocols" use TCP at layer 4 of the OSI model as their "transport protocol". This means that congestion management for iSCSI and NVMe/TCP traffic boils down to congestion management in TCP/IP networks with added considerations for storage operations.

TCP is a key protocol for the success of the Internet. It has been an active subject of research for 50 years. But this chapter focuses only on the relevant aspect of block storage traffic, especially for two types of users. First, those who have Fibre Channel experience but not as much TCP/IP experience. Second, those who have TCP/IP experience but do not have as much experience in handling storage traffic.

This chapter is not a comprehensive reference to TCP. Many details are purposefully skipped or over-simplified to keep the focus on the title of this book and because in-depth content is already written on it. Also, this chapter does not explain the details of the various TCP implementations, such as Vegas, Reno, etc. Refer to the references section for further details.

Because block-storage traffic has the strictest requirements, if a network can deliver these, the same network can also achieve the performance goals of file and object storage traffic. Hence, this chapter is relevant for all kinds of storage regardless of their type—Block, File, or Object, although details are explained only for block-storage protocols (iSCSI and NVMe/TCP).

One of the points that we wish to convey by this book is to apply what has been learned on one transport type (such as Fibre Channel) to another (such as TCP). For organizations that operate all types of transports (Fibre Channel, Lossless Ethernet, and TCP), it is much better to have a consistent congestion detection and troubleshooting methodology and skilled team members regardless of the transport type. These details are explained in the following sections.

Understanding Congestion in TCP Storage Networks

This chapter focuses on TCP transport over lossy Ethernet Networks, which is the default and the most used mode of Ethernet. Technically, TCP can also operate in a lossless Ethernet network, but this is not common, and hence, this chapter only briefly explains it later in the section on *iSCSI and NVMe/TCP in a Lossless Network*. Unless specifically mentioned, "TCP Storage Networks" in this chapter refers to TCP transport over a lossy Ethernet network.

Comparison with Lossless Networks

TCP provides reliable transport. This means that iSCSI and NVMe/TCP layers rely on TCP to deliver data without any loss, duplication, errors, and in the same order as sent by the sender. This does not mean the packets are not lost, duplicated, errored, or get out of order. It just means TCP recovers from these network issues by rearranging the out-of-order packets and retransmitting the lost packets. But detecting the lost packets and their retransmission causes a delay in latency-sensitive storage traffic.

Fibre Channel and TCP’s approaches for achieving reliability and performance are quite different. Fibre Channel fabrics make every attempt of preventing frame drops using B2B flow control that works between directly connected devices. However, under severe congestion or due to bit errors, frames may still be dropped, which are directly seen by the upper layers (SCSI and NVMe) resulting in reinitiating the entire I/O operation. In contrast, TCP retransmits just the lost packets, and hence, the upper layers do not even recognize that the retransmissions have occurred except that the I/O operation took longer than normal.

Another key factor is that TCP uses built-in flow control and congestion control mechanisms to push congestion to the traffic sender. But the real cause of congestion does not go away by itself. In other words, TCP moves congestion, but cannot solve the root cause by itself. Consider a host that requires a read I/O throughput of 100 MB/s. During network congestion, the TCP layer may slow down the traffic to 50 MB/s. As a result, network congestion is controlled but now congestion is moved within the TCP sender. The upper layers, such as SCSI or NVMe, want to send faster but the TCP layer applies backpressure to slow it down. As a result, the application performance is degraded. But remember that the primary purpose of a network is to meet the application requirements. Therefore, the built-in mechanism of TCP should be relied upon only until the

- The source of congestion is determined.
- The cause of congestion is determined.
- The corrective action is known.

- Resources are available to implement a permanent corrective action.

With this perspective, addressing the real cause of network congestion remains mostly the same regardless of the transport type—TCP, Fibre Channel, or lossless Ethernet. These details are explained in the following sections.

How iSCSI and NVMe/TCP Exchange Data

The following are the high-level steps for the end devices to exchange data using iSCSI and NVMe/TCP protocols over a TCP/IP network (Figure 8-1).

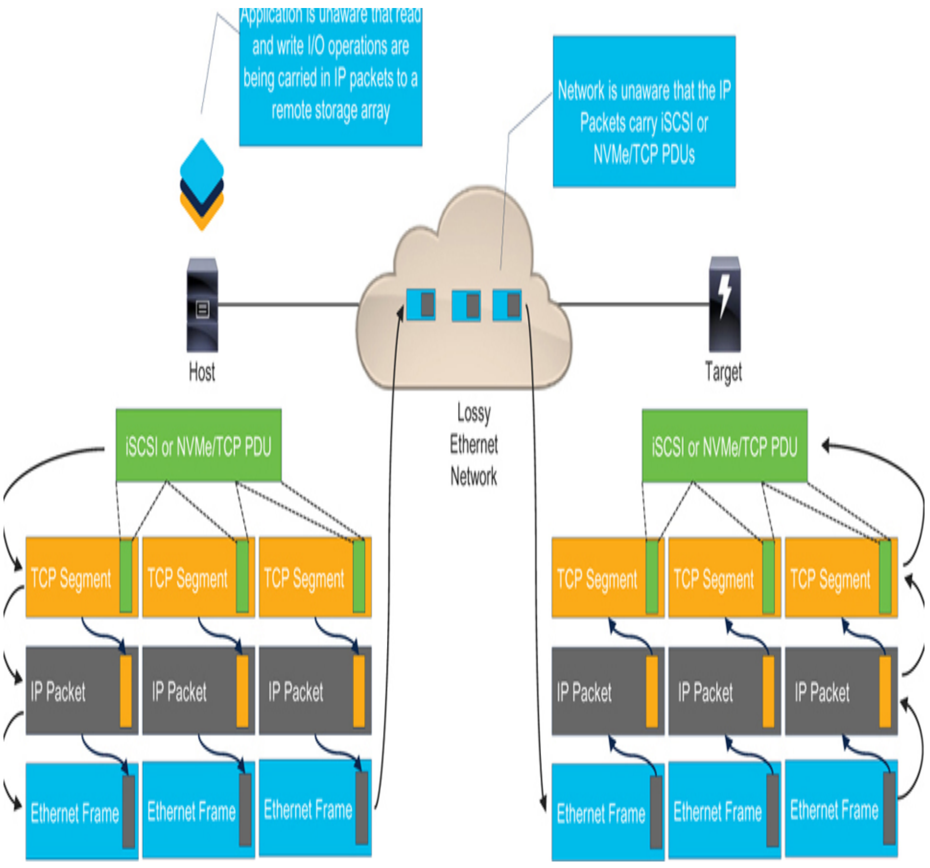

Application is unaware that read and write I/O operations are being carried in IP packets to a remote storage array
Network is unaware that the IP Packets carry iSCSI or NVMe/TCP PDUs
Host
Target
Lossy Ethernet Network
iSCSI or NVMe/TCP PDU
TCP Segment
IP Packet
Ethernet Frame
iSCSI or NVMe/TCP PDU
TCP Segment
IP Packet
Ethernet Frame

Figure 8-1 *iSCSI and NVMe/TCP data exchange via a TCP/IP network*

1. A host (or initiator) and the target must be aware of the IP addresses of each other via a manual configuration or an automated discovery, such as the use of a Centralized

Discovery Controller (CDC) for NVMe/TCP environments.

2. After knowing the IP address of the target, an initiator establishes a TCP connection with the target using a three-way handshake using SYN, SYN-ACK, and ACK. This is no different from the standard approach of establishing TCP connections. For I/O operations, generally, the host (initiator) acts as the TCP client by sending the SYN to the target (controller).
3. The initiator and target may open multiple TCP connections between them. These connections are identified by source IP, source port, destination IP, and destination port. Typically, the TCP server port is preassigned by Internet Assigned Numbers Authority (IANA) based on an upper layer. For example, the iSCSI server (target) uses TCP port 3260, and NVMe/TCP server (controller) uses TCP port 4420. The client (initiator or host) may use an unused port as a source port. By using a different source port number, an initiator can open multiple TCP connections to the same target. A TCP connection on the end devices is identified as a TCP flow on a network. Note that for identifying a network flow, it is important to include the layer 4 protocol, such as TCP, with the port numbers because other protocols at layer 4, such as UDP, may use the same port numbers for different purposes.
4. After establishing TCP connections(s), iSCSI or NVMe/TCP protocols have their specific steps, such as establishing an iSCSI session or NVMe/TCP connection. These steps can be ignored in this conversation to keep the focus on the I/O operations.
5. Finally, when the initiator and target are ready for read and write I/O operations, TCP accepts a stream of bytes from upper layers (iSCSI or NVMe/TCP), divides it into segments, adds a TCP header to each segment, and passes them to the lower layers. The data from the iSCSI and NVMe/TCP layers are in the form of Protocol Data Units (PDU). If PDUs are large, one or more TCP segments

may be created from a PDU. If PDUs are small, multiple PDU may be sent in the same TCP segment. Refer to Chapter 1, the section on *iSCSI* for iSCSI packet format. Refer to Chapter 1, the section on *NVMe/TCP* for NVMe/TCP packet format.

6. The TCP segments are encapsulated in IP (IPv4 or IPv6) packets and then again encapsulated in an Ethernet frame. Refer to Chapter 1, the section *Transmission Control Protocol (TCP)* for the Ethernet frame format. A precise terminology would be to call them TCP segments, IP packets, and Ethernet frames. However, for the sake of simplicity, this chapter uses these terms interchangeably.
7. Finally, the host transmits the Ethernet frame to the network. The switches, in turn, send the packet to the target using the destination MAC address in the Ethernet header and the destination IP address in the IP header. This is no different from the standard approach of forwarding traffic in IP/Ethernet networks.
8. The target receives the frame, decapsulates it through the Ethernet, IP, and TCP layers, and passes the data in the form of PDU to iSCSI or NVMe/TCP layers.

The steps for transporting iSCSI and NVMe/TCP I/O operations via a TCP connection as shown in Figure 8-1 lead to the following conclusions:

1. In remote block storage deployments, the applications are unaware that the storage is physically outside the local enclosure. The host operating system abstracts the details of iSCSI and NVMe/TCP from the applications, and hence, an application just sees a storage volume. In other words, the application reads and writes data to the storage volumes without being aware that those I/O operations are carried over a TCP transport.
2. The network forwards packets to the destination without being aware that a particular packet carries SCSI or NVMe I/O operation.
3. The network can be configured to classify storage traffic in a class and give it preferential treatment using QoS,

such as providing it with a minimum bandwidth guarantee. This classification is typically achieved using the DSCP field in the IP header. The iSCSI and NVMe/TCP layers remain unaware of any QoS on the network. Later, Figure 8-9 shows the IPv4 packet format and the 6-bit DSCP field. Refer to Chapter 7, "Congestion Management in Ethernet Storage Networks," in Table 7-1 for DSCP values.

4. The most crucial factor is that the iSCSI and NVMe/TCP layers rely on the lower layers (TCP, IP, and Ethernet) for the following functions.

   a. **Bit Errors**: If a packet is corrupted, it is dropped. TCP treats this as a lost packet, and retransmits it. Essentially, iSCSI and NVMe/TCP layers rely on TCP for an error-free stream of data, although the upper layers may have additional mechanisms as well. TCP header contains a 16-bit checksum for this purpose (Figure 8-2).

   b. **In-order delivery**: If packets arrive out of order from the network, the TCP destination rearranges them in the correct order and hands off the byte stream to the upper layers. The iSCSI and NVMe/TCP layers remain unaware of the actual order of the arrival of the packets. TCP header contains a 32-bit sequence number for this purpose.

   c. **Network performance**: iSCSI and NVMe/TCP rely on the TCP and IP layers for achieving performance via a network. The network may use the TCP flow information (source and destination port numbers and IP addresses) for load-balancing among the member of an aggregated link (port-channel) or the equal cost multi-path (ECMP) links to achieve uniform network utilization.

   d. **Congestion in the end device**: When a target sends data to an initiator faster than the initiator can process (or vice-versa), the iSCSI and NVMe/TCP layers rely on the TCP flow control mechanism to tackle this condition.

e. **Congestion in the network**: The iSCSI and NVMe/TCP layers rely on the TCP congestion control mechanism to tackle network congestion.

These are end-to-end mechanisms between the TCP connection endpoints, and they work separately for each connection. The following sections explain more details.

Bit Errors in Lossy Ethernet Networks with TCP Transport

As explained in Chapter 1, when a bit stream is exchanged over a network, some bits may be altered resulting in bit errors. These bit errors may occur because of environmental issues, such as temperature and humidity, causing interference with data transmission, faulty cables, faulty SFPs, loose cable connections, loose SFP connections, and accumulated dust on cable end points, SFPs, or patch panels.

A packet is considered corrupted when its contents are changed between a sender and a receiver.

Ethernet CRC

At layer 2 of the OSI model, the Ethernet header (Chapter 1, Figure 1-6) carries a 4-byte Frame Check Sequence (FCS) that carries the result of the cyclic redundancy check (CRC) calculation. Because the source MAC and destination MAC change at every layer 3 hop, the FCS field is re-calculated and updated in the Ethernet header before sending the frame to the next hop. An ingress Ethernet frame is found to have a CRC error when the result of the locally computed CRC calculation does not match the FCS value in the frame.

Although most switches can detect CRC-corrupted frames, their handling varies. A switch that is operating in store-and-forward mode drops CRC-corrupted frames. In contrast, a switch that is operating in cut-through mode cannot drop a corrupted frame because it starts forwarding the frame on the egress port before seeing the end of the frame. Finally, an end device that is the ultimate destination of the frame always drops a corrupted frame.

The scope of the Ethernet FCS field includes the Ethernet header and payload (all fields except the FCS).

TCP Checksum

In some circumstances, a TCP segment may be corrupted before it is encapsulated in the IP header on a sender, although this condition does not necessarily indicate bit errors. These errors are detected by the 16-bit checksum field in the TCP header (Figure 8-2). It works similarly to the Ethernet FCS field, but the scope of TCP checksum includes TCP header and TCP payload.

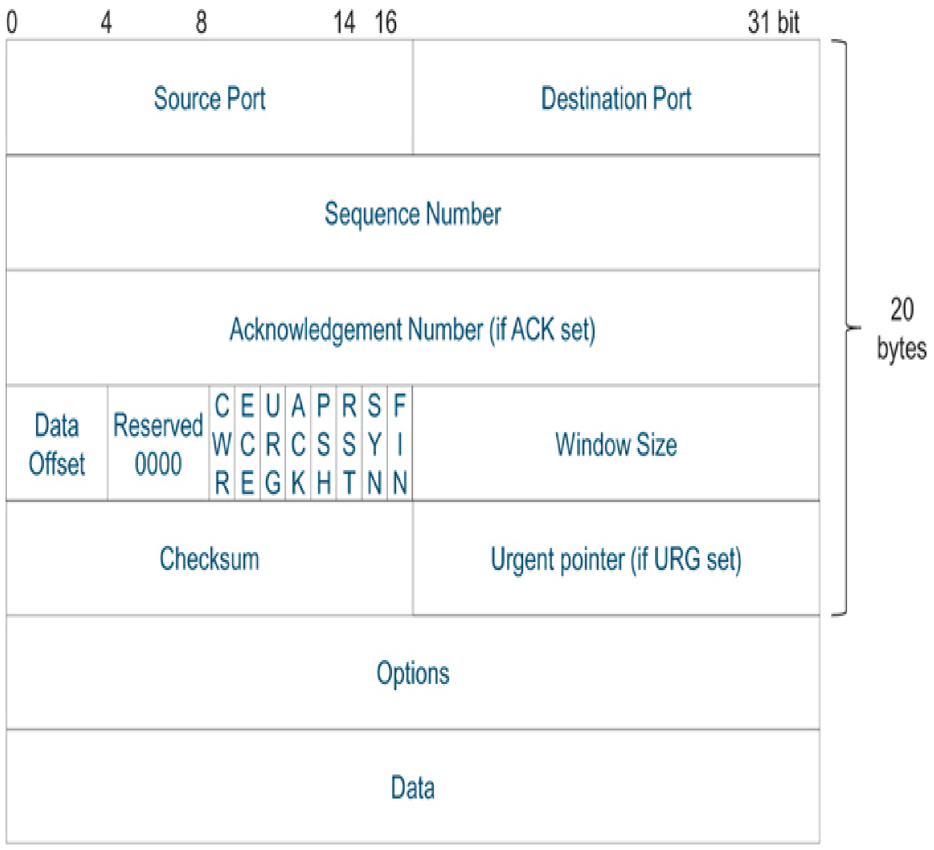


Figure 8-2 *TCP header and segment format*

If a packet is dropped at lower layers due to Ethernet FCS/CRC error, for the TCP layer this is considered a lost packet. If a TCP segment is dropped due to the TCP checksum error, it is still considered a lost packet. In other words, regardless of the location of errors in the packet, the key point to remember is that corrupted packets are discarded/dropped, and TCP recovers them by retransmission.

Comparison with Lossless Networks

Bit errors may corrupt R_RDY primitives in Fibre Channel fabrics (explained in Chapter 2, “Understanding Congestion in Fibre Channel Fabrics”) and Pause frames in lossless Ethernet networks (explained in Chapter 7). This interference with the hop-by-hop flow control mechanisms may lead to congestion or worsen the congestion, which affects many other end devices that share the same network links.

In TCP networks, the effect of bit errors is limited only to the source and the destination (flow) of a corrupted (dropped) frame or packet.

Regardless of this difference among various transport types, you should still detect bit-errors, find their root cause, and make the corrective change as soon as possible.

How TCP Provides Reliable Data Transfer

Performance in TCP storage networks can be increased by detecting and eliminating the reasons for packet drops, which results in invoking its reliability mechanisms. For that, understanding it is the first step.

TCP Timers — RTT, SRTT, RTO

A TCP receiver sends an Acknowledgement (ACK) after it receives a packet. When the TCP sender receives this ACK, it knows that the receiver has received the packet. The time difference between the sent packet and its received ACK is the Round-Trip Time (RTT). For accounting the variation in the network delay, the sender uses this RTT to calculate Smoothed Round-Trip Time (SRTT). The sender then uses this SRTT to calculate the Retransmission Timeout (RTO).

The sender continuously adjusts the RTO based on the SRTT and RTT. In reality, the calculation of RTO is much more involved. Refer to IETF RFC 6298 for further details.

These timers are relevant only at the TCP layer. None of the upper layers (iSCSI or NVMe/TCP) and lower layers (IP and Ethernet) are aware of these.

Timeout-Based Retransmission

A TCP sender starts an RTO timer when it sends a packet. The ACK may not arrive before the expiry of the RTO timer either because the packet is lost or delayed due to congestion. If the ACK is not received before the RTO timer expires, the sender retransmits that packet. This is called timeout-based retransmission.

Waiting for the RTO timer is not an efficient mechanism for maintaining high throughput, as explained next.

Duplicate ACK-Based Retransmission — A.K.A Fast Retransmission

For achieving better performance, TCP uses a retransmission mechanism that is based on three duplicate ACKs.

Consider this example. In a series of segments, let's say 1 – 10, if segments 1 – 5 arrive, the TCP receiver sends ACK with the next expected in-order sequence number, which is segment 6. If segment 6 is lost, but segment 7 arrives, the receiver still sends ACK with the next expected in-order sequence number, which is still segment 6. Then segment 8 arrives, but the receiver still sends an ACK with the same next expected in-order sequence number of 6. Then segment 9 arrives, but the receiver still sends ACK with the next expected in-order sequence number of 6. At this time, the sender would have received three duplicate ACKs (four ACKs with sequence number 6). As a result, it assumes segment 6 is lost and it immediately retransmits the next in-order segment, which is segment 6, without waiting for RTO to expire.

This duplicate ACK-based retransmission scheme is a lot more efficient than waiting for the RTO to expire. Because of its increased efficiency, it is used for fast recovery by TCP congestion control, explained later.

Selective Acknowledgment (SACK)

Some TCP implementations use Selective Acknowledgment (SACK), which increases performance during packet drops by selectively acknowledging only the received packets, and hence transmitting only the lost packets instead of retransmitting all packets after the lost packets.

Duplicate Packets and Ordered Data Transfer

When packets or their ACKs are delayed, but not lost, it results in the TCP sender retransmitting those packets, which ultimately results in the receiver receiving the same packet twice. This problem is solved by using a sequence number on the TCP header (Figure 8-2). A receiver remembers the sequence numbers of the already received packets and if the same sequence number is received soon, it discards the duplicate packet.

The sequence number is also used for rearranging the out-of-order packets. This allows a receiver to hand off the byte stream to the upper layers in the same order as sent by the sender.

Comparison with Fibre Channel Fabrics

In Fibre Channel (and FCoE) fabrics, there is no acknowledgment of each frame and hence, no retransmission of a single lost frame. If a frame is lost, that I/O operation (Exchange) does not complete. Thus, the entire I/O operation is reinitiated after a timeout at SCSI or NVMe layers, which is typically set to 30 seconds. This timeout should not be confused with the TCP RTO timer because even with iSCSI and NVMe/TCP, the upper layers (SCSI and NVMe) are the same and have similar timeouts.

The Fibre Channel header carries an Originator Exchange ID (OX_ID) and Responder Exchange ID (RX_ID) for identifying the frames that belong to the same I/O operation (Exchange). Within an exchange, the sequence count identifies the order of frames within a sequence. Typically, in-order delivery is natively provided by the Fibre Channel fabrics.

TCP Flow Control

Performance in TCP storage networks can be increased by detecting and eliminating the reasons that result in invoking its flow control. For that, understanding it is the first step.

How TCP Flow Control Works

As Figure 8-2 shows, the TCP header carries a 16-bit Window Size, which is used by the traffic receiver to inform the traffic sender about its available free buffers. When the receiver has more free buffers, it can inform the sender to send more traffic by increasing the window size. However, if the receiver's buffers are becoming full, it reduces the window size to slow down the sender. If needed, the receiver can inform a window size of zero to stop the sender from sending any traffic.

As previously explained, while the traffic is sent from the sender to the receiver, ACKs are sent in the reverse direction to inform the sender about the data that the receiver has received. A receiver uses these ACKs to inform the window size to the sender, and hence, the window field in the TCP header can be called the receiver-window.

Comparison with Lossless Networks

Because of the end-to-end flow control mechanism, TCP, by default, handles slow-drain, which is a common type of congestion in lossless networks.

In Fibre Channel fabrics (explained in Chapter 2) and lossless Ethernet networks (explained in Chapter 7), a device causes congestion due to slow-drain when it has a slower processing rate as compared to the rate at which frames are being delivered to it. Instead of dropping the frames due to the lack of free buffers, this slow-drain device slows down all its ingress traffic on the link or in the no-drop traffic class. Unlike TCP, it does not use any end-to-end flow control mechanism to inform the traffic source to reduce its transmission rate.

The concept of TCP receiver-window that's advertised by a receiver is conceptually similar to the Fibre Channel receive B2B credits, although it works differently. A major difference is that the TCP receiver-window is for an end-to-end connection, whereas FC B2B credits are for a link.

End-to-End flow control is also defined in Fibre Channel (Chapter 2), but it was never widely implemented. Imagine that had it been implemented; it would have had many similarities with TCP flow control.

TCP Congestion Control

Performance in TCP storage networks can be increased by detecting and eliminating the reasons that result in invoking its congestion control. For that, understanding it is the first step.

Congestion in a network may drop or delay the packets. In both these conditions, the TCP sender retransmits these packets. However, without checks and balances, sending more packets on an already congested network may lead to a condition called congestion collapse. A related phenomenon is observed when network congestion results in dropping one packet from many I/O operations. The TCP layer will continue to retransmit leading to high network utilization. But the applications cannot proceed until all steps of an I/O operation completes. This concept is typically explained by the availability of "throughput" that lacks "goodput".

For avoiding these and many more issues, TCP uses robust congestion control mechanisms, which are complex and have many variations. This section purposefully skips those details to focus on the subject of this book and because in-depth educational content is already written on them. The focus here is only on a high-level overview of Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery algorithms that are collectively called TCP Congestion Control.

Slow Start

When a sender starts transmission, it does not know how much traffic it can send via a network. There may be speed-mismatches, many-to-one traffic patterns, existing congested links, and many such issues that inhibit the network from sending traffic at the sender's desired rate. Hence, a sender starts slowly and increases its rate every round-trip time (RTT is explained in the earlier section on *TCP Timers — RTT, SRTT, RTO*). This is called the slow start phase (Figure 8-3).

TCP uses a congestion-window to limit the amount of data that a sender sends on a network. This congestion-window is different from the receiver-window that the TCP receiver advertises about its free buffers to the sender. TCP receiver-window is explained in the earlier section on *How TCP Flow Control Works*. The sender uses the lower of the receiver-window and congestion-window to decide how much data it is allowed to send. If there is no network congestion, the congestion-window is the same as the receiver-window. As a result, the sender can send as fast as the receiver can receive.

Another difference between the receiver-window and the congestion-window is that only the receiver-window is advertised by the receiver to the sender, whereas congestion-window is locally calculated by the sender.

Note that the congestion-window is linked to the Sliding Window Protocol in TCP, and hence a window may contain multiple TCP segments. Also, TCP associates its throughput with the window size because the larger the window, the more data is transmitted, which essentially leads to higher throughput.

TCP initializes the congestion-window to one, sends an initial segment, and waits. When the acknowledgment arrives, it increases the congestion-window to 2. Following this scheme, the congestion-window is doubled every RTT, and hence, this scheme achieves an exponential increase in throughput.

By the end of the slow start phase, if no packet loss is detected, the TCP sender is sending at a rate that is desired by its upper layer and as per the receiver's receiving capability as advertised by the receiver-window. When plotted over a long duration, this would be close to a straight line, as shown in Figure 8-3 after the slow-start phase and before the first packet loss was detected.

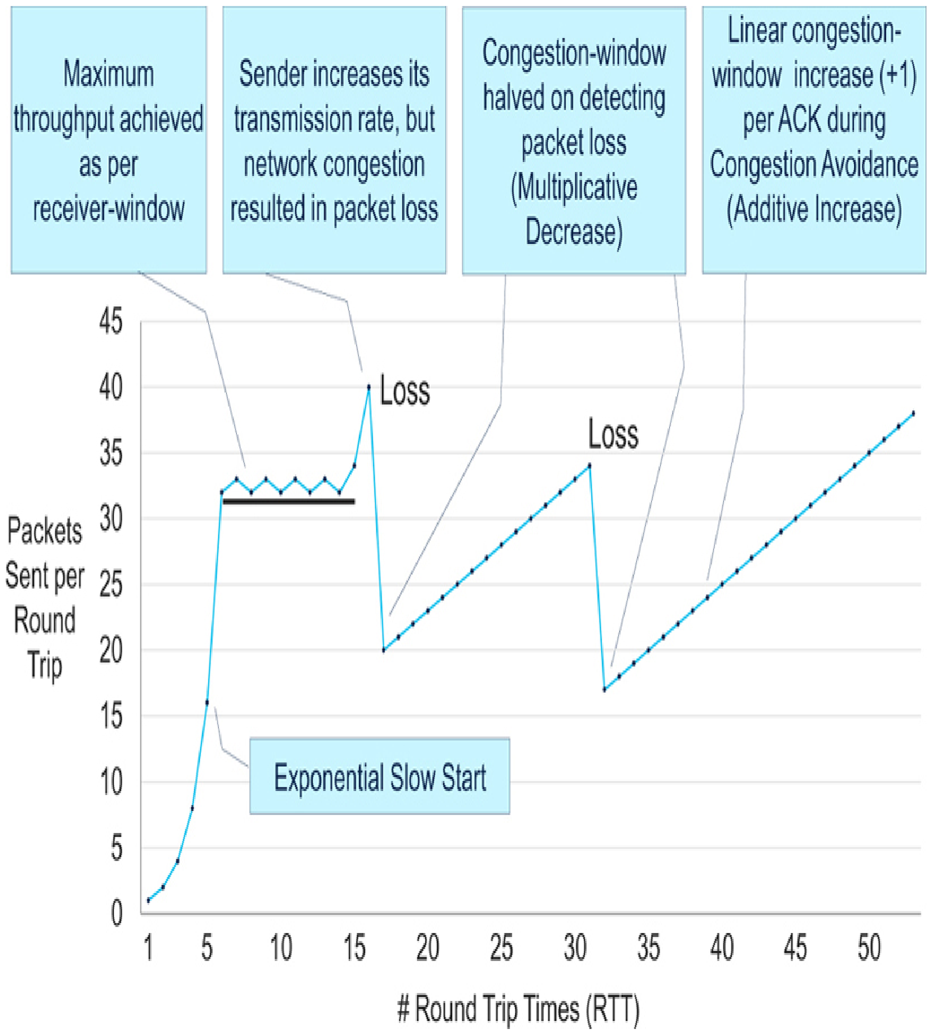


Figure 8-3 *TCP slow start and congestion avoidance with timeout-based retransmission*

Congestion Avoidance

When the sender's upper layer desires to send more data, TCP throughput increases assuming that the receiver-window is not a limiting factor. But this increased rate may cause network congestion leading to a packet drop. The sender reacts to this packet loss by reducing the congestion-window by half and backing off the RTO exponentially. In other words, during congestion, TCP reduces the volume of traffic exponentially and the rate of retransmission exponentially. Meanwhile, the sender retransmits the lost packet.

After reducing the congestion-window by half, the sender may again go through the slow-start phase to scale up the transmission. To avoid increasing the window size too quickly and causing additional congestion, TCP adds one additional restriction. Once the congestion-window reaches one-half of its original size before congestion, TCP enters a congestion avoidance phase and increments the congestion-window by one (not exponential) if all segments in the window have been acknowledged.

Because the congestion-window is reduced by half (multiplicative) under congestion, and increased by one (additive) after congestion, this technique for congestion avoidance is called additive-increase/multiplicative-decrease (AIMD).

The AIMD approach for congestion avoidance results in a plot that looks like a sawtooth. In Figure 8-3, this sawtooth pattern starts after the first packet loss at the 15th RTT. It continues only if the network allows less than the desired throughput of the TCP sender probably because of congestion. If the network is not a limiting factor, the plot would not be in a sawtooth pattern and would rather depend on the traffic pattern of the TCP sender.

Fast Retransmit/Fast Recovery

As the earlier section on *How TCP Provides Reliable Data Transfer* explains, instead of retransmitting a segment after RTO expiry, a sender may retransmit a segment after receiving three duplicate ACKs. This is called Fast Retransmit. When this happens, there is no need to go through the slow-start phase because receiving the duplicate ACKs indicate that the receiver is receiving the traffic, just that it might have lost one or more segments. Hence, the sender uses a fast recovery mechanism to quickly ramp up the throughput.

Comparison with Lossless Networks

Fibre Channel (explained in Chapter 2) and lossless Ethernet networks (explained in Chapter 7) do not use a slow-start mechanism. End devices can instantly send traffic on the network as fast as the upper layers (SCSI or NVMe) desire. This is favorable for storage traffic that is bursty and latency-sensitive. In data-center networks with low round-trip time and high-speed links (and even in general), the term slow-start may be a misnomer because the exponential ramp-up of the TCP traffic is typically very fast.

FC and FCoE do not have AIMD or similar approaches for controlling traffic rates on the end devices. Cisco MDS switches have a similar approach, called Dynamic Ingress Rate Limiting (DIRL), but it controls traffic rate at a per-port level, not per-flow level.

RoCEv2 Congestion Management (RCM) has a provision for the per-flow rate reduction, but that algorithm is not standardized and is implementation dependent. Refer to Chapter 7, the section on *RoCEv2 Congestion Management* for more details on it.

Congestion in TCP Storage Networks

After understanding how TCP provides reliable data transfer, flow control, and congestion control, let’s understand its implications in a storage network.

In Figure 8-4, Host-1 initiates large-size read I/O operations to many targets (only five targets are shown). The targets send data to Host-1 based on the amount of data requested by those read I/O operations. Excessive traffic destined for Host-1, which arrives at the same time (also known as TCP Incast), may result in two types of congestion.

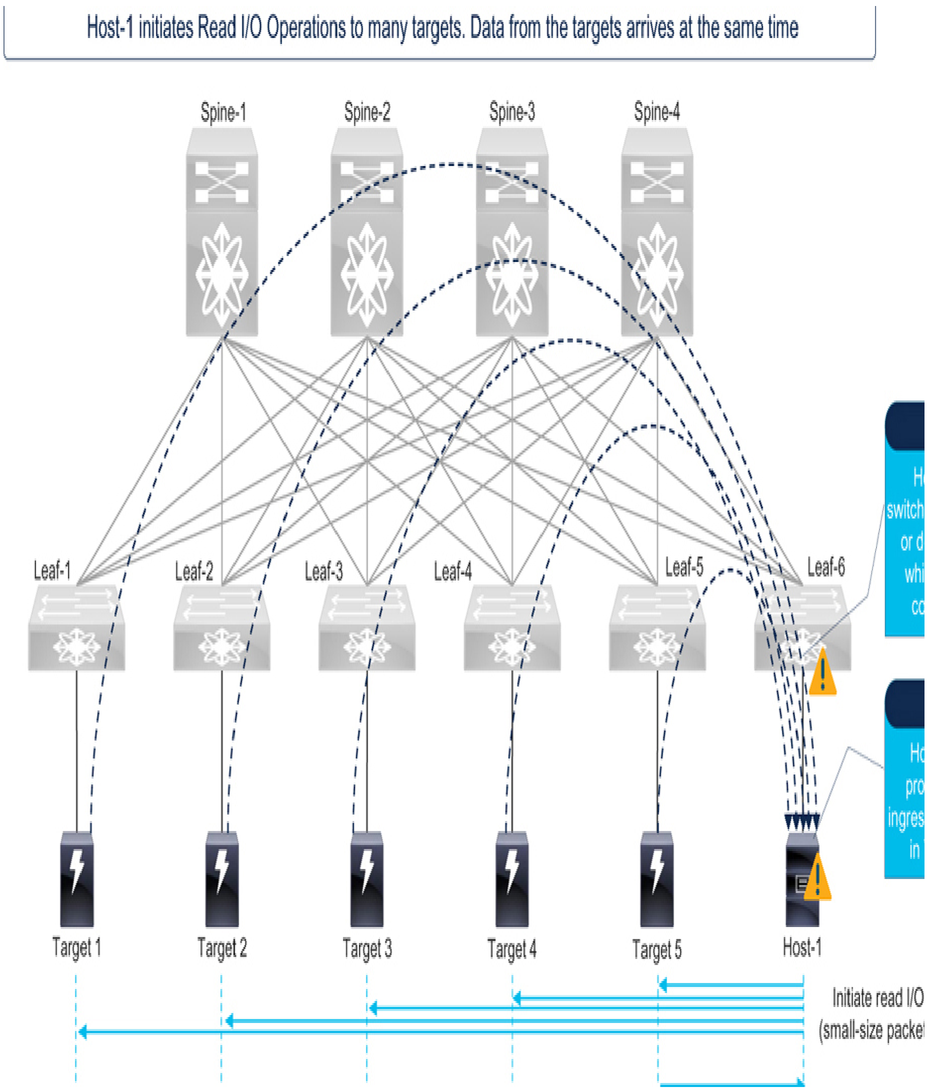


Figure 8-4 *Congestion in TCP Storage Networks*

Congestion Due to Over-utilization of the Host Link

Leaf-6 may receive more traffic from the spine switches than can be sent on the egress port to Host-1. These packets may be delayed while they wait in the buffers (queues) on Leaf-6 or even dropped if the queues are full. Excessive delay or packet drops will result in the targets retransmitting those packets and reducing their transmission rate as per the TCP congestion control mechanism. These retransmissions and rate reductions at the TCP layer degrade I/O performance.

Congestion Due to Over-utilization of the Target Link

The same problem may happen on a target/storage link when multiple hosts initiate large-size write I/O operations to the same target.

