Workbook Overview

Troubleshooting becomes integral part of the updated CCIE R&S lab exam. The new section is a group of loosely correlated trouble tickets. Every ticket has a point value associated with it, and the candidate must obtain 80% of the total section score in order to succeed in this section. Troubleshooting scenario uses a topology separate from the configuration part of the exam and has its own L2 configuration and IP addressing. Section grading is based on the automatic script along with a human hand to confirm the script results. From this information, one may conclude that mastering troubleshooting techniques becomes vital for succeeding in the new exam.

In this new IEWB-RS VOL4 workbook we present you with ten troubleshooting scenarios, each having ten trouble tickets. This amount should be approximately equal to the number of the troubleshooting tasks you will encounter in the actual exam. The topology used for every scenario is the same that we use for all our RS products, including VOL1 (technology-focused labs), VOL2 (configuration mock lab scenarios) and VOL3 (core technologies scenarios).

However, unlike our previous workbooks, we **restrict** access to some of the devices in the lab topology. For every scenario this “restricted” set may be different and it is clearly outlined in the scenario’s baseline. Using this technique we increase the scenario complexity by allowing candidates to see only “one” side of the problem. When looking at the lab diagram, you will clearly see routers not under your control as being displayed in orange color. Also, when you log onto the “restricted” device, it will warn you using a banner message.

In addition to the above restriction, we highly encourage you not using the **`show running-configuration`**, **`show startup-configuration`** commands or any other command that shows you the textual representation of the router’s configuration. This requirement makes you focus on using the show and debugging commands, which is invaluable when troubleshooting the real-world scenarios.

Our ultimate goal is not only prepare you for passing the Troubleshooting section of the CCIE R&S lab exam, but also to teach you a structured troubleshooting approach. As opposed to simple guessing and peeking at the routers running configurations you should learn using the debugging commands and interpreting various show commands output. For every ticket, we are going to follow the same structured procedure to resolve the issue. Here is an outline of this procedure:

1. Build and Analyze the Baseline
2. Analyze the Symptoms (propose hypothesis)
3. Isolate the issue (gather more symptoms)
4. Fix the Issue (by comparing to the Baseline)

We are now going to discuss all these steps in details to give you the basic understanding of the fundamental procedure.

Structured Troubleshooting

Build and Analyze the Baseline.

Since all tickets in a scenario share the same topology, you need to perform this step only once per the whole scenario. Baseline is essentially a picture of the healthy network, which serves as the starting point of any troubleshooting process. In real life, your baseline is the snapshot of your network under "normal" conditions – stable topology, interfaces under normal utilization, devices responding to management requests, users happy etc. In the lab, all you have is the diagram and possibly some additional network description. Additional information might be provided in the trouble ticket itself, but the initial starting point is the diagram.

We recommend making your own diagrams, including the following information:

- IP addressing + IGPs.
- Layer 2 topology.
- BGP diagram.
- IPv6 topology.
- Multicast and Redistribution diagram.

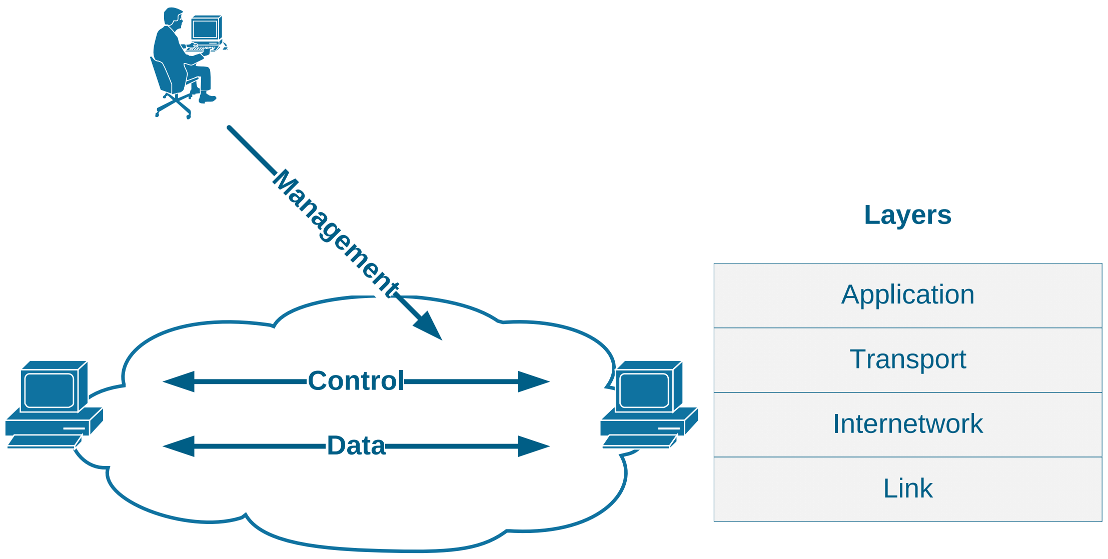
You may enhance your diagrams with any extra information provided, e.g. hints on the network pre-configuration and applications deployed, such as WWW, FTP, SMTP, VoIP and so on. This will help you analyzing symptoms later. You goal at this stage is to get clear picture of the network and discover any potential caveats. Try not using any IOS commands at this point, as this may consume your valuable time and add unneeded information. Overall, don't spend too much time building the baseline – the goal is to spend around 20 minutes. By the end of the baseline analysis phase, you should have clear understanding of the protocols and applications deployed in your network.

When you finished with building the baseline, take a quick look over all tickets in sequence. See if you can make any conclusions based on the ticket information, like marking the potentially broken links or missing information flows. Sometimes this may be obvious from the ticket text and give you extra hint and help when dealing with other tickets.

Analyze the Symptoms.

The ultimate goal of this step is coming up with the initial scope of the problem area and the initial set of hypothesis identifying the root cause. With respect to the CCIE lab, the primary source of the information is trouble ticket itself. The ticket might be formatted in a very simple manner, such as “there is an issue that prevents R1 from communicating with R2” or contain a detailed situation description, for example “at 10:00am this morning customers at Branch 1 site started complaining of poor HTTP performance. Analyzing the NOC action logs, you have noticed that someone was modifying R3’s configuration yesterday, but the change log entry is missing” and so on.

When trying to narrow the initial problem scope, it is helpful to use a reference network model, based on the classic TCP/IP protocol layers. The minimal working network consists of two communicating nodes and communication substrate connecting them. The substrate might be a direct link or a set of other nodes/routers. The network functions in three general planes: data, control and management. The first plane is responsible for forwarding data between the two endpoints, based on the programmed tables. The control plane is responsible for negotiating the data paths and establishing end-to-end connectivity. Example control plane protocols are OSPF, RIP, BGP and so on. The management plane is responsible for enforcing certain policy on the network and monitoring its performance. For example, SNMP and SSH used to carry CLI commands are both management protocols. All planes could be subdivided into four classic layers: Application, Transport, Internetwork and Link.



When analyzing the symptoms, you should select the network elements (e.g nodes, links) that you suspect to belong to the problematic area: e.g. routers on the path between two nodes failing to communicate. At the same time, you should mark the planes and the layers that you suspect to malfunction and possible identify the protocol names at every layer. You need to be aware of the symptoms typical for every layer and remember that an issue at lower layer affects all other overlying layers as well (cascading effect).

Lastly, the most helpful thing to identify the initial problematic area is finding out what prior changes might have been made to the network, if this is mentioned in the ticket. Remember, every change is a potential problem!

Isolate the issue.

The goal of this phase is finding the device/devices and the specific configuration area that might be causing the issue(s).This is the core of the troubleshooting process. You start verifying your initial hypothesis, by trying to narrow the problematic area as small as possible. To start with the process, you need to select either of the three approaches:

Top-down approach

Test application layers across the path that you suspect to be causing the issues. Usually this approach works well when the issue lying in application misconfiguration (e.g. improper IMAP4 settings). This is very helpful in real-life scenarios; however, from the lab perspective this approach is not very useful as most issues will probably be related to the network configuration.

Bottom-up approach

You start by testing physical layer issues of every node in the problematic area. If you don't find any issues, you proceed to the next layer (i.e. networking) and see if there are any deviations from the baseline there. This is the most universal approach, as it starts with the fundamental layer and moves up. However, executing bottom-up search might be routine and time consuming, and thus inappropriate for a small issue.

Divide-and-Conquer approach

This method attempts to reduce the amount of work required by bottom-up search by making a "guess" – picking up the network layer that you suspect to be malfunctioning and testing the devices in the problematic area at this layer. It is common to start with the Internetwork layer and test end-to-end connectivity using the **`ping`** and **`traceroute`** commands. If this layer is healthy, then any underlying layer should be healthy as well, and you may continue searching in the "up" direction. Otherwise, using the above mentioned commands you may further isolate the problematic area and find the specific devices that might be causing the issue.

It is important to remember that during the issue isolating phase you will learn more information and may have to change your initial hypothesis, based on the results. Effectively, the Analyze and Isolate phases are deeply interconnected and depend on each other.

Fix the Issue

At the end of the previous stage you should be dealing with the "hot" area of the problem – devices/links that are malfunctioning or improperly configured. Of course, you should have facts on hands to prove that your hypothesis/guess was valid. Your next step is developing a plan to resolve the problem. Resist the urge or simply going ahead and changing the running configuration – you may effectively introduce more issues then there originally was. Save the original configuration, and type in your "fixup" in the notepad.

Implement the “fixup” step by step – don’t apply changes to many devices at the same time, if you suspect many devices being affected After every change, run verifications to see if the issue has been eliminated or not.

When you’re done, compare the results to the baseline you have built at the first step. If everything seems to match and the symptoms outlined in the ticket no longer persist you may consider the ticket to be resolved. If not, you should re-analyze the initial symptoms and the additional information gathered during the previous steps. The last step could be named as “Verification” step.

Workbook Solutions

Every solution document is formatted in structured manner to show you the flow of the actual troubleshooting process. You will find the sections corresponding to the in-depth analysis of the scenario baseline, diagram drawing and detailed step-by-step troubleshooting for every ticket presented in the scenario. Here is an outline for the solution document structure:

Build and Analyze the Baseline

Layer 2 Diagram.
BGP Diagram.
Multicast and Redistribution.
    Redistribution Loops Analysis.
    Multicast Propagation Analysis.
IPv6 Diagram
Read over the Lab

Solutions

Ticket 1
Analyze the Symptoms
Isolate the Issue
Fix the issue
Verify
...
Ticket 10
Analyze the Symptoms
Isolate the Issue
Fix the issue.
Verify

As you can see, the document follows the exact same path for the troubleshooting process that we outlined before. Every solution is about 50-60 pages long and provides enough details for every ticket, so that you’ll have plenty of material to learn from.

IEWB-RS VOL4 Lab 1

Lab Overview:

The following scenario is a practice lab exam designed to test your skills at troubleshooting Cisco networking devices. Specifically, this scenario is designed to assist you in your preparation for Cisco Systems' CCIE Routing & Switching Lab exam Troubleshooting Section. However, remember that in addition to being designed as a simulation of the actual CCIE lab exam, this practice lab should be used as a learning tool. Instead of rushing through the lab in order to resolve all issues, take the time to apply the structured troubleshooting methodology and improve your strategy.

Lab Instructions:

Prior to starting, ensure that the initial configuration scripts for this lab have been applied. For a current copy of these scripts, see the Internetwork Expert members' site at http://members.INE.com

Refer to the attached diagrams for interface and protocol assignments. Any reference to X in an IP address refers to your rack number, while any reference to Y in an IP address refers to your router number. When not explicitly mentioned, a router's IP address on the segment is based off the router number, Y. For example, R1 will have the IP address of 150.X.100.1 on the subnet 150.X.100.0/24, and SW3 will have the IP address 150.X.100.9 on the same subnet.

Use the name **`cisco`** along with the password of **`cisco`** to access the console line of any device used in the topology.

Lab Do's and Don'ts:

- Do not access the routers that are marked as restricted for your access.
- Do not use the **`show running-config`** or **`show startup-config`** commands or their equivalents when performing troubleshooting.
- Do not change or add any IP addresses from the initial configuration unless required for troubleshooting.
- Do not change any interface encapsulations unless required for troubleshooting.
- Do not change the console, AUX, and VTY passwords or access methods unless otherwise specified.
- Do not use any static routes, default routes, default networks, or policy routing unless otherwise specified.
- Save your configurations often.

Grading:

This practice lab consists of 10 trouble tickets totaling 30 points. A score of 24 points is required to achieve a passing grade. A trouble ticket must be fixed 100% with the requirements given in order to be awarded the points for that ticket. No partial credit is awarded. If a ticket has multiple possible resolutions, choose the solution that best meets the requirements and requires minimal changes. Per the CCIE R&S lab exam requirements, you are required to finish this lab in **`two`** hours.

The tickets generally have no dependencies, unless explicitly stated in the ticket outline, so you may work through them in any order you like. It's up to you to select the tickets that you feel most easy to deal with and manage your time accordingly.

GOOD LUCK!

Baseline

All network devices are configured according to the diagram provided with this scenario. The diagram reflects the proper network configuration, including IP address, IGP protocol settings and BGP AS numbers and serves as your primary source of the information. This section provides the scenario-specific configuration information that you may need during troubleshooting process. Notice that not all of the information may be useful during the troubleshooting process; however, all statement made below reflect the correct network configuration.

Devices under your Control

For this lab, you may only access and modify the configuration of the following devices: R1, R4, R5, R6, SW1 and SW2. Backbone devices BB1, BB2 and BB3 are out of your control per the initial topology configuration. If you refer to the diagram provided, the devices colored in **ORANGE** are out of your control.

Bridging and Switching

- All switches use VTP domain name of CCIE.
- SW1 is VTP server, SW3 & SW4 are VTP clients and SW2 is in VTP transparent mode.
- L3 Etherchannels and point-to-point links are configured between the switches according to the diagram provided with the scenario.
- The following is the list of the trunk links inerconnecting the switches
    - SW1 Fa0/14 to SW2 Fa0/14
    - SW2 Fa0/17 to SW3 Fa0/17
    - SW3 Fa0/19 to SW4 Fa0/19
- SW1 is the STP root bridge for all VLANs. Classic STP (PVST+) is in use.
- The Frame-Relay sub-interfaces of R2 and R3 are configured as point-to-point.

IGP

- Mutual redistribution is configured between RIP and EIGRP on R6.
- Mutual redistribution is configured between OSPF and EIGRP on R3 and SW2.
- The Serial link between R4 and R5 is slow ISDN and should be used as a backup in the rare conditions of the Frame-Relay links failure.
- The network should not have any additional redistribution points.

- IPv6 is configured on the Frame-Relay links between R1 and R2, R2 and R3 as follows:
  - Network 2001:164:X:12::/64 between R1 and R2
  - Network 2001:164:X:23::/64 between R2 and R3
  - The PPP link between R1 and R3 uses the network 2001:164:X:13::/64.
- RIPng is used as IPv6 routing protocol between R1, R2 and R3.
- R1 prefers to reach R3's Loopback100 IPv6 subnet via R2.
- R2 uses point-to-point Frame-Relay sub-interfaces.

BGP

- BGP peering sessions are configured according to the following table:

Device 1	Device 2
R4	BB3
R4	R3
R3	R1
R3	R2
R1	SW2
R1	R2
R2	R6
R6	BB2

- R6 and BB2 peering session is authenticated used the password value of "CISCO".
- AS 300 users cannot use AS 200 as transit to any other AS.
- AS 54 and AS 254 receive only a single summary route representing the whole 164.X.0.0 major subnet.

Multicast

- IP multicast routing is enabled in R2, R3, R4 and SW1.
- PIM is enabled on the following networks:
  - Frame Relay segments between R2 & R3 and R3 & R4.
  - The Ethernet link between R4 and SW1.
  - The VLANs 26, 3, and 7 of R2, R3, and SW1 respectively.
- R3 is the RP for the following multicast groups:
  - 225.10.0.0 - 225.10.255.255
  - 225.26.0.0 - 255.26.255.255
  - 225.42.0.0 - 255.42.255.255

    - 225.58.0.0 - 255.58.255.255
- R4 is the RP for the following multicast groups:
    - 226.37.0.0 - 226.37.255.255
    - 226.45.0.0 - 226.45.255.255
    - 227.37.0.0 - 227.37.255.255
    - 227.45.0.0 - 227.45.255.255
- RP mapping is configured statically through the network.

QoS

- Cisco Unified CallManager server is deployed on VLAN5 and users on VLAN 7 register their SIP phones with the server.
- The Frame-Relay links of R4 and R5 are provisioned at 256Kbps; All routers are configured for FRTS to accommodate this limitation.
- VoIP traffic is being given priority treatment over WAN links based the best-practice recommendations for low-speed links.

Trouble Tickets

Ticket 1: VoIP Quality

- You have received complaints from the users on VLAN7 using their Cisco IP Phones.
- The problem appears to be voice quality degradation when calling the HQ users residing on VLAN 5 subnet.
- According to the corporate QoS policy configuration everything should have been taken care of.
- You looked over the Frame-Relay links and found all of them uncongested.
- To make the problem even harder, users informed you that the voice quality is degraded only one way: from VLAN7 to the HQ.
- Based on the baseline information provide a solution to this problem.

3 Points

Ticket 2: Load-Balancing

- Three switches: SW2, SW3 and SW4 were configured so that traffic from SW1 load-balances to VLAN9 across the links connecting SW2 to SW3 and SW4.
- However, recently you have found that only the path via SW4 is being utilized.
- Accessing the devices under your control only, return the network to the baseline and ensure proper load-balancing.