Comparison with Lossless Networks

A condition similar as Figure 8-4 would cause congestion also in Fibre Channel fabrics (explained in Chapter 2) and lossless Ethernet networks (explained in Chapter 7). Additionally, this would lead to congestion spreading Because Leaf-6 would invoke hop-by-hop flow control to slow down all the traffic from the spine switches on the link or within the no-drop traffic class. This congestion would spread to the source of the traffic (targets) victimizing many devices that share the same network path.

In contrast, congestion does not spread in lossy networks (with or without TCP). This is the biggest difference because other unrelated devices are not affected. Adverse effect is limited to the flows that pass through the congested port. Host-1 is affected partially because some flows may be delayed due to TCP retransmission. But it is still receiving traffic at the full capacity of the link.

The targets, that are sending traffic to Host-1, can send traffic to other hosts without being the victim of the over-utilization of the Host-1-connected switchport. Only the flows that pass through the over-utilized port are affected. To understand these differences further, compare this section to the section on *Congestion in a Lossless Spine-Leaf Network* in Chapter 7.

Congestion Within the Host

Host-1 may have a slower processing rate as compared to the collective rate of traffic sent by the targets. Its link may still have enough capacity, and hence, its switchport on Leaf-6 does not have to store the packets in its buffer any longer than the minimum requirement.

If the source of congestion is on the TCP or any upper layers within Host-1, its internal buffers may fill up. As a result, it advertises a smaller receiver-window to the targets, and therefore the TCP layer on the targets reduces their transmission rates. This is the standard flow control of TCP, which matches the rate of transmission on the targets with the receiving rate of Host-1.

Congestion Within the Targets

The same problem may happen on a target/storage device when multiple hosts initiate concurrent write I/O operations to the same target. Hence, the target (receiver) would advertise a smaller receiver-window to the hosts to reduce their transmission rate.

Comparison with Lossless Networks

In a lossless network, Host-1 is called a slow-drain device and it results in congestion spreading. It invokes the hop-by-hop flow control to slow down all the traffic on the link or in the no-drop class. This hop-by-hop congestion spreads to the traffic source victimizing many devices that share the same network path.

While using TCP, because the flow control is between the source and the destination, the effect is limited to only that TCP connection. Depending upon the source and the cause of congestion within a device, its effect may be limited only to that device or even limited to a few TCP connections while other TCP connections continue to transfer data perfectly well. Congestion does not spread hop-by-hop in a lossy Ethernet network, with or without TCP.

Also, in TCP storage networks, it would not be appropriate to call such devices slow-drain devices because congestion does not spread to the network, even though the root cause is the same. To understand these differences further, compare this section to the section on *Congestion in a Lossless Spine-Leaf Network* in Chapter 7.

Storage I/O Performance Monitoring

Traffic in a storage network is the direct result of an application initiating a read or write I/O operation. Because of this reason, network traffic patterns can be better understood by analyzing the application I/O profile, such as the timing, size, type, and rate of I/O operations. Essentially, the application I/O profile helps in understanding why the network has traffic or congestion.

Chapter 5, "Solving Congestion by Storage I/O Performance Monitoring," explains solving congestion by storage I/O performance monitoring. The same details, at least conceptually, also apply to iSCSI and NVMe/TCP (except the transport protocol differences) because the upper layers (SCSI and NVMe) are the same. Before proceeding, please refer to the following sections in Chapter 5, which are not repeated here for the sake of brevity.

- Chapter 5, the section on *Why Monitor Storage I/O Performance?* explains the basic value of monitoring storage I/O performance.
- Chapter 5, the section on *How and Where to Monitor Storage I/O Performance* explains three locations for storage I/O performance monitoring—hosts, storage arrays, or the network.
- Chapter 5, the section on *Cisco SAN Analytics* explains how Cisco MDS switches monitors storage I/O performance natively within the switches. This capability is called SAN Analytics and it is available only on the Fibre Channel ports. This section also helps in understanding why a similar capability does not exist on Ethernet switches. More on this shortly.
- Chapter 5, the section on *Understanding I/O Flows in a Storage Network* helps in understanding the difference between I/O flows in Fibre Channel fabrics versus TCP networks. Pay special attention to the sub-section on *I/O Flows versus I/O Operations*.
- Chapter 5, the section on *I/O Flow Metrics* helps in understanding how performance of I/O flows can be monitored using various metrics, such as I/O completion time (called Exchange Completion Time in Fibre Channel), IOPS, Throughput, and I/O size.

After understanding these sections from Chapter 5, next, be aware the performance of Ethernet networks carrying iSCSI and NVMe/TCP traffic can be monitored at the following levels:

1. **Port or Traffic Class**: Most end devices and switches report counters such as packets sent/received, bytes sent/received, and dropped packets on a network port or interface. If traffic is classified into multiple classes, these counters can be monitored for each class separately.
2. **Layer 4 (TCP or UDP) Flow**: Flows at layer 4 of the OSI model are identified by 5-tuples—Source IP, Destination IP, Source port, Destination port, and layer-4 protocol (TCP, UDP, etc.). In other words, all iSCSI and NVMe/TCP traffic within a TCP connection belongs to the same TCP flow. Counters such as packets sent/received, bytes sent/received, and dropped packets for each TCP flow can be monitored separately depending upon the capability of the devices in a network.
3. **I/O Flow**: An I/O flow is aware of SCSI or NVMe I/O operations to a storage volume (logical unit for SCSI and Namespace for NVMe). Monitoring performance at an I/O flow level allows calculating how long I/O Operations are taking to complete, their throughput, IOPS, type (read or write), I/O size, etc.

TCP Flow Monitoring versus I/O Flow Monitoring

TCP flow monitoring shouldn’t be confused with I/O flow monitoring because of the following reasons:

1. TCP belongs to the transport layer (layer 4) of the OSI model. It is not aware of the functions of the upper layer protocols such as SCSI and NVMe.
2. Many I/O operations may travel in a single TCP flow. These I/O operations may belong to different I/O flows.

Refer to Chapter 5, the section on *I/O Flows versus I/O Operations*.

3. As mentioned, a TCP flow has its own performance monitoring metrics, such as packets transferred per second, throughput, number of active TCP connections, time taken to establish a TCP connection, etc. This is different from the performance monitoring of I/O flows, such as IOPS, I/O throughput, time taken to complete I/O operations, I/O size, etc.

Unavailability of I/O Flow Monitoring in TCP Storage Networks

At the time of writing, performance monitoring of I/O flows is not available in TCP/IP networks. Ethernet switches may report network latency, which is typically the time spent by a packet in the network. This is not the I/O completion time. Likewise, Ethernet switches may report the throughput of a TCP flow, but this is not read or write I/O throughput or IOPS.

Although the IP addresses of the initiator and the target can be considered to make an IT flow, this flow definition is of little value without knowing the I/O flow metrics.

Even in the future, it is unlikely that Ethernet switches will be able to monitor I/O performance similar to SAN Analytics on Fibre Channel ports on Cisco MDS switches. This is because TCP supports thousands of upper layer protocols each with different port numbers. Enhancing the Ethernet switches to decode just the storage protocols (iSCSI, NVMe/TCP, etc.) and measuring their performance may not justify the additional cost for making these enhancements. This was not a challenge for Cisco MDS switches because Fibre Channel is purposefully built for storage traffic. In contrast, Ethernet networks carry all kinds of traffic, as explained by Figure 1-11 in Chapter 1.

Alternative Approaches

An alternative approach is to monitor storage I/O performance within the hosts or the storage arrays.

Some Ethernet switches (such as Cisco Nexus 9000 switches) monitor the performance of TCP flows. But, as mentioned, this is not the performance monitoring of the I/O flows, which are carried within those TCP flows. Hence, use Ethernet switches for TCP flow monitoring together with I/O flow performance monitoring on hosts and storage arrays. For more details, refer to Chapter 5, the section on *Storage I/O Performance Monitoring in the Host* and *Storage I/O Performance Monitoring in the Storage Array*.

Some monitoring applications in the market can correlate the TCP flow performance data from the network and I/O flow performance data from the hosts and storage arrays. These applications are promising because they gather data where it is available and then focus on solving problems. They are often marketed using AIOps, AI/ML, or similar buzz words. Some examples are listed in the references section. This is not an endorsement, rather an encouragement for finding the right solution or even building your own solution for storage I/O performance monitoring in TCP storage networks.

The later section on *Estimating I/O Performance from TCP Performance* explain another approach that relies on the traffic pattern due to I/O operations in a network.

iSCSI I/O Operations

Figure 8-5 shows iSCSI Read I/O Operation. A host (initiator) initiates a read I/O operation over a TCP connection to the target. The target sends the data to the host in one or more packets based on the amount of data requested by the I/O Operation (I/O size), the Maximum Transmission Unit (MTU) of the network, and the Maximum Segment Size (MSS) of the TCP connection (More details on MTU and MSS later). Finally, the I/O operation completes when the target sends a Response.

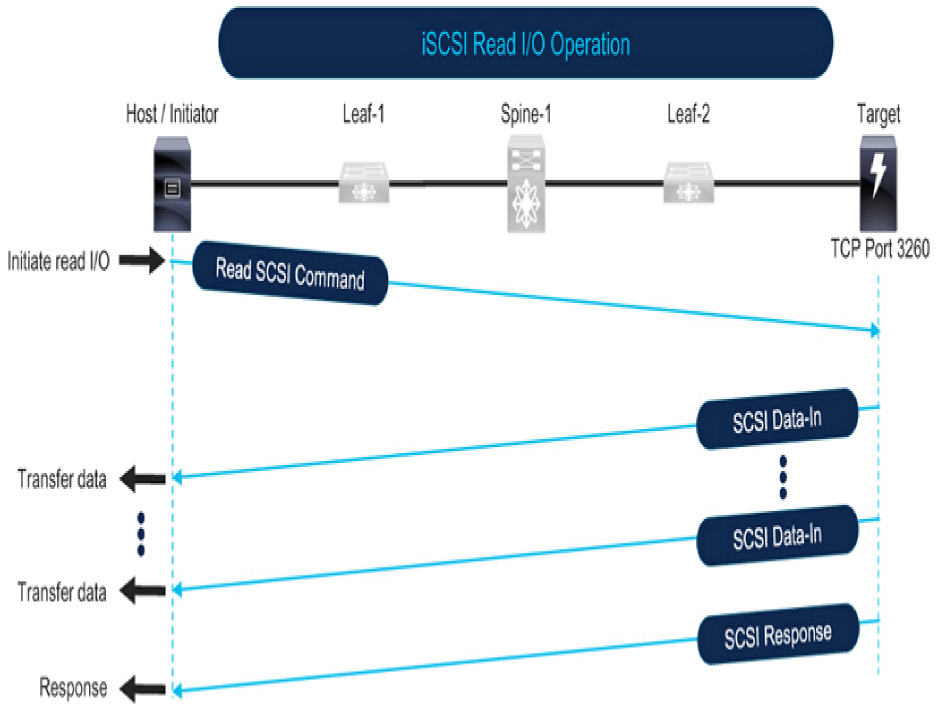


Figure 8-5 *iSCSI Read I/O Operation*

Figure 8-6 shows iSCSI Write I/O Operation. A host (initiator) initiates a write I/O operation over a TCP connection to the target. The target sends a Ready2Transfer (R2T). This is optional. The initiator sends data to the target in one or more packets based on the amount of data requested by the I/O Operation (I/O size), the MTU of the network, and the MSS of the TCP connection. Finally, the I/O operation completes when the target sends a Response.

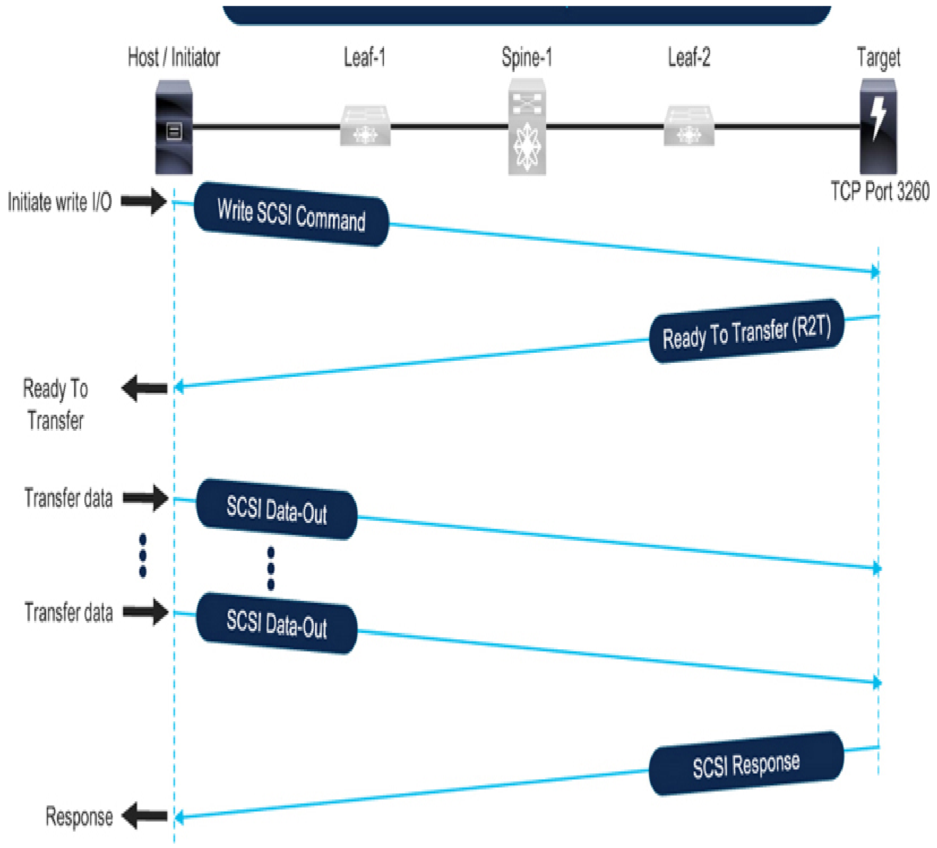
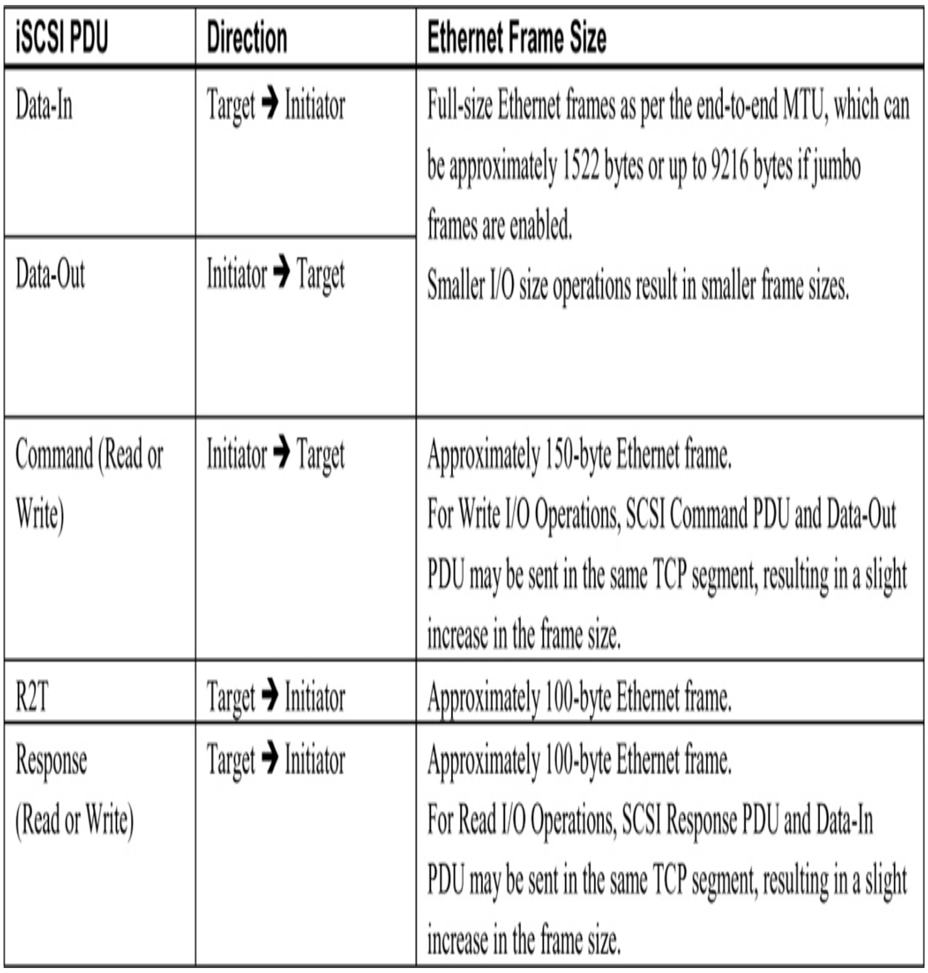


Figure 8-6 *iSCSI Write I/O Operation*

As explained in the earlier section on *How iSCSI and NVMe/TCP Exchange Data*, SCSI Command, R2T, Data-In, Data-Out, and Response are called PDUs at the iSCSI layer. The TCP layer divides them into segments and sends them to the lower layers for adding IP and Ethernet headers before transmitting the frame on the network.

Table 8-1 shows the Ethernet frame sizes and their directions based on the type of iSCSI PDU carried by a frame.

Table 8-1 *Typical size of Ethernet frames for iSCSI read and write I/O operations*



iSCSI PDU	Direction	Ethernet Frame Size
Data-In	Target ➔ Initiator	Full-size Ethernet frames as per the end-to-end MTU, which can be approximately 1522 bytes or up to 9216 bytes if jumbo frames are enabled. Smaller I/O size operations result in smaller frame sizes.
Data-Out	Initiator ➔ Target	
Command (Read or Write)	Initiator ➔ Target	Approximately 150-byte Ethernet frame. For Write I/O Operations, SCSI Command PDU and Data-Out PDU may be sent in the same TCP segment, resulting in a slight increase in the frame size.
R2T	Target ➔ Initiator	Approximately 100-byte Ethernet frame.
Response (Read or Write)	Target ➔ Initiator	Approximately 100-byte Ethernet frame. For Read I/O Operations, SCSI Response PDU and Data-In PDU may be sent in the same TCP segment, resulting in a slight increase in the frame size.

NVMe/TCP I/O Operations

Figure 8-7 shows NVMe/TCP Read I/O Operation. A host (initiator) initiates a read I/O operation over a TCP connection to the target, which is called a controller in NVMe terminology. The target sends data to the host in one or more packets based on the amount of data requested by the I/O Operation (I/O size), the MTU of the network, and the MSS of the TCP connection. Finally, the I/O operation completes when the target sends a Response.

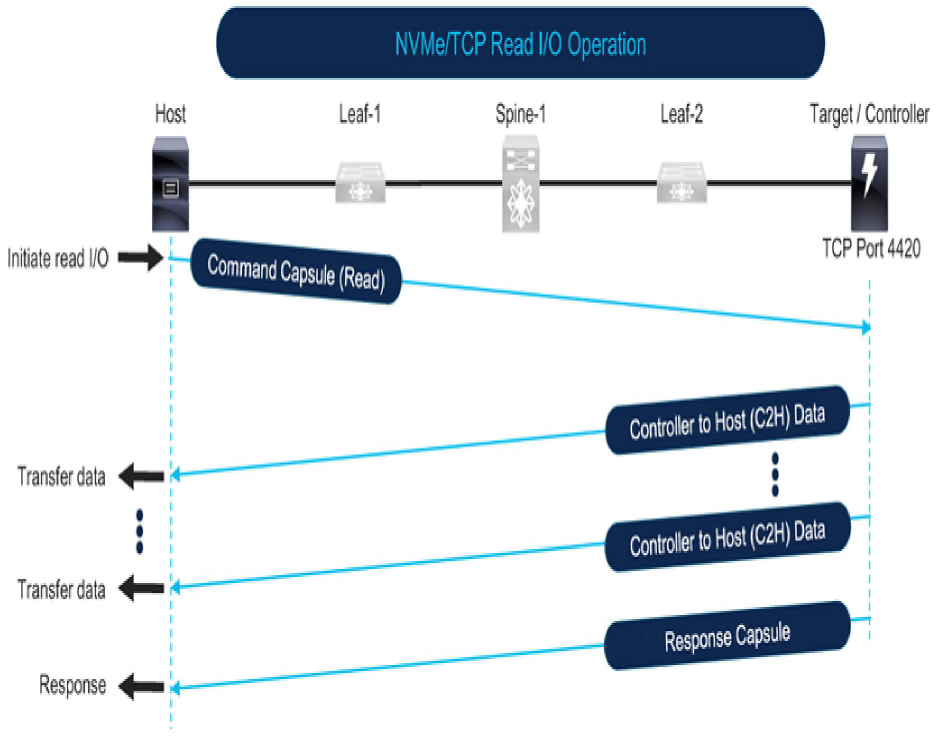


Figure 8-7 *NVMe/TCP Read I/O Operation*

Figure 8-8 shows NVMe/TCP Write I/O Operation. A host (initiator) initiates a write I/O operation over a TCP connection to the target or controller. The target sends a Ready2Transfer (R2T). This is optional. The initiator sends data to the target in one or more packets based on the amount of data requested by the I/O Operation (I/O size), the MTU of the network, and the MSS of the TCP connection. Finally, the I/O operation completes when the target sends a Response.

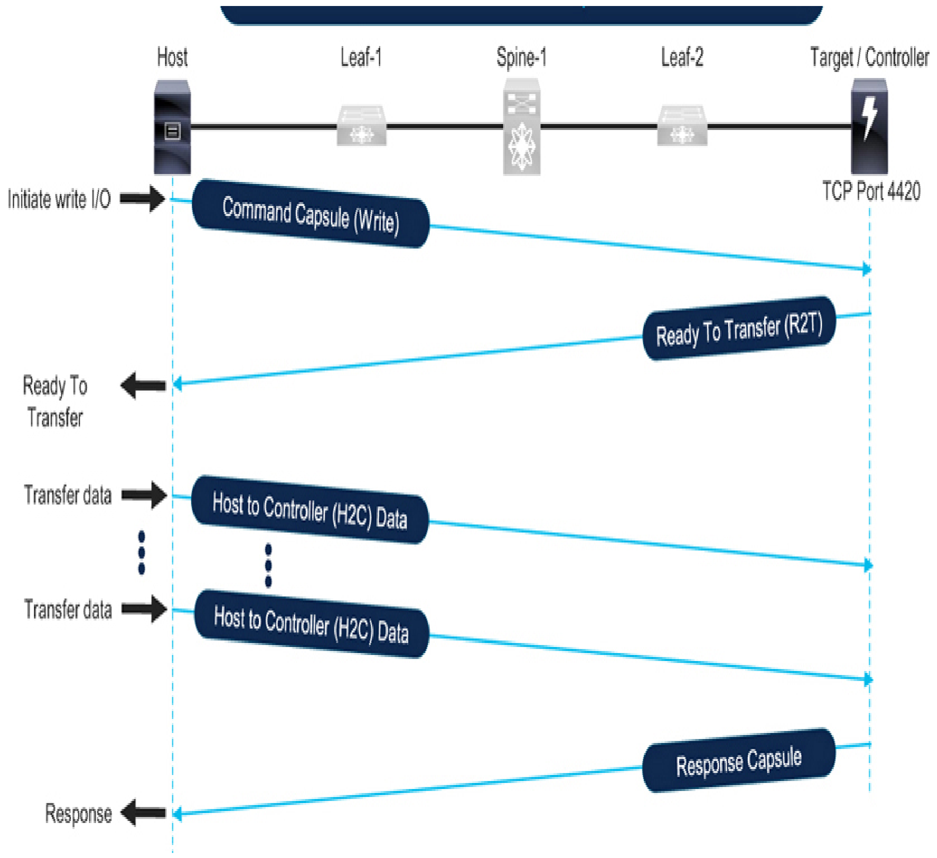
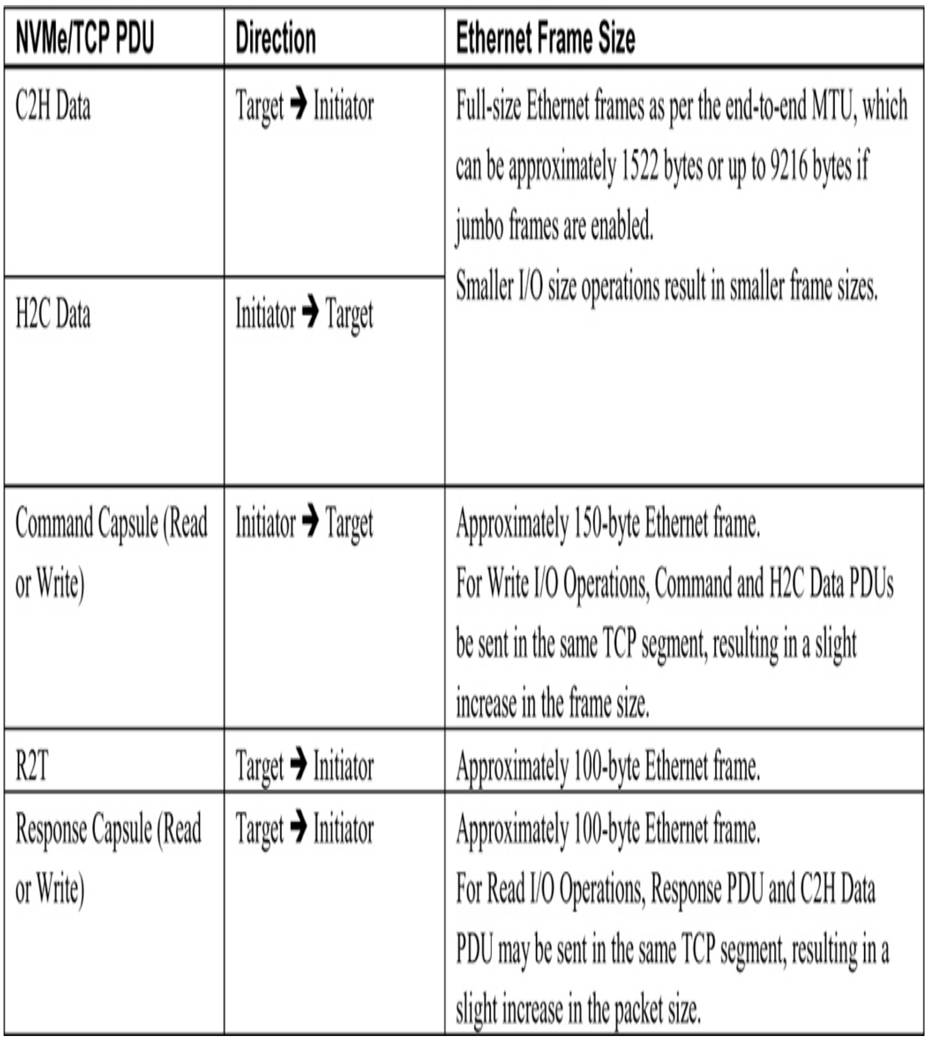


Figure 8-8 *NVMe/TCP Write I/O Operation*

As explained in the earlier section on *How iSCSI and NVMe/TCP Exchange Data*, Command, R2T, C2H Data, H2C Data, and Response are called PDUs at the NVMe/TCP layer. The TCP layer divides them into segments and sends them to the lower layers for adding IP and Ethernet headers before transmitting the frame on the network.

Table 8-2 shows the Ethernet frame sizes and their directions based on the type of NVMe/TCP PDUs carried by an Ethernet frame.

Table 8-2 *Typical size of Ethernet frames for NVMe/TCP read and write I/O operations*



NVMe/TCP PDU	Direction	Ethernet Frame Size
C2H Data	Target ➔ Initiator	Full-size Ethernet frames as per the end-to-end MTU, which can be approximately 1522 bytes or up to 9216 bytes if jumbo frames are enabled. Smaller I/O size operations result in smaller frame sizes.
H2C Data	Initiator ➔ Target	
Command Capsule (Read or Write)	Initiator ➔ Target	Approximately 150-byte Ethernet frame. For Write I/O Operations, Command and H2C Data PDUs be sent in the same TCP segment, resulting in a slight increase in the frame size.
R2T	Target ➔ Initiator	Approximately 100-byte Ethernet frame.
Response Capsule (Read or Write)	Target ➔ Initiator	Approximately 100-byte Ethernet frame. For Read I/O Operations, Response PDU and C2H Data PDU may be sent in the same TCP segment, resulting in a slight increase in the packet size.

Note the following points:

1. At the network layer, iSCSI and NVMe/TCP I/O operations are similar. These are TCP/IP packets with different port numbers. This is by design and explains why a general-purpose TCP/IP network can transport both these protocols simultaneously.
2. iSCSI and NVMe/TCP I/O Operations with the same I/O size results in similar-sized packets.
3. For a read I/O operation, Response PDU may be sent in the same packet as the last Data, if enough space is available in the data packet. In other words, for an iSCSI session, the same packet may carry SCSI Data-In PDU and SCSI Response PDU. For NVMe/TCP connection, the same packet may carry C2H Data PDU and Response PDU. Recall from Chapter 5, the section on *Read I/O Operation*, Fibre Channel has similar optimizations.

4. For a write I/O operation, there may not be a dedicated packet carrying R2T, hence, the first data may be sent in the same packet as the Command PDU. For iSCSI, if the initiator and target support ImmediateData (enabled by default), the initiator can send the first burst without waiting for an R2T. Likewise, for NVMe/TCP, when the connection is established, the controller specifies the maximum number of R2T (MAXR2T). Based on this MAXR2T value, the host may send the first burst without waiting for R2T from the controller. Recall from Chapter 5, the section on *Write I/O Operation*, Fibre Channel has similar optimizations for First Burst.

Correlating I/O Operations, Traffic Patterns, and Network Congestion

Please refer to the following sections in Chapter 5, which are not repeated here for the sake of brevity.

- Compare the section on *I/O Operations and Network Traffic Patterns* in Chapter 5 with the previous section and notice the striking similarities between SCSI and NVMe I/O operations via Fibre Channel and TCP. Because of this reason, they lead to similar traffic patterns and have a similar correlation with network congestion.
- Chapter 5, the section on *Network Traffic Direction* explains traffic on various port types because of read and write I/O operations.
- Chapter 5, the section on *Correlating I/O Operations, Traffic Patterns, and Network Congestion* explains that the key reason for congestion on a host link is the multiple concurrent large-size read I/O operations from that host. Likewise, the key reason for congestion on a storage link is the total amount of data being requested by that storage array via the R2T PDUs.

Comparison with Lossless Networks

As mentioned, the iSCSI and NVMe/TCP I/O Operations are very similar to the SCSI and NVMe I/O operations in a Fibre Channel fabric. The frame sizes are similar and their direction is the same, hence, their effect on the network performance and congestion is the same.

In comparison with RDMA-capable protocols (RoCE, RoCEv2, and iWARP), the frame types are different, but the network traffic patterns are the same. This is because the upper layers (SCSI and NVMe) define the traffic profile, which are the same regardless of different transport types.

Estimating I/O Flow Performance from TCP Flow Performance

As mentioned earlier, at the time of this writing, performance monitoring of I/O flows is not available in Ethernet switches. However, some switches (such as Cisco Nexus 9000) can monitor the performance of TCP flows. Based on the TCP port numbers and the direction of high throughput flows, I/O performance can be estimated.

Remember that with iSCSI and NVMe/TCP, generally the host/initiator initiates a TCP connection to the target/controller for carrying I/O operations. As a result, packets with destination TCP port numbers 3260 (for iSCSI) and 4420 (for NVMe/TCP) are destined for the target/controller. Likewise, packets with source TCP port numbers 3260 (for iSCSI) and 4420 (for NVMe/TCP) are destined for the host/initiator. Because network switches may not inspect the traffic any further than layer 4, it is not possible to find if the packet belongs to a read or a write I/O operation because this information is available only with the header of the upper layer protocols. This issue can be solved by using the port numbers in the TCP header to find the destination of the traffic and analyzing the throughput per TCP flow (different from I/O flow) as explained in the earlier section on *Correlating I/O Operations, Traffic Patterns, and Network Congestion*.

- A high-throughput TCP flow with destination TCP port numbers 3260 (for iSCSI) and 4420 (for NVMe/TCP) is mostly write I/O traffic.
- Likewise, high-throughput TCP flow with source TCP port numbers 3260 (for iSCSI) and 4420 (for NVMe/TCP) is mostly read I/O traffic.

Be aware that some implementations may negotiate different port numbers after the initial discovery. In such cases, traffic for I/O operations may not be carried via TCP connections with standard port numbers. Refer to the product documentation to know the use of any non-standard port numbers and then monitor the throughput for those TCP flows.

Finally, be aware that only the throughput of read and write I/O flows can be estimated using the approach explained in this section. Other metrics, such as IOPS and I/O completion time, require alternative approaches as explained earlier in the section on *TCP Flow Monitoring versus I/O Flow Monitoring*.

IP MTU and TCP MSS Considerations

The Maximum Transmission Unit (MTU) is the maximum amount of data allowed in a packet or frame.

At layer 2, Ethernet has a typical MTU of 1500 bytes. After adding 22 bytes for the header, the Ethernet frame size becomes 1522 bytes (Chapter 1, Figure 1-6).

At layer 3, the IP packet must be less than or equal to 1500 bytes as per Ethernet MTU. The IPv4 header is 20 bytes (Figure 8-9), which results in an IP MTU of 1480 bytes.

At layer 4, TCP negotiates a maximum segment size (MSS) between the endpoints when the connection is established. As its name suggests, MSS is the maximum size of a TCP segment. Because the TCP header is 20 bytes and IP MTU is 1480 bytes, TCP MSS is typically 1460 bytes.

Number of Packets for an I/O Operation

TCP segments carry upper-layer headers, such as iSCSI and NVMe/TCP. The data for an I/O Operation may be sent in one or more packets based on its I/O size, the MTU of the network, and TCP MSS. For example, an I/O size of 1K and MSS of 1460 bytes may send one data packet for a read I/O operation. If enough space is left within the TCP segment, the SCSI or NVMe Response may also be sent within the same TCP segment. This eliminates a dedicated packet just for carrying the response PDU.

If the I/O size is 8K, one segment of 1460 bytes is not enough. Hence, multiple packets are sent.

If the MTU on the network links is increased to 9000 bytes and the TCP connection uses a larger MSS, a single large-size packet can transfer the whole 8K worth of data and possibly also send the SCSI or NVMe Response in the same packet.

Packet Fragmentation

With jumbo packets, the MTU must be correctly set on all the links in the end-to-end data path. If one of the links has a smaller MTU, the packets must be fragmented at the IP layer. The destination must reassemble multiple fragments, which typically is a resource-intensive task resulting in performance degradation.

If the IP header has a Don't Fragment (DF) flag set, instead of fragmentation, these packets may be dropped. When this occurs an ICMP unreachable is normally sent back to the sender indicating the MTU of the link the packet could not traverse but depending on where this is being done may not occur. Such conditions are hard to detect and troubleshoot because the packets that initiate the I/O operations are small-sized, and hence, they reach the destination without any issues. But the I/O operations may never complete because only the large-size data packets are dropped. At the same time, I/O operations with a small I/O size may complete. Another peculiar issue happens when the MTU is not consistently changed on both ends of a link. Only one type of I/O operation (read or write) may be affected because their data frames that are large-sized travel in opposite directions.

Such problems can be detected by the packet drop/discard counters on the network ports. Also, modern TCP implementations may use Path MTU Discovery to automatically detect the least MTU in the end-to-end data path and adjust their MSS automatically. This avoids packet fragmentation, but if a network link has a lower MTU, smaller size TCP segments may limit the performance.

Comparison with Lossless Networks

As Chapter 2 explains, Fibre Channel defines a maximum frame size of 2148 bytes. Users do not have to change the MTU value, and hence, it is never a consideration.

FCoE networks carry an FC frame within an Ethernet frame. Hence, the MTU should be increased to approximately 2300 bytes.

RoCEv2 networks have the same MTU considerations as TCP networks except for the MSS which is relevant only for TCP. The MTU on all the links in the end-to-end data path must be configured as per the maximum size of the packet send by the end devices to avoid fragmentation at the IP layer.

Preventing Congestion in TCP Storage Networks

Unlike earlier chapters, this chapter explains congestion prevention before detection and troubleshooting mechanisms because TCP's congestion control mechanisms are built in. Other mechanisms, like Explicit Congestion Notification (ECN), are standardized and well-adopted in the production networks.

This book uses the term "congestion prevention" as an overall approach to eliminating or reducing congestion in a network so that application performance is acceptable. Do not confuse it with the "congestion control" and "congestion avoidance" mechanisms in TCP. These mechanisms slow down the sender to avoid packet drops, but the real cause of the problem must still be investigated and solved.

Eliminating or Reducing Congestion — An Overview

Chapter 6, "Preventing Congestion in Fibre Channel Fabrics," and Chapter 7, "Congestion Management in Ethernet Storage Networks," explain the high-level approaches to eliminating or reducing congestion in Fibre Channel fabrics and lossless Ethernet networks respectively. This section explains why or why not those approaches are effective in TCP storage networks.

- **Disconnect a culprit device**: In lossless networks, disconnecting a culprit (slow-drain) device eliminates the source of congestion, and therefore results in recovering the network from congestion.

  In TCP storage networks, a slow device uses end-to-end flow control to advertise a smaller receiver-window, which reduces the transmission rate of the sender. This affects only the specific flows to and from the slow device. Because congestion does not spread hop-by-hop in lossy networks, and hence does not affect many other unrelated devices, disconnecting a slow device is not needed for preventing congestion. However, disconnecting a link may still be considered when it has lower-level issues, such as bit errors and low transceiver receive power so that this degraded link does not affect traffic while the investigation continues, and a solution is applied. Refer to Chapter 6, the section on *Congestion Recovery by Disconnecting the Culprit Device* for additional considerations and how to disconnect, if needed.

- **Early dropping of frames**: Lossless networks are not supposed to drop frames. However, under congestion, they may use controlled mechanisms to drop the frames to free up the buffers and reduce backpressure on the entire link or within the no-drop traffic class. Chapter 6, the section on *Congestion Recovery by Dropping Frames* explains such mechanisms in Fibre Channel fabrics (congestion-drop timeout and no-credit-drop timeout).

Chapter 7, the section on *Congestion Recovery by Dropping Frames* explains such mechanisms in lossless Ethernet networks (Pause timeout and PFC watchdog).

Lossy networks (with or without TCP), by default, drop frames under congestion. In fact, packet loss is the most common trigger for TCP to invoke congestion control. Hence, it can be considered that dropping the packets is desirable for TCP, although not entirely correct for storage traffic because the I/O performance is degraded.

- **Traffic Segregation**: In lossless networks, traffic can be segregated into dedicated links or virtual links, or no-drop classes, while each virtual link or no-drop class has dedicated buffers and uses a dedicated hop-by-hop flow control. As a result, congestion in one virtual link or no-drop class does not affect traffic in other virtual links or classes. Chapter 6, the section on *Traffic Segregation* explains this mechanism in Fibre Channel fabrics. Chapter 7, the section on *Multiple no-drop Classes on the Same Link* explains this mechanism in lossless Ethernet networks.

  TCP uses flow control and congestion control mechanisms dedicated to each connection. These are enabled by default and no user action is needed. Traffic of each TCP connection is segregated for the purpose of end-to-end flow control. However, the lower layers (such as the Ethernet link layer) on the end devices and the network links are still shared among all the TCP flows. As a result, congestion caused by a TCP flow may still affect other flows sharing the same network path. For example, excessive traffic on a TCP flow may fill up the switch buffers leading to packet drops even for other TCP flows on the network port. For the purpose of network congestion, traffic can be segregated using QoS. Refer to the section on *QoS Considerations for Dedicated and Shared Storage Networks* for more details.

- **Notifying the end devices about congestion**: Notifying the end devices about congestion allows them to take preventive actions such as lowering their traffic rate.

Different transport types have different implementations of this approach with different degrees of readiness and adoption. Chapter 6, the section on Congestion *Prevention by Notifying the End Devices* provides an overview of these approaches followed by a detailed explanation of its implementation in Fibre Channel fabrics using Fabric Performance Impact Notifications (FPIN) and Congestion Signals. Chapter 7 explains the implementation of this approach for RoCEv2 Congestion Management (RCM).

TCP/IP networks notify the end devices about congestion using Explicit Congestion Notification (ECN). TCP reacts to ECN the same way it reacts to packet loss. ECN results in invoking the TCP congestion control mechanisms to reduce the transmission rate. But ECN may not be enabled by default and requires refining its thresholds. The following sections explain these details.

- **Limiting the traffic going to a congested device**: Limiting the traffic going to a congested device or an over-utilized link can eliminate congestion.
  - Chapter 6, the section on *Congestion Prevention using Rate Limiters on Storage Arrays* explains the use of this approach in Fibre Channel fabric. The same approaches can work also in TCP storage networks based on the capability of the end devices. Note that although TCP has robust built-in mechanisms for flow control and congestion control, this approach can avoid packet drops by avoiding sending too much traffic via the network.
  - Chapter 6, the section on *Congestion Prevention using Dynamic Ingress Rate Limiting* (DIRL) explains how Cisco MDS switches can detect congestion and dynamically adapt the traffic rate to the culprit devices in Fibre Channel fabrics. In TCP storage networks, theoretically, the approach of DIRL can be complementary to the built-in congestion control mechanism. However, no such implementations exist at the time of this writing.

- **Re-designing the network**: Re-designing a network can eliminate or reduce the severity of congestion. For example, converting large storage fabrics having thousands of devices to smaller islands limits the effect of culprit devices within the island. Chapter 6, the section on *Network Design Considerations* provides examples of Fibre Channel fabrics. In TCP storage networks, those examples may not apply directly because congestion does not spread in lossy networks (with or without TCP) and TCP is proven to operate at the scale of the Internet. Network re-design for TCP storage networks typically focuses on having a shared versus dedicated network. Refer to the later section on *QoS Considerations for Dedicated and Shared Storage Networks* for further details.
- **Upgrade**: When a link is highly utilized, adding more capacity is the only solution regardless of the transport type—Fibre Channel, Lossless Ethernet, or TCP. Because TCP already handles many other congestion issues by default, the most common action for a user boils down to upgrading the network capacity. In some cases, links may have non-uniform utilization leading to congestion on only a few links. These details are explained later in the section on *Load-balancing in TCP Storage Networks*.

Congestion Notification in TCP Storage Networks

Congestion Notification in TCP storage networks is achieved by Explicit Congestion Notification (ECN), which is standardized by RFC 3168. As explained earlier, iSCSI and NVMe/TCP layers remain unaware of any such mechanisms.

Solution Components

The success of the approach to congestion prevention by notifying the end devices depends upon the following factors:

1. **Congestion detection**: A switch (or router) must be able to detect the symptoms of congestion.

2. **Sending a notification**: After detecting congestion, the switch must be able to inform the end devices about it.
3. **Receiving a notification**: The end devices must be able to understand the notification from the switches.
4. **Congestion prevention action**: After being notified about congestion in the network the most important component is for the end devices to take preventive actions, such as reducing their rate.

These are the same components as for Fibre Channel fabrics (explained in Chapter 6) and RoCEv2 networks (explained in Chapter 7). The following sections explain Explicit Congestion Notification in TCP/IP networks using the same components to make it easy to compare the implementation among different transport types and for the users to have a consistent detection and troubleshooting workflow.

Explicit Congestion Notification in TCP/IP Networks

When a switch detects congestion, it flags it in the ECN field in the IP header (Figure 8-9). The destination TCP endpoint receives the ECN-flagged packet and reflects this information to the source, which reduces its traffic rate by invoking congestion control as if a packet had been lost.

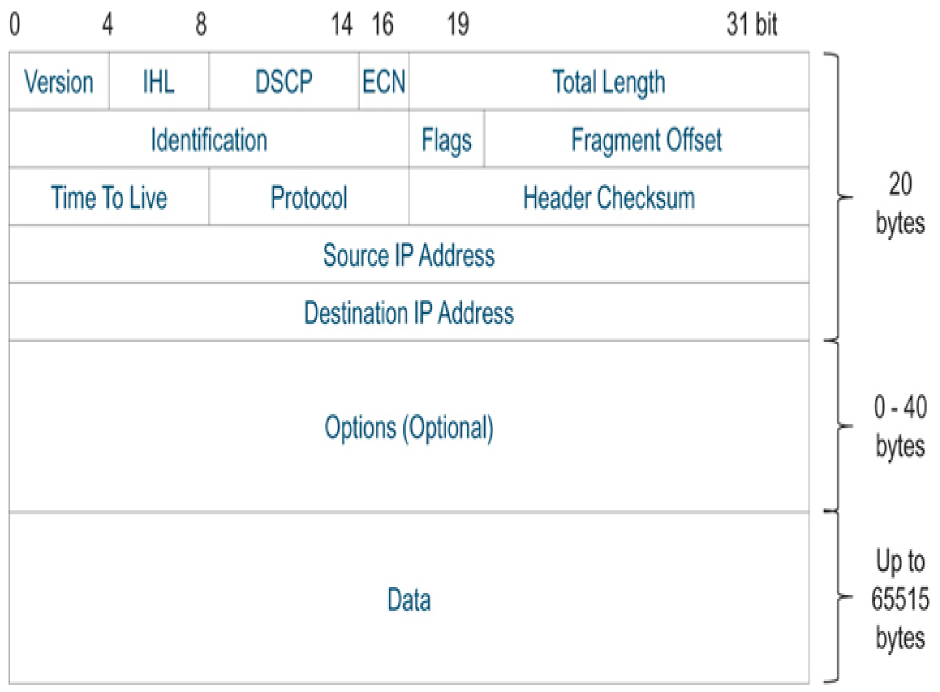


Figure 8-9 *IPv4 packet and header format*

Refer to Figure 8-10, which is a subset of the spine-leaf network explained in Figure 8-4. The following are the high-level steps:

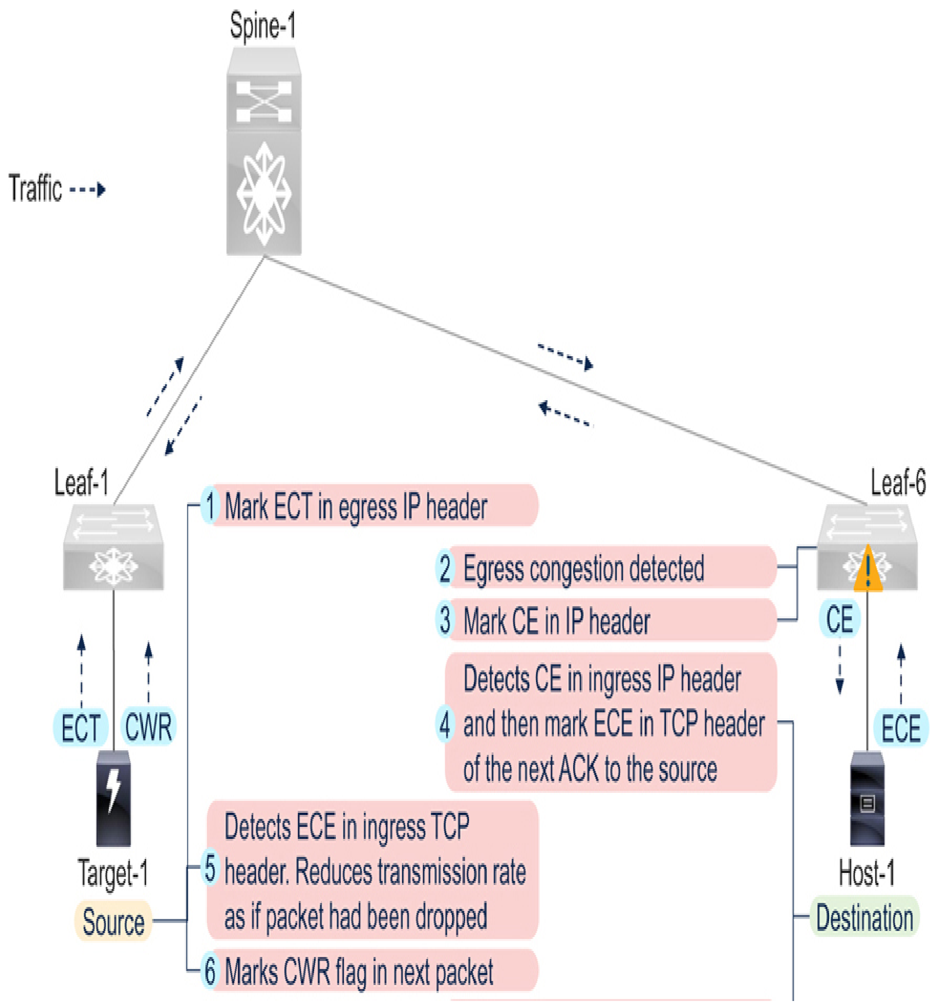

Spine-1
Traffic
Leaf-1
Leaf-6
1 Mark ECT in egress IP header
2 Egress congestion detected
3 Mark CE in IP header
4 Detects CE in ingress IP header and then mark ECE in TCP header of the next ACK to the source
5 Detects ECE in ingress TCP header. Reduces transmission rate as if packet had been dropped
6 Marks CWR flag in next packet
ECT
CWR
CE
ECE
Target-1
Source
Host-1
Destination

Figure 8-10 *Explicit Congestion Notification in TCP/IP Networks*

1. **Willingness to use or not use ECN**: When Target-1 and Host-1 establish a TCP connection (layer 4), they also exchange their willingness to use ECN. If they agree on using ECN, the source marks the ECN-capable Transport (ECT) flag in the IPv4 or IPv6 packets (layer 3) that are sent via this TCP connection. The ECT flag is set with b‘01’ or b‘10’ in the ECN field in the IP header. A value of b‘00’ in the ECN field indicates that the TCP endpoints are not ECN-capable.

2. **Congestion detection**: A switch (Leaf-6) in the end-to-end data path detects congestion when its queue utilization exceeds a configured threshold. The queue utilization increases because the collective ingress rate on one or more ports is faster than the egress rate. More details on this are explained in the *Queue Utilization* section.
3. **Sending a notification**: When the switch detects congestion, it marks the Congestion Experience (CE) flag in the IPv4 or IPv6 header before its queues are full. The CE flag is set with b‘11’ in the ECN field. The congested switchport sets CE flag only for those packets that have the ECT flag (b‘01’ or b‘10’) enabled. Non-ECN-capable packets (ECN value b‘00’) are forwarded unchanged or may even be dropped as per the configuration. Note that the switch does not guarantee that the ECN-capable packets are not dropped. As explained later in section *Queue Utilization*, if the queues become full, any new incoming packets are dropped regardless of their ECN value.
4. **Receiving a notification**: The destination (Host-1) detects network congestion when it receives a packet with the CE flag (ECN field b‘11’). Host-1 need not have any special mechanism to interpret a new kind of packet because the CE flag is part of the standard IP header. However, it must be able to detect and act on the CE flag. When Host-1 receives a CE-flagged packet, it reflects congestion to the source by marking the ECE (ECN-Echo) flag in the TCP header (Figure 8-2) of the next ACK that it sends to the source (Target-1). Because these ACKs are part of the standard TCP, no special packet is needed to reflect the congestion from the destination to the source. Note that the destination learns about network congestion using the CE flag in the IP header (layer 3), but it reflects this to the source using the ECE flag in the TCP header (layer 4).
5. **Congestion prevention action**: The source (Target-1) receives packets with ECE flag in the TCP header. As a result, it reduces the traffic rate of the TCP connection

using the TCP congestion control mechanisms as if the packet had been lost.

6. **Acknowledge receipt by source**: Target-1 marks the CWR (Congestion Window Reduced) flag in the TCP header (Figure 8-2) of the next packet to Host-1 to acknowledge its receipt and reaction to the ECE flag.
7. **Stop notifying the source**: When Host-1 receives packets with the CWR flag, it stops marking egress packets with the ECE flag. Further, if another CE packet is received, the receiver would once again send ACK packet with the ECE flag set.

Comparison with RoCEv2 Networks

RoCEv2 Congestion Management (explained in Chapter 7) uses the same mechanism as ECN in TCP/IP networks. Compare Figure 8-103 with Figure 7-17 in Chapter 7. You can notice that Steps 1 to 4 are the same. The difference is after a destination receives a CE-marked IP packet.

The following are the notable differences between the two approaches:

1. **Notification to the traffic source**: RoCEv2 uses UDP as the layer 4 transport. UDP does not have a concept of a connection. Hence, the upper layer (RoCEv2) must send its dedicated Congestion Notification Packet (CNP) for reflecting the congestion information from the traffic destination to the source. In contrast, TCP uses the ECE flag in its standard header.
2. **Rate-reduction:** UDP does not have a congestion control mechanism. Hence, the upper layer (RoCEv2) may apply a rate-reduction mechanism. In contrast, iSCSI and NVMe/TCP environments benefit from the built-in congestion control algorithms of TCP to achieve rate reduction. Both approaches have their merits. TCP's rate-reduction is standardized, it is widely available and adopted. However, making it optimized for storage traffic may not be possible or not enabled by default. In comparison, RoCEv2's rate-reduction algorithm is not

standardized, and hence, it’s prone to non-availability and partial adoption (similar to Fibre Channel FPIN). However, vendors can choose to implement better algorithms than the standard TCP implementations for handling storage traffic.

Regardless of these architectural differences, congestion detection and troubleshooting mechanisms remain similar for RoCEv2 (UDP transport) and iSCSI, and NVMe/TCP (TCP transport). Find where the links are highly utilized, the queues are filling up, and the ECN counters are incrementing. Those are the sources of congestion and need your attention while the built-in transport mechanisms do their jobs.

Comparison with Fibre Channel Fabrics

Fibre Channel switches (and end devices) send Fabric Performance Impact Notifications (FPIN) and Congestion Signals to inform the end devices about congestion. These are special frames and primitives. Unlike ECN in TCP/IP networks, the Fibre Channel data frames are not marked with congestion information.

The most important difference is that this approach (of notifying the end devices for congestion prevention) is relatively new for Fibre Channel. At the time of this writing, production deployments are unknown. In contrast with TCP storage networks, most production networks support ECN.

Refer to Chapter 6, the section on *Congestion Prevention by Notifying the End Devices* for further details.

ECN Considerations for Block-Storage Traffic

Figure 8-10 provides an oversimplified explanation of ECN in TCP/IP networks. In production storage networks, however, hundreds or thousands of devices may connect to the same network with many-to-one traffic patterns.

Greenfield environments may not have network congestion. Hence, ECN may not activate initially, although it may be enabled or configured. Later, as networks grow and mature, the instances and severity of congestion may increase, which activates the use of ECN.

The following are important considerations for using ECN in TCP storage networks.

1. **Synchronization**: A traffic source that receives the ECE flag in the TCP header is unaware of other sources that send traffic to the same destination. The collective action of multiple such sources may lead to overreaction or under-reaction. This condition is similar to TCP global synchronization. The modern TCP implementations on the end devices and a random ECN-marking scheme (such as Weighted Random Early Detection (WRED)) on the switches should avoid this issue. Monitor the networks for this condition and refine the configuration (such as WRED thresholds) accordingly.
2. **Delayed action**: ECN (and other approaches for preventing congestion by notifying the end devices) may not be effective in some storage environments with burst traffic patterns because of the delay between congestion detection and preventive action. In Figure 8-10, Leaf-6 detects congestion, Host-1 receives the CE-flagged packet, sends the ECE-marked packet to the source, and finally, the source reduces its rate. These steps involve a delay for Leaf-6 to observe a change. By this time the original congestion symptoms may cease by themselves and hence, rate-reduction action invoked by ECN may not be needed anymore. To quote, RFC 3168—"CE packets indicate persistent rather than transient congestion, and hence reactions to the receipt of CE packets should be those appropriate for persistent congestion". When ECN action is delayed, packets are dropped in t@he network due to queue full conditions. Monitor the network for such symptoms and refine the configuration (such as reducing the WRED thresholds) accordingly.
3. **Packet drops**: As previously mentioned, ECN does not guarantee that the packets are not dropped. Because of the delay in observing the ECN rate reduction on the

congestion switchport, its queue may fill up, which ultimately results in packet drops. Refer to the later section on *Switch Buffer Management* for more details.

4. **Configuration complexity**: The success of ECN depends on how well it is configured in a network. Marking the CE flag on the switches typically uses a detection scheme (such as WRED) with thresholds, which should be configured so that the rate-reduction action is observed on the switch before its queues become full. These values also depend on the type of switch because different switches have different architectures and buffer sizes. Another complexity is for the end devices that do not support ECN or may not enable it by default.

These points should not deter ECN from being enabled in a network. Follow the recommendations of the vendors and refine the configuration for your environments. The important point to remember is that just because something works well in newer (greenfield environments), that does not eliminate the need for monitoring congestion symptoms. Problems become visible as the environment grows or matures. Being already aware of the potential problems can help you in faster resolution when and if the problem surfaces.

Switch Buffer Management

Recall that during network congestion, a TCP sender relies on the following events for reducing its transmission rate.

1. When the sender detects packet loss because of the following reasons.

   a. Retransmission timeout (RTO) or

   b. Receiving third duplicate ACK (Fast Retransmission).

2. When the sender receives an ECE-marked TCP packet.

Both events originate at a network port when its queues fill up. This means that the packet drops or marking scheme on a switch can significantly influence TCP's behavior on the end devices, and hence, these schemes are important in handling congestion in TCP networks.

This section explains various states of queues on a network port. These queues are created in the switch buffers, hence the use of the term—Buffer Management. The queues have their own Active Queue Management mechanism as explained later.

Queue Utilization

Figure 8-11 shows a subset of the spine-leaf network explained earlier in Figure 8-4. The Leaf-6 switch receives traffic from the four spine switches on Eth 1/1 – Eth 1/4 (ingress ports) destined for Host-1 that connects to Eth 1/5 (egress port). Ingress ports add packets to the egress queue while the egress port continuously transmits the packets from this queue.

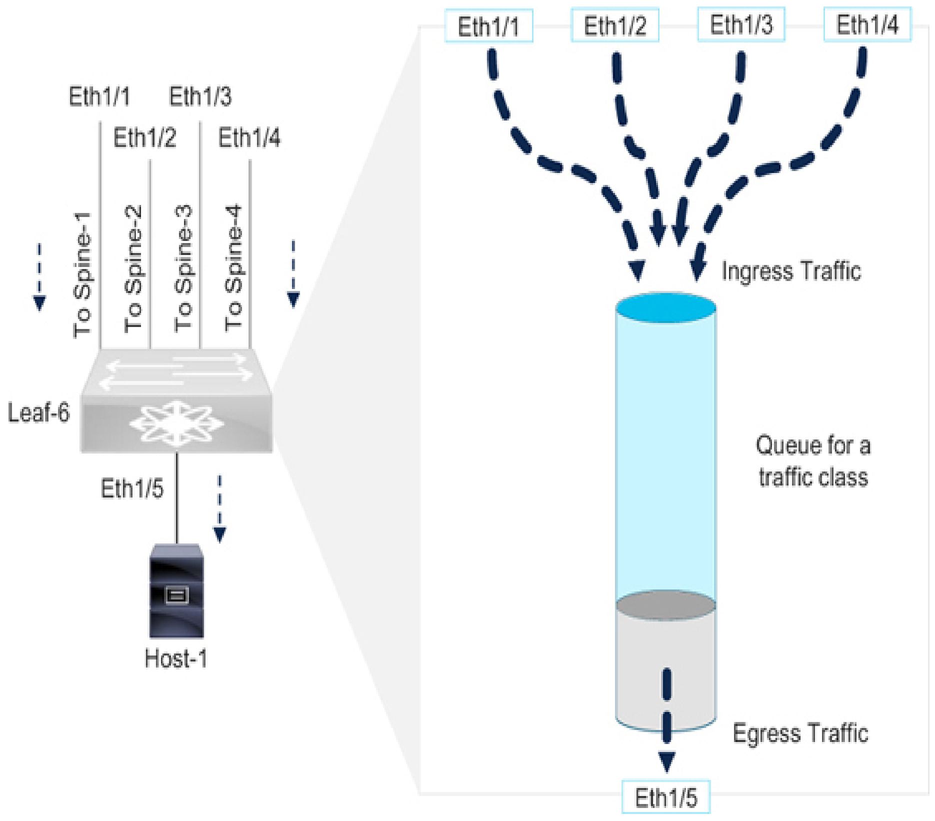


Figure 8-11 *Queue utilization on a switch*

The egress rate is fixed because it is defined by the link speed. If the collective ingress rate is slower than the egress rate, the queue remains empty. However, if the collective ingress rate is faster than the egress rate, the packets are temporarily stored in the queue.

There are many schemes to actively manage the packets in a queue. Together, these are called Active Queue Management. But to understand these schemes, first, let’s explore various states of queue utilization.

Typically, there are multiple egress queues per port, and packets from these queues are transmitted using a scheduling algorithm, such as Round Robin or Deficit Weighted Round Robin (DWRR). For the sake of simplicity, Figure 8-11 shows just one queue that is assigned to the storage traffic. In other words, this section focuses on packet forwarding within a queue, not among multiple queues. More details about the scheduling algorithms can be found in the existing QoS literature mentioned in the references of this chapter.

Figure 8-12 shows various states of queue utilization at an instance.

Empty Queue

The queue is empty in the absence of traffic. Because it has no packets to transmit, its transmission rate is zero (Figure 8-12).

When the queue receives a packet from an ingress port, it immediately starts transmitting it. After this packet is transmitted, the transmission rate again becomes zero until the next packet arrives.

Empty queues typically lead to low network utilization, which is considered a waste of network capacity and investments (cost per Gbps). Consider a 10 GbE port, which takes 1.2 microseconds to transmit a 1500-byte frame. If the queue receives frames every 2.4 microseconds, then it remains empty for 1.2 microseconds after transmitting each frame. This sequence leads to only 50% link utilization. For transmitting at full capacity while keeping the queue empty, the subsequent frames must arrive exactly every 1.2 microseconds.

Low Queue Utilization

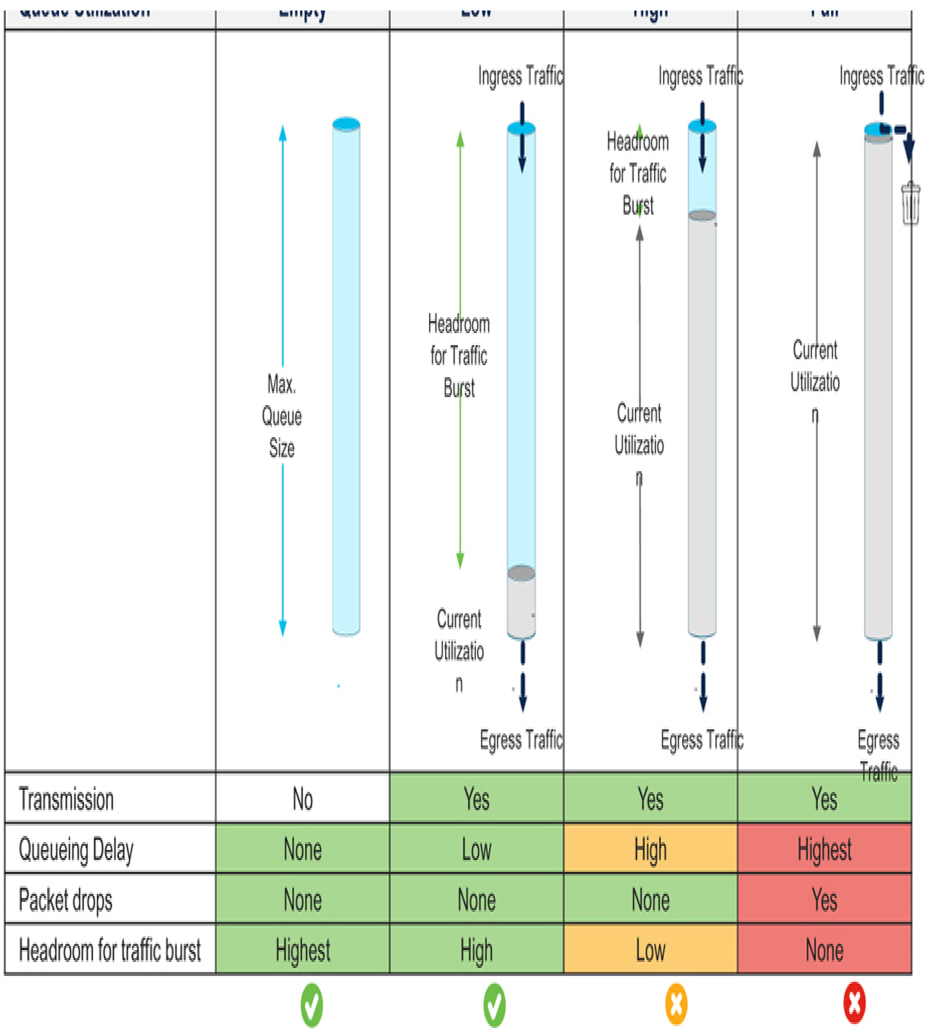
For continuously sending traffic on a port, without wasting its capacity, the next packet should already be in the queue so that it can be transmitted immediately after transmitting the current packet. Considering the example of the same 10 GbE port, production networks cannot guarantee that only 1500-byte packets every 1.2 microseconds. A better approach is to try achieving a low queue utilization instead. The exact value of low utilization depends on the traffic pattern. In general, the lower the better but not zero (Figure 8-12).

High Queue Utilization

The queue utilization increases when the ingress rate is faster than the egress rate. This may happen due to a traffic burst, such as a response to a large-size read or write I/O operation. The egress rate cannot change because packets are already being transmitted at the maximum speed (Figure 8-12).

Full Queue

Buffers are a finite resource. If the ingress rate continues to be faster than the egress rate, the queues are eventually fully utilized. As a result, any new incoming packets are dropped (Figure 8-12).



Transmission	No	Yes	Yes	Yes
Queueing Delay	None	Low	High	Highest
Packet drops	None	None	None	Yes
Headroom for traffic burst	Highest	High	Low	None
	✓	✓	✗	✗

Figure 8-12 *Various states of queue utilization on a network port*

Queue Utilization Considerations

TCP/IP networks typically aim for non-empty queues so that the queues always have packets for transmission without keeping the link idle. This achieves a high transmission rate and returns on investments (cost per Gbps).

As Figure 8-12 shows, non-empty utilization can mean any of the other states—low, high, or even full utilization. In all these states, the link is transmitting at full capacity. But there are some side effects, especially when the queues are highly or fully utilized.

Latency

When queue utilization is high, although no packets are dropped, new packets must wait behind the already-queued packets. This significantly affects the completion times of the I/O operations. For example, on a 10 GbE port with 1500-byte frames, if the queue already has 100 packets, the next packet will be delayed by at least 120 microseconds. This is a long time for a packet to spend within a switch, while the All-flash and NVMe storage arrays can achieve I/O completion times in the order of 100s of microseconds.

Such delays are difficult to detect by measuring only the link utilization because the link transmits at full capacity even when the queue utilization is low with minimal queuing delay. When queue utilization becomes high, the queuing delay increases but the link utilization remains unchanged.

Refer to the later section on *Queue Depth Monitoring and Microburst Detection* that explains how this increased latency can be detected by Cisco Nexus 9000 switches.

Tail Latency

Consider a transactional application that must read some data from the storage array for doing a calculation. This application decides the next step based on the result of that calculation. The application initiates 100 I/O operations for reading the data. While 99 I/O operations are complete, the response packet of the 100th I/O operation is delayed behind many other packets in the queue on a network port. As explained earlier in the sections on *iSCSI I/O Operations* and *NVMe/TCP I/O Operations*, an I/O operation is not complete until all its packets/PDUs are received. This delay may expire an application-level timeout resulting in retrying the entire calculation, which further leads to re-initiating all the 100 I/O operations again.

Overall, increased delay for just 1% of I/O operations stops the application from proceeding. This is like an animal trying to move forward but can't do so if its tail is stuck somewhere, although its entire body has moved forward. Hence, the term "tail latency". Another reason for calling it tail latency is that when the completion time values for these 100 I/O operations are plotted, the shape of the graph has a long tail because of a small number of I/O operations that take much longer than the rest. This is also known as 99th percentile.

High Utilization of the queues on a network port is one of the major reasons for high tail latency. Such issues are difficult to detect by measuring only the link utilization because the network reports high throughput, yet the application performance is degraded. In other words, the network has high “throughput”, but lacks the “goodput”.

Refer to the later section on *Queue Depth Monitoring and Microburst Detection* that explains how this increased latency can be detected by Cisco Nexus 9000 switches.

Headroom for Traffic Burst

When the utilization of a queue is already high, it does not have enough headroom to accommodate traffic bursts. This is even more concerning for storage traffic, which is bursty by nature. As explained in the earlier section on *Storage I/O Performance Monitoring*, a single I/O operation may generate many packets based on its I/O size. All-flash and NVMe storage arrays send the data packets at a high rate resulting in a traffic burst. If the queues on a network port do not have space to accommodate this burst, any new incoming packets are dropped.

Packet Drops

Packets are dropped when the queues are fully utilized regardless of the maximum queue size and the Active Queue Management mechanism (such as WRED). These dropped packets are considered lost by the TCP sender and retransmitted, which ultimately reduces the I/O performance.

Maximum Queue Size

Some switches have large or deep buffer space. As a result, a lot more packets can be stored in the queues, essentially increasing the maximum queue size.

However, simply increasing the queue size without the intelligence of managing them brings the side effect of increased (tail) latency.

Also, buffers are a finite resource. Regardless of their large or deep space, they will eventually get consumed during a bursty TCP Incast (as explained in Figure 8-4).

Finally, understand the difference between having deep buffers versus using them consistently. If these buffers are used all the time, which is like the state of high queue utilization, even the large buffer will not have enough free space to accommodate traffic burst, and hence new incoming packets will still be dropped.

User Actions

After learning the states of queue utilization, the following are the actions that admins and operators can take while using TCP transport for storage traffic.

1. Like any other TCP/IP network, TCP storage networks should also aim for low queue utilization. As Figure 8-12 shows, low queue utilization not only maintains high throughput and return on investments, but also minimizes the queuing delay, and maintains large headroom for traffic burst.

2. For achieving low queue utilization, use an Active Queue Management mechanism, such as WRED, explained later.

3. Regardless of the type of switches in a network, congestion detection, troubleshooting, and prevention remain the same. When the queues are non-empty (low utilization, high utilization, or full utilization), the port is transmitting at full capacity. This condition is explained earlier as congestion due to over-utilization. The solution to this problem is to increase the link speed or add more capacity.

4. Detecting the links that are consistently highly utilized is relatively easy. However, when the average utilization is not high, approaches like Microburst detection (explained later) and ECN counters can detect if the packets are

subjected to queuing delays. If the burst is larger than can be accommodated in the queue at that time, the packets are dropped regardless of the maximum queue size.

5. Despite all these efforts, bursty storage traffic in a large-scale network may not achieve a consistent low queue utilization. In such cases, the queue utilization may move to either end of the spectrum—Empty Utilization or High Utilization. High queue utilization has side effects of increased queuing delay and low headroom. Hence, empty utilization is the next best option.

6. Detecting the difference between low, high, and full queue utilization may be difficult because the link transmits at full capacity in all these states. If this difference cannot be detected, the subsequent action is not possible. Only the empty queue has low link utilization and hence, this is another reason for empty queue utilization to be the next best option after low queue utilization. Note that Advanced features like Mircroburst detection, available on Cisco Nexus 9000 switches, can detect if a queue utilization stays above a threshold for a duration. But not all networks may have such features and finding their thresholds may require additional effort.