3 Points

Ticket 3: BGP Peering

- After the security administrator of AS 300 has changed some “security settings” you found that BGP session between R1 and R3 is not coming up anymore. This is the message you keep seeing on your router’s console:

```
%BGP-3-NOTIFICATION: sent to neighbor 164.1.13.3 4/0 (hold time
expired) 0 bytes
```

- You have limited access to the remote router. So far, the only valuable piece of information you were able to get was the following:

```
Rack1R3#show ip bgp neighbors 164.1.13.1
BGP neighbor is 164.1.13.1,  remote AS 300, external link
  BGP version 4, remote router ID 0.0.0.0
  BGP state = OpenSent
  Last read 00:01:34, last write 00:01:34, hold time is 180,
keepalive interval is 60 seconds
  Message statistics:
    InQ depth is 0
    OutQ depth is 0
                         Sent       Rcvd
    Opens:                 26          1
    Notifications:         23          0
    Updates:                0          0
    Keepalives:           157        156
    Route Refresh:          0          0
    Total:                206        157
  Default minimum time between advertisement runs is 30 seconds

 For address family: IPv4 Unicast
  BGP table version 1, neighbor version 0/0
 Output queue size : 0
  Index 2, Offset 0, Mask 0x4
  2 update-group member
  Outbound path policy configured
  Outgoing update AS path filter list is 1
                                 Sent       Rcvd
  Prefix activity:               ----       ----
    Prefixes Current:               0          0
    Prefixes Total:                 0          0
    Implicit Withdraw:              0          0
    Explicit Withdraw:              0          0
    Used as bestpath:             n/a          0
    Used as multipath:            n/a          0

                                   Outbound    Inbound
  Local Policy Denied Prefixes:    --------    -------
    Total:                                0          0
  Number of NLRIs in the update sent: max 0, min 0

  Connections established 1; dropped 1
  Last reset 01:25:07, due to BGP Notification sent, hold time
expired
  External BGP neighbor may be up to 1 hop away.
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Connection is ECN Disabled, Mininum incoming TTL 254, Outgoing
TTL 255
Local host: 164.1.13.3, Local port: 44156
Foreign host: 164.1.13.1, Foreign port: 179

Enqueued packets for retransmit: 1, input: 0  mis-ordered: 0 (0
bytes)

Event Timers (current time is 0xECDA36):
Timer          Starts    Wakeups            Next
Retrans             7          5        0xECFCDC
TimeWait            0          0             0x0
AckHold             0          0             0x0
SendWnd             0          0             0x0
KeepAlive           0          0             0x0
GiveUp              0          0             0x0
PmtuAger            0          0             0x0
DeadWait            0          0             0x0

iss:  263956977  snduna:  263956978  sndnxt:  263957023
sndwnd:  16384
irs: 1374337120  rcvnxt: 1374337121  rcvwnd:      16384
delrcvwnd:      0

SRTT: 37 ms, RTTO: 1837 ms, RTV: 1800 ms, KRTT: 58784 ms
minRTT: 20 ms, maxRTT: 300 ms, ACK hold: 200 ms
Flags: active open, nagle
IP Precedence value : 6

Datagrams (max data segment is 1460 bytes):
Rcvd: 1 (out of order: 0), with data: 0, total data bytes: 0
Sent: 3 (retransmit: 5, fastretransmit: 0, partialack: 0, Second
Congestion: 0), with data: 1, total data bytes: 45
```

- You cannot reach the other admin via the phone, so it's up to you to use the information you have on hand to fix your side of the connection.

4 Points

Ticket 4: Connectivity Issue

- Another ticket from VLAN7 users. They cannot reach any resource on VLAN 5 – all IP Phones have unregistered, and nothing else works.
- However, they are still able to reach the local resources.
- Using the baseline description as your reference, resolve this issue in optimal manner.

3 Points

Ticket 5: Old Backup

- Accidental R6's power supply failure left you with no choice but quickly looking for a spare router.
- While preparing the new box you found that the latest router backup is dated 3 months old. It's about time to think of a change management system.
- However, for now you loaded the old backup file and now trying to find the missing parts of the puzzle.
- Your main goal is make all routers in EIGRP domain reach all RIP routes announced by BB1 and make sure BGP is working properly.

3 Points

Ticket 6: BGP Prefixes

- Network administrator from AS 254 called you and complained that they cannot reach any of AS 54 prefixes.
- The administrator claims that AS 200 does not advertise these prefixes to AS 254.
- The problem seems to be related to your network configuration, so it's up to you to fix it.

3 Points

Ticket 7: Security Improvement

- After the security administrator enabled "some" security feature for VLAN 55 users in SW3, the users rebooting their PC started complaining they cannot browse anything.
- You asked on the users to run the **`ipconfig /all`** command and figured that rebooted machines get an IP address in the range 169.254.0.0/16, while they should be auto configured via DHCP.
- The DHCP server is configured in R5 and used to work fine prior to the "security improvement".
- The security admin is out for lunch, and you only have a few minutes to fix the issue and cool down the upset users.

3 Points

Ticket 8: BGP Peering

- After some IP configuration changes on VLAN18 you found that BGP peering sessions between R1 and SW2 is no longer active.
- You looked over the latest configuration changes but found nothing that could potentially affect BGP peering – no changes to BGP configuration, no change of IP addressing or new access-lists.
- Fix the problem so that BGP session works again. You are only allowed to change the configurations of R1 and SW2 for this task.

3 Points

Ticket 9: IPv6

- IPv6 users behind R1 cannot reach the Loopback100 address of R3 anymore.
- Per the baseline, there is a primary and backup path, and there are no alarms signalizing any link problems.
- Accessing the router under you control, fix this issue.

3 Points

Ticket 10: Multicast

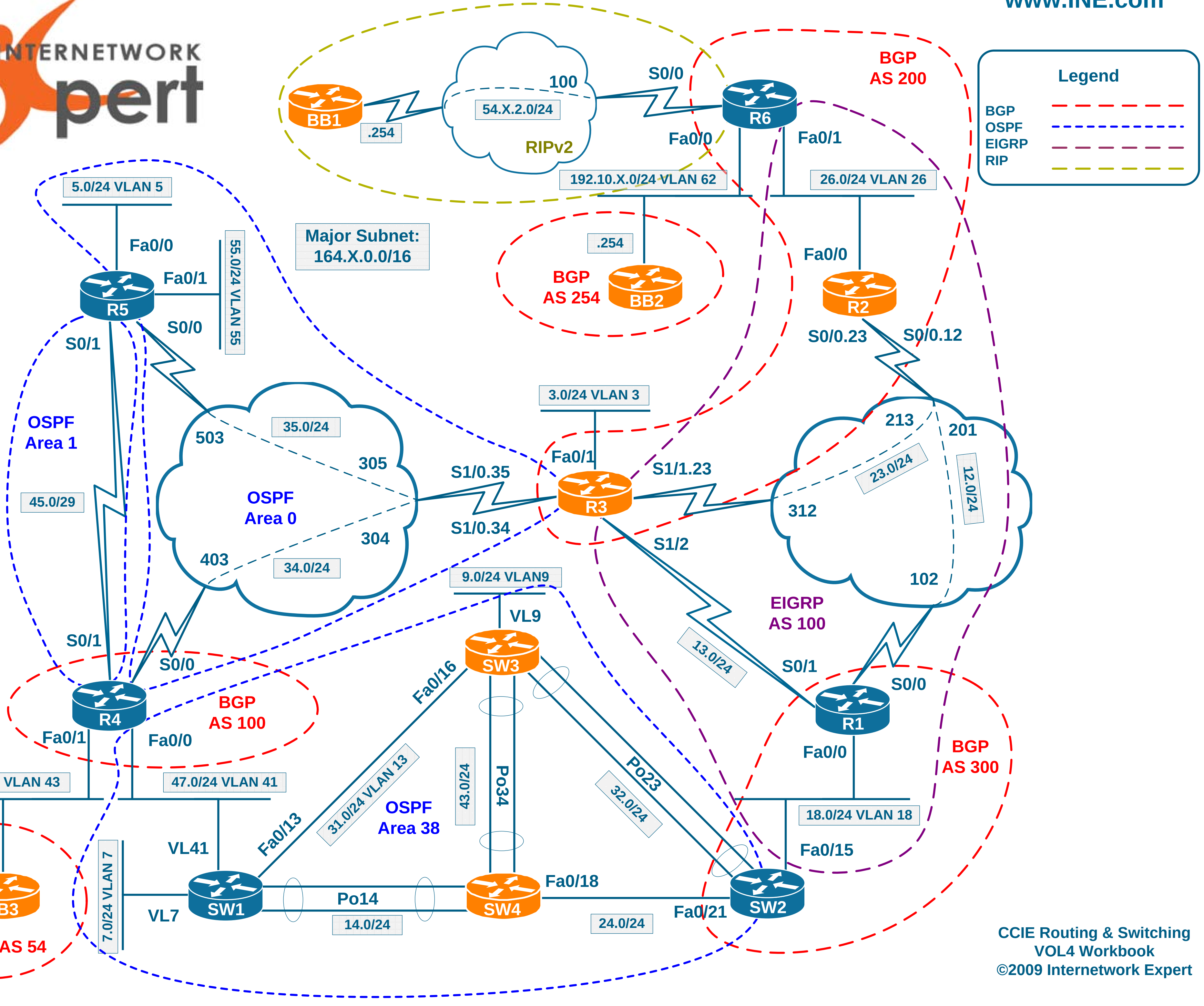
Note: Prior to starting with this ticket make sure you resolved Tickets 4 and 5

- Users on VLAN 41 complain that they cannot receive video feeds from VLAN 26.
- You figured out what channel they are using and found that it maps to the multicast address 226.37.1.1.
- Fix the issue and make sure you can send multicast streams down from R6 to SW1.
- You may use ICMP traffic for testing, as it’s open across the network.

2 Points

www.INE.com

INTERNETWORK expert



CCIE Routing & Switching
VOL4 Workbook
©2009 Internetwork Expert

IEWB-RS VOL4 Lab 1 Solutions

Table of Contents

IEWB-RS VOL4 Lab 1 Solutions 1
Table of Contents 1
Build and Analyze the Baseline 3
Layer 2 Diagram 4
BGP Diagram 5
Multicast and Redistribution 6
Redistribution Loops Analysis 7
Multicast Propagation Analysis 10
IPv6 Diagram 11
Read over the Lab 11
Solutions 13
Ticket 1 13
Analyze the Symptoms 13
Isolate the Issue 13
Fix the issue 17
Verify 17
Ticket 2 19
Analyze the Symptoms 19
Isolate the Issue 19
Fix the Issue 23
Verify 23
Ticket 3 25
Analyze the Symptoms 25
Isolate the Issue 26
Fix the Issue 32
Verify 32
Ticket 4 34
Analyze the Symptoms 34
Isolate the Issue 34
Fix the Issue 38
Verify 38
Ticket 5 40
Analyze the Symptoms 40
Isolate the Issue 40
Fix the Issue 40
Verify 40
Ticket 6 50
Analyze the Symptoms 50
Isolate the Issue 51

Fix the Issue...53
Verify...53
Ticket 7..54
Analyze the Symptoms 54
Isolate the Issue...54
Fix the Issue...54
Verify...55
Ticket 8..56
Analyze the Symptoms 56
Isolate the Issue...56
Fix the Issue...58
Verify...59
Ticket 9..60
Analyze the Symptoms 60
Isolate the Issue...60
Fix the Issue...62
Verify...62
Ticket 10..63
Analyze the Symptom 63
Isolate the Issue...63
Fix the Issue...66
Verify...66

Build and Analyze the Baseline

When you start with the scenario, all you have is the diagram and some textual information on the network baseline. Your goal at this moment is structuring the available information and making additional diagrams. We recommend extra diagrams to outline the following: L2 connection, BGP Peerings, Multicast & Redistribution and IPv6 Topology. Notice that some of these could be combined in a single diagram – for example you may put the Multicast and Redistribution outlines on the initial L3 diagram. This is probably the best way to save your time during the analysis stage. However, we are going to use separate diagrams for the ease of explanation here.

Layer 2 Diagram

This is where our Ethernet L2 connections are outlined (notice, no L3 Etherchannels). We outline the Root Bridge, VTP domains and roles, port numbers and trunk encapsulations. This diagram would helps us finding any L2 mis-configurations. Notice that we only put the routers that have configured Ethernet connections on this diagram.

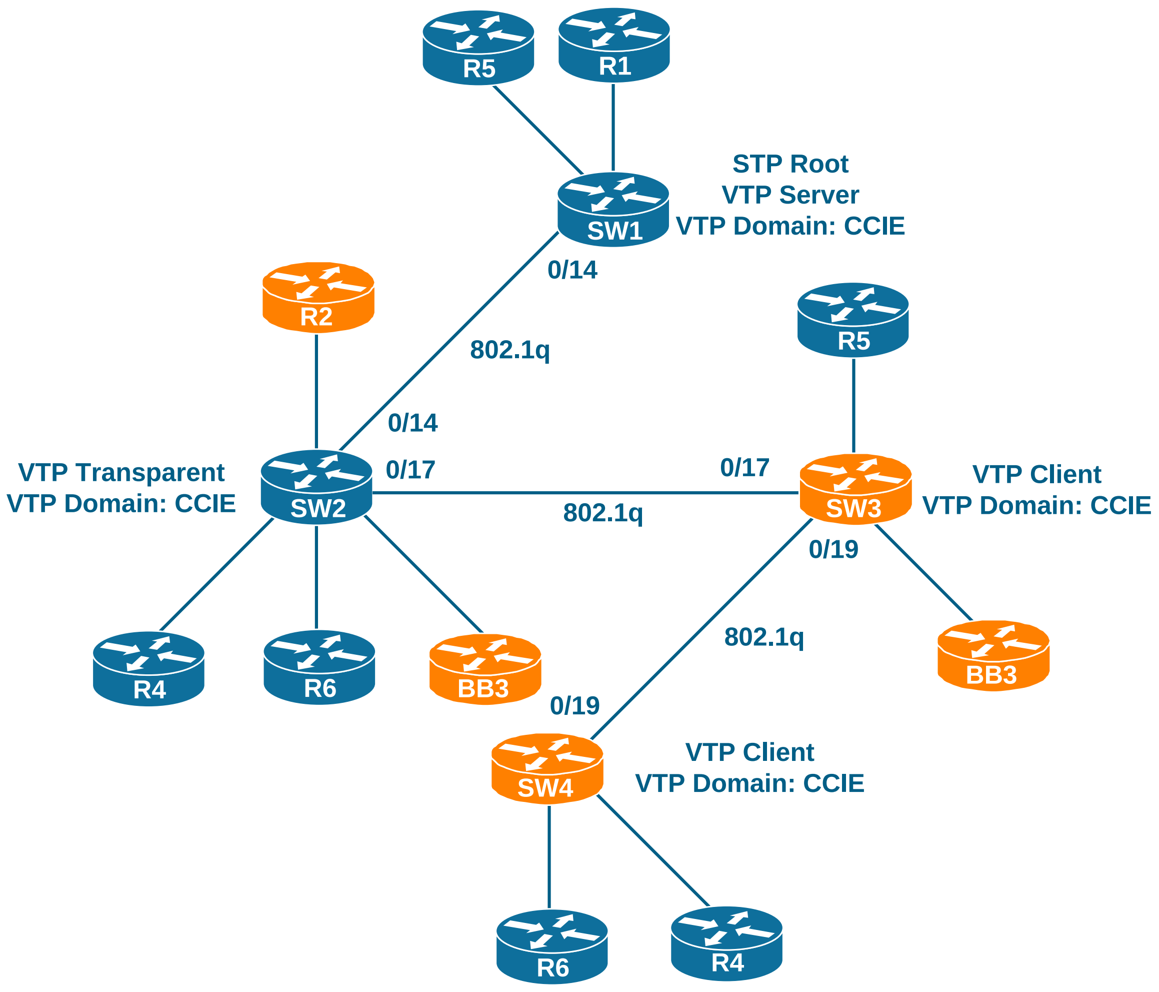


Fig 1

The topology is loopless, and there should be no STP-blocked ports. All VLANs follow the same logical topology, so there is no need to outline logical paths. There does not appear to be any obvious issues with this setup, so we'll just save this diagram for further referencing.

BGP Diagram

Using the information found in the Baseline configuration outline, we may draw a BGP peering diagram. The red arrows mark eBGP peering sessions while blue arrows mark iBGP sessions. Notice that we omit the link IP addressing and use simplified drawing for Ethernet/Serial/FR links.

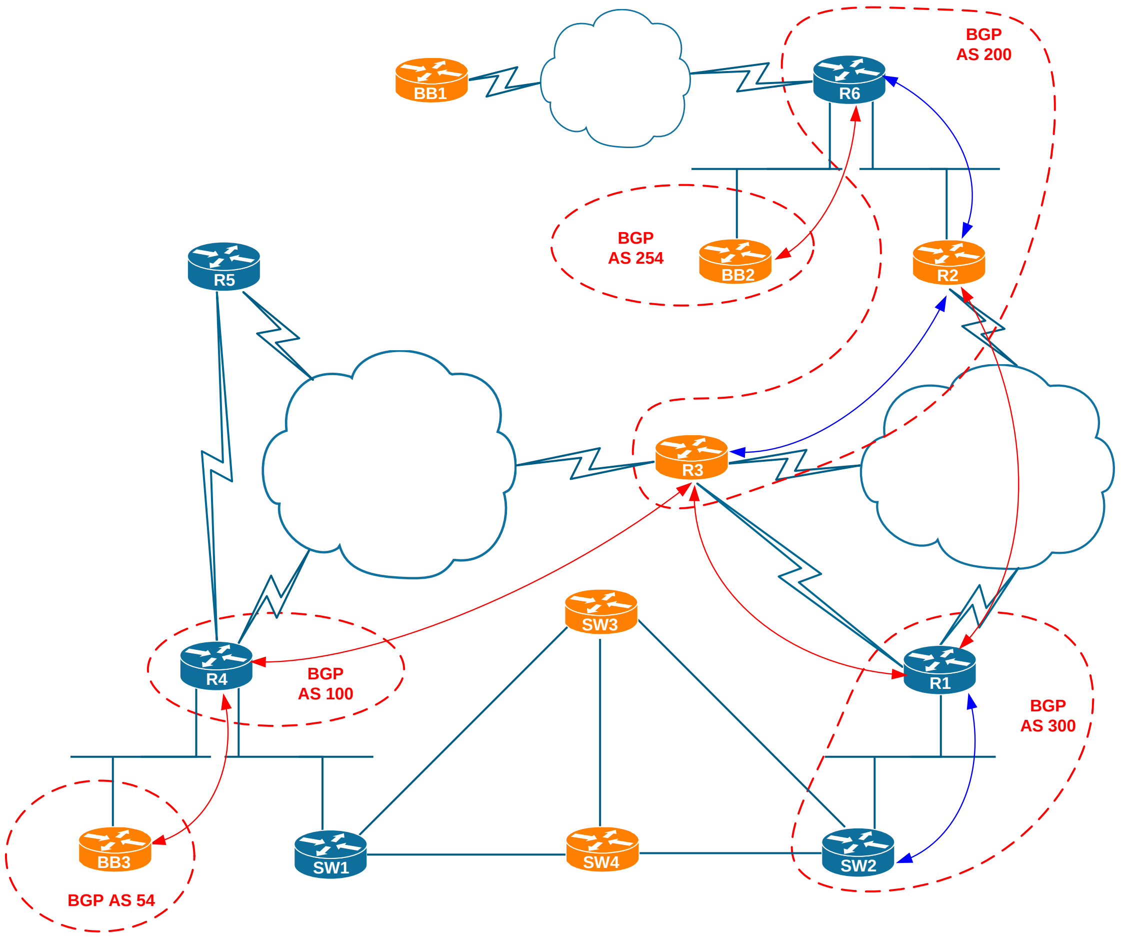


Fig 2

We can see that the eBGP session between R3 and R4 may take either of two IGP paths, and therefore is most-likely sourced using the Loopback0 interfaces for redundancy. There does not appear to be much information on BGP policies, so we'll leave detailed analysis till the moment we need anything related to BGP.