7. In simple terms, for storage traffic, a 60 % link utilization is better than a 90% link utilization. This concept is explained in Chapter 1, the sections on *Defining Full Utilization vs High Utilization vs Over-Utilization* and *Monitoring Full Utilization vs High Utilization vs Over-Utilization*.

Comparison with Lossless Ethernet

Various states of queue utilization that are shown in Figure 8-12 also apply to lossless Ethernet networks.

Note the following points:

1. Figure 8-12 shows only egress queues. Lossless Ethernet ports also have ingress buffers with pause thresholds.

2. When a queue is fully utilized, new incoming packets are dropped regardless of being an ingress queue or egress queue.
3. Hence, most implementations start using the ingress buffers with pause thresholds before the egress queue is fully utilized. This point is illustrated in Chapter 7, Figure 7-18 to Figure 7-23. Pay attention to how ingress buffers with pause thresholds are used only after the egress queue utilization exceeds a certain level but before being completely full because a full queue makes it prone to tail drop.
4. The headroom shown in Figure 8-12 is different from the headroom for sending Pause frames at XOFF threshold. Figure 8-12 shows egress queues, whereas headroom for Pause Threshold is in ingress buffers.
5. Most Active Queue Management mechanisms explained in this chapter also apply to lossless Ethernet networks. The ingress buffers with pause thresholds are managed differently, as explained in Chapter 7, the section on *Pause Threshold and Resume Threshold*.

Comparison with Fibre Channel Fabrics

Low and high queue utilization in Figure 8-12 is similar to low remaining-Tx-B2B-credits on Fibre Channel ports. The full queue utilization state is similar to zero remaining-Tx-B2B-credits. Lossy Ethernet drops new incoming packets when the buffers/queues are full, whereas Fibre Channel fabrics stop any new incoming frames from arriving when their buffers are full using the B2B flow control.

The Active Queue Management mechanisms explained in this section do not apply to Fibre Channel fabrics because they are dedicated only to storage traffic. Also, dropping the packets in the fabric is not desirable.

The closest implementation is the no-credit-timeout feature on Cisco MDS switches, which drops frames when an edge port does not have Tx credits continuously for a duration, such as 50ms. If the idea of the no-credit-drop timeout is applied to the queue utilization states explained in Figure 8-12, packets will be dropped only when the queue utilization continuously stays full for 50 ms or a similar timeout. But without a hop-by-hop flow control to slow down the ingress traffic, new packets are still arriving and when the queue is full, the only option left is to drop these packets regardless of the Active Queue Management. Note that even in Fibre Channel fabrics, new incoming frames are dropped when the remaining-Rx-B2B-credits are zero, although this would be a rare severe protocol error.

Active Queue Management

As previously mentioned, dropping or marking schemes for packets that are waiting in a queue can significantly influence TCP's behavior on the end devices. These schemes are called Active Queue Management (AQM).

Tail Drop

The tail drop scheme drops newly arriving packets when a queue is full. Essentially, this scheme drops the "tail" of a queue which are the packets that have most recently arrived. Calling tail drop an Active Queue Management mechanism is a misnomer because dropping the packets due to the lack of buffer space is the default behavior and there is no "active" scheme needed for it.

Tail drop has an interesting effect on TCP performance. In a case where a single TCP flow arrives when the queue is full, a tail drop results in dropping many packets for that flow, resulting in a significant rate reduction of only that flow.

Another case is when many TCP flows arrive simultaneously. Tail drop results in dropping packets from all of them. As a result, all these flows reduce their rates simultaneously and then increase their rates simultaneously. This results in a cyclic underutilization and over-utilization pattern, called TCP global synchronization.

Random Early Detect (RED)

To avoid TCP global synchronization, packets already queued are dropped (or ECN marked) randomly before the queue is full. This allows only a few TCP connections to reduce their rate.

Weighted Random Early Detection (WRED)

WRED combines the capabilities of RED along with traffic priority, although the implementation may be different on different platforms.

As Figure 8-13 shows, when the egress queue utilization exceeds a WRED minimum threshold, the switch starts dropping or marking the CE flag (b‘11’) in ECN-capable (b‘01’ or b‘10’) packets randomly using a probability denominator. The rate of dropping or marking the packets increases linearly as queue utilization increases. When the queue utilization exceeds the maximum threshold, all packets newly arriving are dropped (tail dropped), or all ECN-capable packets are marked with the CE flag based on the configuration. If the queue still becomes full, all newly arriving packets are tail dropped.

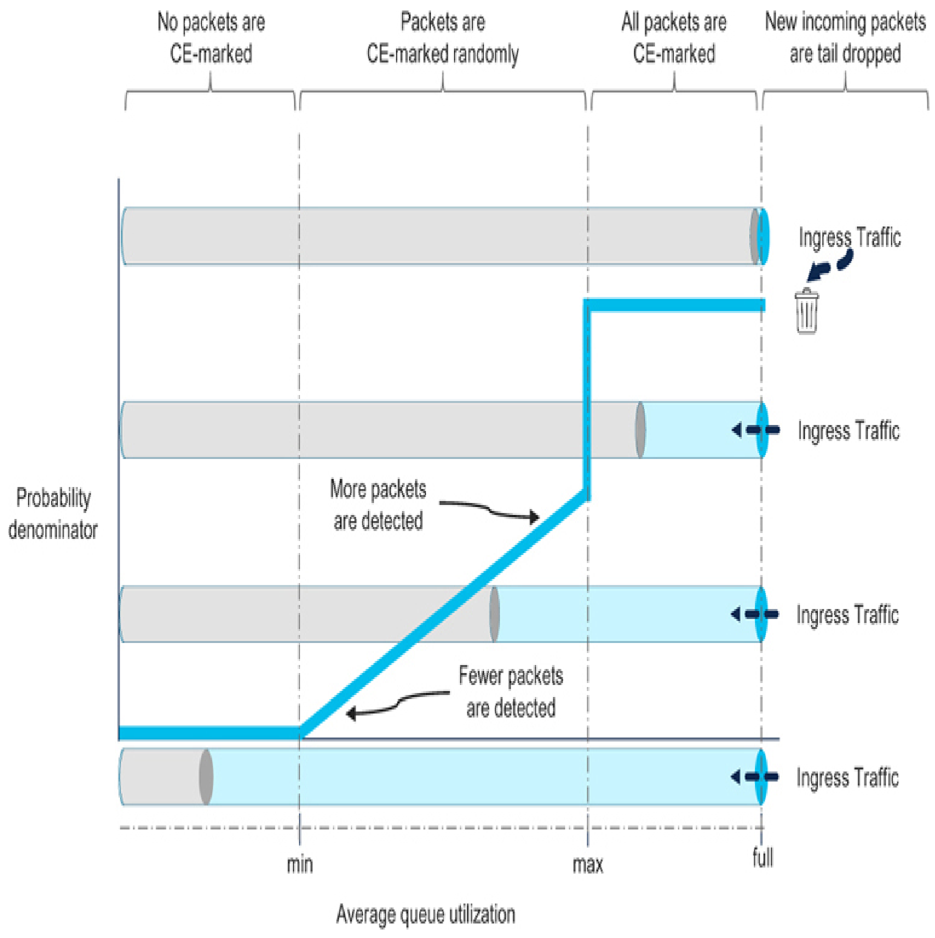

No packets are CE-marked
Packets are CE-marked randomly
All packets are CE-marked
New incoming packets are tail dropped
Ingress Traffic
Ingress Traffic
Ingress Traffic
Ingress Traffic
Probability denominator
More packets are detected
Fewer packets are detected
min
max
full
Average queue utilization

Figure 8-13 *WRED min and max thresholds and probability denominator*

Note the following points:

1. WRED has two components. The first is random detection. The second is action, which happens after the random detection. This action can be configured to drop the packets or mark the packets.

2. The WRED min and max thresholds depend on switch types and their buffer capacities. Different thresholds may be possible for ECN-capable traffic and non-ECN-capable traffic. Refer to the documentation of the products in your environment. Example 8-1 shows configuring WRED min and max thresholds for ECN marking on Cisco Nexus 9000 switches.

3. The WRED min should not be too high so that it delays marking the CE flag. At the same time, it should not be too low that it marks the CE flag too early even if only 1 or 2 packets are in the queue.

4. The difference between WRED min and max thresholds should be large enough for a few flows to be CE-marked and their effect to be observed on the congested ports. If the difference is small and the max threshold is exceeded, all the packets are CE-marked resulting in all the flows reducing their traffic rates.
5. As the section on *ECN Considerations for Block-Storage Traffic* explains, if the rate-reduction action is delayed and meanwhile, if the queue utilization continues to increase leading to a queue full state, all the new incoming packets are tail-dropped. Because of this reason, even a properly working ECN does not guarantee to avoid packet drops, and hence the ECN counters should be treated at the same severity as the packet drop counters. Recall that TCP has the same reaction to packet loss and ECN.

Example 8-1 *Configuring WRED min and max thresholds for ECN marking*

```
policy-map type queuing OUTPUT_Q
    class type queuing c-out-8q-q3
      random-detect minimum-threshold 100 kbytes
                    maximum-threshold 400 kbytes
                    drop-probability 50 weight 0 ecn
```

Approximate Fair Dropping (AFD)

Approximate Fair Drop (AFD) is an Active Queue Management algorithm available on Cisco Nexus 9000 switches. As Figure 8-14 shows, during congestion, AFD acts on long-lived large flows (called elephant flows) and does not impact short flows (called mice flows). The action of AFD can be dropping packets or marking the CE flag in the ECN-capable packets.

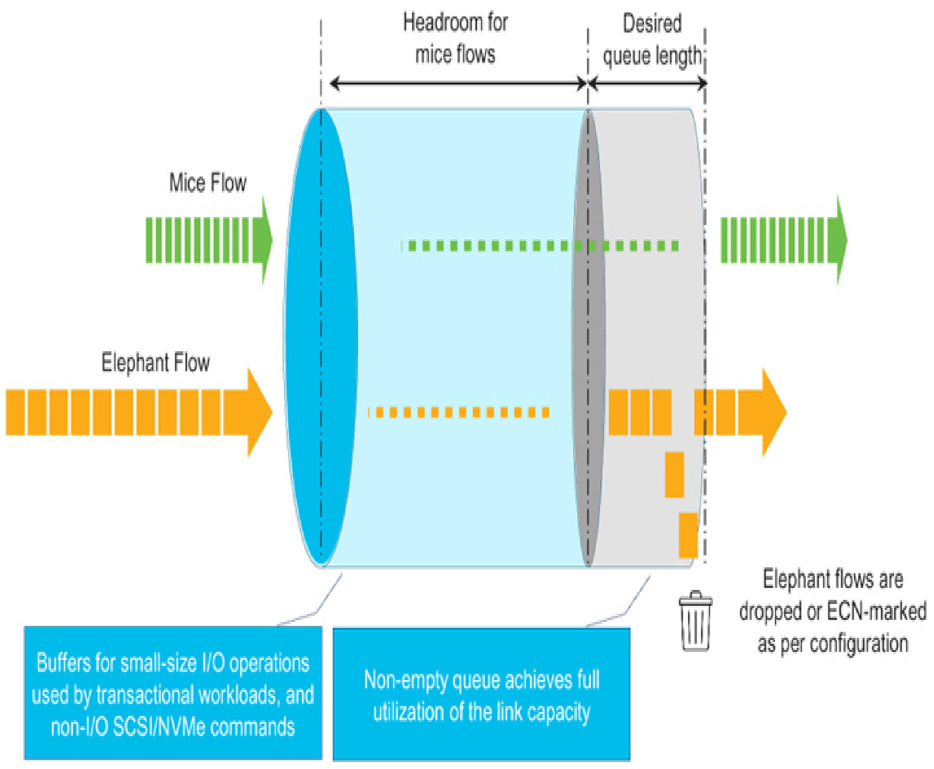

Headroom for mice flows
Desired queue length
Mice Flow
Elephant Flow
Elephant flows are dropped or ECN-marked as per configuration
Buffers for small-size I/O operations used by transactional workloads, and non-I/O SCSI/NVMe commands
Non-empty queue achieves full utilization of the link capacity

Figure 8-14 *AFD mechanism*

Be aware that AFD works on layer 4 flows, such as TCP flows, and not on I/O flows. Refer to the earlier section on *TCP Flow Monitoring versus I/O Flow Monitoring* for the difference between TCP flows and I/O flows.

AFD is different from WRED because AFD is flow-aware, whereas WRED is only class-aware (DSCP). With WRED, all packets (mice flows or elephant flows), have the same probability of dropping or ECN-marking. In contrast, AFD computes its probability based on the arrival rate of the flows, compares it with the computed fair rate, and drops or marks the packets from the elephant flows while not impacting the mice flows.

As explained in the section on *Storage I/O Performance Monitoring*, for iSCSI and NVMe/TCP traffic, data packets are large-sized, whereas command, R2T, and response packets are small-sized. Hence, AFD can prioritize the TCP flows that carry small-size I/O operations, which are typical for transactional applications.

AFD does not permanently categorize a flow as a mice or elephant. The same TCP flow (identified by the 5-tuples) can be categorized as mice or elephant based on its rate during a short duration, such as 50 microseconds. Because of this reason, iSCSI or NVMe/TCP traffic in a TCP flow can be categorized as mice flow when they exchange small-sized I/O operations. However, during large-size I/O operations, the same TCP flow may be categorized as an elephant flow. This behavior benefits transactional workloads that often use smaller I/O sizes and require low latency over the backup applications that are known to use large I/O sizes and require high throughput.

Another benefit of AFD is to avoid closing the connections during severe congestion. Although I/O performance may be affected because of queuing delay, keepalives may pass through because they can be categorized as mice flows. Note that iSCSI sessions may rely on TCP keepalives. NVMe/TCP has its own keepalives. When keepalives are lost, the connection may be closed leading to I/O errors on the end devices. AFD can help to avoid such issues.

Refer to Example 8-2 for configuring AFD thresholds on Cisco Nexus 9000 switches. A flow is categorized as an elephant flow by creating a trap for it (called elephant trap or etrap) when the number of bytes received in it exceeds the 'byte-count'. For a flow to continue to be an elephant flow, the configured 'bandwidth-threshold' number of bytes must be received in the configured 'age-period'.

Example 8-2 *AFD thresholds on Cisco Nexus 9000 switches*

```
switch (config)# hardware qos etrap byte-count
1048555
switch (config)# hardware qos etrap bandwidth-
threshold 500 bytes
switch (config)# hardware qos etrap age-period 50
usec
```

Dynamic Packet Prioritization (DPP)

Dynamic Packet Prioritization is another Active Queue Management algorithm available on Cisco Nexus 9000 switches. As Figure 8-15 explains, using DPP, a specific number of initial packets of a new flow are prioritized. In other words, a new flow always gets priority.

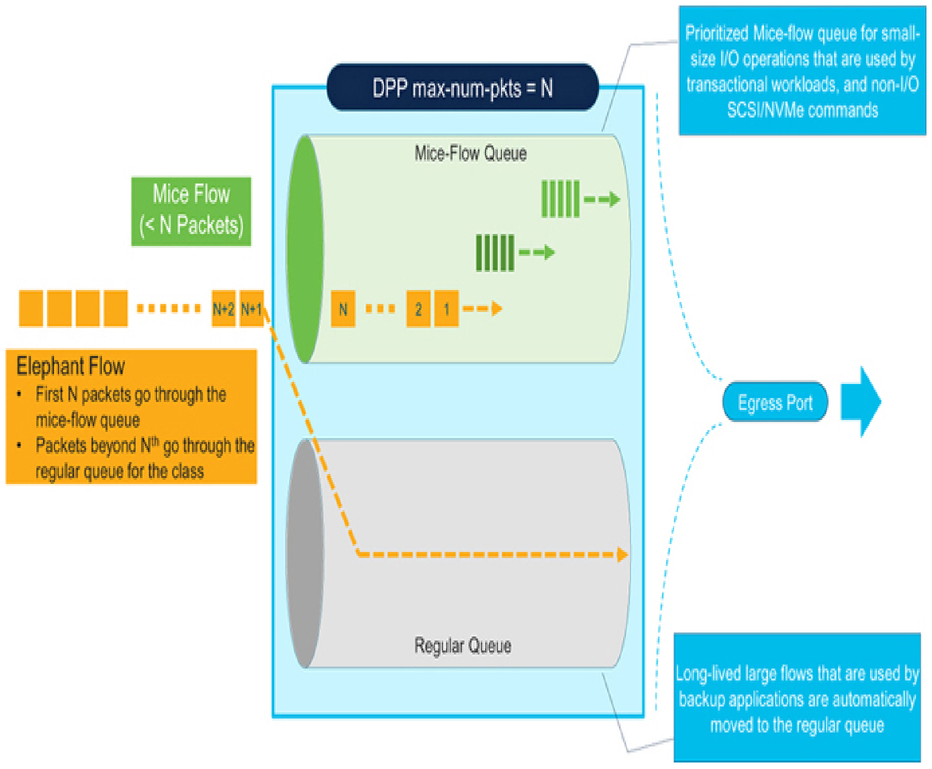

Prioritized Mice-flow queue for small-size I/O operations that are used by transactional workloads, and non-I/O SCSI/NVMe commands
DPP max-num-pkts = N
Mice-Flow Queue
Mice Flow (< N Packets)
N+2
N+1
N
2
1
Elephant Flow
First N packets go through the mice-flow queue
Packets beyond Nth go through the regular queue for the class
Egress Port
Regular Queue
Long-lived large flows that are used by backup applications are automatically moved to the regular queue

Figure 8-15 *DPP mechanism*

Like AFD, DPP also works on layer 4 flows, such as TCP flows, and not on I/O flows. Refer to the earlier section on *TCP Flow Monitoring versus I/O Flow Monitoring* for the difference between TCP flows and I/O flows.

DPP can help with iSCSI and NVMe/TCP traffic because, for an I/O operation, command, R2T (if sent), and response are carried in one packet each. One or more data packets are sent depending upon the size of data requested by the I/O operations. As a result, TCP flows transporting smaller-size I/O operations are prioritized with DPP as compared to the TCP flows that carry traffic for backup job, which typically uses large I/O sizes resulting in a many packets per I/O operation.

Refer to Example 8-3 for configuring DPP thresholds on Cisco Nexus 9000 switches. The initial (max-num-pkts) packets of a new flow are prioritized. A flow is aged out for DPP after the age-period. Later, if packets for the same flow arrive, they will be considered a new flow and their initial (max-num-pkts) packets will be prioritized.

Example 8-3 *DPP thresholds on Cisco Nexus 9000 switches*

```
switch(config)# hardware qos dynamic-packet-
prioritization max-num-pkts 120
switch(config)# hardware qos dynamic-packet-
prioritization age-period 5000 usec
```

To reserve buffer headroom for mice flows or to provide bandwidth fairness among elephant flows in the queue, DPP can be implemented in conjunction with AFD for the same traffic class. The combination of DPP and AFD provides the best handling of simultaneous mice flows mixed with a few elephant flows.

Detecting Congestion in TCP Storage Networks

This section explains detecting the conditions that lead TCP to invoke its reliability mechanisms, congestion control, and flow control. These mechanisms are invoked in response to a condition, but they do not solve the root cause by themselves. The entire reason that TCP invokes its reliability is that some unreliability was detected in the first place. Likewise, the entire reason that TCP invokes its congestion control mechanisms is that there was congestion in the first place. Consider the following cases:

1. The reliability aspect of TCP retransmits the lost packets so that the upper layers, such as iSCSI or NVMe/TCP, still receive the expected data in order. However, this retransmission delays the data transfer leading to increased I/O completion time and reduced IOPS.
2. TCP flow control limits the rate of egress traffic on a sender so that the receiver can process it at that rate. If a host wants to write data to a storage array at 100 MB/s, TCP flow control may reduce the host's transmission rate to 50 MB/s if that is what the storage array can handle.

This reduced throughput degrades application performance.

3. TCP congestion control limits the egress traffic to match the network capacity, even though the receiver can receive at the sender’s rate. For example, if a host wants to write 100 MB/s to a storage array but the network allows only 50 MB/s, TCP congestion control reduces the host’s transmission rate to 50 MB/s. This reduced throughput degrades application performance.

Because iSCSI and NVMe/TCP rely on TCP, detecting congestion in these environments essentially boils down to detecting congestion at the TCP layer. SCSI and NVMe layers typically do not report I/O errors unless the TCP connection is severely affected such as reset or closed.

Source of Congestion within the End Devices

When a TCP receiver is congested, it uses the end-to-end flow control to apply backpressure directly to the source to slow it down. Hence, the investigation should focus on that host device.

Congestion Detection Notes

Note the following points:

1. TCP uses flow control only if the source of congestion lies at the TCP layer or in any of the upper layers, such as iSCSI and NVMe/TCP. For example, if the application is not fast enough to process the ingress traffic because of multiple-concurrent large-size read I/O operations, eventually the TCP layer will have to keep this data in its buffer. Because the receiver’s TCP layer has fewer free buffers, it advertises a smaller (or even zero) receiver-window to the sender.

2. TCP does not use flow control if the source of congestion is within the end device, but below the TCP layer. A common example is virtualized hosts where the virtual

machines and containers are the endpoints of the TCP connections. If the hypervisor has a bottleneck, TCP does not invoke its flow-control mechanism. Such issues are still within the host (and outside the scope of the network), but TCP treats them as network congestion and uses its congestion control mechanisms to handle them.

3. Detecting and troubleshooting these issues depend on the type of end devices. Most TCP stacks are natively available in operating systems, such as Windows, Linux, and ESXi. The detection mechanisms may change if any hardware offload engines are used. Refer to the vendor documentation for more details. In Example 8-4, which shows TCP counters on Linux, TCP Retransmission Timeout (RTO) is 202 milliseconds, Maximum Segment Size (MSS) is 1448 bytes, and congestion-window (cwnd) is 6. Refer to the manual page of Linux **ss** command for a detailed explanation.

Example 8-4 *Output of the Linux* **ss** *command showing TCP counters*

```
[root@localhost ~]# ss -e
ESTAB    0   0    172.22.163.112:60185
172.22.163.208:4420
timer:(keepalive,060ms,0) uid:983 ino:215051683
sk:ffff881fc1718780 <->

ts sack cubic wscale:7,7 rto:202 rtt:1.527/2.26
ato:40 mss:1448 cwnd:6 ssthresh:5
send 45.5Mbps lastsnd:4981 lastrcv:4962 lastack:4981
pacing_rate 91.0Mbps
retrans:0/8724 rcv_rtt:116314 rcv_space:29760
```

Comparison with Lossless Networks

Recall that in Fibre Channel fabrics (explained in Chapter 3, “Detecting Congestion in Fibre Channel Fabrics”) and lossless Ethernet networks (explained in Chapter 7), when the source of congestion is within an end-device, these devices are called slow-drain devices and they are detected using the metrics of the flow-control mechanism, such as the lack of the Tx B2B credits on the Fibre Channel switchport and excessive Pause frames on the Ethernet switchport.

In TCP storage networks, when an end device is the source of congestion, the network ports do not show any indication.

Source of Congestion within the Network

When the source of congestion is within the network, the primary detection approach is to find packet drops. If a network is provisioned for notifying the end devices using ECN, then detecting the ECN counters is another mechanism. ECN does not guarantee there will be no packet drops. Hence, ECN counters should be an early indication of congestion. They should be investigated at the same severity as the packet drops.

Packet Drops or Discards

Note the following points:

1. Dropped packets at an interface/switchport level should be monitored. On Cisco Nexus 9000 switches, NX-OS command **show interface** displays the input and output discarded packets. Refer to Chapter 7, the section on *Frame Drops or Discards* for an example of this command.

2. When iSCSI and NVMe/TCP traffic is assigned to a class, metrics per per-class should also be monitored. On Cisco Nexus 9000 switches, NX-OS command **show queuing interface** (Example 8-5) and **show policy-map interface** (Example 8-6) display dropped packets per traffic class.

2. Packet drops in the storage traffic class and in other classes should be investigated at the same severity. Drops in other classes still indicate a network capacity issue.

3. There are multiple reasons for packet drops.

   a. When packet drops due to queuing/congestion are reported

i. Increase link speed or add additional links.

ii. Verify that traffic is uniformly balanced among multiple links. If links are not uniformly utilized, during the duration of a traffic spike one link may drop the packets, while other links may be underutilized.

b. When packets drop due to corruption and bit errors are reported identify the original link that is causing the corruption and resolve the physical layer issue.

ECN Counters

ECN can prevent congestion in TCP/IP networks by notifying a sender to slow down.

Refer to Example 8-5, which shows the number of ECN-capable packets that were marked with CE-flag. These are referred to as “ECN Pkts” in the **show queuing interface** command on Cisco Nexus 9000 switches.

Example 8-5 *ECN counters on Cisco Nexus 9000 switches in the command* ***show queuing interface***

```
switch# show queuing interface ethernet 1/15
<snip>

+----------------------------------------------------
----------+
|                              QOS GROUP 5
|
+----------------------------------------------------
----------+
|                           |  Unicast
|Multicast        |
+----------------------------------------------------
----------+
|                  Tx Pkts |        16368604|
0|
|                  Tx Byts |     46733248300|
0|
|                 ECN Pkts |         9609408|
0|
|                  ECN Byts |     27614822366|
0|
|              Q Depth Byts |         7819136|
0|
|             WD Flush Pkts |        14107532|
0|
+---------------------------------------------------
----------+
|                              QOS GROUP 6
|
+---------------------------------------------------
----------+
|                           |  Unicast
|Multicast        |
+---------------------------------------------------
----------+
|                   Tx Pkts |               0|
0|
|                   Tx Byts |               0|
0|
| WRED/AFD & Tail Drop Pkts |               0|
0|
| WRED/AFD & Tail Drop Byts |               0|
0|
|              Q Depth Byts |               0|
0|
|        WD & Tail Drop Pkts |               0|
0|+-----------------
---------------------------------------------+
```

Occasional and small increments in ECN counters are fine. However, continuous increments in the ECN counters or increments in high numbers should be treated at the same severity as packet drops.

Link Utilization

Chapter 7, the section on *Link Utilization* explains link utilization monitoring on Cisco Nexus switches and UCS servers.

As mentioned earlier in the section on *Queue Utilization*, networks links that are consistently highly utilized, but are not reporting packet drops, lead to the following conditions:

1. Packets are delayed due to queuing delay resulting in degraded I/O performance.
2. Packets are not dropped and not delayed but if the links are still operating at high utilization, there is not enough capacity left to handle traffic burst. Recall that storage traffic is bursty.

In most production networks, traffic rates are always calculated over a time period, and therefore reported rates are always an average over that time period. Regardless of these states of a network port, if the links are highly utilized, treat them as over-utilized and add more capacity.

Queue Depth Monitoring and Microburst Detection

Queue depth monitoring and microburst detection capture the events that may cause congestion at a lower granularity but are unnoticed by other means due to long polling intervals.

Consider a link that operates at 100% utilization for 100 ms and 0% utilization for the next 900 ms. The reported link utilization during this 1-second interval is averaged to 10% utilization. As a result, this congestion event is not captured by monitoring link utilization. The following are the approaches that can detect this condition:

1. Packet drop counters. But these may increment only if the queues are full (Figure 8-16).
2. ECN counters, if ECN is enabled. But these may increment only if the queue utilization exceeds ECN thresholds. For example, assume that the maximum queue size is 400 KB. Packets are marked with ECN when queue utilization exceeds 100 KB. If queue utilization stays below 100 KB, the link transmits at 100% utilization for 100 ms but there would be no way to detect this condition (Figure 8-16).

Microburst detection and queue depth monitoring capture such conditions (Figure 8-16). In the earlier example, if queue utilization is higher than 30 KB continuously for 80 microseconds, microburst detection can capture this event.

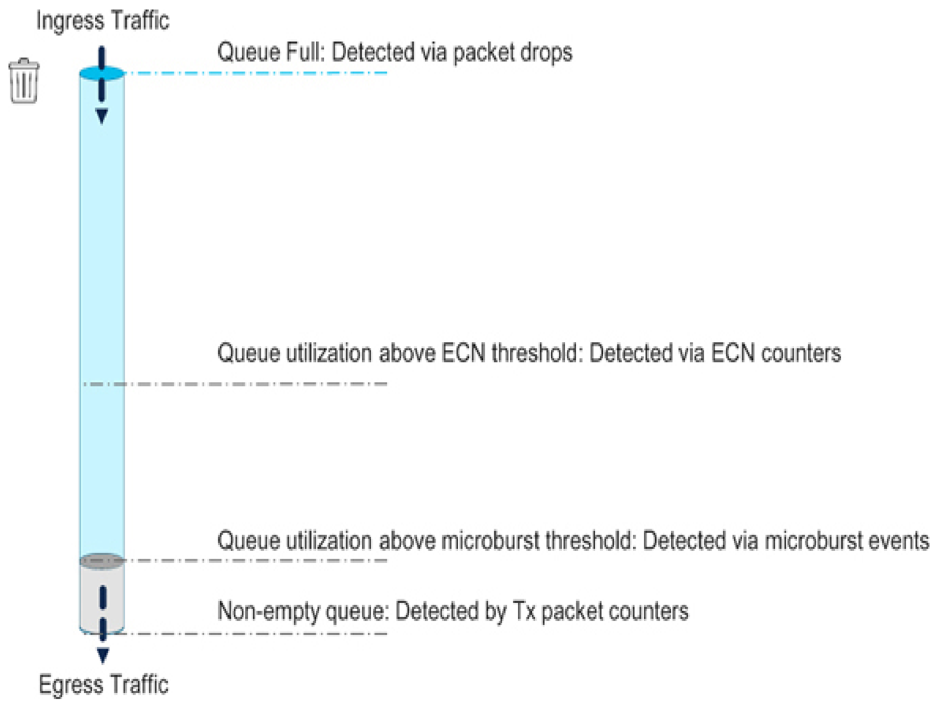

Ingress Traffic
Queue Full: Detected via packet drops
Queue utilization above ECN threshold: Detected via ECN counters
Queue utilization above microburst threshold: Detected via microburst events
Non-empty queue: Detected by Tx packet counters
Egress Traffic

Figure 8-16 *Queue utilization states and their detection mechanisms*

Additionally, it is possible to monitor the instantaneous utilization of the queues. Refer to Example 8-5 that shows Q Depth Byts in the command **show queuing interface** on Cisco Nexus 9000 switches.

Refer to Example 8-6, which shows the output of **show policy-map interface** command on Cisco Nexus 9000 switches. Note the following points:

- WRED (random-detect) is enabled on this queue.
- ECN-capable packets are marked randomly with CE flag when queue utilization is between 100 KB and 200 KB. All ECN-capable packets are marked when queue utilization exceeds 200KB. Because of this configuration, 4459882 packets were marked since the counters were cleared last.
- Despite ECN being operational, queue utilization became full, which is indicated by the 47536 dropped packets.

- The queue utilization is at 51424 bytes (approx. 50K) when this command was executed. This is the instantaneous value, and it is not known how long the queue utilization stayed at this value.
- This queue is configured to detect microburst events ('burst-detect') when the queue utilization exceeds 30K and stays above this value continuously for a minimum burst interval. This burst interval can be 70 to 90 microseconds based on the switch type.

Example 8-6 *Monitoring statistics on Cisco Nexus 9000 switches using* ***show policy-map interface*** *command.*

```
switch# show policy-map interface ethernet 1/15
output detail
<snip>
    Class-map (queuing):   c-out-8q-q5 (match-any)
      bandwidth remaining percent 15
      shape min 1 gbps max 1 gbps
      random-detect minimum-threshold 100 kbytes
                    maximum-threshold 200 kbytes
                    drop-probability 100 weight 0
ecn
      burst-detect rise-threshold 30000 bytes
                   fall-threshold 30000 bytes
      PFC Statistics
       TXPPP Packets: 0
       RXPPP Packets: 0
      PFC WD Statistics       Shutdown Count: 0
       Restore Count: 0
       Flushed Packets: 0
      queue dropped pkts : 0
      queue depth in bytes : 51424
      Ingress queue discard packets : 0
      Ingress queue depth in bytes : 0
       ECN marked packets: 4459882
      Unicast Transmit Packets : 21651482445
      Multicast Transmit Packets : 0
      Unicast Transmit Bytes : 15068405583053
      Multicast Transmit Bytes : 0
      Unicast Dropped Packets : 47536
      Multicast Dropped Packets : 0
      Unicast Dropped Bytes : 135706844
      Multicast Dropped Bytes : 0
      Unicast Depth Bytes : 51424
      Multicast Depth Bytes : 0
<snip>
```

As Example 8-7 shows, use the NX-OS command **show queuing burst-detect** to display microburst events. This output has been split into multiple lines for easy readability. It shows the following:

- Interface Eth 1/15 shows two microburst events for unicast traffic in queue 5 (U5). These events start when the queue utilization exceeds the **rise-threshold** parameter in the **burst-detect** command (Example 8-6). Notice the Start Depth of 29840 bytes, which is close enough to the configured **rise-threshold** of 30000 bytes and the Start Time when this happens. The first microburst starts at 11:55:34.993672. The second microburst starts at 12:17:17.653207.
- During the first microburst event, the Peak Depth of the unicast queue 5 reached to 5390944 bytes at 11:55:35:013740. Likewise, during the second microburst event, the Peak Depth of unicast queue 5 reached to 7497568 bytes at 12:17:17:682402.
- Microburst events end when the queue utilization falls below the **fall-threshold** parameter in the **burst-detect** command. Notice the End Depth of 29952 bytes, which is close enough to the configured fall-threshold of 30000 bytes. The first microburst event ended at 11:59:16:153676 and it lasted 221.16 seconds. Likewise, the second microburst event ended at 12:26:13:771249 and it lasted 536.12 seconds.

Example 8-7 *Microburst events on Cisco Nexus 9000 switches using* ***show queuing burst-detect*** *command.*

```
switch# show queuing burst-detect detail


slot  1
```

```
=======

-----------------------------------------------------
-------
          Microburst Statistics

Flags: E - Early start record, U - Unicast, M -
Multicast
-----------------------------------------------------
-------
Ethernet | Queue |Start Depth|         Start Time
|
Interface|       |  (bytes)  |
|
-----------------------------------------------------
-------
  Eth1/15|  U5   |   29840   | 2023/03/22
11:55:34:993672 |
  Eth1/15|  U5   |   29840   | 2023/03/22
12:17:17:653207 |

-----------------------------------------------------
--------
Ethernet | Queue | Peak Depth |          Peak Time
|
Interface|       |  (bytes)   |
|
-----------------------------------------------------
--------
  Eth1/15|  U5   |  5390944   | 2023/03/22
11:55:35:013740 |
  Eth1/15|  U5   |  7497568   | 2023/03/22
12:17:17:682402 |

-----------------------------------------------------
-------------------
Ethernet | Queue | End Depth  |         End Time
| Duration |
Interface|       |  (bytes)   |
|          |
-----------------------------------------------------
-------------------
```

```
  Eth1/15|  U5   |    29952    | 2023/03/22
11:59:16:153676 | 221.16 s |
  Eth1/15|  U5   |    29952    | 2023/03/22
12:26:13:771249 | 536.12 s |
<snip>
```

In TCP storage networks, occasional microburst events should be fine. However, links that continuously report microburst events, show a higher number of microbursts than other links or show an increasing number of microburst events need investigation.

As the occurrences of such microburst increases, so does the number of congestion events.

Special consideration is needed for links that do not report high utilization but report many microburst events. For example, for a link that reports 80% utilization and occasional microburst, the solution is to increase its speed or add more links. However, a link that reports only 10% utilization but many microburst events, adding more capacity may be difficult to justify. Consider using a different Active Queue Management mechanism, such as AFD and DPP that can prioritize short-lived small flows over long-lived large flows. Also, if the TCP flow-level statistics are available, find if a particular flow that stands out or is frequently reported to drop. Investigate any network design issues for this flow, such as speed mismatch, load-balancing (explained later), etc. Essentially, the aim is to find a flow that carries traffic burst for a very small duration.

Bit Errors

As the section on *Bit Errors in Lossy Ethernet Networks with TCP Transport* explains, depending on where errors are detected, (Ethernet frame or TCP segment), the packet may be dropped at that layer on the end devices. As a result, relevant counters of each layer must be checked. For example, the network port or interface typically reports counters for CRC-corrupted Ethernet frames. However, TCP checksum errors are reported only at the TCP connection endpoints.

Refer to Chapter 7, the section on *Congestion Detection Metrics* and the sub-section *Bit Errors* for detecting bit errors (Ethernet CRC, Stomped CRC, and FEC) on Cisco Nexus switches and UCS servers.

Detecting Congestion on a Remote Monitoring Platform

Remote monitoring platforms can monitor all the ports in a network simultaneously to provide network-wide single-pane-of-glass visibility.

Refer to Chapter 3, the section on *Detecting Congestion on a Remote Monitoring Platform*, which already explained using the following types of monitoring applications:

- An application developed by the device manufacturer/vendor, such as Cisco Nexus Dashboard Fabric Controller (NDFC) and Nexus Dashboard Insights.
- A 3rd party or a custom-developed application, such as the MDS Traffic Monitoring (MTM) App.

Chapter 3 explains *The Pitfalls of Monitoring Network Traffic*. Its sub-section on *Average and Peak Utilization* apply to TCP Storage networks as well.

The remote monitoring platforms can be used for the following additional functions.

Comparative Analysis

Compare the rate of packet drops, ECN Counters, and microburst events per port and per-traffic class. Detect if a few ports have an excessively higher count than others.

The same comparative analysis should be used across all similar entities. For example, compare all the spine ports and detect if a few ports report an excessively high count.

Trends and Seasonality

Packet drops and ECN are important for TCP congestion control mechanisms, and hence, their nominal activity is fine.

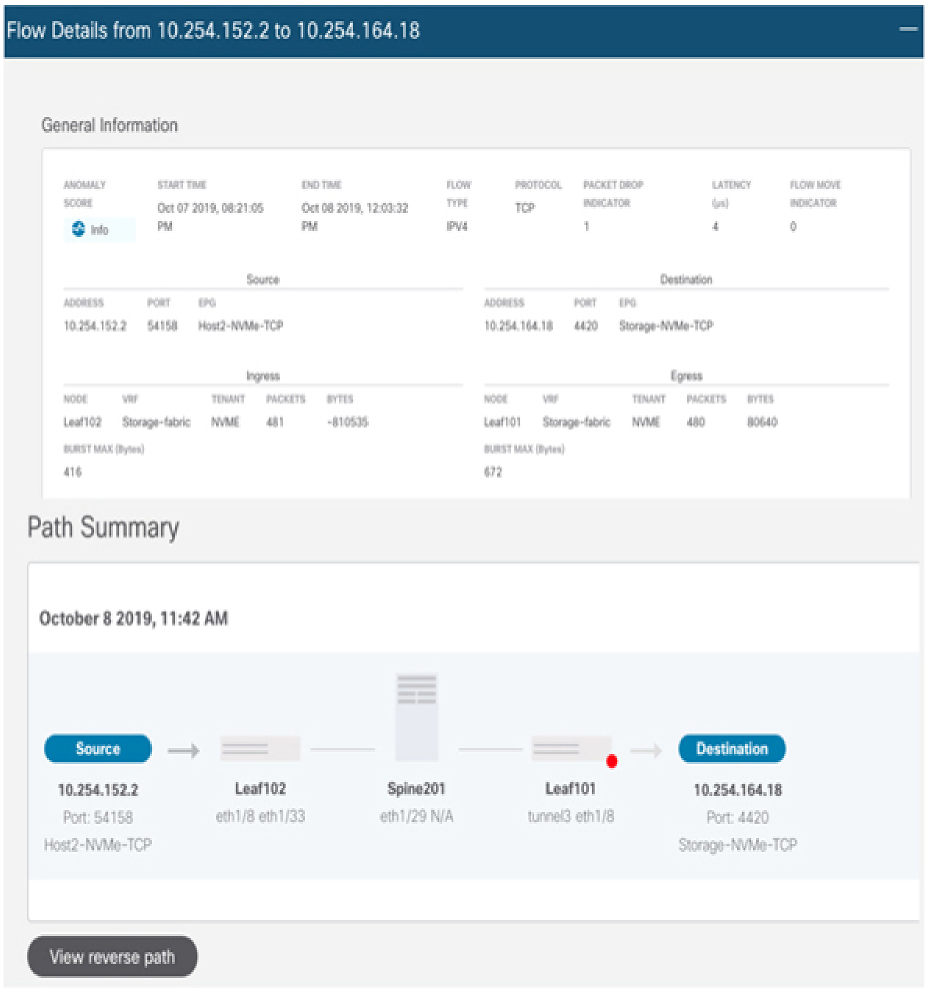
But analyze any spikes and dips carefully. Also, find if the packet drops, ECN counters, and microburst events have been on the rise over the last few days, weeks, or months, although there may not be any sudden spikes. Also, look out for any seasonality, that is, if the spikes are observed during specific hours in a day or days in a week, or even months in a year.

In graphical terms, a straight line with low counts is fine. Occasional spikes are also acceptable. Pay attention to spikes, especially big spikes that are sustained longer.

Congestion Detection using Cisco Nexus Dashboard Insights

Refer to Chapter 7, the section on Congestion Detection using Cisco Nexus Dashboard Insights.

Figure 8-17 shows NVMe/TCP flow as monitored by Cisco Nexus Dashboard Insights. Notice the end-to-end path summary. The red dot on Leaf101 indicates the location of packet drops for this flow.


Flow Details from 10.254.152.2 to 10.254.164.18

General Information

ANOMALY SCORE	START TIME	END TIME	FLOW TYPE	PROTOCOL	PACKET DROP INDICATOR	LATENCY (μs)	FLOW MOVE INDICATOR
Info	Oct 07 2019, 08:21:05 PM	Oct 08 2019, 12:03:32 PM	IPV4	TCP	1	4	0

Source

ADDRESS	PORT	EPG
10.254.152.2	54158	Host2-NVMe-TCP

Destination

ADDRESS	PORT	EPG
10.254.164.18	4420	Storage-NVMe-TCP

Ingress

NODE	VRF	TENANT	PACKETS	BYTES
Leaf102	Storage-fabric	NVME	481	-810535

BURST MAX (Bytes)
416

Egress

NODE	VRF	TENANT	PACKETS	BYTES
Leaf101	Storage-fabric	NVME	480	80640

BURST MAX (Bytes)
672

Path Summary

October 8 2019, 11:42 AM

Source	Leaf102	Spine201	Leaf101	Destination
10.254.152.2	eth1/8 eth1/33	eth1/29 N/A	tunnel3 eth1/8	10.254.164.18
Port: 54158				Port: 4420
Host2-NVMe-TCP				Storage-NVMe-TCP

View reverse path

Figure 8-17 *NVMe/TCP flow monitoring in Cisco Nexus Dashboard Insights*

Cisco Nexus Dashboard Insights has features that solve many other types of networking problems. The best place to learn about it is the Cisco documentation, especially the Cisco Nexus Dashboard Insights User Guide.

Metric Export Mechanisms

The mechanism for exporting metrics is a major consideration for a custom-developed application or a script. Using command-line outputs and SNMP has been historically common but using APIs has become the norm now. For low-granularity metric export at scale, streaming telemetry is seeing rapid adoption.

Cisco Nexus Dashboard Fabric Controller and Nexus Dashboard Insights select the best export mechanism by default.

Most details that are explained in Chapter 3 and Chapter 7, the section on *Metric Export Mechanisms* apply to TCP Storage networks as well. Pay special attention to the *Recommendations* for metric exports as explained in Chapter 3.

The field of network telemetry and analytics is evolving fast. We recommend referring to the product documentation and release notes for knowing the capabilities of products in your environment and how to use them.

Troubleshooting Congestion in TCP Storage Networks

Troubleshooting congestion in TCP storage networks is different from lossless Ethernet networks and Fibre Channel fabrics because TCP typically runs on a lossy Ethernet network without any hop-by-hop flow control. Because congestion does not spread hop-by-hop in lossy networks (with or without TCP), there is no need to chase congestion to its source as it's needed in lossless networks. Packets are dropped at a congested port and the adverse effect of congestion is confined only to the flows that pass through that port.

Goals

The end-to-end flow control and congestion control mechanisms of TCP are enabled by default and no user intervention is needed. But these mechanisms, including ECN, are not invoked until needed. When these mechanisms are invoked, the following are the troubleshooting goals.

1. **Identify the source (Culprits) and cause of congestion**: The cause of congestion is most certainly the lack of capacity on a network link, inability to handle traffic burst, or bit errors.

2. **Identify the affected devices (Victims)**: When a network link does not have enough capacity, all flows passing through that link may be affected because their packets may be dropped or delayed. Unlike lossless networks, only the flows that pass through the congested ports are affected. Other links in the traffic path are not affected. Also, the source and destination of the affected flows can continue to send/receive traffic to/from other devices via non-congested links.

Congestion Severities and Levels

Chapter 4, "Troubleshooting Congestion in Fibre Channel Fabrics," defines congestion severities and levels for Fibre Channel fabrics, and Chapter 7, the section on *Troubleshooting Congestion in Lossless Ethernet Networks* defines them for lossless Ethernet networks. Those definitions may not directly apply to TCP networks because the metrics for detecting congestion are different.

TCP Storage networks can categorize congestion in the following levels.

1. **Mild Congestion (Level 1)**: Increased I/O Latency but no frame drops and no I/O errors on the end devices.
   a. Detect with ECN-counters, high link utilization, and microburst detection on the network links and CE and ECE-marked packets, out-of-order packets, increase in RTT, SRTT, and RTO on the end devices.
2. **Moderate Congestion (Level 2)**: Increased I/O Latency and Frame drops but no I/O errors on the end devices.
   a. Detect with per-port and per-traffic-class packet drops on the network links and retransmissions on the end devices.
3. **Severe Congestion (Level 3)**: Increased I/O Latency, Frame drops, TCP connection failure, and I/O errors on the end devices.
   a. This may happen when the TCP connection resets or closes or a host cannot reach the remote storage

volume due to the loss of control messages, such as keepalives. Detect these events using per-port and per-class packet drops in large numbers on the network links. These events are also logged at the upper layers (SCSI and NVMe) on the end devices.

Methodology

As previously discussed, we recommend troubleshooting congestion in a decreasing level of severity. For example, verify that the end devices can establish TCP connections and that these connections do not close randomly due to severe congestion, routing loops, or misconfigurations. Then, verify that they can exchange traffic at the expected rate and find any packet drops. Finally, troubleshoot if the performance is not as expected or if the I/O responses are delayed.

These congestion severities are not standardized. Their sole purpose is to help a user in developing a solid workflow, which helps in detecting and troubleshooting congestion quickly and accurately. You can customize it for your environment.

While investigating a performance degradation problem on the end devices, users may notice TCP transmissions.

OK, TCP is reporting retransmissions, what do I do now?

If TCP is getting lower than expected performance, even though its endpoints know about lost packets, it has no idea where in the end-to-end path the packets were lost or the reason for the dropped packets. Many times, it comes down to scouring the switches and routers in the end-to-end IP path. This requires understanding the layer 3 and layer 2 devices along the way looking, even hoping to see an interface with input CRC errors or queuing drops. These steps can be simplified by using Cisco Nexus Dashboard Insights or similar remote monitoring platforms. But in general, this is an arduous task and it becomes even more difficult if an Internet Service Provider (ISP) is involved because once the packets enter their network, the local network and the storage teams no longer have visibility or control. The ISP must be relied on, which is somewhat an opaque process. Be prepared for some convincing because the non-storage traffic may not be reporting problems. But the storage traffic may be degraded due to higher throughput requirements. Be prepared to demonstrate the metrics that are being reported by the TCP endpoints with as much specificity as possible.

If this situation is compared to Fibre Channel, it's still mostly true that end devices don't know where packets (frames) are lost but the B2B link level flow control pinpoints pretty well where the congestion is in the fabric and where it has spread too. Also, with the release of FPINs, there is some notification to end devices of where frames were discarded (Frame Discard FPIN) or which links have CRC errors (Link Integrity FPIN). Those FPINs contain the PWWNs of both the detecting and affected (adjacent) ports involved. But the use of FPIN is dependent on the readiness of an environment. Refer to Chapter 2, the section on *Congestion Prevention by Notifying the End Devices* for further details.

Load-balancing in TCP Storage Networks

IP networks typically use flow-based load balancing scheme (or optionally a per-packet-based load balancing scheme) for achieving uniform utilization of members of an aggregated link (port-channel) or ECMP links. Here, the flow refers to the 5-tuples—source IP, destination IP, layer 4 protocol (UDP, TCP, etc.), layer 4 source port, and layer 4 destination port. As a result, all the iSCSI or NVMe/TCP traffic in a TCP connection belongs to the same flow and hence takes the same network path.

This behavior is different from Fibre Channel or FCoE fabrics that typically use exchange-based load-balancing. All frames of an I/O operation are uniquely identified by Exchange IDs (carried within the Fibre Channel header). As a result, traffic of different I/O operations between the same endpoints is uniformly distributed among member links of a port-channel or ECMP links.

On the contrary, in TCP/IP networks, traffic for even multiple concurrent I/O operations carried by a TCP flow always takes the same network path. This scheme is more likely to cause non-uniform link utilization as compared to the exchange-based load balancing scheme of Fibre Channel, and therefore, it needs special attention. Metrics and symptoms of congestion detection should be separately monitored on all the available links. This is even more important in a clos network design with multiple spine switches that are connected to the leaf switches via ECMP links because not only the links on one switch must be compared, but also the links on multiple spine switches must be compared.

If a particular TCP flow is causing high I/O throughput, consider spreading its traffic across multiple TCP flows if the end devices allow increasing the number of TCP connections. A new TCP connection uses a different source port, and hence its traffic is classified as a different flow, which takes a different link than the earlier TCP connections. Number of TCP connections in an iSCSI session depends on the MaxConnections parameter. Its default value is 1, but it can be increased based on the capability of the iSCSI initiator and target. NVMe/TCP, by default, creates one TCP connection per I/O Submission Queue and Completion Queue pair. Refer to the documentation of the products in your environment for further details.

QoS Considerations for Dedicated and Shared Storage Networks

A network is called a dedicated storage network when it carries only storage traffic, such as iSCSI and/or NVMe/TCP. But most deployments share the same Ethernet network for storage and other traffic. In these shared storage networks, QoS is configured for providing fair network utilization to each traffic class. For example, traffic may be classified using the DSCP field in the IP header, assigned to a storage traffic class, and allocated a minimum bandwidth guarantee. This ensures that traffic in other classes does not consume the full capacity of the link under congestion.

Effect of Other Traffic Classes on Storage Traffic Class

While using QoS in a shared storage network, when the storage traffic class is assigned a minimum bandwidth, such as 50% of the link capacity, be aware that this is not the maximum bandwidth. A traffic class can consume up to 100% of the link capacity when other classes do not have traffic. However, when other classes have traffic, the storage traffic class is limited to its bandwidth allocation (assuming 50%). This may cause sporadic poor performance due to packet drops.

It is a common misconception that after configuring QoS, traffic in other classes does not affect the storage traffic class. It depends on how a problem is approached. Consider a 10 GbE link with 5 Gbps allocated to storage traffic (iSCSI) and 5 Gbps allocated to other traffic. A host needs 8 Gbps of I/O throughput, which can be achieved via this link when there is no other traffic. However, when traffic in other class increase to 4 Gbps, the iSCSI traffic is limited to 6 Gbps by dropping the packets. Although congestion on the network port is controlled, the iSCSI application is affected because its I/O throughput degraded from 8 Gbps to 6 Gbps. The TCP layer is applying the backpressure to the upper layers in the end devices. In other words, congestion is moved from the network to the host.

In this condition, the duration of performance degradation of the iSCSI application matches with the increased traffic in the other traffic class.

This certainly adds a complication to troubleshooting congestion in shared storage networks. Monitoring per-class traffic and per-class congestion separately as well as combined at a link level can help in better understanding and detecting such problems.

Refer to Chapter 7, the section on the *Effect of Lossy Traffic on no-drop Class*, and the case studies for more details.

Configuring versus Operating a Shared Storage Network

Another important consideration is the operational state of a network. Just because a network is configured for sharing, doesn't necessarily mean that storage and other traffic flow through it simultaneously. For example, assume you configured a network with 50% bandwidth allocated to iSCSI and NVMe/TCP traffic and 50% bandwidth for other traffic. If there is no iSCSI and NVMe/TCP traffic at that time, at the data layer this network is no different than any other Ethernet network. On the other hand, if there is no traffic in the other classes, this network operates like a dedicated storage network. In other words, it is "configured" as a shared storage network but not "operating" as a shared storage network.

QoS Expertise

Users with a Fibre Channel background may find it interesting to know that QoS is an integral part of transporting storage traffic via Ethernet networks (lossless or lossy).

Even if an Ethernet network is dedicated to storage traffic and no other traffic is sent on it, configuring QoS is still needed. This is because you would still need to mark and classify the storage traffic, allocate minimum bandwidth to it, and apply this configuration consistently across the entire network. Even if you decide to allocate 100% bandwidth to the storage traffic, an overall QoS design and configuration is still needed because Active Queue Management (such as WRED and AFD), ECN, and other features are enabled via QoS.

Even if a GUI application or automation framework simplifies the initial QoS configuration, understanding its parameters is still needed for troubleshooting congestion in TCP storage networks.

In Chapter 7, the steps explained in the section on *Configuring Lossless Ethernet* are the same for TCP Storage networks using lossy Ethernet. A major difference for TCP is that its traffic class won't be assigned to a no-drop queue and PFC won’t be enabled for it. But PFC is only a small section of QoS configuration.

FCoE, RoCE, iSCSI, and NVMe/TCP in the Same Network

Ethernet networks can simultaneously transport lossless and lossy traffic. FCoE traffic can be assigned to a no-drop class. RoCE traffic can be assigned to another no-drop class. PFC must be enabled for no-drop classes that need lossless behavior.

iSCSI and NVMe/TCP traffic can be assigned to a separate class without PFC, and hence, packets may be dropped in this class. Assume that a minimum of 20% bandwidth is guaranteed to each of these three storage traffic classes. The rest 40% is assigned to other traffic.

Chapter 7, the section on *FC and FCoE in the Same Network* and *Multiple no-drop classes on the same Link* explains the details of congestion troubleshooting in no-drop classes. In addition to those details, note the following points:

1. Troubleshoot congestion in each no-drop class separately.
2. PFC Pause frames in no-drop classes do not affect the iSCSI or NVMe/TCP traffic in a separate class.
3. If the link utilization is high, it may result in spreading the congestion in each no-drop class separately toward the traffic source. However, in the iSCSI and NVMe/TCP traffic class, packets may be dropped, or ECN counters may increment.
4. As previously mentioned, traffic in FCoE, RoCE, and iSCSI or NVMe/TCP traffic classes can consume the entire link capacity in the absence of traffic in other classes. When the throughput of a class increases, other classes may be limited to their minimum bandwidth guarantee.
5. Packet drop is a symptom of congestion regardless of the traffic class.

Using multiple lossless and lossy traffic classes in a network is technically possible, but it makes congestion management more complex because of contention of network resources, such as bandwidth and buffers. Chapter 7, the section on *Multiple no-drop Classes on the Same Link* explains one such scenario. These environments should be carefully designed, monitored, and optimized as explained in this book.

iSCSI and NVMe/TCP in a Lossless Network

Some environments may use lossless Ethernet networks for iSCSI and NVMe/TCP traffic. In other words, the TCP traffic between iSCSI and NVMe/TCP initiators and targets is assigned to a no-drop traffic class with PFC. The reason for doing this is to avoid packet drops in the network due to congestion. In the absence of packet loss, TCP need not use its AIMD approach for congestion control which results in a sawtooth pattern of performance. This is explained earlier in the section on *TCP Congestion Control*. Hence, PFC with TCP allows for achieving consistent performance for iSCSI and NVMe/TCP.

In contrast, the reason for using lossy network for TCP is that TCP usually relies on packet drops to invoke its congestion control mechanisms. If packet drops are prevented by PFC in a lossless network, a TCP sender will continue to send more traffic, which will eventually lead to congestion spreading.

This section does not make a case for using a lossless versus lossy network (or vice-versa) for iSCSI and NVMe/TCP. In our experience, lossless networks for iSCSI are less common, but some environments may still use it. This book aims to educate users about using each network type and the pros and cons to each.

Congestion management differs significantly for lossy and lossless networks. The focus of this chapter is only on lossy network for TCP transport. If your environment uses lossless network for TCP traffic, then this chapter and Chapter 7 collectively explain congestion detection, troubleshooting, and prevention.

A key factor is the use of ECN for TCP traffic in lossless networks. The overall concept is explained in the earlier section on *Explicit Congestion Notification in TCP/IP Networks*. But in lossless networks, it works more like as explained in Chapter 7, the section on *Congestion Notification in Routed Lossless Ethernet Networks*.

iSCSI and NVMe/TCP with VXLAN

Virtual Extensible LAN (VXLAN) extends Layer 2 domains over a Layer 3 data center network. The end devices do not know that their traffic is passing through VXLAN tunnels. Also, the traffic classification based on the DSCP field in the IP header and ECN bits are preserved in VXLAN. Because of these reasons, everything that has been explained in this chapter applies to VXLAN environments as well. Refer to Chapter 7, the section on *Lossless Traffic with VXLAN* for a brief overview of VXLAN and how traffic classification and ECN bits are preserved.

Fibre Channel over TCP/IP (FCIP)

FCIP (defined in RFC 3821) is used for replication and backup services among data center sites. Storage arrays in these sites connect to a Fibre Channel fabric, whereas the sites have IP connectivity among them. FCIP transports Fibre Channel frames via a TCP connection, essentially merging the remote fabrics. The FCIP endpoints are switches that have Fibre Channel and Ethernet ports, such as Cisco MDS 9220i. They encapsulate (and decapsulate) FC frames within FCIP, TCP, IP, and Ethernet headers. Refer to Chapter 1, the section on FCIP for FCIP packet format.

TCP Optimizations for Storage Traffic on Cisco FCIP Switches

Those who have experience in operating FCIP may recall that the TCP stack on FCIP switches is enhanced for storage traffic. This achieves high-performance requirements for synchronous and asynchronous replication. The following are these enhancements.

1. **MTU**: Because the maximum Fibre Channel frame size is 2148 bytes, configuring IP MTU of 2300 bytes (after accounting for the headers) on all the links between the FCIP end points avoids fragments, and therefore achieves better performance (Example 8-8).

2. **Selective Acknowledgements (SACK)**: SACK increases performance during packet drops by selectively acknowledging only the received packets, and hence the sender retransmits only the lost packets instead of retransmitting all the packets after the lost packets even if these packets might have reached the receiver. By default, SACK is enabled on Cisco FCIP switches, although Example 8-8 shows explicit configuration.

2. **Aggressive slow-start and recovery during congestion**: A common practice for FCIP traffic is to allocate a dedicated capacity on the WAN between the TCP endpoints. For example, 1 Gbps may be dedicated to FCIP traffic. Hence, there is no need for TCP to go through the typical slow-start because it can instantly consume the entire 1 Gbps (max-bandwidth-mbps in Example 8-8). Also, when a packet is dropped, the congestion window is reduced to lower the throughput only by a small value, such as 90% of the max-bandwidth (min-available-bandwidth-mbps in Example 8-8). These changes significantly improve TCP performance as compared to the standard implementation. Figure 8-18 compares the performance of a standard TCP implementation with the TCP implementation on Cisco FCIP switches.

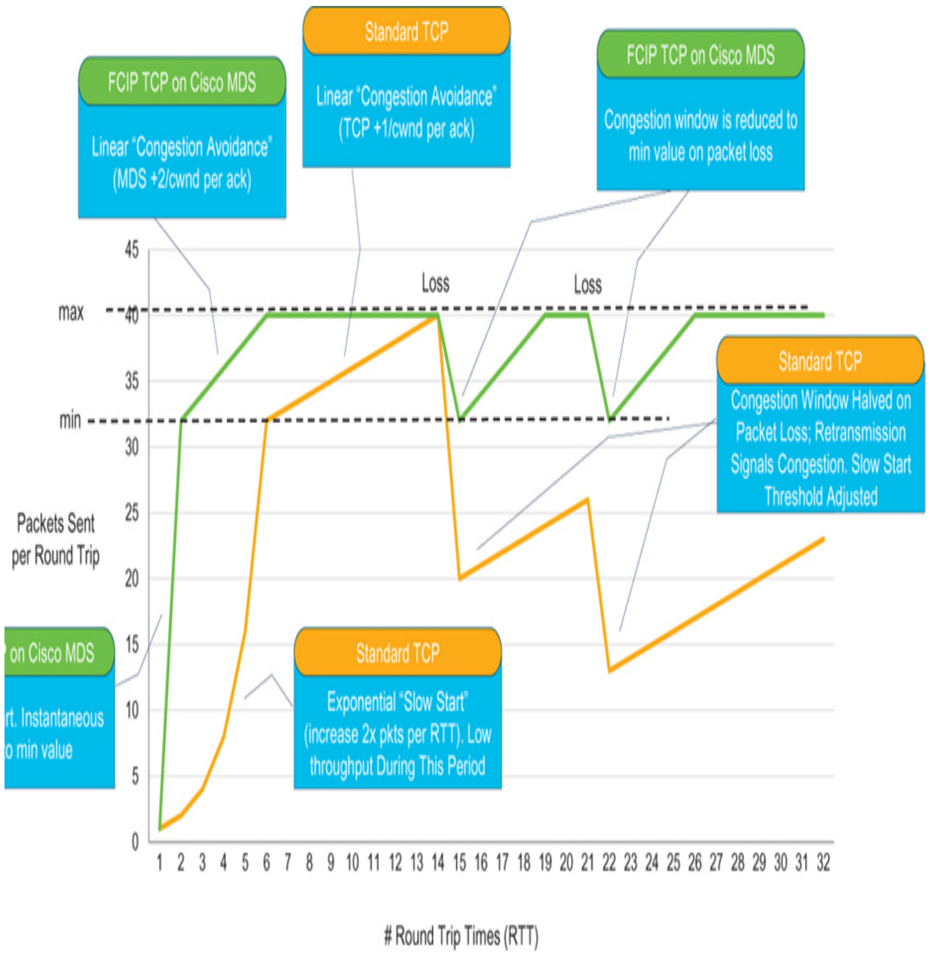


Figure 8-18 *Enhanced TCP implementation on Cisco FCIP switches*

4. **Configuration of the expected RTT**: Allowing the RTT to be configured allows a TCP scale option to be negotiated during the TCP 3-way handshake. This allows for an exponentially larger TCP window size to ensure that the FCIP interface can reach its full max-bandwidth.
5. **Number of TCP Connections**: By default, Cisco FCIP switches create two TCP connections per FCIP link. One is used for data frames, whereas the other is used for control frames. The performance of FCIP can be increased by increasing the number of TCP connections to five (Example 8-8).

Example 8-8 *TCP optimizations on Cisco FCIP Switches*

```
!
fcip profile 1
  port 5001
  ip address a.b.c.1
  tcp max-bandwidth-mbps 1000 min-available-
bandwidth-mbps 900 round-trip-time-ms
1
  tcp sack-enable
!
interface fcip1
  use-profile 1
  peer-info ipaddr a.b.c.2 port 5001
  tcp-connections 5
  no shutdown
!
interface IPStorage1/1
  ip address a.b.c.1 255.255.255.0
  switchport mtu 2300
  no shutdown
!
```

The TCP stack on Cisco FCIP switches relies on packet drops for congestion avoidance and does not use ECN because the WAN links that carry FCIP traffic are typically owned by an Internet Service Provider. Not all the devices in the end-to-end traffic path may support ECN or may not transport ECN bits correctly while tunneling the packets.

Detecting Congestion on FCIP links

As mentioned earlier, FCIP end points encapsulate/decapsulate Fibre Channel frames to TCP/IP packets for sending them over WAN. As Figure 8-19 shows, if a packet loss is detected, the TCP layer on the FCIP endpoints (Switch-1 or Switch-2) retransmits the packets. But if excessive packets are lost, TCP sender reduces its rate to a much lower value. In this condition, Switch-1 must equalize the rate of traffic received on Fibre Channel ports to the rate of transmission on the TCP connections. For this rate equalization, Switch-1 uses B2B flow control on the Fibre Channel ports. This results in Rx credits unavailability on Fibre Channel port on Switch-1 and Tx credits unavailability on its directly connected neighbor.

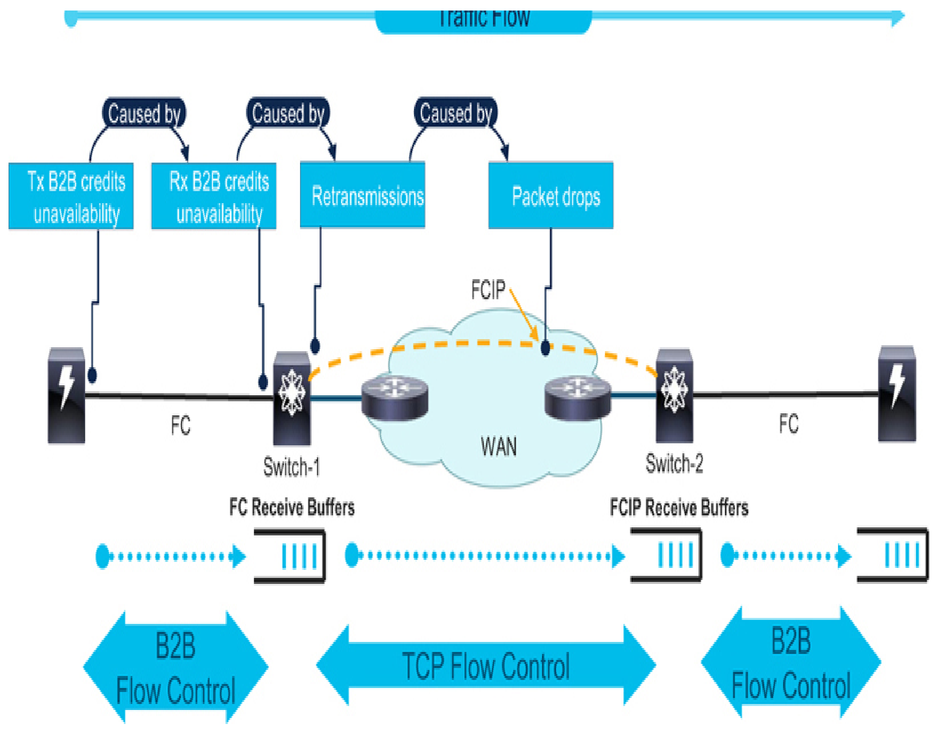


Figure 8-19 *Detecting congestion caused by FCIP links*

Detecting congestion on FC links remains the same as explained in Chapter 3.

In this condition, although congestion has spread to Fibre Channel fabric, the cause of this congestion is the sluggish performance of TCP connection, and its source is somewhere in the WAN link. Refer to the earlier section on *Congestion Severities and Levels* and *Methodology* for details about how you should find the source of this congestion.

Cisco FCIP switches provides multiple ways to measure the performance of TCP connections. For example, to find retransmissions on FCIP link, use **show ips stats tcp interface** command (Example 8-9) on Cisco MDS switches.

Example 8-9 *TCP stats for FCIP connections*

```
show ips stats tcp interface ipstorage1/1

TCP statistics for port IPStorage1/1
    TCP send stats
      1309156584 segments, 2728716752884 bytes
      1302369617 data, 6786957 ack only packets
      292 control (SYN/FIN/RST), 0 probes, 10 window
updates
```

```
      0 segments retransmitted, 0 bytes
      0 retransmitted while on ethernet send queue,
71907 packets split
      18460 delayed acks sent
    TCP receive stats
      678581062 segments, 39819000 data packets in
sequence, 5133261092 bytes in
sequence
      656915228 predicted ack, 39818282 predicted
data
      0 bad checksum, 0 multi/broadcast, 0 bad
offset
      0 no memory drops, 0 short segments
      0 duplicate bytes, 0 duplicate packets
      0 partial duplicate bytes, 0 partial duplicate
packets
      0 out-of-order bytes, 0 out-of-order packets
      0 packet after window, 0 bytes after window
      0 packets after close
      656956974 acks, 2728672844760 ack bytes, 0 ack
toomuch, 0 duplicate acks
      0 ack packets left of snd_una, 0 non-4 byte
aligned packets
      0 window updates, 0 window probe
      292 pcb hash miss, 292 no port, 0 bad SYN, 0
paws drops
    TCP Connection Stats
      0 attempts, 0 accepts, 0 established
      0 closed, 0 drops, 0 conn drops
      0 drop in retransmit timeout, 0 drop in
keepalive timeout
      0 drop in persist drops, 0 connections drained
    TCP Miscellaneous Stats
      918593 segments timed, 656956974 rtt updated
      0 retransmit timeout, 0 persist timeout
      0 keepalive timeout, 0 keepalive probes
    TCP SACK Stats
      0 recovery episodes, 0 data packets, 0 data
bytes
      0 data packets retransmitted, 0 data bytes
retransmitted
      0 connections closed, 0 retransmit timeouts
```

Typically, give special attention to more than 0.05% of retransmissions, which can be calculated by Percentage retransmission = (segments retransmitted / data segments) x 100. Look for other indications of a problem in the IP network, such as duplicate packets, out-of-order packets, and duplicate acks.

Check On-Board Failure Logging (OBFL) for 60 second retransmit rate exceeding 0.05% and measured SRTT exceeding configured RTT (Example 8-10)

Example 8-10 *TCP retransmission rate and SRTT increase on OBFL*

```
show logging onboard error-stats

----------------------------
Module: 1 show clock
----------------------------
2022-09-02 13:42:56
---------------------------------
Module: 1 error-stats
---------------------------------
----------------------------------------------------
----------------------------
 ERROR STATISTICS INFORMATION FOR DEVICE: FCIP MAC
----------------------------------------------------
----------------------------
Notes:
     - Sampling period is 60 seconds
     - TCP_RETRANS_RATE_XCD_THRESH logged only when
       Delta retransmit rate > retxmt threshold
     - TCP_SRTT_XCD_CONF_RTT logged only when
       SRTT shown in count column > 1.3 * configured
RTT

----------------------------------------------------
---------------------------
Interface |Error Stat Counter Name      |Delta
|Count    |Time Stamp
<tcp_con> |                             |ReTrans%
|or       |MM/DD/YY HH:MM:SS
```

```
          |                            |or SRTT%
|SRTT us  |
----------------------------------------------------
---------------------------
fcip12<0> |TCP_SRTT_XCD_CONF_RTT        |36.661
|42000    |09/02/22 00:22:18
fcip12<1> |TCP_SRTT_XCD_CONF_RTT        |39.915
|43000    |09/02/22 00:22:18
fcip12<1> |TCP_RETRANS_RATE_XCD_THRESH  |5.479    |4
|09/02/22 00:22:18
```

The 24/10 SAN Extension Module on MDS 9700 supports TxWait and RxWait counters for FCIP. TxWait on Ethernet (IPStorage) ports on this module indicate egress congestion on one or more FCIP interfaces on that port. Likewise, RxWait on Ethernet (IPStorage) ports on this module indicates ingress congestion on one or more FCIP interfaces on that port.

Refer to Example 8-11. This output is similar to the TxWait output for FC interfaces. In this output, Eth X/1 is associated with IPStorage X/1, Eth X/2 is associated with IPStorage X/2, and so on (X is the module number). Note that the Ethernet ports on the faceplate of this module are called IPStorage ports and are written in short as IPS ports.