Multicast and Redistribution

We usually group these two technologies on the same diagram because they both depend on IGP protocols. Thus, the diagram should have IGPs outlined in addition to the Multicast and redistribution marking. We mark the links that have PIM enabled using the **RED** color. This makes it easy to see where the multicast traffic may actually flow. Notice that we copied some interface names from the main L3 diagram so we know the multicast-enabled interfaces by name.

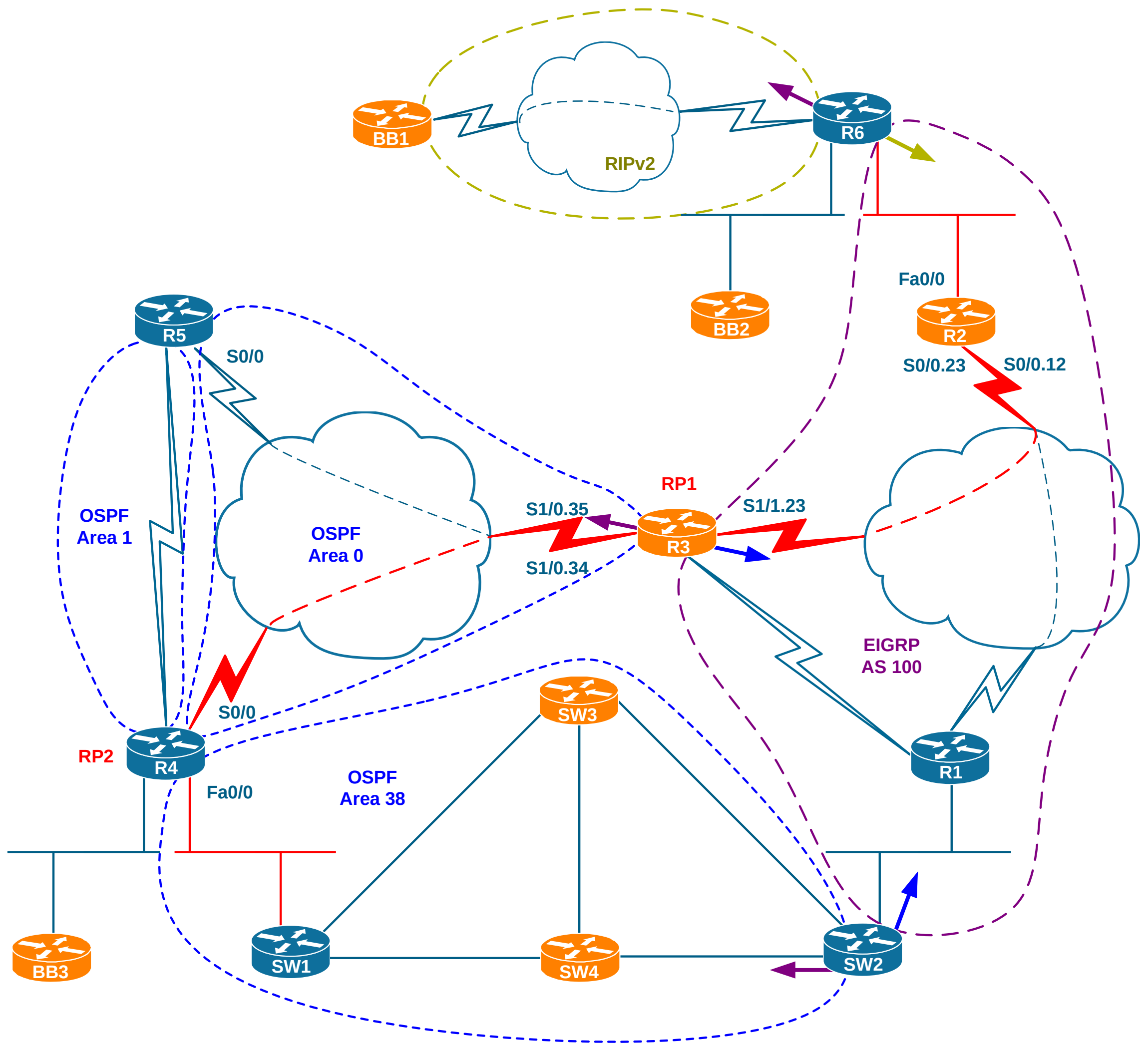


Fig 3

You can also see the PIM SM RPs outlined on this diagram. The detailed group to RP mappings are not provided due to the high amount of group ranges.

Redistribution Loops Analysis

Look at the diagram on Fig 3. We use arrows on the redistribution points to show the direction of the protocol information being injected and color to specify the protocol being redistributed. For instance, if you look at R3, you will see EIGRP being injected into OSPF and OSPF being injected into EIGRP based on the arrow colors. If there are no multiple points of redistribution, your work is already done. However, if the redistribution appears to be complex, some detailed analysis may be required.

Prior to starting analysis, notice that there are three IGP domains in the topology: OSPF, EIGRP AS 100 and RIP. Let's see if there are possibilities for routing loops. By definition, routing loops occur when routing information re-enters the same routing domain twice, i.e. "returns back". To see if that's possible with any of our routing domains, we use the following "tracing" procedure.

The idea of the "tracing" procedure is building the path that routing information from a selected IGP domain may take through the topology and see if that path loops to itself. Here is the tracing algorithm, consisting of two nested loops, exhausting all routing domains and information flows.

Loop 1: Find Routing Information Flows.
Input: Routing Domains, Redistribution Points.
Output: Beginnings of Information flows.

Pick up a domain and see where the information flows out of it. Use the arrows on the diagram you created. Record the points of exit and the destination domain. Use the diagram you created for reference.

Loop 2: Find Looping Routing Information Flows.
Input: Starting redistribution points for the Flows.
Output: Looped Flows, Points of Intersection.

Pick up the first information flow and trace it through the topology across all applicable redistribution points. Find the flows that loops and the redistribution point where this happens.

Rule 1: A flow may re-enter the domain it already traversed if the following is true:

a) The AD of the source domain is lower than AD of the destination domain. E.g. RIP AD is lower than EIGRP External AD.
b) There is no filtering to prevent it.

Notice that "re-entering" may only happen at a redistribution point.

End Rule 1.

Trace every flow until it's stopped by **`Rule 1`** or stub domain or looped to itself. Record the redistribution points where looping occurs.

End Loop 2

Continue to the next IGP domain and extract the routing information flows.

End Loop 1

Let's see how this work for our scenario. Start with any routing domain used in this scenario, for example take EIGRP AS 100. The information from EIGRP AS 100 leaks in three directions via three redistribution points at R6, R3 and SW2 respectively. This is clearly seen from the diagram.

1. The first flow exiting via R6 enters "stub" RIP domain and does not form a loop.
2. The second flow via R3 flows across the OSPF domain down to SW1 and finally comes to SW2. At this point EIGRP 100 information may enter back the domain of origin. However, OSPF External AD of 110 is higher than EIGRP native AD of 90, and thus the flow does not loop.
3. The third flow via SW2 is symmetric to the flow via R3 and thus does not end in a loop, unless ADs are tuned for some reason.

If we look at the OSPF and EIGRP domains we see that they are symmetrical to each other. Furthermore, OSPF redistributed into EIGRP has AD of 170 by default, which is way higher than OSPF's native AD of 110. Thus, we may quickly conclude based on the symmetry assumption that OSPF information will not loop back to OSPF either.

The last domain left in this scenario is RIP. There is just one flow out of this domain – the one injected into EIGRP AS 100. The flow further splits into two branches entering the OSPF domain via R3 and SW2. Let's trace the first branch via R3. The information flows down to SW2 and we compare the ADs at that point. OSPF default external AD of 110 is preferred over EIGRP external AD of 170 used for RIP routes. Thus, the information *loops* back to the same domain it has already passed through! Using the symmetry consideration, we may conclude that the second RIP flow branch via SW2 could re-enter EIGRP domain via R3.

Note

If you find difficulties tracing the information flows on the combined diagram, you can make a separate one, consisting of just IGP domains and redistribution points, such as one provided below: You can see the "RIP" flow being traced across the domains on the figure below

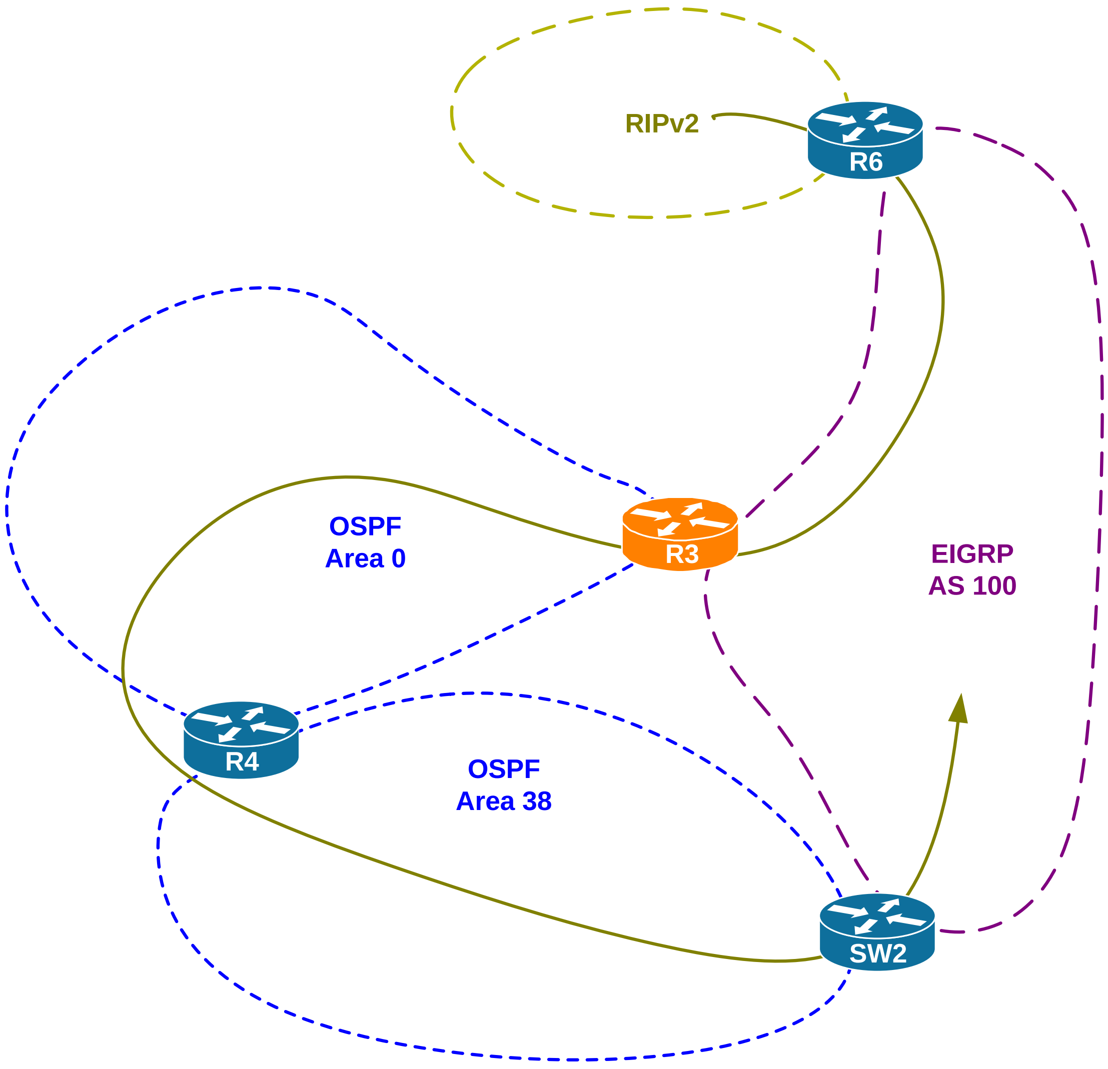


Fig 4

Based on the analysis performed, we conclude that some sort of filtering should be implemented at R3 and SW2: using tags, access-list or AD manipulation. In the simplest case you would just increase OSPF External AD over 170 on R2 and SW2, thus preventing the looped flows per **Rule 1**.

Since R3 is not under our control, we should be mostly concerned with SW2. However, keep in mind that loop-prevention might be already done for you.

Multicast Propagation Analysis

Refer to **Fig 3** when analyzing the multicast section. In our scenario, the multicast network spans two IGP domains. This may potentially result in suboptimal routing and RPF failures. Multicast will always flow across R3, no matter if it’s sourced in OSPF or EIGRP 100 domain. Thus we need to watch out and make sure that OSPF domain prefers reaching the EIGRP domain via R3 and vice versa in order to avoid RPF failures. This is just a note – the baseline configuration might be already taking care of this for us. Lastly, since RPs are configured statically, there should be no issue with RP information propagation.

IPv6 Diagram

The IPv6 part of this scenario appears to be simple. Just one IGP, and IPv6 is only configured on three routers. Plus, only one router is under our control, so we may expect not to have any major issues with IPv6.

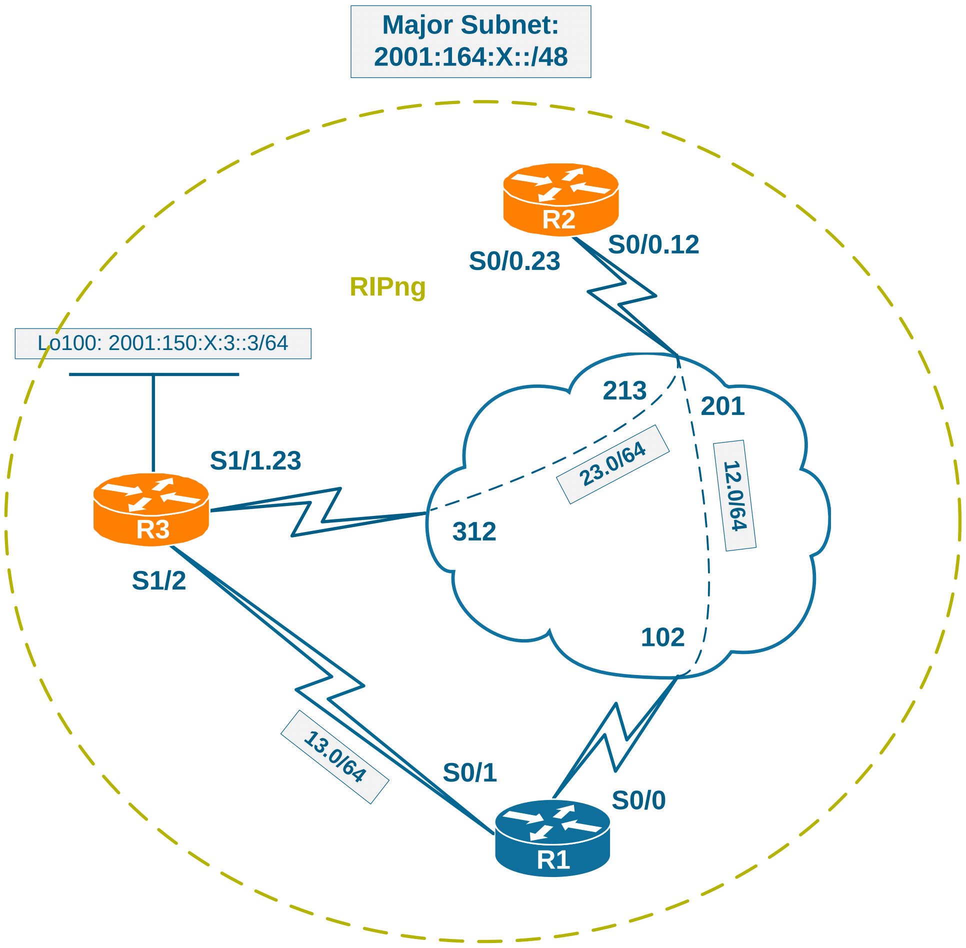


Fig 5

Read over the Lab

Our last step is looking through the tickets and seeing If we can extract any potentially useful information. If you look at Ticket 1 and Ticket 4 you will see that they deal with the same segment – VLAN 7. Furthermore, both tickets deal with the flows to/from VLAN55, and Ticket 4 says there are end-to-end issues, while Ticket 1 mentions only QoS problems. This “overlap” between the tickets means we should be careful when performing isolation for Ticket 1, and don take VLAN 7 in consideration.

Ticket 5 prompts us that we may have problems communicating with R6 at all. We’re not sure about the extent of the problem, but we need to account for that when dialing with other tickets.

Tickets 3 and 8 warn us that there could be some problems in communications between R1/R3 and R1/SW2. This may affect IGP routing, so we should watch out.

Lastly, Ticket 10 explicitly states that it depends on two other tickets. This probably makes it the latest candidate to deal with, especially if you look at the point value associated with the ticket.

Solutions

Ticket 1

Analyze the Symptoms

No one complains on the end-to-end connectivity breakouts. Thus, it appears that Layer 3 is not the source of the problem. The problem could not be caused by link issues, as the voice quality towards VLAN 7 is not affected. The asymmetric behavior prompts that there might be something different in the paths that traffic take between VLAN 5 and VLAN 7. For example, if the voice traffic from VLAN 7 to VLAN 5 flows over the slow backup link, the quality may be severely affected. This is our first hypothesis. Then, there could be some QoS configuration differences in R4 or R5 based on the same asymmetry fact. This is our second hypothesis. The initial problem area involves R3, R4 and R5. However, R3 is not under our control so we're only limited to R4 and R5.

`Hypothesis 1:` Traffic routing asymmetry.
`Hypothesis 2:` QoS configuration asymmetry.
`Problem Scope`: R4 and R5.

Isolate the Issue

As we already dropped off link issues, we're using the Divide and Conquer approach. Now we start with some network layer testing to probe our first hypothesis. Our primary tool here is **`traceroute`**:

```
Rack1R4#traceroute 164.1.5.5

Type escape sequence to abort.
Tracing the route to 164.1.5.5

  1 164.1.34.3 32 msec 97 msec 28 msec
  2 164.1.35.5 60 msec *  56 msec
Rack1R4#

Rack1R5#traceroute 164.1.47.4

Type escape sequence to abort.
Tracing the route to 164.1.47.4

  1 164.1.35.3 28 msec 32 msec 33 msec
  2 164.1.34.4 56 msec *  56 msec
```

The paths look symmetric, so it's highly unlikely that our first hypothesis is valid. We may continue to our second hypothesis.

First, quickly check the QoS settings in R4 and R5. At the very minimum, voice should be given priority treatment, and fragmentation should be applied as the links are low-rate.

```
Rack1R5#show frame-relay pvc 503

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 503, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0

  input pkts 810           output pkts 676          in bytes 95786
  out bytes 59152          dropped pkts 0           in pkts dropped 0
  out pkts dropped 0                out bytes dropped 0
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0          in DE pkts 0             out DE pkts 0
  out bcast pkts 665       out bcast bytes 58704
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
  pvc create time 01:37:09, last time pvc status changed 01:36:46
  fragment type end-to-end fragment size 320
  cir 256000    bc   2560      be 0         limit 320    interval 10
  mincir 256000    byte increment 320   BECN response no  IF_CONG no
  frags 680       bytes 59152     frags delayed 8         bytes delayed
1504
  shaping inactive
  traffic shaping drops 0
  service policy LLQ
 Serial0/0: DLCI 503 -

  Service-policy output: LLQ

    Class-map: VoIP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: access-group name VoIP
      Queueing
        Strict Priority
        Output Queue: Conversation 40
        Bandwidth 200 (kbps) Burst 5000 (Bytes)
        (pkts matched/bytes matched) 0/0
        (total drops/bytes drops) 0/0

    Class-map: class-default (match-any)
      1 packets, 84 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
  Output queue size 0/max total 600/drops 0
```

The relevant output values are highlighted. We can see that the link is shaped to 256K – while this is not the optimal value, it should be OK in most cases. We see traffic being fragmented according to the fragment size of 320 bytes. Let's check if this fragment will take 10ms to serialize:

`delay = frag_size/link_rate = 320*8/256000=10ms`

To make sure the fragmentation is actually working, check that the number of fragments increment when you ping across the cloud. Clear the FR statistics prior to doing this:

```
Rack1R5#clear frame-relay counter interface serial 0/0

Rack1R5#ping 150.1.3.3 size 1500

Type escape sequence to abort.
Sending 5, 1500-byte ICMP Echos to 150.1.3.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 481/484/489
ms
Rack1R5#show frame-relay fragment
interface                 dlci frag-type  size in-frag    out-frag
dropped-frag
Se0/0                     503  end-to-end 320  27         25         0
```

Thus everything appears to be good. Let's see if we have any differences at R4.

```
Rack1R4#show frame-relay pvc 403

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 403, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0

  input pkts 1220          output pkts 1148         in bytes 119868
  out bytes 85692          dropped pkts 0           in pkts dropped 0
  out pkts dropped 0                out bytes dropped 0
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0          in DE pkts 0             out DE pkts 0
  out bcast pkts 1023      out bcast bytes 78084
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
  pvc create time 01:39:49, last time pvc status changed 01:39:37
  fragment type end-to-end fragment size 320
  cir 256000    bc   2560      be 0         limit 320    interval 10
  mincir 256000    byte increment 320   BECN response no  IF_CONG no
  frags 1152      bytes 85692     frags delayed 7         bytes delayed
1488
  shaping inactive
  traffic shaping drops 0
  service policy LLQ
 Serial0/0: DLCI 403 -

  Service-policy output: LLQ

    Class-map: VoIP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: access-group name VoIP
      Queueing
```

```
        Output Queue: Conversation 41
        Bandwidth 200 (kbps)Max Threshold 64 (packets)
        (pkts matched/bytes matched) 0/0
        (depth/total drops/no-buffer drops) 0/0/0

    Class-map: class-default (match-any)
      1148 packets, 85656 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
  Output queue size 0/max total 600/drops 0
```

The settings appear to be the same, the CIR, the fragment size, the voice bandwidth. Let's check if the fragmentation is working the same way.

```
Rack1R4#ping 150.1.3.3 size 1500

Type escape sequence to abort.
Sending 5, 1500-byte ICMP Echos to 150.1.3.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 480/480/481
ms

Rack1R4#show frame-relay fragment
interface                dlci frag-type  size in-frag    out-frag
dropped-frag
Se0/0                    403  end-to-end 320  30         27         0
```

So everything appears to be the same, but still there is something missing. What if we take a closer look at the service-policy attached in the "problematic" direction.

```
  Service-policy output: LLQ

    Class-map: VoIP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: access-group name VoIP
      Queueing
        Output Queue: Conversation 41
        Bandwidth 200 (kbps)Max Threshold 64 (packets)
        (pkts matched/bytes matched) 0/0
        (depth/total drops/no-buffer drops) 0/0/0

    Class-map: class-default (match-any)
      1148 packets, 85656 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
  Output queue size 0/max total 600/drops 0
```

If you compare it to the service-policy applied in the "right" direction you will see

```
  Service-policy output: LLQ

    Class-map: VoIP (match-all)
```

```
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: access-group name VoIP
      Queueing
        Strict Priority
        Output Queue: Conversation 40
        Bandwidth 200 (kbps) Burst 5000 (Bytes)
        (pkts matched/bytes matched) 0/0
        (total drops/bytes drops) 0/0
```

That the policy from R4 to R5 does not use LLQ, but uses simple bandwidth reservation. This configuration has two effects: first, it doest no guarantee best service to voice packets, and secondly it disables interleaving for VoIP packets and data packet fragments. Thus we found that our second hypothesis was valid.

Conclusion: Pay due diligence and look through the command output carefully. When facing “asymmetry” try comparing the “show” commands output and see if you can find any mismatches.

Fix the issue

```
R4:
policy-map LLQ
  class VoIP
    no bandwidth
    priority 200
```

Verify

Simply check that the VoIP queue now uses priority service (notice the Conversation ID of 40, which corresponds to the priority queue on a low-speed interface)

```
Rack1R4#show policy-map interface serial 0/0
 Serial0/0: DLCI 403 -

  Service-policy output: LLQ

    Class-map: VoIP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: access-group name VoIP
      Queueing
        Strict Priority
        Output Queue: Conversation 40
        Bandwidth 200 (kbps) Burst 5000 (Bytes)
        (pkts matched/bytes matched) 0/0
        (total drops/bytes drops) 0/0

    Class-map: class-default (match-any)
      188 packets, 22090 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
```

Ticket 2

Analyze the Symptoms

This is a typical end-to-end problem, where one node cannot communicate to another in proper manner. Let's see what our routing table at SW2 shows up:

```
Rack1SW2#show ip route 164.1.9.0
Routing entry for 164.1.9.0/24
  Known via "ospf 1", distance 110, metric 21, type intra area
  Redistributing via eigrp 100
  Advertised by eigrp 100 metric 10000 1000 1 255 1500 route-map OSPF-
>EIGRP
  Last update from 164.1.24.10 on FastEthernet0/21, 00:05:04 ago
  Routing Descriptor Blocks:
  * 164.1.24.10, from 150.1.9.9, 00:05:04 ago, via FastEthernet0/21
      Route metric is 21, traffic share count is 1
```

Apparently, the link is being advertised by SW3 and SW2 uses the path via SW4 to reach it. The problem scope includes three devices: SW2, SW3 and SW4. There are two things that come to mind immediately: something is wrong with the link between SW2 and SW4 or link metric are not configured properly.

Hypothesis 1: Issues with the link between SW2 and SW4.
Hypothesis 2: Improper metric configuration.
Initial problem scope: SW2, SW3 and SW4.

Isolate the Issue

Using the "divide-and-conquer" approach we start by testing the EtherChannel link health. This will probe our first hypothesis.

```
Rack1SW2#ping 164.1.23.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 164.1.23.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 109/114/118
ms
```

OK this rules out most physical issues. The next obvious check is seeing if we have OSPF neighbors on the link:

```
Rack1SW2#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address
Interface
150.1.9.9         1   FULL/BDR        00:00:32    164.1.32.9      Port-
channel23
150.1.10.10       1   FULL/DR         00:00:31    164.1.24.10
FastEthernet0/21
```

The first hypothesis does not appear to be valid so far as even the neighbors came up. Let’s see if we have any luck researching the second guess – the mismatched metrics. Let’s see the OSPF metric over the active path:

```
Rack1SW2#show ip route 164.1.9.0 | inc metric
  Known via "ospf 1", distance 110, metric 21, type intra area
  Advertised by eigrp 100 metric 10000 1000 1 255 1500 route-map OSPF-
>EIGRP
      Route metric is 21, traffic share count is 1
```

We record down the number of 21. Now let’s check what path SW4 chooses to reach VLAN9:

```
Rack1SW2#traceroute 164.1.9.9

Type escape sequence to abort.
Tracing the route to 164.1.9.9

  1 164.1.24.10 0 msec 8 msec 0 msec
  2 164.1.43.9 0 msec *  0 mse
```

This appears to be good, as SW4 prefer the direct link to SW3. Now let’s compare the cost of the direct link between SW2 and SW4 to the cost of the path being used now:

```
Rack1SW2#show ip ospf interface port-channel 23 | inc Cost
  Process ID 1, Router ID 150.1.8.8, Network Type BROADCAST, Cost: 20
```

So it’s 20 and the cost to reach VLAN9 via SW4 is 21. Most likely the cost of VLAN9 interface is simply “1”. To make sure it is, we do the following check – look for all links advertised by SW3 and see their metrics

```
Rack1SW2#show ip ospf database router adv-router 150.1.9.9

            OSPF Router with ID (150.1.8.8) (Process ID 1)

                Router Link States (Area 38)

  LS age: 1952
  Options: (No TOS-capability, DC)
  LS Type: Router Links
  Link State ID: 150.1.9.9
  Advertising Router: 150.1.9.9
  LS Seq Number: 80000011
  Checksum: 0xF5A
  Length: 96
  Number of Links: 6

<snip>

    Link connected to: a Stub Network
```

```
     (Link ID) Network/subnet number: 164.1.9.0
     (Link Data) Network Mask: 255.255.255.0
      Number of TOS metrics: 0
       TOS 0 Metrics: 1

<snip>
```

It appears to be that both paths – via SW4 and direct - have the same cost to reach VLAN9 from SW2. Thus, our both hypothesis appear to be invalid. Here is the situation that we have now:

- No link errors preventing OSPF neighbor adjacencies from coming up.
- No mismatching link metrics that prevent equal cost load-balancing
- OSPF selects just one path during SPF for some reason

The next step would be trying to look at the OSPF topology and see why it does not include direct path into calculations or selection. We are going to look at every router's (SW2, SW3 and SW4) router type LSA and see if there is anything wrong with those. We start with SW2 – notice that in the output that follows only relevant links are shown, the links that connect SW2 to SW3 and SW4:

```
Rack1SW2#show ip ospf database router self-originate

            OSPF Router with ID (150.1.8.8) (Process ID 1)

                Router Link States (Area 38)

  LS age: 887
  Options: (No TOS-capability, DC)
  LS Type: Router Links
  Link State ID: 150.1.8.8
  Advertising Router: 150.1.8.8
  LS Seq Number: 8000000C
  Checksum: 0xFA86
  Length: 48
  AS Boundary Router
  Number of Links: 2

    Link connected to: a Transit Network
     (Link ID) Designated Router address: 164.1.32.8
     (Link Data) Router Interface address: 164.1.32.8
      Number of TOS metrics: 0
       TOS 0 Metrics: 20

    Link connected to: a Transit Network
     (Link ID) Designated Router address: 164.1.24.10
     (Link Data) Router Interface address: 164.1.24.8
      Number of TOS metrics: 0
       TOS 0 Metrics: 10
```

Here we have two transit networks connected to SW3 and SW4. When looking at the database, notice the link type. In our case, both links are transit, meaning this router is not along on those and see some neighbors. Now it's time to check the relevant entries in SW4's database:

Note

The router-ID is usually the loopback IP address. If you don't know the router ID you may find it using the command **`show ip ospf neighbor`** or peek it from the database output.

```
Rack1SW2# show ip ospf database router adv-router 150.1.10.10

            OSPF Router with ID (150.1.8.8) (Process ID 1)

                Router Link States (Area 38)

  LS age: 1480
  Options: (No TOS-capability, DC)
  LS Type: Router Links
  Link State ID: 150.1.10.10
  Advertising Router: 150.1.10.10
  LS Seq Number: 8000000D
  Checksum: 0x55C4
  Length: 72
  Number of Links: 4

    Link connected to: a Transit Network
     (Link ID) Designated Router address: 164.1.43.10
     (Link Data) Router Interface address: 164.1.43.10
      Number of TOS metrics: 0
       TOS 0 Metrics: 10

    Link connected to: a Transit Network
     (Link ID) Designated Router address: 164.1.24.10
     (Link Data) Router Interface address: 164.1.24.10
      Number of TOS metrics: 0
       TOS 0 Metrics: 10
<snip>
```

At networks appear to be in order – both are marked as transit an OSPF SPF should account for these links as valid entries in the topology. Let's check the database of SW3 once again:

```
Rack1SW2# show ip ospf database router adv-router 150.1.9.9

            OSPF Router with ID (150.1.8.8) (Process ID 1)

                Router Link States (Area 38)

  LS age: 1323
  Options: (No TOS-capability, DC)
  LS Type: Router Links
  Link State ID: 150.1.9.9
  Advertising Router: 150.1.9.9
  LS Seq Number: 80000012
  Checksum: 0xD5B
  Length: 96
  Number of Links: 6

    Link connected to: a Transit Network
     (Link ID) Designated Router address: 164.1.43.10
     (Link Data) Router Interface address: 164.1.43.9
      Number of TOS metrics: 0
       TOS 0 Metrics: 10

    Link connected to: another Router (point-to-point)
     (Link ID) Neighboring Router ID: 150.1.8.8
     (Link Data) Router Interface address: 164.1.32.9
      Number of TOS metrics: 0
       TOS 0 Metrics: 20

    Link connected to: a Stub Network
     (Link ID) Network/subnet number: 164.1.32.0
     (Link Data) Network Mask: 255.255.255.0
      Number of TOS metrics: 0
       TOS 0 Metrics: 20

<snip>
```

The above shown are the links connecting to SW2 and SW4. Notice the link to SW4 looks normal – as a regular transit network. However, the link to SW2 is marked as “point-to-point” plus it generates a “stub network” entry. Even though the adjacency is UP, the link types at every end seem to mismatch, because SW2 treats the same link as “transit” subnet. The reason why is quickly deducted from the “point-to-point” keyword found in the first entry – SW3 must be configured for link type point-to-point on this interface! This is what dangerous about OSPF adjacencies – they come up as soon as timer values match. However, if the topological views of the link aren’t the same, the OSPF will never be able to have a complete graph of the network connection.

Conclusion: Don’t trust the OSPF adjacencies! If you see some path being ignore while adjacencies are up, there might be something wrong with the topology!

Fix the Issue

```
SW2:
interface Port-Channel 23
 ip ospf network point-to-point
```

Verify

Check that SW2 now has two equal-cost paths to reach VLAN9. One path via SW3 and another via SW4:

```
Rack1SW2#show ip route 164.1.9.9
```

```
Routing entry for 164.1.9.0/24
  Known via "ospf 1", distance 110, metric 21, type intra area
  Redistributing via eigrp 100
  Advertised by eigrp 100 metric 10000 1000 1 255 1500 route-map OSPF-
>EIGRP
  Last update from 164.1.24.10 on FastEthernet0/21, 00:03:13 ago
  Routing Descriptor Blocks:
    164.1.32.9, from 150.1.9.9, 00:03:13 ago, via Port-channel23
      Route metric is 21, traffic share count is 1
  * 164.1.24.10, from 150.1.9.9, 00:03:13 ago, via FastEthernet0/21
      Route metric is 21, traffic share count is 1
```

Ticket 3

Analyze the Symptoms

The ticket clearly mentions a BPG peering issue. The good thing is that problem scope is limited to just the link between the two routers. As it was working before the changes, we may assume link is still healthy. An extra check would never hurt:

```
Rack1R1#ping 164.1.13.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 164.1.13.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/29/32 m
```

Good, so we already did some isolation and found the network layer is OK. The problem might be related to the next layer – transport. So what do we know about BGP peering establishment?

- BGP uses TCP as underlying transport, destination port 179.
- Both peers attempt to establish TCP connections simultaneously.
- BGP uses IP TTL of 1 to establish BGP peering on directly connected interfaces.
- Source and destination IP addresses must match the ones configured in the **`neighbor`** command.
- BGP sources session of the primary IP address on the interface.

Everything that happens after the transport session establishment is usually reflected in BGP notification messages – e.g. mismatched BGP AS numbers. We don have some logging message – from the one provided in the scenario it is apparent that BGP cannot successfully exchange KEEPALIVE messages at the very least. The second output shows us BGP State “OpenSent” meaning BGP was able to establish the TCP session and send at least one packet out. Plus, we know the remote peer is using our correct interface IP address.

```
Rack1R3#show ip bgp neighbors 164.1.13.1
BGP neighbor is 164.1.13.1,  remote AS 300, external link
  BGP version 4, remote router ID 0.0.0.0
  BGP state = OpenSent
```

So the issue is somewhere at the transport level. But we don’t have any working hypothesis yet – more information needed. The first check you may run on a BGP session using telnet to simulate a TCP connection:

```
Rack1R1#telnet 164.1.13.3 179
Trying 164.1.13.3, 179 ... Open
```

```
fff
ff
434ewr
ÿÿÿÿÿÿÿÿ

[Connection to 164.1.13.3 closed by foreign host]
```

That shows we were able to establish a TCP connection on port 179 and exchange some data. That means there are no traffic filters on the path to block the connection based on just L3 and L4 settings. So our only hypothesis is – something does not match in the configurations that block transport connection from getting properly established.

Hypothesis: Configuration mismatch
Initial problem domain: R1 and R3.