Example 8-11 *TCP retransmission rate and SRTT increase on OBFL*

```
MDS9710-1# show logging onboard rxwait module 2
---------------------------------
 Module: 2 rxwait count
---------------------------------
----------------------------
Module: 2 show clock
----------------------------
2022-04-25 15:41:33
---------------------------------
Module: 2 rxwait
---------------------------------
Notes:
     - Sampling period is 20 seconds
     - Only rxwait delta >= 100 ms are logged
```

```
----------------------------------------------------
----------------------------
| Interface      | Delta RxWait Time     | Congestion
| Timestamp
|
|                | 2.5us ticks | seconds |
|
|
----------------------------------------------------
----------------------------
| Eth2/1(VL3)    |  7794923    |   19    |       97%
| Mon Apr 25 15:41:28 2022
| Eth2/1(VL3)    |  7797819    |   19    |       97%
| Mon Apr 25 15:41:08 2022
| Eth2/1(VL3)    |  7794669    |   19    |       97%
| Mon Apr 25 15:40:48 2022
| Eth2/1(VL3)    |  7796037    |   19    |       97%
| Mon Apr 25 15:40:28 2022
```

Modified TCP Implementations

Standard TCP with ECN detects the presence of congestion and reacts to it by reducing the congestion window as if packets had been dropped. In data center environments, where the links are of high capacity and end-to-end delay is low, the standard TCP congestion control mechanisms, such as reducing the congestion window by half, maybe an overreaction leading to underutilization of the links. Also, ECN conveys only the presence of congestion. It does not convey the extent or severity of congestion.

Hence, like TCP implementation on Cisco FCIP switches, modified TCP implementations may be available on the end devices for intra-data center communication. Data Center TCP (DCTCP) is one such example. It is described in an information RFC 8257, but it is not standardized.

DCTCP changes traditional ECN processing by estimating the fraction of bytes that encounter congestion rather than simply detecting that some congestion has occurred. DCTCP then scales the TCP congestion window based on this estimate.

This method achieves high-burst tolerance, low latency, and high throughput within a data center network.

The only difference between a DCTCP sender and a standard TCP sender is in how each reacts to receiving an ACK with the ECN-Echo flag set. This is explained earlier in the section on *Explicit Congestion Notification in TCP/IP Networks*. Other features of TCP such as slow start, additive increase in congestion avoidance, or recovery from packet loss are left unchanged. While standard TCP reduces its window size by half, DCTCP reduces the window only slightly during low severity of congestion. When congestion is severe, DCTCP reduces the window to half, just like standard TCP.

Although modified TCP implementations, such as DCTCP, may deliver better performance, their use must be carefully analyzed because of the following reasons:

1. **Configuration overhead**: DCTCP or similar modified TCP implementations may not be enabled by default. Hence, using them involves a configuration overhead of enabling them on all devices.
2. **Expertise**: Deploying DCTCP or similar modified implementations needs relevant expertise in the form of professional services or developing the expertise in-house.
3. **Hybrid environments**: An end device that uses DCTCP or similar non-standard implementations may need to communicate with the devices that do not support DCTCP. This requires a dual TCP stack, which further increases the configuration overhead.
4. **Co-existence with standard TCP flows**: DCTCP and standard TCP flows do not coexist well in the same network. As a workaround, DCTCP flows can be segregated using the DSCP field in the IP header and assigned to separate traffic classes. This further increases the configuration overhead.
5. **Slow Start:** DCTCP uses the standard slow start algorithm of standard TCP, which may not be considered

adequate for bursty storage traffic by some users, hence there may be need of another TCP implementation.

One of the major reasons in favor of using TCP as a transport of storage protocols (iSCSI and NVMe/TCP) is its ubiquitous nature, which means it can work out of the box with minimal changes. This benefit is not available when using modified TCP implementation, such as DCTCP.

The decision of using a modified TCP implementation depends on the requirements of a network. If you decide to use it, the details that this chapter explains for detecting, troubleshooting, and preventing congestion will still be applicable.

Summary

This chapter explains congestion management in TCP/IP storage networks for transporting block-storage protocols, primarily, iSCSI and NVMe/TCP, although the details are relevant for all kinds of storage regardless of their type — Block, File, or Object.

The “storage protocols” use TCP as their “transport protocol”, which means that congestion management for iSCSI and NVMe/TCP traffic boils down to congestion management in TCP/IP networks.

TCP has robust flow-control and congestion-control mechanisms, which move the congestion from the network to the end devices. But the real cause of congestion does not magically go away. To achieve the low latency and burst requirements of storage traffic, congestion issues must be understood, detected, and reduced so that the application performance is acceptable.

Congestion in TCP storage networks can be detected by packet drops, ECN counters, bit errors, high link utilization, and microburst events. Troubleshooting can be quick and accurate when using remote monitoring platforms that can analyze the trends and seasonality, perform comparative analysis, provide single-pane-of-glass visibility in the end-to-end path, and generate alerts to notify the users proactively.

The underlying cause of the congestion can be understood by the application I/O profile, such as the timing, size, type, and rate of I/O operations. Essentially, the application I/O profile helps in understanding why the network has traffic or congestion. Correlating I/O Operations with traffic patterns and network congestion greatly enhances the congestion detection and troubleshooting workflow.

Understanding QoS is an important skill to have because most of the Active Queue Management mechanisms (such as WRED and AFD) and ECN are configured by Modular QoS CLI or similar. While configuring the Active Queue Management mechanisms, the goal should be to achieve low queue utilization to minimize the queuing latency and maximize the headroom for traffic bursts. The next best option is to achieve an occasional empty queue utilization. Essentially, for storage networks, prefer an underutilized link over an overutilized link.

Throughout this chapter, comparisons are explained with Fibre Channel fabrics that have decades of history of transporting storage traffic in all kinds of environments across the world. All these transport types have the same upper layers—SCSI and NVMe—as their source of traffic. Although TCP networks that use lossy Ethernet do not spread the congestion towards the traffic source, it is beneficial to learn from the long history of Fibre Channel and apply the knowledge to TCP storage networks to prevent congestion issues proactively.

References

- Comer, Douglas E. (2006). Internetworking with TCP/IP: Principles, Protocols, and Architecture. Vol. 1 (5th ed.). Prentice Hall. ISBN 978-0-13-187671-2.
- Transmission Control Protocol (TCP) - https://datatracker.ietf.org/doc/html/rfc9293

- TCP Congestion Control - https://datatracker.ietf.org/doc/rfc5681/
- TCP Selective Acknowledgment Options - https://datatracker.ietf.org/doc/rfc2018/
- Computing TCP's Retransmission Timer - https://datatracker.ietf.org/doc/html/rfc6298
- Data Center TCP (DCTCP): TCP Congestion Control for Data Centers - https://www.rfc-editor.org/rfc/rfc8257.html
- Data Center TCP (DCTCP) - SIGCOMM'10, August 30–September 3, 2010, New Delhi, India. (https://people.csail.mit.edu/alizadeh/papers/dctcp-sigcomm10.pdf)
- TCP/IP Illustrated, Vol. 1: The Protocols (Addison-Wesley Professional Computing Series)
- TCP/IP Illustrated: The Implementation
- TCP/IP Illustrated: v. 3: TCP for Transactions, HTTP, NNTP and the Unix Domain Protocols (Addison-Wesley Professional Computing Series)
- R. Griffith, Y. Chen, J. Liu, A. Joseph, and R. Katz. Understanding TCP incast throughput collapse in datacenter networks. In WREN Workshop, 2009 (https://dl.acm.org/doi/10.1145/1592681.1592693)
- RFC 7143: Internet Small Computer System Interface (iSCSI) Protocol (Consolidated) - https://datatracker.ietf.org/doc/html/rfc7143
- RFC 4171: Internet Storage Name Service (iSNS) https://datatracker.ietf.org/doc/html/rfc4171
- NVM Express TP8009 Automated Discovery of NVMe-oF Discovery Controllers for IP Networks
- NVM Express TP8009 Centralized Discovery Controller (CDC)
- NVM Express® Base Specification 2.0. Available from https://nvmexpress.org/

- NVM Express over Fabrics, Revision 1.1. Available from http://www.nvmexpress.org.
- NVM Express® TCP Transport Specification. Available from https://nvmexpress.org/
- NVM Express® RDMA Transport Specification. Available from http://www.nvmexpress.org.
- Intelligent Buffer Management on Cisco Nexus 9000 Series Switches White Paper: https://www.cisco.com/c/en/us/products/collateral/switches/nexus-9000-series-switches/white-paper-c11-738488.html
- Storage Area Network Extension Design and Operation, Cisco Live, Las Vegas, 2016.
- Virtana Infrastructure Performance Management: https://www.virtana.com/products/infrastructure-performance-management/
- Augtera NVMe/TCP Congestion Detection Solution Brief: https://augtera.com/wp-content/uploads/2022/10/Augtera-NVME-White-Paper.pdf
- RFC 3821: Fibre Channel Over TCP/IP (FCIP); https://www.rfc-editor.org/rfc/rfc3821.html
- End-to-End QoS Network Design: Quality of Service for Rich-Media & Cloud Networks (Networking Technology) 2nd Edition. Cisco Press. ISBN-13: 978-1587143694
- IETF Recommendations Regarding Active Queue Management: https://www.rfc-editor.org/rfc/rfc7567.html

Chapter 9. Congestion Management in Cisco UCS Servers

This chapter covers the following topics

- Cisco UCS Architecture, traffic flow, and flow control
- Understanding congestion in a UCS domain
- The UCS Traffic Monitoring (UTM) App and its use in detecting and troubleshooting congestion in UCS servers
- Case studies

The focus of this chapter is explaining the use of a remote monitoring platform (such as UTM) for detecting congestion in Cisco UCS Servers. UTM demonstrates that even minimal congestion metrics can detect most (if not all) congestion issues when the data is correlated and analyzed carefully. This chapter, however, does not intend to make you an expert in Cisco UCS. Additionally, the examples in this chapter consider only UCS Manager. The relatively new Intersight Managed Mode (IMM) is not explained, although a similar workflow can be used.

Cisco UCS Architecture

Cisco Unified Computing System (UCS) has a 3-layer architecture (Figure 9-1).

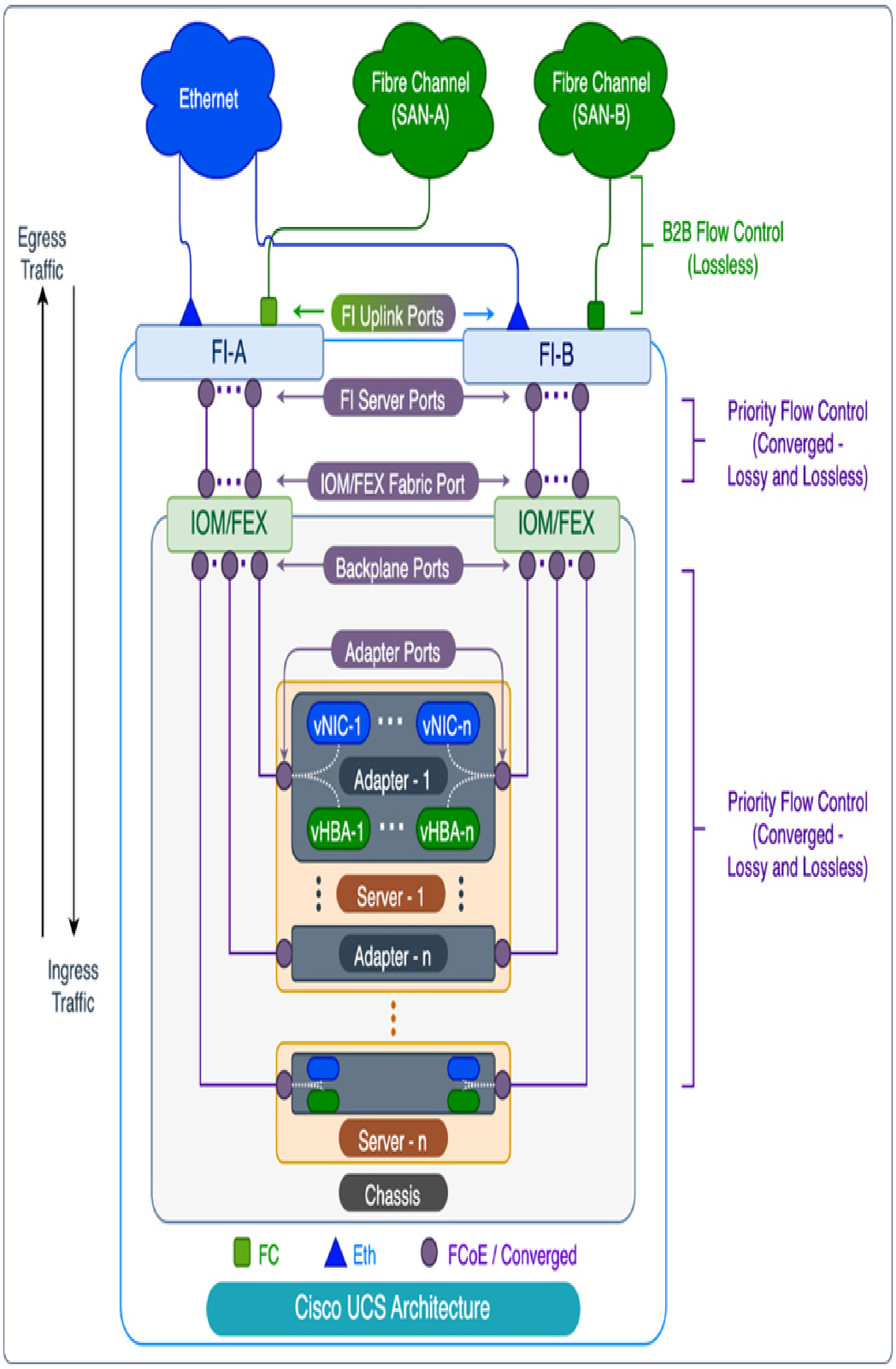


Figure 9-1 *Cisco UCS architecture*

1. **Fabric Interconnects (FI)**: Fabric Interconnects make the top layer in the UCS architecture. Physically, FIs are standalone switches with Ethernet and Fibre Channel ports. These are typically deployed in a pair (FI-A and FI-B) to provide redundancy. The ports on the FIs that connect to the Ethernet networks and Fibre Channel fabrics are called Uplink Ports, whereas the ports towards servers are called Server Ports. FIs are based on various

Nexus switches, hence, their NX-OS commands are similar.

2. **IO Modules (IOM) or Fabric Extenders (FEX)**: IOMs/FEXs make the middle layer in the UCS architecture. The ports on IOM/FEX that connect to the FI Server Ports are called Fabric Ports, whereas those that connect to the servers are called Backplane Ports. The Fabric Ports are also known as Network Interfaces (NIFs) and the Backplane Ports are also known as Host Interfaces (HIFs). Similar to the FIs, the IOM/FEX are also deployed in a pair, and each connects to FI-A or FI-B. Physically, the IOM/FEX are inside a UCS chassis, or they can be standalone switches.
3. **Servers**: Servers make the bottom layer in the UCS architecture. Each Server has one or more Adapters, called Virtual Interface Cards (VICs). These VICs typically create multiple Virtual HBAs (vHBAs) for Fibre Channel traffic and Virtual NICs (vNICs) for Ethernet traffic. vNICs are also created for RoCE traffic. The Ports on these Adapters connect to the IOM/FEX Backplane Ports. Typically, there are at least two adapter ports for connecting to each IOM/FEX, which further connect to FI-A and FI-B. In some cases, the servers may directly connect to the FI Server Ports, eliminating the middle layer of IOM/FEX. Physically, the blade servers are inside a UCS chassis, whereas the rack servers have their own chassis.

UCS Domain

A pair of FIs makes a UCS domain. It can have up to 20 chassis. Each chassis can have up to 8 blade servers. Therefore, a UCS domain can have up to 160 servers, although some of these may be blade servers and some may be rack servers.

Traffic Flow in a UCS Domain

As Figure 9-1 shows, egress traffic from a server takes the following path.

1. Egress Fibre Channel traffic exits via vHBAs and Ethernet traffic exits via vNICs.
2. Exits Servers via Adapter Ports.
3. Enters via IOM/FEX Backplane Ports and exits via IOM/FEX Fabric Ports.
4. Enters via FI Server Ports and finally exits the UCS domain via FI Uplink Ports.

Ingress traffic to a server takes the reverse path.

1. Ingress traffic enters via FI Uplink Ports and exits via FI Server Ports.
2. Enters via IOM/FEX Fabric Ports and exits via IOM/FEX Backplane Ports.
3. Enters Servers via Adapter Ports.
4. Fibre Channel traffic enters via vHBAs and Ethernet traffic enters via vNICs.

These traffic paths should make clear that Server ingress traffic is egress on FI Server Ports and IOM/FEX Backplane Ports. Likewise, Server egress traffic is ingress on FI Server Ports and IOM/FEX Backplane Ports.

A Server Adapter allows the creation of multiple Virtual Interfaces (VIF). Different VIFs are pinned to different FI Uplinks based on their type. For example, vHBAs are pinned to Fibre Channel Uplinks and vNICs are pinned to Ethernet Uplinks. An Adapter may also create vNICs for RoCE/RoCEv2, which typically uses PFC for lossless transport via a shared Ethernet network that also carries lossy traffic. Multiple VIFs share the same Adapter Port(s) for sending and receiving traffic.

The Uplink Ports can be physical (such as FC 1/1 or Eth 1/2) or they can be aggregated to port-channels (such as LAN-PC-1 or SAN-PC-1).

Flow Control in a UCS Domain

As Figure 9-1 shows, the use of flow control on the FI Uplink Ports depends on their type. Fibre Channel Uplink Ports use B2B flow control. Ethernet Uplink Ports do not use a hop-by-hop flow control for connecting to a lossy Ethernet network. However, they may be configured for PFC for transporting lossy and lossless traffic on the converged links. FCoE, RoCE, and RoCEv2 are the key use cases for converged FI Uplink Ports.

The internal links in a UCS domain are configured for PFC. In other words, the FI to IOM/FEX (Server) links and IOM/FEX to Adapter (Backplane) links are converged Ethernet links. The lossless traffic (such as Fibre Channel (FCoE) and RoCE traffic) is assigned to no-drop classes which are flow controlled, whereas lossy traffic is assigned to other classes which are not flow-controlled.

Although the internal links are always configured for PFC, whether the flow control is used or not depends on the type of traffic on those links. If no server in a UCS domain has a vHBA and no vNIC is configured for RoCE (or similar that needs lossless Ethernet transport), there will not be any traffic in the no-drop classes. Therefore, under congestion, there won't be any Pause frames reported on any ports. Refer to Chapter 7, "Congestion Management in Ethernet Storage Networks," for details on Pause frames and PFC.

Understanding Congestion in a UCS Domain

Detecting and troubleshooting congestion in a UCS domain is the same as explained in Chapter 7, "Congestion Management in Ethernet Storage Networks," and Chapter 8, "Congestion Management in TCP Storage Networks". Lossless traffic uses hop-by-hop flow control. As a result, congestion spreads to the traffic source victimizing other unrelated traffic. Lossy traffic does not use hop-by-hop flow control, and hence, it is dropped under congestion.

The cause of the congestion can be any of the following.

1. **Slow-drain**: Slow-drain happens when a device has a slower processing rate as compared to the rate at which the frames are being delivered to it. When a Server in a UCS domain becomes a slow-drain device, it results in the Adapter Ports sending PFC Pause frames to the IOM/FEX Backplane Ports.
2. **Over-utilization of a Link**: Congestion due to over-utilization happens when a port is already transmitting at its maximum capacity, yet it has more frames than can be transmitted on it. When a Server in a UCS domain causes congestion due to over-utilization, it results in the Server receiving a large amount of traffic without sending PFC Pause frames to the IOM/FEX. The IOM/FEX in turn sends PFC Pause frames to the FI to slow down the traffic. The congestion then spreads to the storage networks via the FI uplink ports either by sending PFC Pause frames (Ethernet) or withholding/slowing R_RDYs (Fibre Channel).
3. **Bit errors on a Link**: Bit errors may interfere with hop-by-hop flow control resulting in congestion or worsening existing congestion.
4. **Excessive Length of a Link**: Links with hop-by-hop flow control (such as B2B flow control and PFC) require a minimum number of buffers (or credits) for the speed, frame size, and distance of the link to be able to transmit at the full capacity on a link. However, this factor can be ignored in a UCS domain because the links are only a few meters long within the same rack or to neighboring racks.

These are the same causes as explained in Chapter 1, and then in Chapter 2 for Fibre Channel fabrics and Chapter 7 for lossless Ethernet networks. This chapter expands on these as they apply to Cisco UCS.

Detecting Congestion in a UCS Domain

Detecting congestion in a UCS domain is like doing it in any other converged network. From the perspective of hop-by-hop flow control, the IOM/FEX should be considered as the edge switches for the Servers, which connect to the FIs using ISLs. The FIs may further connect to a storage device directly or via a network.

Because traffic (and therefore congestion) are directional, the symptoms of ingress congestion are in the reverse direction as compared to egress congestion.

This section explains detecting congestion for ingress and egress traffic in a UCS domain using the terms ‘excessive Pause frames’ or ‘reduced rate of B2B credits (R_RDY)’. Be aware that there are typically some Pause frames always flowing to achieve hop-by-hop flow control. Understand the level of ‘normal’ congestion that is present so it can be determined what is above the normal levels or excessive.

Ingress Congestion

In a UCS domain, ingress congestion typically starts at a server because it has slower processing rate as compared to the rate at which the frames are being delivered to it (slow-drain) or the IOM/FEX Fabric Ports receive traffic faster than can be sent on their Backplane Port to the Servers (over-utilization). A slow-drain server can be detected by excessive Pause frames sent by its Adapter Ports to the IOM/FEX Backplane port. Congestion due to over-utilization can be detected by high egress utilization of the IOM/FEX Backplane Port and a lack of PFC Pause frame sent by the Adapter ports to the IOM/FEX Backplane port. Both these causes of congestion result in IOM/FEX Fabric Ports sending Pause frames to the FI Server Ports to slow down the ingress traffic. This congestion spreads further from the FI Uplink Ports to their neighbors. If the Uplink is a Fibre Channel port, it reduces the rate of sending of B2B credits (R_RDY). However, if the Uplink is a converged Ethernet port, it sends Pause frames to slow down the ingress lossless traffic.

Egress Congestion

When congestion is against egress traffic, the culprit is generally outside the UCS domain. In this condition, the Fibre Channel FI Uplink Ports receive B2B credits at a reduced rate, whereas the converged Ethernet Uplink Ports receive excessive Pause frames. Consequently, the FI Server Ports send Pause frames to the IOM/FEX Fabric Ports. This congestion spreads toward the Servers resulting in sending Pause frames from IOM/FEX Backplane Ports to Server Adapter Ports.

In this case, the source and cause of the congestion can be found using the troubleshooting methodology explained in Chapter 4, "Troubleshooting Congestion in Fibre Channel Fabrics," and Chapter 7, "Congestion Management in Ethernet Storage Networks". The scope of this chapter is limited only to finding the FI Uplink Port in the traffic path when troubleshooting egress congestion.

In our experience, egress congestion in a UCS domain is not as frequent as the ingress congestion scenarios.

Congestion Between FI Server Ports and IOM/FEX Fabric Ports

A less common scenario is when the links between FI Server Ports and IOM/FEX Fabric Ports are over-utilized. As mentioned, from the perspective of flow control, consider these links as ISLs. Although no one server receives or sends excessive traffic by itself, collective traffic of multiple servers may lead to over-utilization of these links. Depending on the traffic direction, it may lead to ingress congestion or egress congestion. The key difference to remember is that server ingress traffic is egress on FI Server Ports, whereas Server egress traffic is ingress on FI Server Ports.

Besides that, this condition is less common but when it happens, it spreads congestion the same way on the FI Uplink Ports for congestion against ingress traffic and on the IOM/FEX Backplane Ports for congestion against egress traffic.

UCS Congestion Detection Notes

Note the following points for detecting congestion in a UCS domain.

1. At the time of this writing, UCS does not detect the duration for which transmission was stopped due to B2B flow control on Fibre Channel Uplink Ports and due to PFC on any Ethernet port. In other words, UCS does not report TxWait or RxWait. A recent development may result in reporting of TxWait on the UCS VICs in the future. However, we are not sure if this new development is released before this writing. Refer to the Cisco release notes for the latest information.

2. At the time of this writing, the Fibre Channel Uplink Ports on the FIs do not increment the “B2B credits transitions to/from zero” counter. Although the NX-OS command **show interface fc counters** may show “B2B credit transitions from zero”, these values do not increment and always show zero. Therefore, we recommend detecting congestion on the peer of the FI Uplink Ports, such as Cisco MDS switches.

3. The number of Pause frames is the only mechanism that UCS has for detecting congestion due to slow-drain. The count includes both Pause and Un-Pause (see 8b below). These are shown by NX-OS command **show interface priority-flow-control**. Refer to Chapter 7, the section on *Number of Pause Frames* for the output of this command and how to use it.

4. The initial firmware releases of the 4th generation (6400) and 5th generation (6500) FIs do not report Pause frames on the IOM/FEX Backplane Ports under the NX-OS command **show interface priority-flow-control**. The shown values stay at zero and do not increment. However, the later releases resolved this issue. Refer to the Cisco release notes for more details. The key point to understand is that before relying on this output, verify that the values increment and do not stay at zero, although zero is a technically valid value.

5. While monitoring Pause frames on the Server Adapter Ports that connect to the IOM/FEX Backplane Ports, be

aware that Pause frames are counted per port or link, and not per vNIC or vHBA.

6. Ingress Pause must be correlated with egress traffic on a port. Likewise, egress Pause must be correlated with ingress traffic on a port.
7. Monitoring Pause frames on just one end of a link should be enough.
8. Just the number of Pause frames is not a strong symptom for detecting congestion because
    a. A nominal rate of the Pause frames does not necessarily indicate congestion.
    b. The Pause frames may stop the traffic (non-zero quanta) or they may resume (Un-Pause) the traffic (zero quanta). However, just the number of Pause frames does not convey how many stopped the traffic and how many resumed the traffic.
    c. UCS shows only an accumulated value of Pause frames. It is not known when their value incremented.

    Chapter 7 explains these limitations in the section *Troubleshooting Congestion in Spine-Leaf Topology*. Chapter 7 also explains that these limitations can be overcome up to an extent by using a remote monitoring platform, which continuously polls the number of Pause frames to maintain a time and date-stamped history. Refer to the section *Troubleshooting Congestion using a Remote Monitoring Platform* to understand how these platforms can perform comparative and time-based analysis to find the source and cause of congestion.

The earlier chapters explain these details with the help of NX-OS commands, examples, and case studies, such as:

1. Chapter 6, the section *Case study 1 — A Bank Avoided Congestion by Traffic Segregation* explains a congestion segregation technique using Cisco UCS Servers.
2. Chapter 7, the sections *FC and FCoE in the Same Network* and *Multiple no-drop Classes on the Same Link* apply directly to Cisco UCS Servers.

In addition to monitoring the congestion counters, tracing the traffic path within a UCS domain requires an understanding of various commands, which may differ based on the model type. These details are explained in the official Cisco documentation and are outside the scope of this book. The focus of this chapter is on detecting and troubleshooting congestion using the UCS Traffic Monitoring (UTM) App.

The UCS Traffic Monitoring (UTM) App

UTM is a full-blown traffic monitoring application for Cisco UCS environments. It was developed by Paresh Gupta (co-author of this book).

UTM has been immensely successful in finding the source and cause of congestion in UCS domains across hundreds of production environments across the world. This is because it offers ready-made comparative and time-based analysis (trending and seasonality) and correlates server traffic with the FI Uplink Ports.

UTM has many use cases, such as:

1. Faster troubleshooting
2. Congestion monitoring
3. Capacity planning
4. Auditing
5. Traffic optimization, load-balancing, port-channel verification, etc.
6. Assisted change management
7. Reporting
8. Traffic trends and seasonality
9. Errors, Drops, CRC, FCS, etc.

But this chapter focuses only on congestion detection and troubleshooting use cases.

The Journey of UTM

The first conversation about UTM started in the Spring of 2019 when Craig Schaff, a Technical Solutions Architect in the Cisco sales organization, called Paresh Gupta, a Technical Marketing Engineer in the Data Center Networking Business Unit at Cisco. Craig had many customers who were looking for traffic visibility within their UCS domains because of long-standing performance and congestion issues.

Paresh was not part of the UCS business unit, but these were still Cisco customers with whom Paresh has been working in his Technical Marketing Engineering role. So, he spent the following weekend and evenings learning about the Cisco UCS XML APIs, watched a few Cisco Live presentations, and got help from other UCS experts (such as Eric Williams and John McDonough). The result—an initial prototype of UTM was ready by the following Friday that could visualize the vNIC and vHBA traffic.

Paresh showed this initial prototype to Craig who arranged meetings with his customers to seek initial feedback. During Cisco Live, San Diego in June 2019, UTM was discussed with the first would-be user from a multinational retail company. Their interest and feedback were amazingly positive. But it was clear that a prototype is far from being used in production environments.

In July 2019, Paresh re-wrote the entire app with many enhancements primarily focused on enterprise-class performance and user experience resulting in the birth of version 0.2. Over the next few months, by word of mouth, it spread as an "app to monitor UCS servers". Many large enterprises tried it in their production environments. This is when, with the help of Art Scrimo, a Product Manager in the Data Center Networking Business Unit at Cisco, the app was named UTM.

In December 2019, Paresh published UTM on GitHub so that anybody could use it for free. In the following months, it achieved a global reach and started getting deployed in all kinds of organizations across the world. Many users started pouring in their feedback by opening issues on GitHub and reaching out directly. The wide adoption of UTM kept the motivation alive during late nights and weekends.

Many organizations allowed access to their production environments to help develop UTM. As a result, version 0.4 was released in April 2020 which added support for UCS mini, S-series, 10+ use-cases, and 25+ features. In June 2021, version 0.6 was released with 20+ features, mostly focusing on the proactive resolution of issues. This release also completely redesigned the Congestion Monitoring dashboard.

UTM has come a long way from its initial prototype to a full-blown enterprise-class monitoring application. As of this writing, the UTM project on GitHub has earned 73 stars, 21 forks, 95 total issues, and 58 resolved issues. The UTM OVA has been downloaded thousands of times.

Many people have contributed directly or indirectly to the success of UTM. Of course, Paresh Gupta developed it. But this wasn’t possible without the help from users, peers, and his family.

Today, UTM is monitoring thousands of servers across the globe in many large enterprises. Its users include large financials, global banks, energy utility companies, software giants, logistics companies, healthcare providers, retail companies, government agencies, and so on. The largest known production deployment of UTM has been running for more than three years without missing even a single heartbeat while monitoring 4000+ servers in 40+ UCS domains.

The reason that UTM is the primary focus of this chapter is that many users who could not solve congestion issues for months were able to detect the source and cause of congestion within minutes using UTM. In a similar way, UTM can help many more users. Even if you do not use UTM, there is a lot to learn from it for developing monitoring applications because UTM has demonstrated that even minimal information, when correlated and analyzed in a meaningful way, can become extremely useful. These details are explained in the following sections.

Getting Started with UTM

To use UTM, it must be installed and collecting data for a while.

UTM is available at https://github.com/paregupt/ucs_traffic_monitor under the MIT license. It has two installation options. The first do-it-yourself option requires manually installing all the required packages. The second and easier option is to deploy the all-inclusive OVA. After installation, add a UCS Manager read-only account to UTM to start monitoring the UCS servers. These steps are explained on the GitHub link and the UTM installation steps at https://www.since2k7.com/blog/2020/02/29/cisco-ucs-monitoring-using-grafana-influxdb-telegraf-utm-installation/#Installing_UTM_using_OVA.

UTM Architecture

UTM has a collector that polls metrics from the UCS domains every 60 seconds (by default) using the XML APIs and NX-OS commands over the SSH session. The metrics are correlated and stored in a time-series database (InfluxDB). Finally, the use cases are presented using Grafana.

An Overview of Using UTM

UTM follows the 3-level architecture of the UCS Servers. It offers ready-made use cases that are grouped in their relevant dashboards.

1. **Locations dashboard**: Allows monitoring of multiple UCS domains globally or per location. This helps in finding if servers in a particular location are more congested or utilized than others.
2. **UCS Domain Overview dashboard**: This provides inventory and resource utilization use cases such as speed distribution, and port states.

3. **UCS Domain Traffic dashboard**: This provides use cases for traffic and congestion monitoring in a specific domain, such as correlating FI Uplink Port traffic with server traffic.
4. **UCS Chassis Traffic dashboard**: Allows investigating a chassis in detail.
5. **Server Traffic Dashboard**: Allows investigating a server in detail such as vHBA and vNIC traffic, congestion on the IOM/FEX backplane port to a server, load-balancing, and VIF path mapping between Server Adapter Port and FI Uplink Port.
6. **Congestion Monitoring Dashboard**: Allows detecting congestion points among thousands of UCS servers.

Troubleshooting Congestion Using UTM

Troubleshooting a problem typically starts under the following conditions:

1. **When you know where to start:** In this condition, the problem description has a location. For example, if application performance is degraded, go to the “Server Traffic” dashboard to monitor its hosting server. Likewise, if a link is congested, go to the “UCS Domain Traffic” dashboard to investigate the affected link. Later, this chapter explains case studies that started when it was known where to start troubleshooting and the goal was to find the cause and solution of the problem.
2. **When you do not know where to start:** A problem location is not known when doing a general health check or when the users report generic performance issues with many servers. In such cases, the troubleshooting workflow may start on the following UTM dashboards in no specific order:

    a. Congestion Monitoring dashboard: Find any server or port that stands out from the others. Refer to the section on *Proactively Detecting Congestion Due to Slow-Drain* for further details.

b. Locations dashboard or UCS Domain Traffic dashboard: Find any highly utilized servers or ports. Also, find any bit errors, discards, link errors, etc. Refer to the section on *Proactively Detecting Congestion Due to Over-Utilization* for further details.