Isolate the Issue

The problem seems to be complicated. We might consider using "heavy" artillery and trace the actual TCP transport connection. Here is what we would do

- Enable TCP transactions debugging using the command **debug ip tcp transactions.**
- Enable TCP packet dumps by creating an access list as follows and enable the debug command:

  ```
  access-list 100 permit tcp any host 164.1.13.1
  access-list 100 permit tcp host 164.1.13.3 any

  Debug ip packet detail 100 dump
  ```

- Enable BGP events debugging using the command **debug ip bgp**.

You will see output similar to the following:

Note

This is our local SYN being sent to the remote side – R1 attempts active open.

```
TCP: sending SYN, seq 415664179, ack 0
TCP0: Connection to 164.1.13.3:179, advertising MSS 1460
IP: tableid=0, s=164.1.13.1 (local), d=164.1.13.3 (Serial0/1), routed
via FIB
IP: s=164.1.13.1 (local), d=164.1.13.3 (Serial0/1), len 44, sending
    TCP src=48285, dst=179, seq=415664179, ack=0, win=16384 SYN

07E3CF50:                   45C0002C 5E6C0000          E@.,^l..
07E3CF60: 0106F899 A4010D01 A4010D03 BC9D00B3  ..x.$...$...<..3
```

```
07E3CF70: 18C68833 00000000 60024000 97D50000  .F.3....`.@..U..
07E3CF80: 020405B4                             ...4
TCP0: state was CLOSED -> SYNSENT [48285 -> 164.1.13.3(179)]
```

Note

The following is the SYN packet we receive from the other party. There is no ACK flag set, so either the other side did not see our SYN packet or we're opening simultaneously.

```
IP: tableid=0, s=164.1.13.3 (Serial0/1), d=164.1.13.1 (Serial0/1),
routed via RIB
IP: s=164.1.13.3 (Serial0/1), d=164.1.13.1 (Serial0/1), len 44, rcvd 3
    TCP src=36826, dst=179, seq=1406017359, ack=0, win=16384 SYN

07E3BB50:          FF030021 45C0002C 86FB0000      ...!E@.,.{..
07E3BB60: FE06D309 A4010D03 A4010D01 8FDA00B3  ~.S.$...$....Z.3
07E3BB70: 53CE1F4F 00000000 60024000 F2740000  SN.O....`.@.rt..
07E3BB80: 020405B4                             ...4
```

Note

The local side processes SYN, replies with a SYN ACK packet. R1 processes the passive open from R3:

```
TCB85684DD8 created
TCP0: state was LISTEN -> SYNRCVD [179 -> 164.1.13.3(36826)]
TCP: tcb 85684DD8 connection to 164.1.13.3:36826, peer MSS 1460, MSS is
516
TCP: sending SYN, seq 1923456560, ack 1406017360
TCP0: Connection to 164.1.13.3:36826, advertising MSS 1460
```

Note

The remote side sends an ACK to our SYN ACK packet – just match the sequence numbers in the packets to see that.

```
IP: tableid=0, s=164.1.13.3 (Serial0/1), d=164.1.13.1 (Serial0/1),
routed via RIB
IP: s=164.1.13.3 (Serial0/1), d=164.1.13.1 (Serial0/1), len 40, rcvd 3
    TCP src=36826, dst=179, seq=1406017360, ack=1923456561, win=16384
ACK
07CA3310:          FF030021 45C00028 86FC0000      ...!E@.(.|..
07CA3320: FE06D30C A4010D03 A4010D01 8FDA00B3  ~.S.$...$....Z.3
07CA3330: 53CE1F50 72A59E31 50104000 F94A0000  SN.Pr%.1P.@.yJ..
07CA3340:
```

Note

The local router now considers the TCP connection to be established.

```
TCP0: state was SYNRCVD -> ESTAB [179 -> 164.1.13.3(36826)]
TCB8446B3E0 callback, connection queue = 1
TCB8446B3E0 accepting 85684DD8 from 164.1.13.3.36826
```

Note

We receive the first BGP message from R3 – which is supposed to be the OPEN message. The connection is considered to be passively open. Below you can see some of the debugging output related to TCP transaction and BGP message.

```
IP: tableid=0, s=164.1.13.3 (Serial0/1), d=164.1.13.1 (Serial0/1),
routed via RIB
IP: s=164.1.13.3 (Serial0/1), d=164.1.13.1 (Serial0/1), len 85, rcvd 3
    TCP src=36826, dst=179, seq=1406017360, ack=1923456561, win=16384
ACK PSH

07E3C190:          FF030021 45C00055 86FD0000      ...!E@.U.}..
07E3C1A0: FE06D2DE A4010D03 A4010D01 8FDA00B3  ~.R^$...$....Z.3
07E3C1B0: 53CE1F50 72A59E31 50184000 3CDB0000  SN.Pr%.1P.@.<[..
07E3C1C0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF  ................
07E3C1D0: 002D0104 00C800B4 96010303 10020601  .-...H.4........
07E3C1E0: 04000100 01020280 00020202 00        .............

BGP: 164.1.13.3 passive open to 164.1.13.1
```

Note

The below output is for our router closing the active connection, which it has originated initially. Our side sends a BGP OPEN message to the peer and processes the OPEN message received from the peer.

```
TCP0: state was SYNSENT -> CLOSED [48285 -> 164.1.13.3(179)]
TCB 0x8496ABE8 destroyed
BGP: 164.1.13.3 went from Active to Idle
BGP: 164.1.13.3 went from Idle to Connect
TCB8446B3E0 setting property TCP_IN_TTL (23) 844A33A0
TCB8446B3E0 setting property TCP_OUT_TTL (24) 844A33A0
TCB85684DD8 setting property TCP_OUT_TTL (24) 8446B2DA
BGP: 164.1.13.3 rcv message type 1, length (excl. header) 26
BGP: 164.1.13.3 rcv OPEN, version 4, holdtime 180 seconds
BGP: 164.1.13.3 went from Connect to OpenSent
BGP: 164.1.13.3 sending OPEN, version 4, my as: 300, holdtime 180
seconds
BGP: 164.1.13.3 rcv OPEN w/ OPTION parameter len: 16
```

```
BGP: 164.1.13.3 rcvd OPEN w/ optional parameter type 2 (Capability) len
6
BGP: 164.1.13.3 OPEN has CAPABILITY code: 1, length 4
BGP: 164.1.13.3 OPEN has MP_EXT CAP for afi/safi: 1/1
BGP: 164.1.13.3 rcvd OPEN w/ optional parameter type 2 (Capability) len
2
BGP: 164.1.13.3 OPEN has CAPABILITY code: 128, length 0
BGP: 164.1.13.3 OPEN has ROUTE-REFRESH capability(old) for all address-
families
BGP: 164.1.13.3 rcvd OPEN w/ optional parameter type 2 (Capability) len
2
BGP: 164.1.13.3 OPEN has CAPABILITY code: 2, length 0
BGP: 164.1.13.3 OPEN has ROUTE-REFRESH capability(new) for all address-
families
BGP: 164.1.13.3 rcvd OPEN w/ remote AS 200
BGP: 164.1.13.3 went from OpenSent to OpenConfirm
BGP: 164.1.13.3 send message type 1, length (incl. header) 45
```

Note

If you will follows the debugging output after this, you will see that R3 continues sending the same OPEN message to R1 – like it has never received the OPEN CONFIRM from R1 or R1's OPEN message. R1 discards every repeating segment from R3 as bad.

```
IP: tableid=0, s=164.1.13.3 (Serial0/1), d=164.1.13.1 (Serial0/1),
routed via RIB
IP: s=164.1.13.3 (Serial0/1), d=164.1.13.1 (Serial0/1), len 85, rcvd 3
    TCP src=36826, dst=179, seq=1406017360, ack=1923456561, win=16384
ACK PSH

07CA3E50:          FF030021 45C00055 86FD0000      ...!E@.U.}..
07CA3E60: FE06D2DE A4010D03 A4010D01 8FDA00B3  ~.R^$...$....Z.3
07CA3E70: 53CE1F50 72A59E31 50184000 3CDB0000  SN.Pr%.1P.@.<[..
07CA3E80: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF  ................
07CA3E90: 002D0104 00C800B4 96010303 10020601  .-...H.4........
07CA3EA0: 04000100 01020280 00020202 00        .............


TCP0: bad seg from 164.1.13.3 -- outside window: port 179 seq
1406017360 ack 1923456561 rcvnxt 1406017405 rcvwnd 16339 len 45
164.1.13.1:179 <---> 164.1.13.3:36826   congestion window changes

IP: tableid=0, s=164.1.13.3 (Serial0/1), d=164.1.13.1 (Serial0/1),
routed via RIB

IP: s=164.1.13.3 (Serial0/1), d=164.1.13.1 (Serial0/1), len 106, rcvd 3
07CA7310:          FF030021 45C0006A 35B80000      ...!E@.j58..
07CA7320: FE06240F A4010D03 A4010D01 E63B00B3  ~.$.$...$...f;.3
07CA7330: 236C9312 A173EF88 50194000 09DB0000  #l..!so.P.@..[..
07CA7340: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF  ................
07CA7350: 002D0104 00C800B4 96010303 10020601  .-...H.4........
07CA7360: 04000100 01020280 00020202 00FFFFFF  ................
07CA7370: FFFFFFFF FFFFFFFF FFFFFFFF FF001503  ................
```

```
07CA7380: 0400                                ..

TCP0: bad seg from 164.1.13.3 -- outside window: port 179 seq 594318098
ack 2708729736 rcvnxt 594318165 rcvwnd 16318 len 66
TCP0: Data repacketized, seq 2708729736, sent 85 byte
TCP0: timeout #8 - timeout is 58784 ms, seq 2708729736
TCP: (179) -> 164.1.13.3(58939)

IP: tableid=0, s=164.1.13.3 (Serial0/1), d=164.1.13.1 (Serial0/1),
routed via RIB
IP: s=164.1.13.3 (Serial0/1), d=164.1.13.1 (Serial0/1), len 85, rcvd 3

07CA3090:          FF030021 45C00055 3EE10000      ...!E@.U>a..
07CA30A0: FE061AFB A4010D03 A4010D01 723400B3  ~..{$...$...r4.3
07CA30B0: 6D51674C 5A8405E0 50184000 A9740000  mQgLZ..`P.@.)t..
07CA30C0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF  ................
07CA30D0: 002D0104 00C800B4 96010303 10020601  .-...H.4........
07CA30E0: 04000100 01020280 00020202 00        .............

TCP0: bad seg from 164.1.13.3 -- outside window: port 179 seq
1834051404 ack 1518601696 rcvnxt 1834051449 rcvwnd 16339 len 45
TCP0: timeout #4 - timeout is 29392 ms, seq 1518601696
TCP: (179) -> 164.1.13.3(29236)
```

This does not clear much, but what conclusions could be made from the above output?

- Our TCP connection active open attempts fail – the other side never responds to our TCP SYNs.
- Passive open succeeds but all data exchange is being rejected by peer after this.
- Yet the Telnet TCP connection to port 179 works well.

It is apparent that something is wrong with our configuration, but it's not clear what. Even the heavy debugging analysis does not help much. Let's try to step back and use the information provided in the task. Compare the **`show ip bgp neighbor`** output on R1 with the output provided in the scenario.

Note

From the first part of the output, we may see that connection is stuck in OpenSent state. That means the other side considers transport connection open, but never seen an open message from the other side. Hold time and the keepalive interval values are set at their defaults. Advertisement interval is 30 seconds though that does not matter much here.

```
Rack1R3#show ip bgp neighbors 164.1.13.1
BGP neighbor is 164.1.13.1,  remote AS 300, external link
  BGP version 4, remote router ID 0.0.0.0
  BGP state = OpenSent
  Last read 00:01:34, last write 00:01:34, hold time is 180, keepalive
interval is 60 seconds
  Message statistics:
    InQ depth is 0
    OutQ depth is 0
                         Sent       Rcvd
    Opens:                 26          1
    Notifications:         23          0
    Updates:                0          0
    Keepalives:           157        156
    Route Refresh:          0          0
    Total:                206        157
  Default minimum time between advertisement runs is 30 seconds
<snip>
```

Note

Here is the part relevant to the transport connection. A connection has been established but later dropped. The other peer may be 1 hop away. The transport connection status is ESTAB(lished). Minimum incoming TTL is 254 and outgoing TTL is 255. Local host and remote host are taken from the link addressing.

```
  Connections established 1; dropped 1
  Last reset 01:25:07, due to BGP Notification sent, hold time expired
  External BGP neighbor may be up to 1 hop away.
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Connection is ECN Disabled, Mininum incoming TTL 254, Outgoing TTL 255
Local host: 164.1.13.3, Local port: 44156
Foreign host: 164.1.13.1, Foreign port: 179
```

Let's hold for a moment. As mentioned previous, BGP by default uses TTL of 1 when establishing eBGP connection. However, the output says that remote party expects TTL of 254 and sets outgoing TTL to 255. Let's check the same output at R3's side:

```
Rack1R1#show ip bgp neighbors 164.1.13.3
BGP neighbor is 164.1.13.3,  remote AS 200, external link
  BGP version 4, remote router ID 150.1.3.3
  BGP state = OpenConfirm
  Last read 00:00:39, last write 00:00:39, hold time is 180, keepalive
interval is 60 seconds
  Message statistics:
    InQ depth is 0
    OutQ depth is 0
                         Sent       Rcvd
    Opens:                  3          3
    Notifications:          1          0
    Updates:                0          0
    Keepalives:            30         28
    Route Refresh:          0          0
    Total:                 34         31
  Default minimum time between advertisement runs is 30 seconds

<snip>

  Connections established 1; dropped 1
  Last reset 00:05:34, due to BGP Notification sent, hold time expired
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Connection is ECN Disabled, Mininum incoming TTL 0, Outgoing TTL 1
Local host: 164.1.13.1, Local port: 179
Foreign host: 164.1.13.3, Foreign port: 12943
```

Now we see were the settings mismatch. We should have probably used the information in the scenario first, before spending so much time on heavy debugging! Now it becomes apparent that the security feature R3 is using is General TTL Security Mechanism. The reason why the initial connection from R3 to R1 succeeds is because R1 responds to the initial handshake using the SAME TTL it found in the initial SYN packet. However, you may recall the following output in the TCP debugs

```
TCB8446B3E0 setting property TCP_IN_TTL (23) 844A33A0
TCB8446B3E0 setting property TCP_OUT_TTL (24) 844A33A0
TCB85684DD8 setting property TCP_OUT_TTL (24) 8446B2DA
```

That sets the outgoing TTL to 1 after the connection has been established. This is reason why R3 would reject OPEN confirmation sent from R1 after the connection establishment. Also, for the active opens, R1 uses TTL value of 1. This is why all active opens from R1 to R3 fails at all time. But why did Telnet session work normally? This is because by default outgoing TCP connections uses TTL of 255, which perfectly matched the settings at R3. So that could have been a clue… if we knew where to look at!

Conclusion: Before digging into heavy debugging, try paying more attention to the information provided in the scenario; even if the output is lengthy, pay due diligence and read through it.

Fix the Issue

R1:
```
router bgp 300
 neighbor 164.1.13.3 ttl-security hops 1
```

This command will enforce R3 to use TTL of 255 for outgoing packets an only accept the packets with TTL of 254 from the remote peer.

Verify

The connection is now in established state and everything looks alright.

```
Rack1R1#show ip bgp neighbors 164.1.13.3
BGP neighbor is 164.1.13.3,  remote AS 200, external link
  BGP version 4, remote router ID 150.1.3.3
  BGP state = Established, up for 00:00:23
  Last read 00:00:23, last write 00:00:23, hold time is 180, keepalive
interval is 60 seconds
  Neighbor capabilities:
    Route refresh: advertised and received(old & new)
    Address family IPv4 Unicast: advertised and received
  Message statistics:
    InQ depth is 0
    OutQ depth is 0
                         Sent       Rcvd
    Opens:                  5          5
    Notifications:          2          0
    Updates:                0          0
    Keepalives:            34         31
    Route Refresh:          0          0
    Total:                 41         36
  Default minimum time between advertisement runs is 30 seconds

 For address family: IPv4 Unicast
  BGP table version 1, neighbor version 1/0
 Output queue size : 0
  Index 2, Offset 0, Mask 0x4
  2 update-group member
                                 Sent       Rcvd
  Prefix activity:               ----       ----
    Prefixes Current:               0          0
    Prefixes Total:                 0          0
    Implicit Withdraw:              0          0
    Explicit Withdraw:              0          0
    Used as bestpath:             n/a          0
    Used as multipath:            n/a          0

                                   Outbound    Inbound
  Local Policy Denied Prefixes:    --------    -------
    Total:                                0          0
  Number of NLRIs in the update sent: max 0, min 0

  Connections established 2; dropped 1
  Last reset 00:12:06, due to BGP Notification sent, hold time expired
  External BGP neighbor may be up to 1 hop away.
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Connection is ECN Disabled, Mininum incoming TTL 254, Outgoing TTL 255
Local host: 164.1.13.1, Local port: 41279
Foreign host: 164.1.13.3, Foreign port: 179

<snip>
```

Ticket 4

Analyze the Symptoms

We start right away with some isolation – seeing if we can ping between the R5 and SW1. Apparently we can't:

```
Rack1R5#ping 164.1.47.7 timeout 1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 164.1.47.7, timeout is 1 seconds:
.....
Success rate is 0 percent (0/5)
```

Let's run traceroute and see where the communications break:

```
Rack1R5#traceroute 164.1.47.7

Type escape sequence to abort.
Tracing the route to 164.1.47.7

  1 164.1.35.3 32 msec 32 msec 28 msec
  2 164.1.34.4 60 msec 61 msec 56 msec
  3  *  *
```

This allows us to quickly limit the problem scope – it seems to involve just R4 and SW1. So our initial hypothesis – something's wrong with the path between R4 and SW1.

Isolate the Issue

Just to make sure, we ping between R4 and SW1. The ping fails, which confirms our initial hypothesis.

```
Rack1R4#ping 164.1.47.7 timeout 1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 164.1.47.7, timeout is 1 seconds:
.....
Success rate is 0 percent (0/5)
```

Obviously, OSPF adjacency is down as well.

```
Rack1R4#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address
Interface
150.1.3.3         0   FULL/  -        00:00:38    164.1.34.3
Serial0/0
150.1.5.5         0   FULL/  -           -        164.1.45.5
OSPF_VL0
150.1.5.5         0   FULL/  -        00:00:34    164.1.45.5
Serial0/1
```

Now we descend to the link level and check for physical or any Layer 2 issues. To do this, we check our Layer 2 diagram and see how R4 connects to SW1. Refer back to Fig 1 to see that it takes R4 to cross both SW2 and SW1. So we start by checking all segments on the path, using the command **`show cdp neighbors`**.

```
Rack1R4#show cdp neighbors fastEthernet 0/0
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater

Device ID        Local Intrfce     Holdtme    Capability  Platform
Port ID
Rack1SW2         Fas 0/0            151        R S I      WS-C3560- Fas
0/4

Rack1SW1#show cdp neighbors fastEthernet 0/14
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater, P -
Phone

Device ID        Local Intrfce     Holdtme    Capability  Platform
Port ID
Rack1SW2         Fas 0/14          141          R S I     WS-C3560- Fas
0/14
```

From the CDP output it's clear that R4 has no problems communicating with SW4, nor SW1 has any physical issues communicating to SW2. Now that we checked that physical connectivity is OK, we may want to move up to Layer 2 and check two other things at Ethernet level: VLANs and STP. Before we do that, let's make a quick shortcut and ensure there is no filtering configured in SW1, SW2 or R4:

```
Rack1SW1#show ip interface fastEthernet 0/14
FastEthernet0/14 is up, line protocol is up
  Inbound  access list is not set

Rack1SW1#show vlan filter vlan 41

Rack1SW1#

Rack1SW2#show ip interface fastEthernet 0/14
FastEthernet0/14 is up, line protocol is up
  Inbound  access list is not set

Rack1SW2#show ip interface fastEthernet 0/17
FastEthernet0/17 is up, line protocol is up
  Inbound  access list is not set
Rack1SW2#

Rack1SW2#show vlan filter vlan 41

Rack1SW2#
```

```
Rack1R4#show ip interface fastEthernet 0/0 | inc access
  Outgoing access list is not set
  Inbound  access list is not set
  IP access violation accounting is disabled
```

This is the standard check that may help you save time in many real-life situations. Since there is no filtering, let's check the VLAN status in the switches on the path:

```
Rack1SW1#show vlan id 41

VLAN Name                             Status    Ports
---- -------------------------------- --------- ----------------------
-
41   VLAN0041                         active    Fa0/14

VLAN Type  SAID       MTU   Parent RingNo BridgeNo Stp  BrdgMode Trans1
Trans2
---- ----- ---------- ----- ------ ------ -------- ---- -------- ------
-
41   enet  100041     1500  -      -      -        -    -        0
0

Remote SPAN VLAN
----------------
Disabled

Primary Secondary Type              Ports
------- --------- ----------------- ------------------------------------

Rack1SW2#show vlan id 41

VLAN Name                             Status    Ports
---- -------------------------------- --------- ----------------------
-
41   VLAN0041                         active    Fa0/4, Fa0/14, Fa0/17

VLAN Type  SAID       MTU   Parent RingNo BridgeNo Stp  BrdgMode Trans1
Trans2
---- ----- ---------- ----- ------ ------ -------- ---- -------- ------
-
41   enet  100041     1500  -      -      -        -    -        0
0

Remote SPAN VLAN
----------------
Disabled

Primary Secondary Type              Ports
------- --------- ----------------- ------------------------------------
```

The VLAN is active on the ports that we need. So there are no mis-configurations or missing information. Thus, the last check we may apply is look over the STP instance for VLAN41. Obviously, there are no loops in the topology, so there are no blocked ports. However, it never hurts to double-check the STP status. Keeping in mind that SW1 is the root we look at the STP status in SW2:

```
Rack1SW2#show spanning-tree vlan 41

VLAN0041
  Spanning tree enabled protocol ieee
  Root ID    Priority    4137
             Address     000f.f76d.ac80
             Cost        19
             Port        19 (FastEthernet0/17)
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    32809  (priority 32768 sys-id-ext 41)
             Address     001f.2711.d580
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time 300

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- -----------------------
-
Fa0/4               Desg FWD 19        128.6    P2p
Fa0/14              Desg FWD 19        128.16   P2p
Fa0/17              Root FWD 19        128.19   P2p
```

However, we se something wrong: the root port points toward SW3, not SW1! For some reason, SW1 is not the root bridge for this STP instance. However, this should not prevent communications under normal configuration. Let's get back to SW1 and see the situation there.

```
Rack1SW1#show spanning-tree vlan 41

VLAN0041
  Spanning tree enabled protocol ieee
  Root ID    Priority    8233
             Address     001f.6d94.7b80
             This bridge is the root
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    8233   (priority 8192 sys-id-ext 41)
             Address     001f.6d94.7b80
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time 300

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- -----------------------
-
Fa0/14              Desg BKN*19        128.16   P2p *ROOT_Inc
```

The link to SW2 is blocking on this VLAN saying the port is in "Root Inconsistent" state. This usually means the root guard is blocking this port. Check if this is true:

```
Rack1SW1#show spanning-tree vlan 41 interface fastEthernet 0/14 detail
 Port 16 (FastEthernet0/14) of VLAN0041 is broken  (Root Inconsistent)
```

```
   Port path cost 19, Port priority 128, Port Identifier 128.16.
   Designated root has priority 8233, address 001f.6d94.7b80
   Designated bridge has priority 8233, address 001f.6d94.7b80
   Designated port id is 128.16, designated path cost 0
   Timers: message age 3, forward delay 0, hold 0
   Number of transitions to forwarding state: 1
   Link type is point-to-point by default
   Root guard is enabled on the port
   BPDU: sent 20, received 871
```

So the issue is that some other bridge claiming itself as root for VLAN41. If you look at the following output once again

```
Rack1SW2#show spanning-tree vlan 41

VLAN0041
  Spanning tree enabled protocol ieee
  Root ID    Priority    4137
             Address     000f.f76d.ac80
             Cost        19
```

You will notice that the “wrong” root has the priority set to 4096+41, which means the root bridge priority is 4096. The additional 41 is due to the “extended system-ID” feature, which steals some bits from the priority field to make the bridge ID unique by concatenating it with the VLAN number.

Conclusion: When you have a problem that most likely relates to some L1/L2 misconfiguration, use the bottom-up approach. However, “shortcuts” never hurt, as they allow for catching some common issues quickly. And last but not least – even in loop-less topology STP could be causing issues!

Fix the Issue

Configure SW1 as the root bridge for VLAN 41 with a better priority – zero. This will keep the configuration aligned with the baseline. Alternatively, you may remove the root guard, but this will violate the security and the baseline requirement.

```
SW1:
spanning-tree vlan 41 priority 0
```

Verify

Check the OSPF neighbors on R4 and confirm that we have end-to-end connectivity now:

```
Rack1R4#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address
Interface
```

```
150.1.3.3         0   FULL/  -        00:00:39    164.1.34.3
Serial0/0
150.1.5.5         0   FULL/  -           -        164.1.45.5
OSPF_VL0
150.1.5.5         0   FULL/  -        00:00:35    164.1.45.5
Serial0/1
150.1.7.7         1   FULL/BDR        00:00:38    164.1.47.7
FastEthernet0/0

Rack1R4#ping 164.1.47.7

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 164.1.47.7, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
Rack1R4#

Rack1SW1#show spanning-tree vlan 41

VLAN0041
  Spanning tree enabled protocol ieee
  Root ID    Priority    41
             Address     001f.6d94.7b80
             This bridge is the root
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    41     (priority 0 sys-id-ext 41)
             Address     001f.6d94.7b80
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time 300

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- ----------------------
Fa0/14              Desg FWD 19        128.16   P2p
```

Ticket 5

Analyze the Symptoms

We have a stereotypic scenario here – something is broken, and we know the issue is localized to just one router. Based on the baseline information, we need to recover the router to the proper state. Since we have no idea of the level of damage applied, we just start bottom-up, checking the link status and per-link reachability. We'll be fixing issues as we progress through the analysis in cycles, so all troubleshooting phases will be effectively merged together.

Isolate the Issue

Fix the Issue

Verify

The first command we might like to run is the following:

```
Rack1R6#show ip interface brief
Interface                  IP-Address      OK? Method Status
Protocol
FastEthernet0/0            192.10.1.6      YES manual administratively
down down
Serial0/0                  54.1.2.6        YES manual up
up
FastEthernet0/1            164.1.26.6      YES manual up
up
Loopback0                  150.1.6.6       YES manual up
up
```

It allows us detecting any interfaces that are in down state. So far we already spotted that R6's connection to BB2 is shut down. The IP addressing on the interfaces matches the diagram. So we fix the first issue:

```
R6:
interface FastEthernet 0/0
 no shutdown
```

OK, now let's run a quick connectivity test on all links, pinging our directly connected neighbors.

```
Rack1R6#ping 192.10.1.254

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.10.1.254, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/5/8 ms

Rack1R6#ping 54.1.2.6

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 54.1.2.6, timeout is 2 seconds:
```

```
.....
Success rate is 0 percent (0/5)

Rack1R6#ping 164.1.26.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 164.1.26.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

Everything looks fine with except to R6, which is connected via the Frame-Relay link. We focus on this connection:

```
Rack1R6#show interfaces serial 0/0
Serial0/0 is up, line protocol is up
  Hardware is PowerQUICC Serial
  Internet address is 54.1.2.6/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation FRAME-RELAY, loopback not set
  Keepalive set (10 sec)
  LMI enq sent  100, LMI stat recvd 100, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent  0, LMI upd sent  0
  LMI DLCI 1023  LMI type is CISCO  frame relay DTE
  FR SVC disabled, LAPF state down
  Broadcast queue 0/64, broadcasts sent/dropped 234/0, interface
broadcasts 39
  Last input 00:00:00, output 00:00:03, output hang never
  Last clearing of "show interface" counters 00:16:45
  Input queue: 2/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
     Conversations  0/1/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec
  5 minute input rate 1000 bits/sec, 2 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     1685 packets input, 142603 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     334 packets output, 16204 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
     0 output buffer failures, 0 output buffers swapped out
     2 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

As the protocol is up, and LMI exchange occurs (based on the LMI couters), we must assume the protocol is healthy and we receive the DLCIs mapped to the interface. The next thing we should check for Frame-Relay are the FR mappings:

```
Rack1R6#show frame-relay map
Serial0/0 (up): ip 0.0.0.0 dlci 401(0x191,0x6410)
              broadcast,
              CISCO, status defined, active
Serial0/0 (up): ip 0.0.0.0 dlci 301(0x12D,0x48D0)
```

```
              broadcast,
              CISCO, status defined, active
Serial0/0 (up): ip 0.0.0.0 dlci 201(0xC9,0x3090)
              broadcast,
              CISCO, status defined, active
Serial0/0 (up): ip 0.0.0.0 dlci 101(0x65,0x1850)
              broadcast,
              CISCO, status defined, active
Serial0/0 (up): ip 0.0.0.0 dlci 51(0x33,0xC30)
              broadcast,
              CISCO, status defined, active
Serial0/0 (up): ip 54.1.2.255 dlci 100(0x64,0x1840), static,
              broadcast,
              CISCO, status defined, active
```

A bunch of mappings to self, which is not good, but should not affect overall connectivity as the 0.0.0.0 addresses are mapped to the unused DLCIs. We are particularly interested in the last static mapping. It's easy to notice that an incorrect IP is being mapped to DLCI 100. It should be 54.X.2.254 not .255 for the last octet. Based on this, we fix this issue:

```
R6:
interface Serial 0/0
 no frame-relay map ip 54.1.2.255 100
 frame-relay map ip 54.1.2.254 100 broadcast
```

Now everything works and we can ping BB1:

```
Rack1R6#ping 54.1.2.254

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 54.1.2.254, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 32/35/41 ms
```

It appears we're done with the per-link connectivity. We now can go ahead and check our IGPs. We start with the RIP, as the most simple protocol.

```
Rack1R6#show ip route rip

Rack1R6#
```

This means we don't receive anything via RIP. What we could do now, is check if there are any RIP updates getting to the router. We enable debug output to the console, as it may be disabled by default.

```
Rack1R6#debug ip rip
RIP protocol debugging is on

Rack1R6#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1R6(config)#logging console debugging

RIP: ignored v2 packet from 54.1.3.254 (illegal version)
```

```
RIP: ignored v2 packet from 54.1.2.254 (illegal version)
RIP: ignored v2 packet from 54.1.10.254 (illegal version)
RIP: ignored v2 packet from 54.1.1.254 (illegal version)
```

We quickly see the issue – the other party sends us RIPv2 packets but the local router appears to be configured for version 1. Let's check this hypothesis:

```
Rack1R6#show ip protocols | beg \"rip\"
Routing Protocol is "rip"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Sending updates every 30 seconds, next due in 13 seconds
  Invalid after 180 seconds, hold down 180, flushed after 240
  Redistributing: eigrp 100, rip
  Default version control: send version 1, receive any version
    Interface             Send  Recv  Triggered RIP  Key-chain
    Serial0/0             1     1
  Automatic network summarization is not in effect
  Maximum path: 4
  Routing for Networks:
    54.0.0.0
  Routing Information Sources:
    Gateway         Distance      Last Update
    54.1.2.254           120      00:04:41
  Distance: (default is 120)
```

And indeed, the RIP process is set to send and receive only version 1 on the Frame-Relay interface. We go ahead and configure RIP for version 2:

```
R6:
router rip
 version 2
```

But for some reason, we're still getting the same debugging output complaining on the old version.

```
RIP: ignored v2 packet from 54.1.10.254 (illegal version)
RIP: ignored v2 packet from 54.1.3.254 (illegal version)
RIP: ignored v2 packet from 54.1.2.254 (illegal version)
RIP: ignored v2 packet from 54.1.1.254 (illegal version)
```

We check RIP configuration once again and see the following:

```
Rack1R6#show ip protocols | beg \"rip\"
Routing Protocol is "rip"
<snip>
  Default version control: send version 2, receive version 2
    Interface             Send  Recv  Triggered RIP  Key-chain
    Serial0/0             2     1
  Automatic network summarization is not in effect
  Maximum path: 4
  Routing for Networks:
    54.0.0.0
  Routing Information Sources:
    Gateway         Distance      Last Update
    54.1.2.254           120      00:05:36
  Distance: (default is 120)
```

We got stuck in the RIP feature that allows an intermix of versions used on different interfaces. We need to configure the RIP on the particular interface to receive version 2 as well:

```
R6:
interface Serial 0/0
 ip rip receive version 2
```

This fixes the issue as now we can see the RIP routes from BB1:

```
Rack1R6#show ip route rip
R    212.18.1.0/24 [120/1] via 54.1.2.254, 00:00:11, Serial0/0
R    212.18.0.0/24 [120/1] via 54.1.2.254, 00:00:11, Serial0/0
R    212.18.3.0/24 [120/1] via 54.1.2.254, 00:00:11, Serial0/0
R    212.18.2.0/24 [120/1] via 54.1.2.254, 00:00:11, Serial0/0
```

During the troubleshooting you may have noticed the following output on your console:

```
%TCP-6-BADAUTH: Invalid MD5 digest from 192.10.1.254(36413) to
192.10.1.6(179)
%TCP-6-BADAUTH: Invalid MD5 digest from 192.10.1.254(36413) to
192.10.1.6(179)
```

This output clearly states that we’re having an issue with the authentication of our BGP session to BB2. As we know the correct password should be “CISCO” we fork from our main troubleshooting process and fix this issue as well:

```
R6:
router bgp 200
 neighbor 192.10.1.254 password CISCO
```

And that fixes this particular BGP session problem – the session is not in “Active” state.

```
Rack1R6#show ip bgp summary | inc 192.10.1.254|Neigh
Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
192.10.1.254    4   254       5       4        4    0    0 00:00:18
3
```

So now we’re back to checking our IGPs. The next protocol is EIGRP. We start by checking EGRIP adjacencies:

```
Rack1R6#show ip eigrp neighbors
IP-EIGRP neighbors for process 100
```

Something is wrong. We check the global protocol settings to see what this might be:

```
Rack1R6#show ip protocols
Routing Protocol is "eigrp 100"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Default networks flagged in outgoing updates
  Default networks accepted from incoming updates
  EIGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
  EIGRP maximum hopcount 100
  EIGRP maximum metric variance 1
  Redistributing: eigrp 100, rip
  EIGRP NSF-aware route hold timer is 240s
  Automatic network summarization is not in effect
  Maximum path: 4
  Routing for Networks:
    150.1.6.6/32
    164.1.26.6/32
  Passive Interface(s):
    FastEthernet0/0
    Serial0/0
    FastEthernet0/1
    Loopback0
    VoIP-Null0
  Routing Information Sources:
    Gateway         Distance      Last Update
  Distance: internal 90 external 170
```

We see all interfaces configured as “Passive”! That gives us a clue what to do – enable EIGRP on the interface connecting R6 to R2:

```
R6:
router eigrp 100
 no passive-interface FastEthernet 0/1
```

And this brings back the neighbor and populates our database with the EIGRP routes:

```
Rack1R6#show ip eigrp neighbors
IP-EIGRP neighbors for process 100
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q
Seq
                                            (sec)         (ms)
Cnt Num
0   164.1.26.2              Fa0/1             14 00:00:28   13   300  0
69
Rack1R6#

Rack1R6#show ip route eigrp
D EX 119.0.0.0/8 [170/2430976] via 164.1.26.2, 00:00:31,
FastEthernet0/1
D EX 118.0.0.0/8 [170/2430976] via 164.1.26.2, 00:00:31,
FastEthernet0/1
D EX 117.0.0.0/8 [170/2430976] via 164.1.26.2, 00:00:31,
FastEthernet0/1
```

```
D EX 116.0.0.0/8 [170/2430976] via 164.1.26.2, 00:00:31,
FastEthernet0/1
D EX 115.0.0.0/8 [170/2430976] via 164.1.26.2, 00:00:31,
FastEthernet0/1
D EX 114.0.0.0/8 [170/2430976] via 164.1.26.2, 00:00:31,
FastEthernet0/1
D EX 113.0.0.0/8 [170/2430976] via 164.1.26.2, 00:00:31,
FastEthernet0/1
D EX 112.0.0.0/8 [170/2430976] via 164.1.26.2, 00:00:31,
FastEthernet0/1
<snip>
```

Now we can check if the EIGRP router sees the RIP prefixes:

```
Rack1R1#show ip route eigrp | inc 212.18
Rack1R1#
```

It does not. Maybe R1 learns it from another protocol with a better AD? No.

```
Rack1R1#show ip route 212.18.1.0
% Network not in table
```

Make sure no filtering is applied to the routing process:

```
Rack1R1#show ip protocols
Routing Protocol is "eigrp 100"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Default networks flagged in outgoing updates
  Default networks accepted from incoming updates
  EIGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
  EIGRP maximum hopcount 100
  EIGRP maximum metric variance 5
  Redistributing: eigrp 100
  EIGRP NSF-aware route hold timer is 240s
  Automatic network summarization is not in effect
  Maximum path: 4
  Routing for Networks:
    150.1.1.1/32
    164.1.12.1/32
    164.1.13.1/32
    164.1.18.1/32
  Routing Information Sources:
    Gateway         Distance      Last Update
    164.1.13.3            90      00:00:03
    164.1.12.2            90      00:00:03
    164.1.18.8            90      00:00:03
  Distance: internal 90 external 170
```

Let’s check and see if those routes appear in the topology table of the source router – R6, i.e. if they are being redistributed into EIGRP from RIP:

```
Rack1R6#show ip eigrp topology 212.18.1.0 255.255.255.0
IP-EIGRP (AS 100): Topology entry for 212.18.1.0/24
```

```
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is
2560000256
  Routing Descriptor Blocks:
  54.1.2.254, from Redistributed, Send flag is 0x0
      Composite metric is (2560000256/0), Route is External
      Vector metric:
        Minimum bandwidth is 1 Kbit
        Total delay is 10 microseconds
        Reliability is 1/255
        Load is 1/255
        Minimum MTU is 1
        Hop count is 0
      External data:
        Originating router is 150.1.1.1 (this system)
        AS number of route is 0
        External protocol is RIP, external metric is 1
        Administrator tag is 0 (0x00000000)
```

That means R6 actually redistributes the routes. We cannot access R2 or R3, but we can use a quick “cheat” trick: Create a static route for missing prefixes via R2 and see if we they actually route to it:

```
R1:
ip route 212.18.1.0 255.255.255.0 150.1.2.2

Rack1R1#traceroute 212.18.1.1

Type escape sequence to abort.
Tracing the route to 212.18.1.1

  1 164.1.13.3 16 msec 16 msec 16 msec
  2 164.1.23.2 40 msec 40 msec 44 msec
  3 54.1.2.254 56 msec
    164.1.26.6 44 msec *
```

The traceroute reaches R6, meaning that the border router has actually passed them to R3/R2 and they are unlikely to filter it (otherwise the scenario would make no sense). Don’t forget to remove the static route after this test!