Congestion Troubleshooting Workflow in UTM

Chapter 4 explains a congestion troubleshooting methodology in lossless networks that requires chasing the source of congestion in the traffic direction. Using the same methodology in a UCS domain, as Figure 9-2 illustrates, troubleshooting congestion against ingress traffic has the following typical workflow:

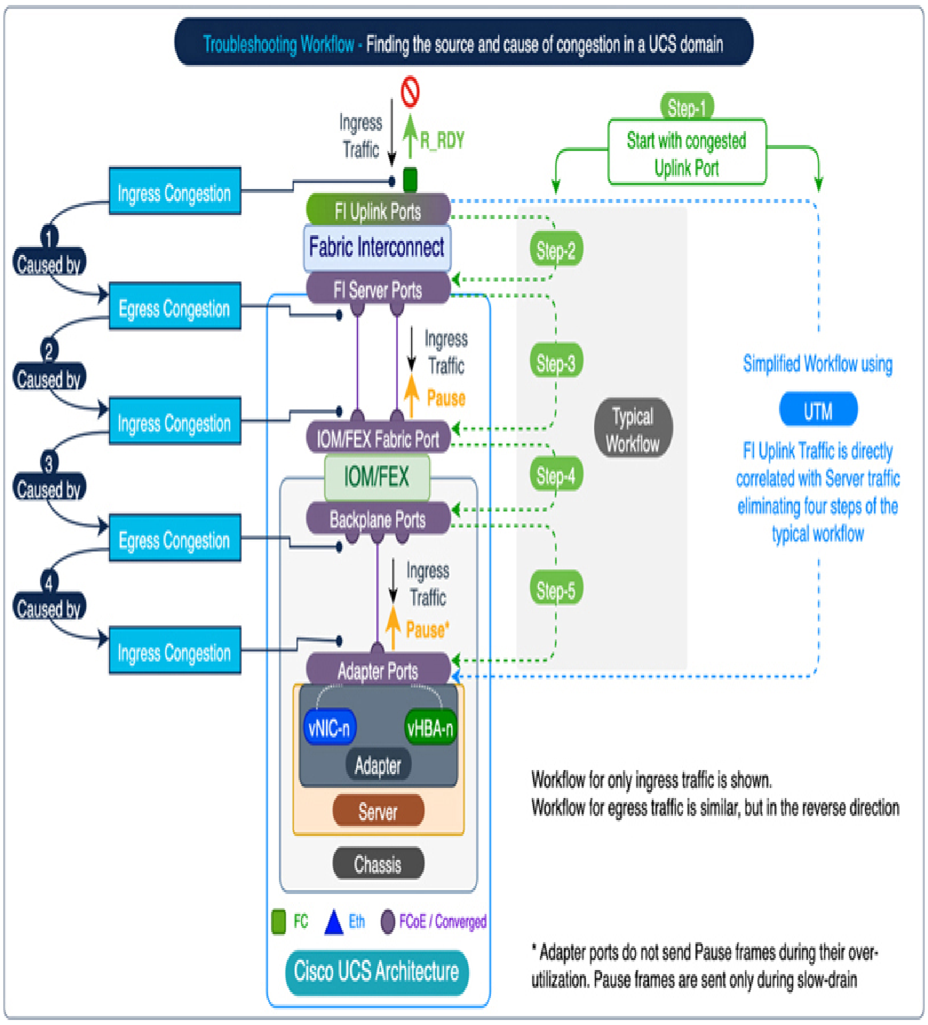

Troubleshooting Workflow - Finding the source and cause of congestion in a UCS domain
Ingress Traffic
R_RDY
Step-1
Start with congested Uplink Port
Ingress Congestion
1 Caused by
Egress Congestion
2 Caused by
Ingress Congestion
3 Caused by
Egress Congestion
4 Caused by
Ingress Congestion
FI Uplink Ports
Fabric Interconnect
FI Server Ports
Ingress Traffic
Pause
IOM/FEX Fabric Port
IOM/FEX
Backplane Ports
Ingress Traffic
Pause*
Adapter Ports
vNIC-n
vHBA-n
Adapter
Server
Chassis
Step-2
Step-3
Step-4
Step-5
Typical Workflow
Simplified Workflow using
UTM
FI Uplink Traffic is directly correlated with Server traffic eliminating four steps of the typical workflow
FC
Eth
FCoE / Converged
Cisco UCS Architecture
Workflow for only ingress traffic is shown.
Workflow for egress traffic is similar, but in the reverse direction
* Adapter ports do not send Pause frames during their over-utilization. Pause frames are sent only during slow-drain

Figure 9-2 *Congestion troubleshooting workflow in a UCS domain*

Step-1: Find the FI Uplink Port with congestion against ingress traffic.

Step-2: Find FI Server Ports with egress congestion. For this, a user must monitor Pause frames over a duration, take a delta, and find the ports with maximum increase.

Step-3: Find the IOM/FEX Fabric Ports that are connected to the congested FI Server Port. These ports would show ingress congestion.

Step-4: Find the IOM/FEX Backplane Ports with egress congestion.

Step-5: Finally, find the Server(s) that are connected to the congested IOM/FEX Backplane Ports and monitor their traffic per vHBA and vNIC.

Troubleshooting congestion against egress traffic has a similar workflow but in the reverse direction.

Doing these steps manually for up to 160 servers in a UCS domain while a congestion event is ongoing is tedious, hence it does not lead to a correct resolution for most users.

This is where UTM shines because it offers a simplified workflow. It makes all the calculations automatically and correlates the FI Uplink Port traffic with the Server traffic. In other words, it allows going directly from the 1st step to the 5th step after, and therefore it reduces troubleshooting from days to minutes.

There are a few more reasons for using the simplified workflow of UTM for finding the source and cause of congestion instead of using the typical 5-step workflow as shown in Figure 9-2.

1. In our experience, most congestion issues in a UCS domain originate at the servers, and hence, we recommend investigating them first. The simplified workflow in UTM has improvised over the years to try to

find the problems where they are typically found across many other production deployments.

2. In some cases, however, congestion may be caused by the FI Server Ports when the collective traffic of multiple servers leads to the over-utilization of these links. For this condition also, UTM offers a simplified workflow in the form of ready-made use cases on the UCS Domain Traffic dashboard. This again saves a significant amount of time as compared to following the typical workflow.
3. Finally, at the very basic level, congestion against ingress traffic in a UCS domain happens under the following conditions.
   a. At least one FI Server Port observes egress congestion due to over-utilization.
   b. At least one FI Server Port observes egress congestion due to slow-drain (ingress Pause).
   c. At least one IOM/FEX Backplane Port observes egress congestion due to over-utilization. This would lead to (b) because congestion spreads against traffic direction.
   d. At least one IOM/FEX Backplane Port observes egress congestion due to slow-drain (ingress Pause). This would also lead to (b) because congestion spreads against traffic direction.

   Most of these conditions can be detected on Server Adapter Ports that connect to the IOM/FEX Backplane Ports (c and d, which lead to b), and thus we recommend focusing first on them. The over-utilization of the FI Server Ports (a) should be investigated if no server is found to cause congestion. This workflow has proven to be a significant time saver.

Proactively Detecting Congestion Due to Slow-Drain

UTM offers a dedicated Congestion Monitoring dashboard for proactively detecting slow-drain when the source of congestion is not known. In one click, it finds the most congested servers and ports. In two clicks, it finds the exact time of congestion.

Figure 9-3 shows panels from the UTM Congestion Monitoring dashboard. The bar graphs show the culprit chassis and the culprit server in that chassis. Only the top 10 items are shown that help to focus on the most critical congestion events. To use it effectively, pay attention to the large values that are colored red. In Figure 9-3, first, investigate congestion against egress traffic because its Pause frame count (130589) is much larger than the Pause frame count (2723) for congestion against ingress traffic. The top row shows Chassis-2 connected to FI-B via Server Port 1/08 is causing the most congestion. Then, the table below it shows the Server in Chassis-2 as the most severe source of congestion because it sent more Pause frames (230116) than any other Server.

The time-series graphs help in finding the exact time of congestion and the number of times when congestion happened. In Figure 9-3, some names have been intentionally changed and blurred.

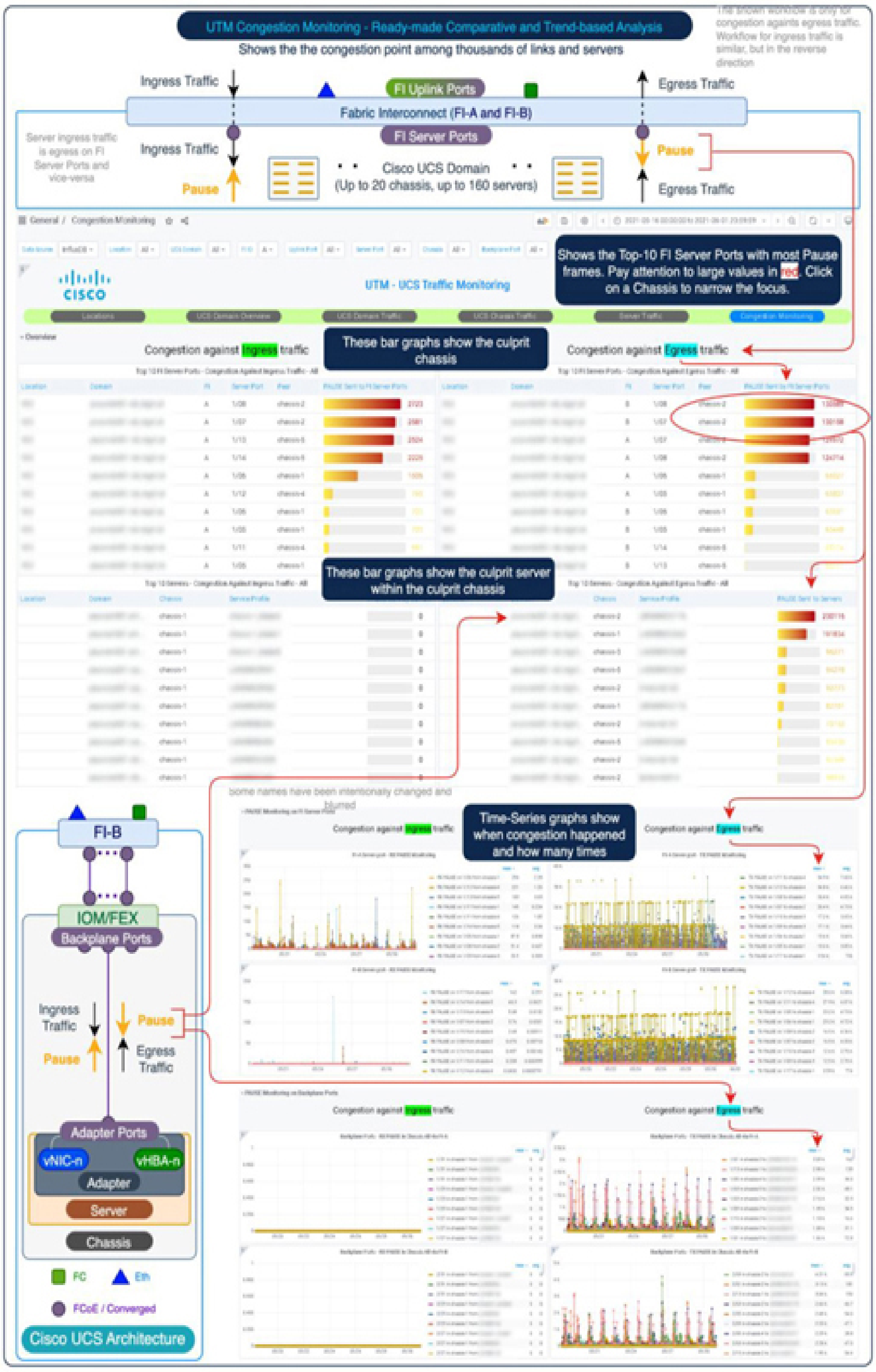

UTM Congestion Monitoring - Ready-made Comparative and Trend-based Analysis
Shows the the congestion point among thousands of links and servers
The shown workflow is only for congestion agaisnt egress traffic. Workflow for ingress traffic is similar, but in the reverse direction
Ingress Traffic
FI Uplink Ports
Egress Traffic
Fabric Interconnect (FI-A and FI-B)
FI Server Ports
Server ingress traffic is egress on FI Server Ports and vice-versa
Pause
Cisco UCS Domain
(Up to 20 chassis, up to 160 servers)
Shows the Top-10 FI Server Ports with most Pause frames. Pay attention to large values in red. Click on a Chassis to narrow the focus.
UTM - UCS Traffic Monitoring
Congestion against Ingress traffic
These bar graphs show the culprit chassis
Congestion against Egress traffic
These bar graphs show the culprit server within the culprit chassis
Some names have been intentionally changed and blurred
Time-Series graphs show when congestion happened and how many times
FI-B
IOM/FEX
Backplane Ports
Adapter Ports
vNIC-n
vHBA-n
Adapter
Server
Chassis
FC
Eth
FCoE / Converged
Cisco UCS Architecture

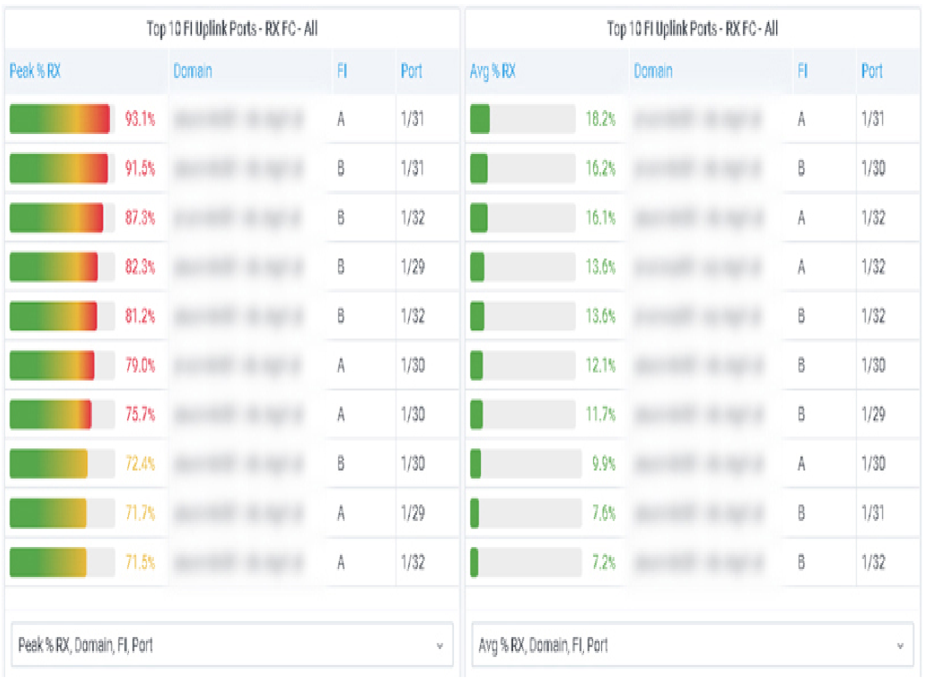
Figure 9-3 *Congestion Monitoring dashboard in UTM*

As mentioned, the absolute count of Pause frames may not be conclusive but the fact that a server is sending more Pause frames than others indicates that this server should be investigated first. Also, the spike in Pause frames is a strong indication of congestion if the time of this spike matches with other events, such as application errors or alerts from the upstream network.

For congestion against ingress traffic, this investigation should lead to a culprit server, which can be further investigated using its dedicated “Server Traffic” dashboard. For congestion against egress traffic, this investigation should lead to FI Uplink Ports. Based on their type (Fibre Channel, lossy Ethernet, or converged Ethernet) find the source of congestion using the troubleshooting methodology explained in Chapter 4 and Chapter 7.

Proactively Detecting Congestion Due to Over-Utilization

The UTM Locations dashboard shows the top 10 FI Uplink Ports and servers that send and receive the most traffic across all the monitored UCS domains (Figure 9-4).



Top 10 FI Uplink Ports - RX FC - All

Peak % RX	Domain	FI	Port
93.1%	[redacted]	A	1/31
91.5%	[redacted]	B	1/31
87.3%	[redacted]	B	1/32
82.3%	[redacted]	B	1/29
81.2%	[redacted]	B	1/32
79.0%	[redacted]	A	1/30
75.7%	[redacted]	A	1/30
72.4%	[redacted]	B	1/30
71.7%	[redacted]	A	1/29
71.5%	[redacted]	A	1/32

Peak % RX, Domain, FI, Port

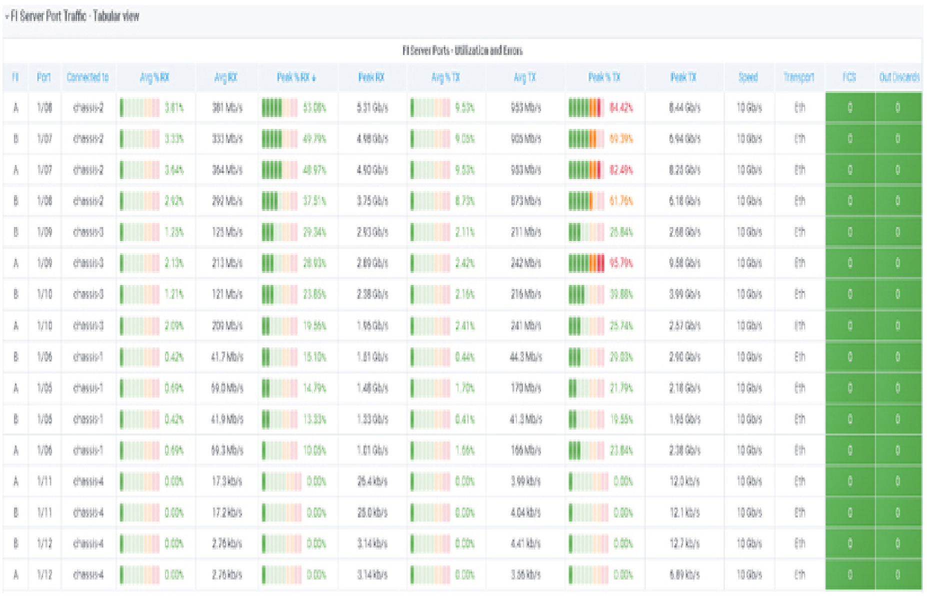
Top 10 FI Uplink Ports - RX FC - All

Avg % RX	Domain	FI	Port
18.2%	[redacted]	A	1/31
16.2%	[redacted]	B	1/30
16.1%	[redacted]	A	1/32
13.6%	[redacted]	A	1/32
13.6%	[redacted]	B	1/32
12.1%	[redacted]	B	1/30
11.7%	[redacted]	B	1/29
9.9%	[redacted]	A	1/30
7.6%	[redacted]	B	1/31
7.2%	[redacted]	B	1/32

Avg % RX, Domain, FI, Port

Figure 9-4 *Top 10 utilized FI Uplink Ports on UTM Locations dashboard*

The UCS Domain Traffic dashboard shows the peak and average utilization of FI Server Ports in a UCS domain (Figure 9-5). By default, the bar graphs change their color from green to orange to red as their utilization increases. These tables (Figure 9-6 and Figure 9-7) also show CRC errors to identify the links with bit errors and discards for identifying the links with packet drops.

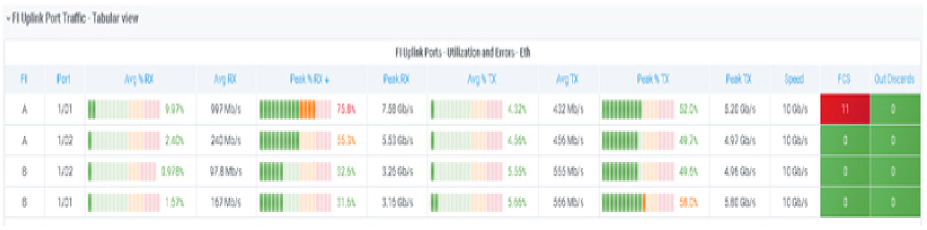


▾ FI Server Port Traffic - Tabular view

FI Server Ports - Utilization and Errors

FI	Port	Connected to	Avg % RX	Avg RX	Peak % RX ↓	Peak RX	Avg % TX	Avg TX	Peak % TX	Peak TX	Speed	Transport	FCS	Out Discards
A	1/08	chassis-2	3.81%	381 Mb/s	53.08%	5.31 Gb/s	9.53%	953 Mb/s	84.42%	8.44 Gb/s	10 Gb/s	Eth	0	0
B	1/07	chassis-2	3.33%	333 Mb/s	49.79%	4.98 Gb/s	9.05%	905 Mb/s	69.39%	6.94 Gb/s	10 Gb/s	Eth	0	0
A	1/07	chassis-2	3.64%	364 Mb/s	48.97%	4.90 Gb/s	9.53%	953 Mb/s	82.49%	8.25 Gb/s	10 Gb/s	Eth	0	0
B	1/08	chassis-2	2.92%	292 Mb/s	37.51%	3.75 Gb/s	8.73%	873 Mb/s	61.76%	6.18 Gb/s	10 Gb/s	Eth	0	0
B	1/09	chassis-3	1.23%	123 Mb/s	29.34%	2.93 Gb/s	2.11%	211 Mb/s	26.84%	2.68 Gb/s	10 Gb/s	Eth	0	0
A	1/09	chassis-3	2.13%	213 Mb/s	28.93%	2.89 Gb/s	2.42%	242 Mb/s	95.79%	9.58 Gb/s	10 Gb/s	Eth	0	0
B	1/10	chassis-3	1.21%	121 Mb/s	23.85%	2.38 Gb/s	2.16%	216 Mb/s	39.88%	3.99 Gb/s	10 Gb/s	Eth	0	0
A	1/10	chassis-3	2.09%	209 Mb/s	19.56%	1.96 Gb/s	2.41%	241 Mb/s	25.74%	2.57 Gb/s	10 Gb/s	Eth	0	0
B	1/06	chassis-1	0.42%	41.7 Mb/s	15.10%	1.51 Gb/s	0.44%	44.3 Mb/s	29.03%	2.90 Gb/s	10 Gb/s	Eth	0	0
A	1/05	chassis-1	0.69%	69.0 Mb/s	14.79%	1.48 Gb/s	1.70%	170 Mb/s	21.79%	2.18 Gb/s	10 Gb/s	Eth	0	0
B	1/05	chassis-1	0.42%	41.9 Mb/s	13.33%	1.33 Gb/s	0.41%	41.3 Mb/s	19.55%	1.95 Gb/s	10 Gb/s	Eth	0	0
A	1/06	chassis-1	0.69%	69.3 Mb/s	10.05%	1.01 Gb/s	1.66%	166 Mb/s	23.84%	2.38 Gb/s	10 Gb/s	Eth	0	0
A	1/11	chassis-4	0.00%	17.3 kb/s	0.00%	26.4 kb/s	0.00%	3.99 kb/s	0.00%	12.0 kb/s	10 Gb/s	Eth	0	0
B	1/11	chassis-4	0.00%	17.2 kb/s	0.00%	28.0 kb/s	0.00%	4.04 kb/s	0.00%	12.1 kb/s	10 Gb/s	Eth	0	0
B	1/12	chassis-4	0.00%	2.76 kb/s	0.00%	3.14 kb/s	0.00%	4.41 kb/s	0.00%	12.7 kb/s	10 Gb/s	Eth	0	0
A	1/12	chassis-4	0.00%	2.76 kb/s	0.00%	3.14 kb/s	0.00%	3.56 kb/s	0.00%	6.89 kb/s	10 Gb/s	Eth	0	0

Figure 9-5 *FI Server Port traffic and errors on UCS Domains Traffic dashboard*

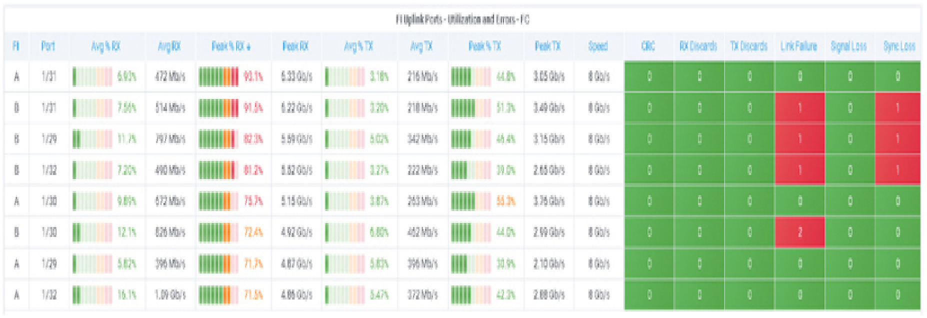


▾ FI Uplink Port Traffic - Tabular view

FI Uplink Ports - Utilization and Errors - Eth

FI	Port	Avg % RX	Avg RX	Peak % RX ↓	Peak RX	Avg % TX	Avg TX	Peak % TX	Peak TX	Speed	FCS	Out Discards
A	1/01	9.97%	997 Mb/s	75.8%	7.58 Gb/s	4.32%	432 Mb/s	52.0%	5.20 Gb/s	10 Gb/s	11	0
A	1/02	2.40%	240 Mb/s	55.3%	5.53 Gb/s	4.56%	456 Mb/s	49.7%	4.97 Gb/s	10 Gb/s	0	0
B	1/02	0.978%	97.8 Mb/s	32.6%	3.26 Gb/s	5.55%	555 Mb/s	49.6%	4.96 Gb/s	10 Gb/s	0	0
B	1/01	1.67%	167 Mb/s	31.6%	3.16 Gb/s	5.66%	566 Mb/s	58.0%	5.80 Gb/s	10 Gb/s	0	0

Figure 9-6 *Ethernet FI Uplink Port traffic and errors on UCS Domains Traffic dashboard*



FI Uplink Ports - Utilization and Errors - FC

FI	Port	Avg % RX	Avg RX	Peak % RX ↓	Peak RX	Avg % TX	Avg TX	Peak % TX	Peak TX	Speed	CRC	RX Discards	TX Discards	Link Failure	Signal Loss	Sync Loss
A	1/31	6.93%	472 Mb/s	93.1%	6.33 Gb/s	3.18%	216 Mb/s	44.8%	3.05 Gb/s	8 Gb/s	0	0	0	0	0	0
B	1/31	7.56%	514 Mb/s	91.5%	6.22 Gb/s	3.20%	218 Mb/s	51.3%	3.49 Gb/s	8 Gb/s	0	0	0	1	0	1
B	1/29	11.7%	797 Mb/s	82.3%	5.59 Gb/s	5.02%	342 Mb/s	46.4%	3.15 Gb/s	8 Gb/s	0	0	0	1	0	1
B	1/32	7.20%	490 Mb/s	81.2%	5.52 Gb/s	3.27%	222 Mb/s	39.0%	2.65 Gb/s	8 Gb/s	0	0	0	1	0	1
A	1/30	9.89%	672 Mb/s	75.7%	5.15 Gb/s	3.87%	263 Mb/s	55.3%	3.76 Gb/s	8 Gb/s	0	0	0	0	0	0
B	1/30	12.1%	826 Mb/s	72.4%	4.92 Gb/s	6.80%	462 Mb/s	44.0%	2.99 Gb/s	8 Gb/s	0	0	0	2	0	0
A	1/29	5.82%	396 Mb/s	71.7%	4.87 Gb/s	5.83%	396 Mb/s	30.9%	2.10 Gb/s	8 Gb/s	0	0	0	0	0	0
A	1/32	16.1%	1.09 Gb/s	71.5%	4.86 Gb/s	5.47%	372 Mb/s	42.3%	2.88 Gb/s	8 Gb/s	0	0	0	0	0	0

Figure 9-7 *Fibre Channel FI Uplink port traffic and errors on UCS Domains Traffic dashboard*

These tables show the average as well as the peak utilization. When monitoring the performance of the last 24 hours, the average values show a mean utilization over the last 24 hours, whereas the peak values show the maximum of the mean utilization over the 60-second monitoring intervals. Because of this reason, peak utilization should be preferred, and the use of average utilization should be avoided for congestion monitoring. The average utilization has other use cases such as for verifying load balancing. UTM still shows peak and average utilization to send a reminder of the importance of peak utilization and how misleading the average utilization can be. As Figure 9-7 shows, on FI-A, FC 1/31's peak utilization was 93.1%, which might have caused congestion due to over-utilization. However, relying on the average utilization of only 6.93% will not detect a potential congestion event. This concept was explained in Chapter 1, the sections on Defining Full *Utilization vs High Utilization vs Over-Utilization* and *Monitoring Full Utilization vs High Utilization vs Over-Utilization*. Figure 9-7 reinforces the same concept.

Even the peak utilization in UTM is really the mean utilization over the 60-second monitoring interval. This is still a long time, especially when the all-flash and NVMe storage arrays have I/O response times in hundreds of microseconds. Because of this reason, all instances of high utilization should be treated at the same severity as congestion due to over-utilization.

Another possibility is that UTM may not show over-utilized links for a shorter duration. For example, consider a link that's 100% utilized for 30 seconds, but idle during the following 30 seconds. This link will show 50% peak utilization due to the minimum polling interval of 60 seconds, and thus, it will not be colored in orange and red, and therefore may not even show up in the top-10 utilized links. Congestion in such cases can be detected by

1. Packet drops for lossy traffic, or
2. Pause frames on the upstream links for lossless traffic, or
3. Both

Packet drops are shown under the row with the title "Errors" on the Locations dashboard and the UCS Domain Traffic dashboard. The Pause frames are shown on the Congestion Monitoring dashboard and all the relevant dashboards.

Case Study 1 — Finding the Cause and Source of Congestion in a UCS Domain

A user reported that many MDS switchports often generate TxWait alerts. These switchports were connected to the UCS FI. Figure 9-8 shows their topology and problem description. They wanted to know what the TxWait alert meant, what caused it, and what was its solution.

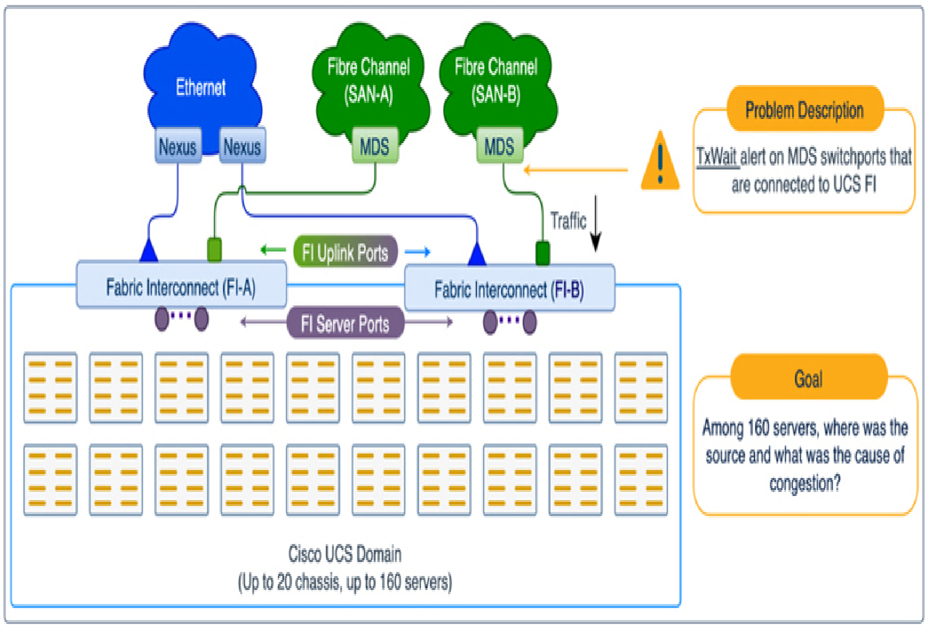

Ethernet
Nexus
Nexus
Fibre Channel (SAN-A)
MDS
Fibre Channel (SAN-B)
MDS
Problem Description
TxWait alert on MDS switchports that are connected to UCS FI
Traffic
FI Uplink Ports
Fabric Interconnect (FI-A)
Fabric Interconnect (FI-B)
FI Server Ports
Goal
Among 160 servers, where was the source and what was the cause of congestion?
Cisco UCS Domain
(Up to 20 chassis, up to 160 servers)

Figure 9-8 *TxWait alert on MDS switchports that are connected to UCS FI*

Background

TxWait is a metric that calculates the duration of Tx B2B credit unavailability on the MDS switchports. The user had enabled automatic alerting at 30% TxWait using the Port-Monitor feature on MDS switches. The TxWait alerts indicated that MDS switchports were unable to transmit for 30% of the duration between the rising and the falling alerts. Refer to Chapter 3, “Detecting Congestion in Fibre Channel Fabrics,” for more details on TxWait and Port-Monitor.

The FI Uplink was a Port-Channel (SAN-PC-1) with four member links each operating at 8GFC. In this case, the user knew the FI Uplink Ports that were connected to MDS. If this is not known, use the **show interface** command on MDS and match the “Peer port WWN” with the “Port WWN” of the FI uplink port. This is demonstrated in Chapter 4, Example 4-17.

Investigation

TxWait on the MDS switchports indicated the UCS FI reduced the rate of sending of B2B credits (R_RDY) for the traffic that MDS was sending to it. This was a case of congestion against ingress traffic in this UCS domain.

As explained earlier in the section *Congestion Troubleshooting Workflow*, a typical troubleshooting workflow would require investigating the FI Server Ports, finding their connected IOM/FEX Fabric Ports, then finding the congested Backplane ports, and finally, finding the culprit servers. However, UTM already correlates the traffic path and utilization, and thus offers a simplified workflow that eliminates these steps.

In this case, because a starting point was already known, the next step was to find the Servers, which were the ultimate destination for the traffic. Had no culprit server been found, the fallback option would have been to find a slow-drain Server using the Congestion Monitoring dashboard. This workflow is explained in Case Study 2.

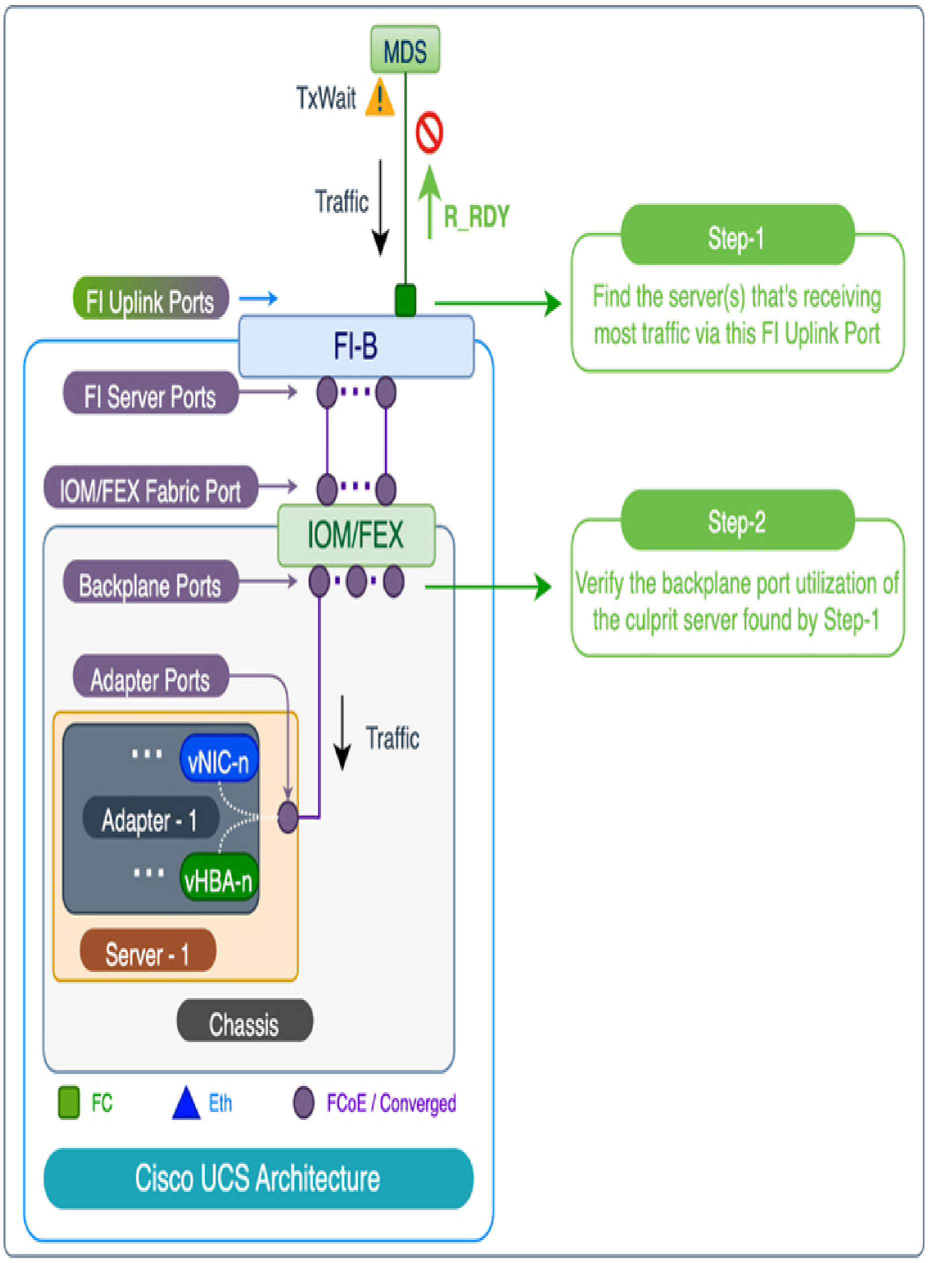


Figure 9-9 *Simplified troubleshooting workflow using UTM*

Step — 1

In UTM, on the “UCS Domains Traffic” dashboard, the user expanded the row with the title “FI Uplink Port Traffic Distribution per Server”. By default, it should be the second row from the top. This shows the utilization of the FI Uplink Ports and their usage per server. Then, at the top of the page, they narrowed the scope to the UCS domain under investigation, FI-A or FI-B, and the Uplink (SAN-PC-1).