The last thing we could do before getting totally lost is seeing if R1 actually receives these updates. You may use the noisy command debug ip eigrp 100 to see EIGRP route related events. After configuring this command, clear one of the EIGRP neighbors.

```
Rack1R1#debug ip eigrp 100
IP-EIGRP Route Events debugging is on
Rack1R1#clear ip eigrp neighbor 164.1.12.2
```

Among the massive amount of debugging output you may notice the following:

```
IP-EIGRP(Default-IP-Routing-Table:100): ExtS 212.18.0.0/24 M 2560514816
- 2560000000 514816 SM 2560002816 - 2560000000 2816
```

```
IP-EIGRP(Default-IP-Routing-Table:100): ExtS 212.18.3.0/24 M 2560514816
- 2560000000 514816 SM 2560002816 - 2560000000 2816
IP-EIGRP(Default-IP-Routing-Table:100): ExtS 212.18.2.0/24 M 2560514816
- 2560000000 514816 SM 2560002816 - 2560000000 2816
IP-EIGRP(Default-IP-Routing-Table:100): ExtS 212.18.1.0/24 M 2560514816
- 2560000000 514816 SM 2560002816 - 2560000000 2816
```

Which means R1 receives these prefixes via the UPDATE packet. So the issue is somehow linked to R1's configuration. Since there is no filtering involved, we may run through the checklist for EIGRP prefixes not being installed in the routing table:

- Administrative Distance
- Distribute-List filtering
- The same router ID filtering External Routes

The last thing sounds interesting, as our routes are external. We go ahead and check the EIGRP router ID of R6:

```
Rack1R6#show ip eigrp topology
IP-EIGRP Topology Table for AS(100)/ID(150.1.1.1)
```

We hit the spot, they match, and this prevents R1 from installing the external prefixes. This is an additional loop-prevention mechanism in EIGRP to make sure external routes do not enter the domain twice. Apply the fix:

```
R6:
router eigrp 100
 eigrp router-id 150.1.6.6

And now we see all external routes in R1's table:

Rack1R1#show ip route eigrp | inc 212.18
D EX 212.18.1.0/24 [170/2560514816] via 164.1.12.2, 00:00:07, Serial0/0
D EX 212.18.0.0/24 [170/2560514816] via 164.1.12.2, 00:00:07, Serial0/0
D EX 212.18.3.0/24 [170/2560514816] via 164.1.12.2, 00:00:07, Serial0/0
D EX 212.18.2.0/24 [170/2560514816] via 164.1.12.2, 00:00:07, Serial0/0
```

The last thing left to do is check the BGP peering session health. We already fixed the peering to BB2. We check all session's status at once now:

```
Rack1R6#show ip bgp summary |  beg Neig
Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
164.1.26.2      4   200       8       6       24    0    0 00:01:11
10
192.10.1.254    4   254      51      52       24    0    0 00:46:47
3
```

So it appears we're good. Let's just trace from R1 to the prefixes learned from BB1 in order to make sure we have end-to-end connectivity:

```
Rack1R1#traceroute 212.18.1.1

Type escape sequence to abort.
Tracing the route to 212.18.1.1

  1 164.1.12.2 29 msec 28 msec 28 msec
  2 164.1.26.6 28 msec 28 msec 28 msec
  3 54.1.2.254 505 msec *  44 msec
```

Ticket 6

Analyze the Symptoms

Initial guesses may be as follows: Something's wrong with AS 100 configuration – e.g. no peering session with AS 200 or filtering configured. We gather more symptoms by checking if AS 100 advertises any routes to AS 200.

```
Rack1R4#show ip bgp summary | beg Neigh
Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
150.1.3.3       4   200     230     233       14    0    0 03:45:16
3
163.1.13.1      4   300       0       0        0    0    0 never
Idle
204.12.1.254    4    54     121     121       14    0    0 01:52:10
10

Rack1R4#show ip bgp neighbors 150.1.3.3 advertised-routes
BGP table version is 14, local router ID is 150.1.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 28.119.16.0/24   204.12.1.254             0             0 54 i
*> 28.119.17.0/24   204.12.1.254             0             0 54 i
*> 112.0.0.0        204.12.1.254                           0 54 50 60 i
*> 113.0.0.0        204.12.1.254                           0 54 50 60 i
*> 114.0.0.0        204.12.1.254                           0 54 i
*> 115.0.0.0        204.12.1.254                           0 54 i
*> 116.0.0.0        204.12.1.254                           0 54 i
*> 117.0.0.0        204.12.1.254                           0 54 i
*> 118.0.0.0        204.12.1.254                           0 54 i
*> 119.0.0.0        204.12.1.254                           0 54 i
*> 205.90.31.0      150.1.3.3                              0 200 254 ?
*> 220.20.3.0       150.1.3.3                              0 200 254 ?
*> 222.22.2.0       150.1.3.3                              0 200 254 ?

Total number of prefixes 13
```

Good, now let's see if R6 receives these prefixes:

```
Rack1R6#show ip bgp neighbors 164.1.26.2 routes

Total number of prefixes 0
```

Nope, it does not. So there is something in between the routers of AS 200 that we cannot access and R6. Our problematic area is narrowed down to R2, R3 and R6.

Hypothesis: Misconfiguration between R6, R2 and R3
Problem Scope: R2, R3 and R6. Note that R2 should reflect routes from R3 to R6.

Isolate the Issue

First we check if there is any filtering applied at R6 to the prefixes received from R2:

```
Rack1R6#show ip protocols | beg bgp
Routing Protocol is "bgp 200"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  IGP synchronization is disabled
  Automatic route summarization is disabled
  Unicast Aggregate Generation:
    164.1.0.0/16       summary-only
  Neighbor(s):
    Address          FiltIn FiltOut DistIn DistOut Weight RouteMap
    164.1.26.2
    192.10.1.254
  Maximum path: 1
  Routing Information Sources:
    Gateway         Distance      Last Update
    192.10.1.254          20      01:31:52
    164.1.26.2           200      00:13:03
  Distance: external 20 internal 200 local 200
```

Let’s check the per-neighbor statistics. There is an important part of it that shows the amount of prefixes being exchange and the results of the local policy may have affected incoming routes.

```
Rack1R6#show ip bgp neighbors 164.1.26.2
BGP neighbor is 164.1.26.2,  remote AS 200, internal link
<snip>

 For address family: IPv4 Unicast
  BGP table version 94, neighbor version 94/0
 Output queue size : 0
  Index 2, Offset 0, Mask 0x4
  2 update-group member
  NEXT_HOP is always this router
                                 Sent       Rcvd
  Prefix activity:               ----       ----
    Prefixes Current:               3          0
    Prefixes Total:                12         40
    Implicit Withdraw:              9         20
    Explicit Withdraw:              0         20
    Used as bestpath:             n/a          0
    Used as multipath:            n/a          0

                                   Outbound    Inbound
  Local Policy Denied Prefixes:    --------    -------
    Suppressed duplicate:                 0         20
    CLUSTER_LIST loop:                    n/a         39
    ORIGINATOR loop:                      n/a         12
    Bestpath from this peer:               60        n/a
    Total:                                 60         71
  Number of NLRIs in the update sent: max 3, min 3

  Connections established 1; dropped 0
  Last reset never
<snip>
```

We see the total of 40 prefixes have been received. At the same time, the policy deny statistics shows that a bunch of then have been dropped due to CLUSTER_LIST and ORIGINATOR loop. This means that the local router received paths that have the local BGP router ID in the CLUSTER_LIST attribute or in the ORIGINATOR attribute. These attributes are used for loop prevention in Route-Reflector hierarchies. So our hypothesis is: Someone in AS 200 uses the same BGP Router ID as we do.

We could confirm our hypothesis using the following debug of the incoming BGP updates:

```
Rack1R6#debug ip bgp 164.1.26.2 updates
BGP updates debugging is on for neighbor 164.1.26.2 for address family:
IPv4 Unicast

Rack1R6#clear ip bgp 164.1.26.2 soft in
Rack1R6#
BGP: 164.1.26.2 RR in same cluster. Reflected update dropped
BGP(0): 164.1.26.2 rcv UPDATE w/ attr: nexthop 150.1.4.4, origin i,
localpref 100, metric 0, originator 150.1.3.3, clusterlist 150.1.6.6,
path 100 54, community , extended community

BGP(0): 164.1.26.2 rcv UPDATE about 28.119.17.0/24 -- DENIED due to:
reflected from the same cluster;
BGP(0): 164.1.26.2 rcv UPDATE about 28.119.16.0/24 -- DENIED due to:
reflected from the same cluster;
BGP(0): 164.1.26.2 rcv UPDATE about 119.0.0.0/8 -- DENIED due to:
reflected from the same cluster;
BGP(0): 164.1.26.2 rcv UPDATE about 118.0.0.0/8 -- DENIED due to:
reflected from the same cluster;
<snip>
```

From this output we can also see that R2 uses the cluster-ID of 150.1.6.6 which happens to be the local router ID. And by default, the local cluster-ID is built from the router-ID So now we know how to fix the problem.

Conclusion: Typos do happen when people configure router-IDs statically (which is not a bad idea!). In some cases, you may deliberately set the same Cluster-ID for RRs in the same cluster. However, watch out for setting BGP cluster-ID in non RRs!

Also, note how the idea of detecting the “duplicates” is used to locate potential loops in many protocols. We’ve just seen the same idea used with EIGRP. In BGP, it’s used even more frequently, if you recall the checks for AS_PATH loops.

Fix the Issue

Configure R6 to use a Cluster-ID different from its router-ID:

```
R6:
router bgp 200
 bgp cluster-id 150.1.66.66
```

Verify

Check that AS 254 have the routes from AS 54 advertised by R6:

```
Rack1R6#show ip bgp neighbors 192.10.1.254 advertised-routes
BGP table version is 114, local router ID is 150.1.6.6
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
r>i28.119.16.0/24   150.1.4.4                0    100      0 100 54 i
r>i28.119.17.0/24   150.1.4.4                0    100      0 100 54 i
r>i112.0.0.0        150.1.4.4                0    100      0 100 54 50
60 i
r>i113.0.0.0        150.1.4.4                0    100      0 100 54 50
60 i
r>i114.0.0.0        150.1.4.4                0    100      0 100 54 i
r>i115.0.0.0        150.1.4.4                0    100      0 100 54 i
r>i116.0.0.0        150.1.4.4                0    100      0 100 54 i
r>i117.0.0.0        150.1.4.4                0    100      0 100 54 i
r>i118.0.0.0        150.1.4.4                0    100      0 100 54 i
r>i119.0.0.0        150.1.4.4                0    100      0 100 54 i

Total number of prefixes 10
```

Ticket 7

Analyze the Symptoms

We don't have enough information to analyze yet. Maybe if we could simulate a DHCP client we could get some more? Let's configure SW2 with an SVI interface for VLAN55 requesting a DHCP IP address.

Isolate the Issue

Enable the following DHCP server debugging in R5 (you may collect the debugging output in the logging buffer)

```
Rack1R5#debug ip dhcp server packet
Rack1R5#debug ip dhcp server events
```

And now configure SW2 as a DHCP client.

```
SW2:
interface Vlan 55
 ip address dhcp
```

Now look at the log messages collected:

```
DHCPD: inconsistent relay information.
DHCPD: relay information option exists, but giaddr is zero.
DHCPD: inconsistent relay information.
DHCPD: relay information option exists, but giaddr is zero.
DHCPD: inconsistent relay information.
DHCPD: relay information option exists, but giaddr is zero
```

This means R5 receives the DHCP packets. However, the "giaddr" field is present in the packets and has the value of all zeros "0.0.0.0". This field carries the information about the DHCP relay that forwarded the packet and should be non-zero per RFC. As a security pre-caution, IOS routers drop such DHCP packets by default. Based on the fact that everything was working before, we may conclude that some DHCP related feature in SW3 modifies the DHCP packets. The obvious answer is that this is DHCP Snooping feature, which is known to insert the information option and the giaddr field of zero, as the switch does not actually "relay" the DHCP packet. Since we cannot modify the switch configuration, we have to go ahead to configure the router to accept such packets.

Conclusion: Some Cisco features are just not designed to work with each other smoothly!

Fix the Issue

The following command instructs R5's DHCP server to accept packets even with the giaddr field of zero.

R5:

```
ip dhcp relay information trust-all
```

Verify

Make sure SW2 has configured the IP address dynamically:

```
Rack1SW2#show ip interface vlan 55
Vlan55 is up, line protocol is up
  Internet address is 164.1.55.1/24
  Broadcast address is 255.255.255.255
  Address determined by DHCP
```

And confirm this address was allocated by R5:

```
Rack1R5#show ip dhcp binding
Bindings from all pools not associated with VRF:
IP address          Client-ID/              Lease expiration
Type
                    Hardware address/
                    User name
164.1.55.1          0063.6973.636f.2d30.    Jul 09 2009 07:39 PM
Automatic
                    3031.662e.3237.3131.
                    2e64.3563.352d.566c.
                    3535
```

Ticket 8

Analyze the Symptoms

Start by gathering more information on the problem. Check the BGP sessions status:

```
Rack1SW2#show ip bgp summary
BGP router identifier 150.1.8.8, local AS number 300
BGP table version is 3, main routing table version 3

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
164.1.18.1      4   300      12      13        0    0    0 00:02:54
Active
```

Get more detailed information:

```
Rack1SW2#show ip bgp neighbors 164.1.18.1
BGP neighbor is 164.1.18.1,  remote AS 300, internal link
  BGP version 4, remote router ID 0.0.0.0
  BGP state = Active
<snip>
  Connections established 1; dropped 1
  Last reset 00:05:00, due to BGP Notification sent, hold time expired
  Transport(tcp) path-mtu-discovery is enabled
  No active TCP connection
```

So that means SW2 cannot establish the TCP connection; Let’s check the link status:

```
Rack1SW2#ping 164.1.18.1 timeout 1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 164.1.18.1, timeout is 1 seconds:
.....
Success rate is 0 percent (0/5)
```

OK, so we suppose there is some link problem. This is our working hypothesis, and the problematic area is limited to SW2, SW1 and R1 – per L2 connectivity diagram you may see that SW1 connects SW2 and R1.

Isolate the Issue

Start at the physical layer. Check the links status of SW2 and R1:

```
Rack1SW2#show interfaces fastEthernet 0/15
FastEthernet0/15 is up, line protocol is up (connected)
  Hardware is Fast Ethernet, address is 001f.2711.d5c2 (bia
001f.2711.d5c2)
  Internet address is 164.1.18.8/24

Rack1R1#show interfaces fastEthernet 0/0
```

```
FastEthernet0/0 is up, line protocol is down
  Hardware is AmdFE, address is 000f.23d5.5220 (bia 000f.23d5.5220)
  Internet address is 164.1.18.1/24
  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
```

It appears the link between R1 and SW1 is down. Since the link is not in admininistratively down state, that means something is misconfigured in SW1. Let's go there and check the switchport of R1:

```
Rack1SW1#show interfaces fastEthernet 0/1
FastEthernet0/1 is down, line protocol is down (err-disabled)
  Hardware is Fast Ethernet, address is 001f.6d94.7b83 (bia
001f.6d94.7b83)
```

The port is being shut down due to an error-disable problem. There could be a few potential problems to cause this, and you need to check the log messages to see the real cause of the problem.

```
Rack1SW1#show logging
Syslog logging: enabled (0 messages dropped, 0 messages rate-limited, 0
flushes, 0 overruns, xml disabled, filtering disabled)

No Active Message Discriminator.



No Inactive Message Discriminator.


    Console logging: disabled
    Monitor logging: level debugging, 0 messages logged, xml disabled,
                     filtering disabled
    Buffer logging:  disabled, xml disabled,
                    filtering disabled
    Exception Logging: size (4096 bytes)
    Count and timestamp logging messages: disabled
    File logging: disabled
    Persistent logging: disabled
    Trap logging: level informational, 206 message lines logged
```

It appears that logging to the buffer and console has been disabled! Let's enable it back and try replicating the issue.

```
Rack1SW1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1SW1(config)#logging console debugging
Rack1SW1(config)#int fa 0/1
Rack1SW1(config-if)#shutdown
Rack1SW1(config-if)#no shutdown
Rack1SW1(config-if)#
%LINK-3-UPDOWN: Interface FastEthernet0/1, changed state to up
```

```
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1,
changed state to up

%PM-4-ERR_DISABLE: psecure-violation error detected on Fa0/1, putting
Fa0/1 in err-disable state
Rack1SW1(config)#
%PORT_SECURITY-2-PSECURE_VIOLATION: Security violation occurred, caused
by MAC address 0000.0c07.ac01 on port FastEthernet0/1.

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1,
changed state to down
```

OK so we can see that the port is being shut down due to port-security violation. We check the port-security configuration on the port we're dealing with:

```
Rack1SW1#show port-security interface fastEthernet 0/1
Port Security              : Enabled
Port Status                : Secure-shutdown
Violation Mode             : Shutdown
Aging Time                 : 0 mins
Aging Type                 : Absolute
SecureStatic Address Aging : Disabled
Maximum MAC Addresses      : 1
Total MAC Addresses        : 0
Configured MAC Addresses   : 0
Sticky MAC Addresses       : 0
Last Source Address:Vlan   : 0000.0c07.ac01:18
Security Violation Count   : 1
```

The limit of the MACs is set to one – plus, we know the offending MAC address 0000.0c07.AC01. Normally, a router uses just one MAC address on a port. If you remember the technology basics well, the offending MAC address is used by HSRP. Thus, the security violation may be caused by R1 running HSRP and sending packets using the virtual MAC for standby group 1.