UTM presents two graphs side-by-side. One is for the ingress traffic and the other is for the egress traffic. Because congestion is directional, it was important to focus only on the ingress traffic. Investigating the egress traffic would not have led to finding the source and cause of congestion.

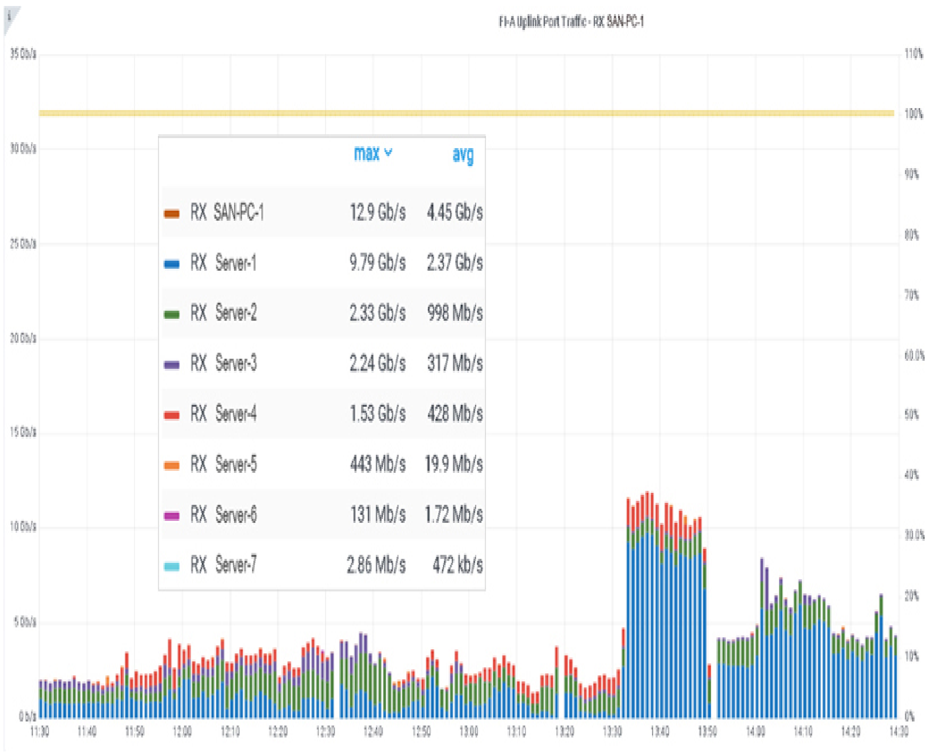
Figure 9-10 shows this graph.

1. The left y-axis shows the utilization of the FI Uplink Port in Gbps.
2. The right y-axis shows its percentage utilization.
3. The x-axis shows time.
4. The legends show the max (peak) and average utilization of the FI Uplink Port and traffic for servers that receive traffic via this port.
5. The bar graphs are stacked. Each bar shows utilization per server via the FI Uplink Port. The collective utilization of these servers is the cause of traffic on this Uplink Port.

The following conclusions were drawn from this chart.

1. Max (peak) ingress utilization of SAN-PC-1 was 12.9 Gbps (first row in the legends).
2. The top consumer of this link was Server-1, with a max (peak) utilization of 9.79 Gbps (second row in the legends). The utilization spiked at 13:33 hours and remained high until 13:50 hours. These times matched with the TxWait alert generated by the MDS switchports.

In reality, this graph shows all the servers (up to 160) that may be using this Uplink Port. However, it is natural to focus on the top consumer, which was Server-1. Also, Figure 9-10 shows a trimmed-down list of only a few servers and their names have been intentionally changed.



	max ⌄	avg
🟧 RX SAN-PC-1	12.9 Gb/s	4.45 Gb/s
🟦 RX Server-1	9.79 Gb/s	2.37 Gb/s
🟩 RX Server-2	2.33 Gb/s	998 Mb/s
🟪 RX Server-3	2.24 Gb/s	317 Mb/s
🟥 RX Server-4	1.53 Gb/s	428 Mb/s
🟧 RX Server-5	443 Mb/s	19.9 Mb/s
🟪 RX Server-6	131 Mb/s	1.72 Mb/s
🟦 RX Server-7	2.86 Mb/s	472 kb/s

Figure 9-10 *Step-1: Find the servers that are the top consumers of the FI Uplink Port Step - 2*

For investigating a server, the user opened the “Server Traffic” dashboard and narrowed the scope to Server-1 by selecting it in the Service Profile drop-down. Then they expanded the row with the title “Backplane port % Utilization per vIF”. This shows the vHBA and vNIC that are contributing to the utilization of the Backplane Port.

As previously mentioned, UTM presents two graphs side-by-side for ingress and egress traffic. As the UCS architecture in Figure 9-1 shows, server ingress traffic is egress on the Backplane Ports. Therefore, in this case, only the egress (TX) graphs were investigated.

Figure 9-11 shows this graph.

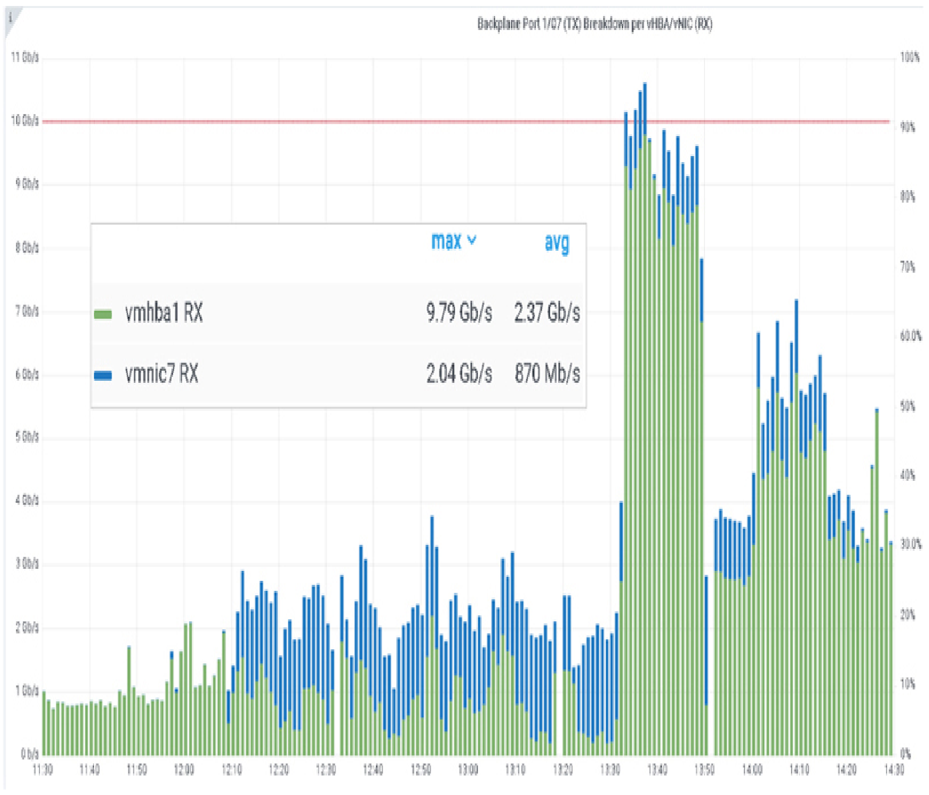


Figure 9-11 *Step-2: Find the Backplane Port utilization of the server found in step-1*

1. The left y-axis shows the utilization of the IOM/FEX Backplane Port in Gbps.
2. The right y-axis shows its percentage utilization.
3. The x-axis shows time.
4. The legends show the max (peak) and average utilization of the vHBA and vNIC that receive traffic via Backplane Port 1/07 (shown in chart header).
5. The bar graphs are stacked. Each bar shows utilization per vNIC and vHBA via the Backplane Port. The collective utilization of these VIFs is the cause of traffic on this Backplane Port.

The following conclusions were drawn from this chart.

1. Backplane Port 1/07 was operating at 10 Gbps — Indicated by the red threshold line.
2. The top consumer of this link was vmhba1 with a max (peak) utilization of 9.79 Gbps. The max (peak) utilization because of vmnic7 was 2.04 Gbps.

This investigation showed that the backplane port (1/07) was highly utilized between 13:33 hours and 13:50 hours. These times matched with the high-utilization of the FI Uplink Port (SAN-PC-1) as shown in Figure 9-10 and the TxWait alert generated by the MDS switchports confirming that the over-utilization of the Server-1 Backplane Port was the cause of congestion.

Note:

Figure 9-11 shows vNIC's name as vmnic7 and vHBA's name as vmhba1. These names are defined by the users and UTM automatically shows them as configured in UCS Manager.

Conclusion

The source of congestion was Server-1. The cause of congestion was the over-utilization of the backplane port that was connected to Server-1. When IOM received more traffic than can be sent on this backplane port, it invoked PFC to slow down the traffic from the FI Server Ports. In turn, the FI reduced the rate of sending of the B2B credits (R_RDYs) to the upstream MDS switch to slow down the ingress traffic. This resulted in the TxWait alert that started this whole investigation.

Solution

The solution was to increase the capacity of the Server-1 backplane port. To increase the network capacity, increasing the link speed is the preferred approach. However, in this case, the user had a spare adapter (VIC) that they used until the newer adapter with a faster speed arrived.

Case Study 1 — Summary

A UCS domain can have up to 160 servers, each with two or more backplane ports. Ingress congestion on any of these ports would result in egress congestion (TxWait) on the upstream MDS switch. This was the cause of congestion in this case.

Without UTM, it was almost impossible to solve this problem because UCS does not log the metrics with time and date stamps. Even if the problem was ongoing, finding the servers that were the top consumer of FI Uplink Port and manually calculating the utilization of their backplane ports would be an extremely tedious task if at all attempted.

Using UTM, it took just two steps and less than 20 minutes to find the source and the cause of the congestion. As shown in Step-1, the end-to-end correlation between the FI Uplink Ports and Server traffic was the key to solving this problem.

Case Study 2 — Congestion due to Slow-Drain on the Backplane Port

In Case Study 1, the root cause of congestion was found to be the over-utilization of the backplane port that was connected to Server-1.

If no backplane ports had shown over-utilization, the next step would have been to investigate congestion due to slow-drain. These servers should be on the list of the Top 10 Servers on the Congestion Monitoring dashboard. In other words, the workflow can start from the Domain Traffic dashboard using the Uplink Port or Congestion Monitoring dashboard. Both these workflows should lead to finding a slow-drain server.

Investigation

After finding a slow-drain server, go to the Server Traffic dashboard and expand the row with the title “Congestion Monitoring on Backplane Ports”. These graphs compare traffic and Pause frames in the reverse direction. As explained in Case Study 1, while troubleshooting TxWait on the MDS switchports, focus on ingress traffic in the UCS domain, which is egress on the IOM/FEX Backplane Ports.

Figure 9-12 shows this graph.

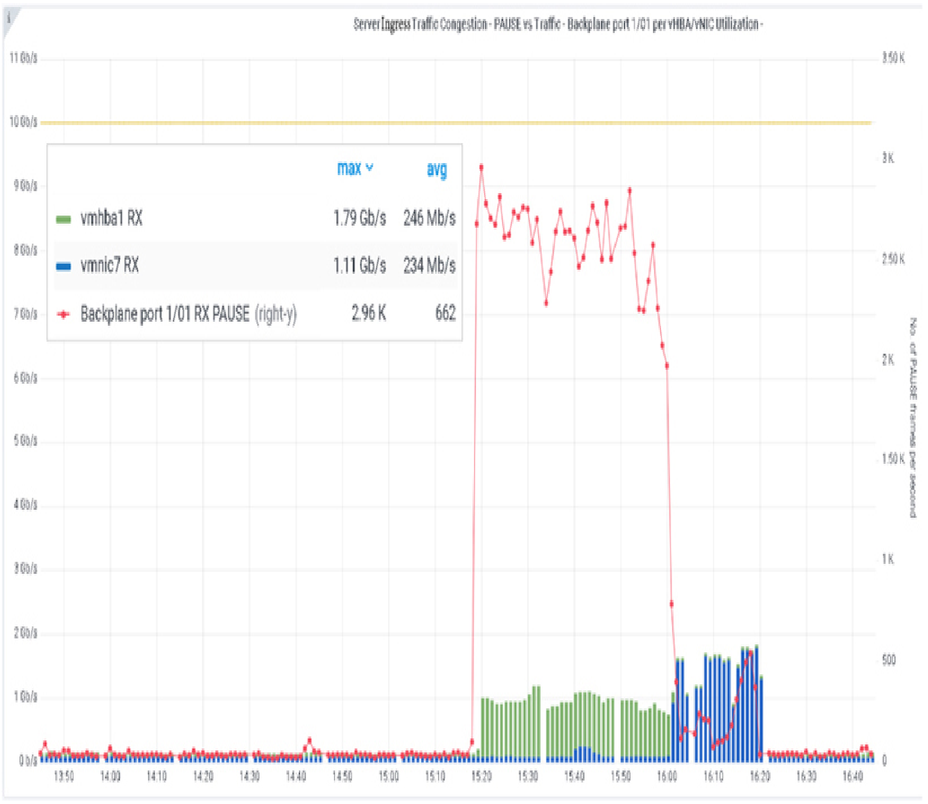


Figure 9-12 *Finding slow-drain by correlating traffic and Pause frames in the reverse direction*

1. The left y-axis shows the utilization of the Backplane Port in Gbps.
2. The right y-axis shows the number of Pause frames per second sent by the Server to the Backplane Port against ingress traffic.
3. The x-axis shows time.
4. The legends show the max (peak) and average utilization of the vHBA and vNIC that receive traffic via Backplane Port 1/01 (shown in chart header) and the number of Pause frames in the reverse direction.
5. The bar graphs are stacked. Each bar shows utilization per vNIC and vHBA via the Backplane Port. The collective utilization of these VIFs is the cause of traffic on this Backplane Port. As mentioned, this is measured on the left y-axis.
6. The dotted-line shows the number of Pause frames received per second by the Backplane Port from the

Server Adapter Port. As mentioned, this is measured on the right y-axis.

Conclusions

The following conclusions were drawn.

1. Backplane Port 1/01 was operating at 10 Gbps — Indicated by the yellow threshold line.
2. The max (peak) utilization of vmhba1 was 1.79 Gbps, whereas that of vmnic7 was 1.11 Gbps. As mentioned, vmhba1 and vmnic7 are the user-defined names of vHBA and vNIC respectively in the UCS Manager.
3. The number of Pause frames per second spiked from almost zero to approximately 3000 at 15:20 hours and it sustained until 16:00 hours. This is a strong indication of congestion due to slow-drain because this time also matched with the time of the TxWait alert on MDS switchports. The time of these events are different from that of Case Study 1. Note that the absolute number of Pause frames is not conclusive because the duration of the traffic pause is not known. However, this trend-based analysis correlated with the TxWait alert is a strong indication of congestion due to slow-drain on this Backplane Port.

   a. Note that the Pause frames spiked when the traffic to vmhba1 increased between 15:20 and 16:00 hours. This is Fibre Channel (or FCoE) traffic that uses hop-by-hop flow control (lossless).

   b. At 16:00 hours, traffic to vmhba1 reduced, and traffic to vmnic7 increased. But the number of Pause frames is fewer because the vmnic7 traffic does not use hop-by-hop flow control (lossy).

   c. Note that at 16:10 hours, the link utilization is higher than between 15:20 and 16:00 hours. Yet, there is no congestion. This is a crucial demonstration of why traffic should be monitored at per-port and per-class levels and Pause frames in the reverse direction.

It is not known why this server slowed down the ingress traffic. It can be the lack of resources (such as CPU or memory), traffic patterns, or both.

Possibly the same server may not cause slow-drain during a different hour in the day even with a lot more traffic. This is demonstrated in Figure 9-13. The max (peak) utilization of the Backplane Port is approximately 90%, collectively caused by vmnic7 and vmhba1. Yet, there are no Pause frames.

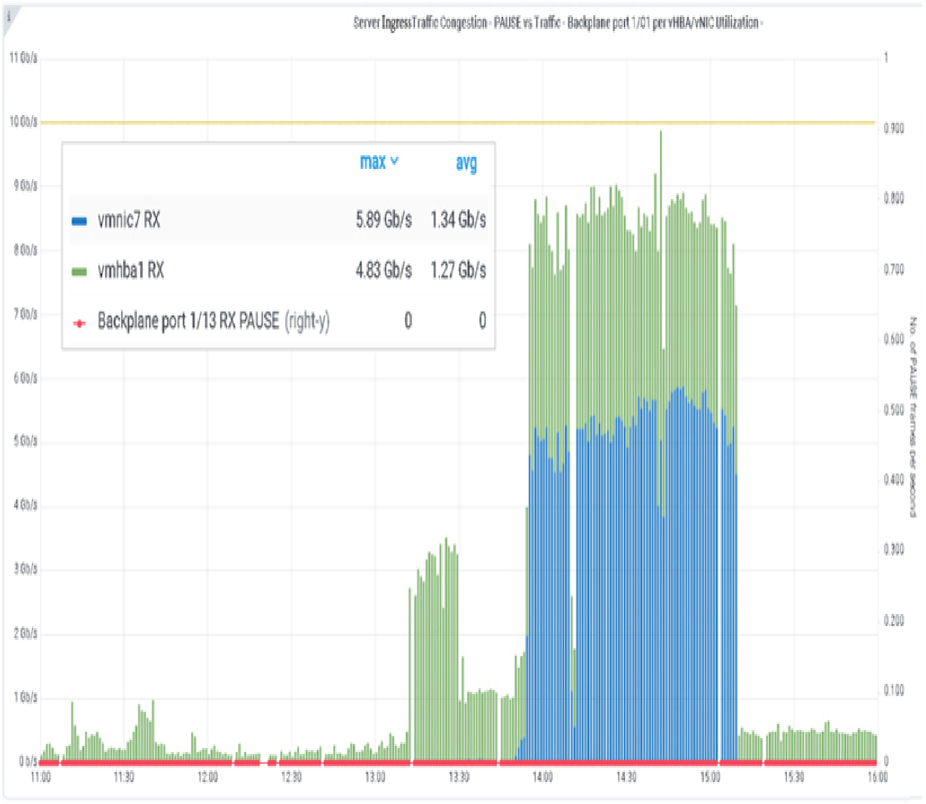


Figure 9-13 *More traffic, but no Pause frames*

Case Study 2 — Summary

In this case, it was found that the root cause of TxWait alerts on the upstream MDS switchport was caused by a slow-drain server. This is different from case Study 1, where the cause of congestion was over-utilization. Because the source and cause of congestion are not unknown until investigated, either of the two UTM workflows can be used. The first option is to start at the UCS Domain Traffic dashboard and find the server that consumes the most capacity of the FI Uplink Port. The second option is to find the most congested server on the Congestion

Monitoring dashboard. Both these options should take you to the Server Traffic dashboard for investigating a culprit server.

Case Study 3 — Non-Uniform Utilization of FI Uplink Ports

A user reported that their UCS servers might have been underperforming. On one end, the application owners suspected storage performance issues. On the other end, the storage teams point the bottleneck back to the servers. In between, the compute team believed to have ample capacity on the UCS servers.

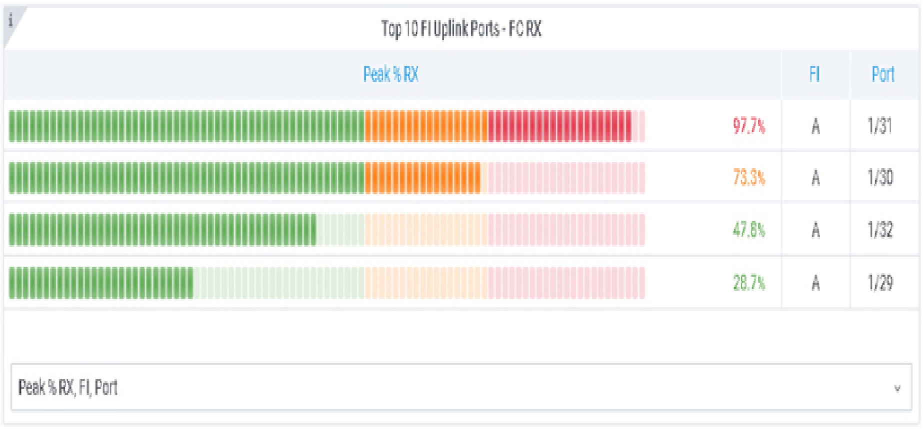
The user wanted to investigate this long-standing problem of performance degradation and blame games among different teams.

Investigation

This was an open-ended problem, and thus there was no specific location to start the investigation.

Because performance issues were highlighted, first, the UTM "Congestion Monitoring" dashboard was investigated, but no specific server or link stood out.

Next, on the "UCS Domain Traffic" dashboard, the utilization of the FI Uplink Ports was verified in the panel with the title "Top 10 FI Uplink Ports". As Figure 9-14 shows, although there were four uplinks, their utilization was not uniform. The peak utilization of FC 1/31 on FI-A was 97.7% while FC 1/29 was only 28.7% utilized.



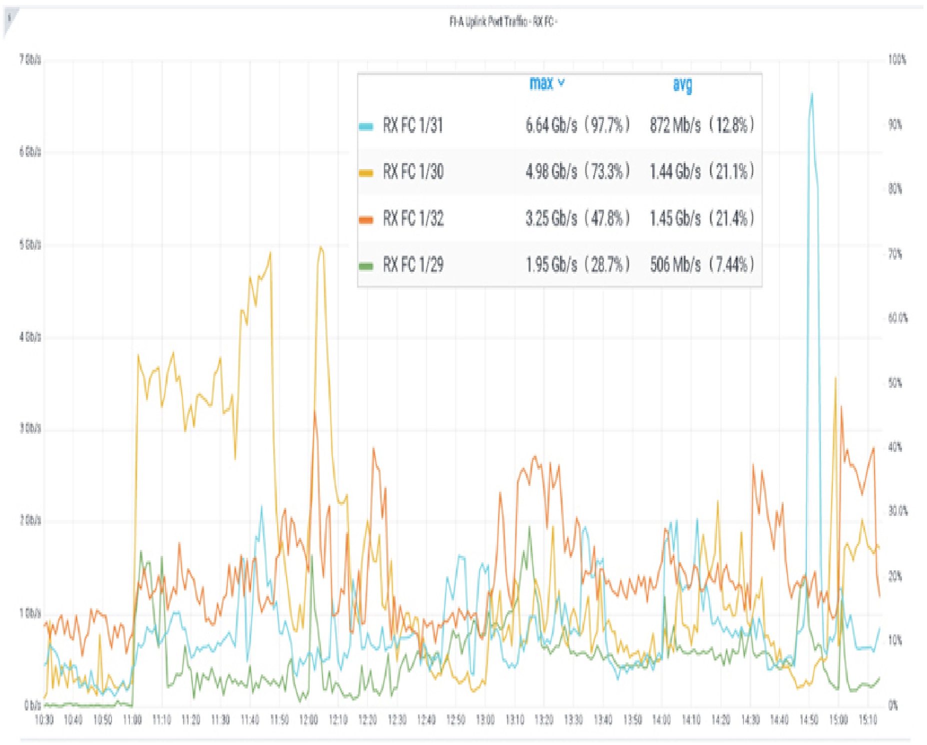
Top 10 FI Uplink Ports - FC RX

Peak % RX	FI	Port
97.7%	A	1/31
73.3%	A	1/30
47.8%	A	1/32
28.7%	A	1/29

Peak % RX, FI, Port

Figure 9-14 *UCS Domain Traffic dashboard showing FI Uplink Port non-uniform utilization*

The tabular view helps to locate an issue quickly, but it does not show the exact time and the duration of high utilization, which can be found using the time-series graphs under the row with the title “FI Uplink Port Traffic - Graphical view”. As Figure 9-15 shows, at 14:50 hours, FC 1/31 was highly utilized for a few minutes. The left y-axis shows utilization in Gbps, whereas the right y-axis shows utilization in percentage.



	max ˅	avg
🟦 RX FC 1/31	6.64 Gb/s (97.7%)	872 Mb/s (12.8%)
🟨 RX FC 1/30	4.98 Gb/s (73.3%)	1.44 Gb/s (21.1%)
🟧 RX FC 1/32	3.25 Gb/s (47.8%)	1.45 Gb/s (21.4%)
🟩 RX FC 1/29	1.95 Gb/s (28.7%)	506 Mb/s (7.44%)

Figure 9-15 *Time-series view of FI Uplink Port utilization*

The four Fibre Channel uplinks, each operating at 8GFC provided a collective capacity of 32GFC. But their non-uniform utilization indicated that probably these four links were not aggregated in a port-channel. This was also suspected by Figure 9-14, where individual physical members were shown. Had these links been aggregated, only the port-channel would have shown in that panel. The lack of a port-channel was finally confirmed using the row with the title “Port-channel Traffic Distribution”, which did not show any port-channel interface with Fibre Channel Uplink Ports.

Conclusion

The Fibre Channel FI Uplink Ports were not aggregated in a port-channel. As a result, each server was limited to the data rate of only one 8GFC link to which it was pinned (or used for its Fabric Login (FLOGI)). When one server received excessive traffic, it led to high utilization of its pinned uplink while other links were under-utilized.

Not just the performance of this server impacted, but the high utilization of its pinned uplink might have also caused congestion due to over-utilization. Because the uplinks were highly utilized in the ingress direction, the congestion might have spread upstream from the UCS domain while the internal links of the UCS domain stayed clear of congestion. This explains why investigating the Congestion Monitoring dashboard, in the beginning, did not show any prominent symptoms of congestion. The internal UCS links had ample available capacity because the ingress traffic via one FI to a server was limited only to the maximum data rate of 8GFC. As Chapter 2 explains, the data rate of 8GFC links is 6.8 Gbps.

Solution

The solution was to aggregate the four Fibre Channel uplinks in a port-channel. This would allow the servers to use the port-channel for FLOGI, and thus each server would be able to use the entire 32GFC capacity instead of being limited only to 8GFC.

Case Study 3 — Summary

This case may seem simple while reading it, but the user struggled for months to find this root cause. They could not catch the high-utilization alert in UTM by orange and red colors because they had changed the default view from peak to average utilization. As explained in the section on *Proactively Detecting Congestion Due to Over-Utilization*, relying on the average utilization misled them for months. Also, it was difficult for them to believe that the Uplinks were over-utilized despite having enough capacity. Only after seeing the traffic utilization in UTM, they realized the practical significance of port-channel and how a lack of it led to performance degradation and potential congestion.

Case Study 4 — Congestion due to Multipathing I/O Imbalance

A user reported that their UCS domain often becomes the cause of congestion due to over-utilization. As Figure 9-16 shows, they detected this condition via the Tx-datarate alert generated by the MDS switchports (MDS-1) that connect to the FI Uplink Ports. At the same time, the upstream MDS switch (MDS-2) shows TxWait alert, confirming that the Tx-datarate indeed caused congestion.

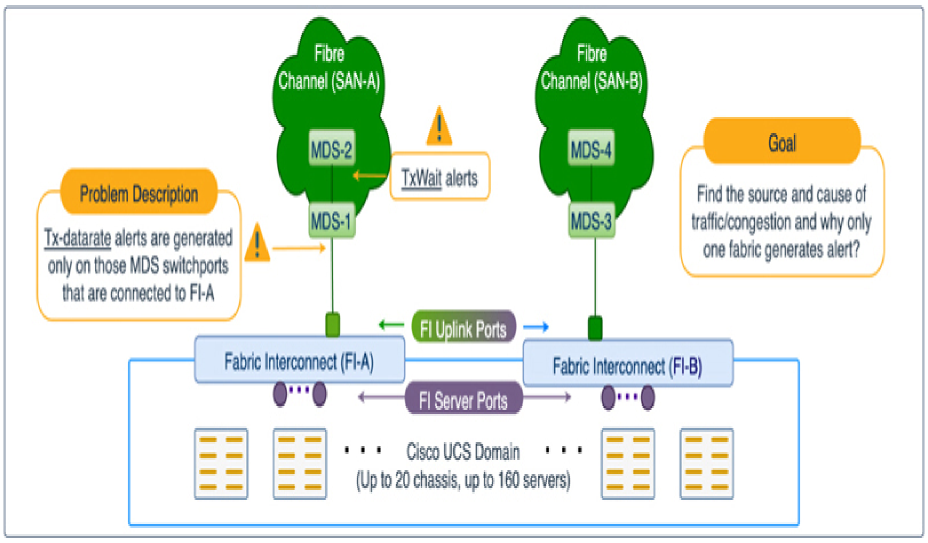

Fibre Channel (SAN-A)
MDS-2
MDS-1
Fibre Channel (SAN-B)
MDS-4
MDS-3
TxWait alerts
Problem Description
Tx-datarate alerts are generated only on those MDS switchports that are connected to FI-A
Goal
Find the source and cause of traffic/congestion and why only one fabric generates alert?
FI Uplink Ports
Fabric Interconnect (FI-A)
Fabric Interconnect (FI-B)
FI Server Ports
Cisco UCS Domain
(Up to 20 chassis, up to 160 servers)

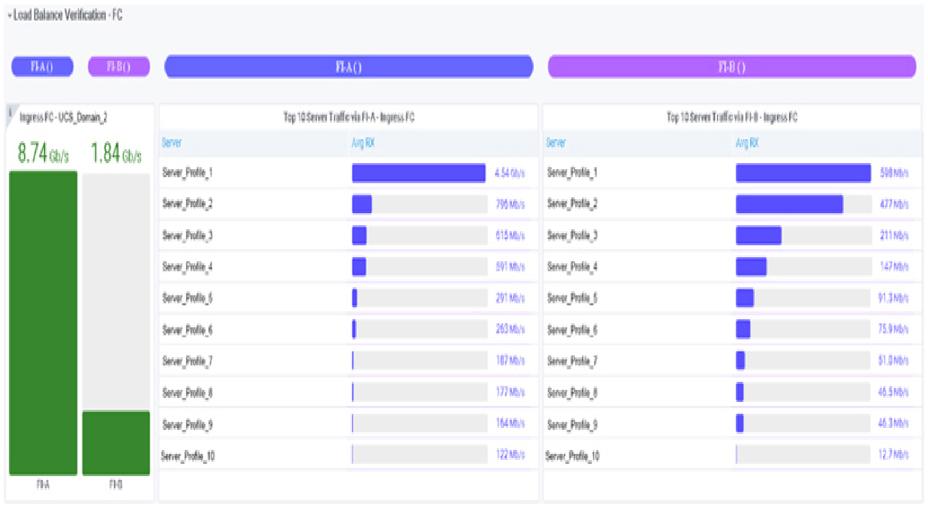
Figure 9-16 *Traffic imbalance between FI-A and FI-B*

The user also reported that the Tx-datarate alert (and congestion) were reported in only SAN-A that was connected to FI-A. SAN-B did not report a similar issue. They wanted to find the source and the cause of traffic that was the cause of congestion due to over-utilization.

Investigation

For investigating such traffic imbalance issues, UTM by default correlates the server traffic with their paths across FI-A and FI-B and provides a ready-made panel for “Load Balance Verification”. It is available on the “UCS Domain Traffic” dashboard.

Refer to Figure 9-17. The vertical bar graphs on the left show that FI-A's average utilization was 8.74 Gbps, whereas FI-B's utilization was only 1.84 Gbps. This was a clear imbalance and it also explained why only the MDS that was connected to FI-A generated Tx-datarate alert. The horizontal bar graphs on the right show the servers that were the top consumers of each FI's capacity. Server_profile_1 received 4.54 Gbps via FI-A, whereas it received only 598 Mbps via FI-B. Many other servers show a similar traffic imbalance.



Top 10 Server Traffic via FI-A - Ingress FC

Server	Avg RX
Server_Profile_1	4.54 Gb/s
Server_Profile_2	796 Mb/s
Server_Profile_3	615 Mb/s
Server_Profile_4	591 Mb/s
Server_Profile_5	291 Mb/s
Server_Profile_6	263 Mb/s
Server_Profile_7	187 Mb/s
Server_Profile_8	177 Mb/s
Server_Profile_9	164 Mb/s
Server_Profile_10	122 Mb/s

Top 10 Server Traffic via FI-B - Ingress FC

Server	Avg RX
Server_Profile_1	598 Mb/s
Server_Profile_2	477 Mb/s
Server_Profile_3	211 Mb/s
Server_Profile_4	147 Mb/s
Server_Profile_5	91.3 Mb/s
Server_Profile_6	75.9 Mb/s
Server_Profile_7	51.0 Mb/s
Server_Profile_8	46.5 Mb/s
Server_Profile_9	46.3 Mb/s
Server_Profile_10	12.7 Mb/s

Figure 9-17 *FI-A and FI-B traffic balance verification*

Conclusion

Tx-datarate alert and congestion due to over-utilization were reported in only one fabric because of multipathing I/O misconfiguration on the servers.

Solution

This issue was resolved by correcting the multipathing misconfiguration on the servers.

Case Study 4 — Summary

The problem description made it quite evident that the congestion was because of traffic imbalance. However, finding the culprits among 160 servers, each with multiple vHBAs is a tedious task. UTM simplified this investigation to a single click with its Load Balance Verification.

Summary

This chapter explains the use of the UCS Traffic Monitoring (UTM) App in detecting and troubleshooting congestion in Cisco UCS Servers. UTM correlates the server traffic with its path and offers ready-made use cases based on comparative and time-based analysis. The Congestion Monitoring dashboard shows the source and cause of congestion in one click. In two clicks, it provides the exact time of congestion.

UTM has been immensely successful in solving long-standing congestion issues in hundreds of production environments across the world. This demonstrates that even if the network devices provide minimal congestion metrics, such as only the number of Pause frames but no TxWait, correlating and analyzing the available data can still detect most congestion issues, if not all. For the users of Cisco UCS servers, this chapter is directly relevant. For others, this chapter explains the practical techniques of detecting and troubleshooting congestion in converged systems.

References

- UTM Project on GitHub: https://github.com/paregupt/ucs_traffic_monitor/
- UTM installation: https://www.since2k7.com/blog/2020/02/29/cisco-ucs-monitoring-using-grafana-influxdb-telegraf-utm-installation/#Installing_UTM_using_OVA
- Cisco Live Breakout Session — Traffic Monitoring and Engineering for UCS (BRKCOM-2004)
- Cisco Live Breakout Session — UCS Performance Troubleshooting (BRKCOM-3002)
- Cisco Live Breakout Session — UCS Networking —Deep Dive (BRKINI-2025)
- UCS Tech Talk: Demystifying Monitoring for Cisco UCS Manager and Standalone C-Series Servers - https://www.youtube.com/watch?v=oE0liSmc1-4

- Cisco UCS Monitoring Resource Handbook — https://community.cisco.com/t5/unified-computing-system-knowledge-base/cisco-ucs-monitoring-resource-handbook/ta-p/3646478
- UCS Tech Talk: Demystifying Monitoring for Cisco UCS Manager and Standalone C-Series Servers — https://community.cisco.com/t5/unified-computing-system-knowledge-base/ucs-tech-talk-demystifying-monitoring-for-cisco-ucs-manager-and/ta-p/3648084
- Grafana project on GitHub: https://github.com/grafana/grafana
- InfluxDB project on GitHub: https://github.com/influxdata/influxdb