Conclusion: Sometimes irrelevant configuration may cause cascading effect and break a lot of things!

Fix the Issue

Generally, this problem has two solutions: configuring the switches to allow more MAC addresses, or configuring the routers to use the hardware (BIA) address for HSRP packets. Since we are not allowed to modify the switches configuration for this task, we re-configure R1:

```
R1:
interface FastEthernet 0/0
 standby use-bia
```

Notice that the above command is not supported for HSRP implementation in the Catalyst switches. After the fix, you need to shutdown and un-shutdown the switchport to remove the error-disable state:

```
SW1:
Interface FastEthernet 0/1
 shutdown
 no shutdown
```

Remember that the option of using BIA has disadvantages. For example, if you're changing the active forwarder, it needs to send a gratituous ARP packet to update all nodes ARP caches. Sometimes this may not work, and the hosts may attempt using the old MAC address.

Verify

Confirm that connectivity has been restored:

```
Rack1SW2#show ip bgp summary
BGP router identifier 150.1.8.8, local AS number 300
BGP table version is 4, main routing table version 4
1 network entries using 117 bytes of memory
1 path entries using 52 bytes of memory
2/1 BGP path/bestpath attribute entries using 280 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 449 total bytes of memory
BGP activity 2/1 prefixes, 2/1 paths, scan interval 60 secs

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
164.1.18.1      4   300      17      15        4    0    0 00:00:36
1
Rack1SW2#ping 164.1.18.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 164.1.18.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/9 ms
```

Ticket 9

Analyze the Symptoms

Luckily, IPv6 domain in this scenario is really small. Thus, we may safely assume there is something wrong in R1 as the other routers are out of our control. First, we check if the issue really exists:

```
Rack1R1#ping 2001:150:1:3::3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:150:1:3::3, timeout is 2
seconds:
.....
Success rate is 0 percent (0/5)
```

OK, we can't ping the R3's Loopback100 interface. The ticket states that there are no link issues, but it never hurts to double check:

```
Rack1R1#show ipv6 interface brief
FastEthernet0/0            [up/up]
Serial0/0                  [up/up]
    FE80::20F:23FF:FED5:5220
    2001:164:1:12::1
Serial0/1                  [up/up]
    FE80::20F:23FF:FED5:5220
    2001:164:1:13::1
Loopback0                  [up/up]
```

There appear to be no physical problems. You may ping the IPv4 addresses on these links to make sure everything is OK:

```
Rack1R1#ping 164.1.12.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 164.1.12.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/57/61 ms

Rack1R1#ping 164.1.13.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 164.1.13.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/29/32 ms
```

And so we proceed to isolating the IPv6 misconfiguration problem, which is our only and the primary hypothesis.

Isolate the Issue

Let's check if we can ping across the links using the IPv6 addresses:

```
Rack1R1#ping 2001:164:1:12::2
```

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:164:1:12::2, timeout is 2
seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/56/57 ms

Rack1R1#ping 2001:164:1:13::3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:164:1:13::3, timeout is 2
seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/28 ms
```

The addressing appears to be correct. Let’s see if routing is in order:

```
Rack1R1#show ipv6 route rip
IPv6 Routing Table - 8 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS
summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF
ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
R   2001:150:1:3::/64 [120/3]
     via FE80::202, Serial0/0
R   2001:164:1:23::/64 [120/2]
     via FE80::202, Serial0/0
```

We see the both prefixes attached to R3 reachable via R1. However, for some reason we cannot reach the Loopback100. The next thing we may want to check is whether we can reach the next-hop IPv6 address for the prefix in question. The next hop to reach 2001:150:1:3::3 is FE80::202. Let’s try to ping it:

```
Rack1R1#ping fe80::202
Output Interface: Serial0/0
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to FE80::202, timeout is 2 seconds:
Packet sent with a source address of FE80::20F:23FF:FED5:5220
.....
Success rate is 0 percent (0/5)
```

The address does not respond. This means either of two: we don’t have this link-local address properly mapped on the Frame-Relay interface or the other side does not map our IPv6 link-local address correctly. The second issue is impossible, because R2 uses point-to-point subinterface. So we check the local mapping for FE80::202.

```
Rack1R1#show frame-relay map
Serial0/0 (up): ipv6 FE80::2 dlci 102(0x66,0x1860), static,
              CISCO, status defined, active
Serial0/0 (up): ipv6 2001:164:1:12::2 dlci 102(0x66,0x1860), static,
              broadcast,
              CISCO, status defined, active
Serial0/0 (up): ip 164.1.12.2 dlci 102(0x66,0x1860), static,
              broadcast,
              CISCO, status defined, active
```

We have the map for FE80::2 but not or FE80::202! This appears to be the root cause of the problem.

Conclusion: Be careful with IPv6 routing on NBMA interfaces. Next-hop address uses the link-local format, and you need to make sure these addresses are properly mapped, as IPv6 does not support mechanisms such as Inverse-ARP.

Fix the Issue

We simply add the mapping to R2 for the discovered Link-Local address used as RIP-NG update source in R2:

```
R1:
interface Serial 0/0
 frame-relay map ipv6 FE80::202 102
```

Verify

Make sure we can reach the Loopback100 address via R2 now:

```
Rack1R1#ping 2001:150:1:3::3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:150:1:3::3, timeout is 2
seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 72/98/117
ms

Rack1R1#trace 2001:150:1:3::3

Type escape sequence to abort.
Tracing the route to 2001:150:1:3::3

  1 2001:164:1:12::2 45 msec 44 msec 44 msec
  2 2001:150:1:3::3 48 msec 88 msec 44 msec
```

Ticket 10

Analyze the Symptom

From the multicast distribution diagram you may remember that we though of some RPF issues that may happen during redistribution. This makes us think of R4 as a potential RPF failure point. We quickly check if R4 sees the route to VLAN26 via R3:

```
Rack1R4#show ip route 164.1.26.0
Routing entry for 164.1.26.0/24
  Known via "ospf 1", distance 110, metric 10
  Tag 390, type extern 2, forward metric 390
  Last update from 164.1.34.3 on Serial0/0, 00:06:40 ago
  Routing Descriptor Blocks:
  * 164.1.34.3, from 150.1.3.3, 00:06:40 ago, via Serial0/0
      Route metric is 10, traffic share count is 1
      Route tag 390
```

So there should not be any issues at R4, which is the only potential source of the problem. Therefore, the whole path between R6 and VLAN41 is under suspicion. However, we only control a few routers there, so the issues is somewhere in R4, since this is where all traffic converges!

In order to troubleshoot multicast issue, the best thing is simulating the multicast flow from the source segment to destination, and then tracing the potential issues. So far, we note that group 226.37.1.1 maps to RP R4 per the baseline information. The next step is setting up R6 as a multicast source and SW1 as a multicast sink.

Isolate the Issue

Here is the configuration for R6 and SW1. Notice that R6 is configured to flood all multicast traffic out of VLAN 26 interface. We configure static RP in R6, just in case if it's the DR for the segment, which it should be based on the highest IP address. SW1 is configured to accept the multicast packets for the group in question.

```
R6:
ip multicast-routing
!
ip pim rp-address
!
interface FastEthernet 0/1
 ip pim dense-mode

SW1:
interface VLAN 41
 ip igmp join-group 226.37.1.1
```

After this, we check the multicast routing table of R4. At this point, we should have the (*,G) entry for our group, as SW1 should have joined the shared tree.

```
Rack1R4#show ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C -
Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry,
       X - Proxy Join Timer Running, A - Candidate for MSDP
Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 226.37.1.1), 00:31:53/00:03:21, RP 150.1.4.4, flags: SJC
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse-Dense, 00:31:51/00:03:21

(*, 224.0.1.40), 00:31:54/00:02:49, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Loopback0, Forward/Sparse-Dense, 00:31:54/00:00:00
    FastEthernet0/0, Forward/Sparse-Dense, 00:31:55/00:00:00
```

The incoming interface is set to Null, because the is no traffic source for this group and this is just the shared tree. We now start sending multicast traffic from R6.

```
Rack1R6#ping 226.37.1.1 repeat 100

Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 226.37.1.1, timeout is 2 seconds:
.....
```

At this point, we should check if the DR has registered the source with the RP. Registration is performed using unicast PIM messages.

```
Rack1R4#show ip mroute 226.37.1.1
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C -
Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry,
       X - Proxy Join Timer Running, A - Candidate for MSDP
Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group
Outgoing interface flags: H - Hardware switched, A - Assert winner
```

```
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 226.37.1.1), 00:35:17/00:02:41, RP 150.1.4.4, flags: SJC
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse-Dense, 00:35:15/00:02:56
```

There is nothing on the RP about the traffic source. As there should not be any RPF failures per our expectations, we start PIM messages debugging in R6:

```
Rack1R4#debug ip pim
PIM debugging is on

PIM(0): v2, PIM message 164.1.26.6 explicitly denied on Serial0/0
Rack1R4#
PIM(0): v2, PIM message 164.1.26.6 explicitly denied on Serial0/0
Rack1R4#
PIM(0): v2, PIM message 164.1.26.6 explicitly denied on Serial0/0
Rack1R4#
PIM(0): v2, PIM message 164.1.26.6 explicitly denied on Serial0/0
```

So we see that the PIM register messages from R6 are being rejected by R4, most likely due to the policy configuration. Seeing the interface name in the messages, we check PIM configuration on this interface:

```
Rack1R4#show ip pim interface serial 0/0 detail
Serial0/0 is up, line protocol is up
  Internet address is 164.1.34.4/24
  Multicast switching: process
  Multicast packets in/out: 8/0
  Multicast TTL threshold: 0
  PIM: enabled
    PIM version: 2, mode: sparse-dense
    PIM DR: 164.1.34.4 (this system)
    PIM neighbor count: 0
    PIM Hello/Query interval: 60 seconds
    PIM Hello packets in/out: 338/767
    PIM State-Refresh processing: enabled
    PIM State-Refresh origination: disabled
    PIM NBMA mode: disabled
    PIM ATM multipoint signalling: disabled
    PIM domain border: disabled
    PIM neighbor filter: 1
  Multicast Tagswitching: disabled
```

We see two interesting things: first, there are no adjacent neighbors; second, there is neighbor filter applied to the interface. This might be our chance to fix the problem. We inspect the access-list mentioned in the output:

```
Rack1R4#show ip access-lists 1
Standard IP access list 1
    10 deny   any (241 matches)
```

And we notice that this list denies anything. Apparently, this looks like the misconfiguration. We go ahead and fix it.

Conclusion: In some situations, having fewer devices to deal with would greatly reduces the amount of the checks that you have to apply.

Fix the Issue

```
R4:
interface Serial 0/0
 no ip pim neighbor-filter 1

%PIM-5-NBRCHG: neighbor 164.1.34.3 UP on interface Serial0/0

Rack1R4#show ip pim neighbor
PIM Neighbor Table
Mode: B - Bidir Capable, DR - Designated Router, N - Default DR
Priority,
      S - State Refresh Capable
Neighbor          Interface                Uptime/Expires    Ver   DR
Address
Prio/Mode
164.1.47.7        FastEthernet0/0          00:44:36/00:01:28 v2    1 /
DR S
164.1.34.3        Serial0/0                00:00:16/00:01:28 v2    1 /
S
```

Verify

OK so it’s about time to check if our solution is working now.

```
Rack1R6#ping 226.37.1.1 repeat 100

Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 226.37.1.1, timeout is 2 seconds:

Reply to request 0 from 164.1.47.7, 136 ms
Reply to request 1 from 164.1.47.7, 120 ms
Reply to request 2 from 164.1.47.7, 117 ms
Reply to request 3 from 164.1.47.7, 120 ms
Reply to request 4 from 164.1.47.7, 116 ms
```

Now let’s see how R4 multicast routing table should look like in this situation:

```
Rack1R4#show ip mroute 226.37.1.1
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C -
Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry,
       X - Proxy Join Timer Running, A - Candidate for MSDP
Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group
```

```
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 226.37.1.1), 00:46:03/00:03:02, RP 150.1.4.4, flags: SJC
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse-Dense, 00:46:01/00:03:02

(164.1.26.6, 226.37.1.1), 00:03:36/00:00:11, flags: T
  Incoming interface: Serial0/0, RPF nbr 164.1.34.3
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse-Dense, 00:03:36/00:03:02
```

There are two trees. One tree is shared, and used for the sources that just start sending traffic. The other tree has been initiated by the router closest to the receiver – this is the shortest tree to the multicast source, which actually transports the multicast traffic.

IEWB-RS VOL4 Lab 2

Lab Overview:

The following scenario is a practice lab exam designed to test your skills at troubleshooting Cisco networking devices. Specifically, this scenario is designed to assist you in your preparation for Cisco Systems' CCIE Routing & Switching Lab exam Troubleshooting Section. However, remember that in addition to being designed as a simulation of the actual CCIE lab exam, this practice lab should be used as a learning tool. Instead of rushing through the lab in order to resolve all issues, take the time to apply the structured troubleshooting methodology and improve your strategy.

Lab Instructions:

Prior to starting, ensure that the initial configuration scripts for this lab have been applied. For a current copy of these scripts, see the Internetwork Expert members' site at http://members.INE.com

Refer to the attached diagrams for interface and protocol assignments. Any reference to X in an IP address refers to your rack number, while any reference to Y in an IP address refers to your router number. When not explicitly mentioned, a router's IP address on the segment is based off the router number, Y. For example, R1 will have the IP address of 150.X.100.1 on the subnet 150.X.100.0/24, and SW3 will have the IP address 150.X.100.9 on the same subnet.

Use the name **`cisco`** along with the password of **`cisco`** to access the console line of any device used in the topology.

Lab Do's and Don'ts:

- Do not access the routers that are marked as restricted for your access.
- Do not use the **`show running-config`** or **`show startup-config`** commands or their equivalents when performing troubleshooting.
- Do not change or add any IP addresses from the initial configuration unless required for troubleshooting.
- Do not change any interface encapsulations unless required for troubleshooting.
- Do not change the console, AUX, and VTY passwords or access methods unless otherwise specified.
- Do not use default routes, default networks, or policy routing unless otherwise specified.
- Save your configurations often.

Grading:

This practice lab consists of 10 trouble tickets totaling 30 points. A score of 24 points is required to achieve a passing grade. A trouble ticket must be fixed 100% with the requirements given in order to be awarded the points for that ticket. No partial credit is awarded. If a ticket has multiple possible resolutions, choose the solution that best meets the requirements and requires minimal changes. Per the CCIE R&S lab exam requirements, you are required to finish this lab in **`two`** hours.

The tickets generally have no dependencies, unless explicitly stated in the ticket outline, so you may work through them in any order you like. It’s up to you to select the tickets that you feel most easy to deal with and manage your time accordingly.

GOOD LUCK!

Baseline

All network devices are configured according to the diagram provided with this scenario. The diagram reflects the proper network configuration, including IP address, IGP protocol settings and BGP AS numbers and serves as your primary source of the information. This section provides the scenario-specific configuration information that you may need during troubleshooting process. Notice that not all of the information may be useful during the troubleshooting process; however, all statement made below reflect the correct network configuration.

Devices under your Control

For this lab, you may only access and modify the configuration all devices with except to SW1, SW3 and SW4. Backbone devices BB1, BB2 and BB3 are out of your control per the initial topology configuration. If you refer to the diagram provided, the devices colored in **ORANGE** are out of your control.

Bridging and Switching

- All switches are in transparent VTP mode and use VTP domain name of CCIE.
- The following is the list of the trunk links interconnecting the switches:
  - SW1's interfaces Fa0/13, Fa0/14, and Fa0/15 and SW2's interfaces Fa0/13, Fa0/14, and Fa0/15.
  - SW1's interfaces Fa0/19, Fa0/20, and Fa0/21 and SW4's interfaces Fa0/13, Fa0/14, and Fa0/15.
  - SW2's interfaces Fa0/16, Fa0/17, and Fa0/18 and SW3's interfaces Fa0/16, Fa0/17, and Fa0/18.
  - SW3's interfaces Fa0/19, Fa0/20, and Fa0/21 and SW4's interfaces Fa0/19, Fa0/20, and Fa0/21.
- SW1 is the STP root bridge for all VLANs. Classic STP (PVST+) is in use.
- The Frame-Relay sub-interfaces of R2 and R3 are configured as point-to-point.

IGP

- Mutual redistribution is configured between RIP and OSPF on R3 and R4.
- IPv6 is configured on the Frame-Relay link between R1 and R2 using the network 2001:141:X:12::/64
- IPv6 is configured on R2, R5 and SW2 using the IPv6 prefix 2001:141:X:25::/64.
- RIPng is used as IPv6 routing protocol between R1, R2 and R3.
- OSPFv3 Area 0 is configured on the Frame-Relay link while OSPFv3 Area 1 is configured on the Ethernet segment.
- New Loopback interfaces with the IP addresses 2001:150:X:Y::Y/64 have been created on R5 and SW2 and advertised into OSPFv3 Area 1.

BGP

- BGP peering sessions are configured according to the following table:

Device 1	Device 2
R6	BB1
R6	R2
R1	BB2
R1	R2
R2	R5
R5	SW2
R5	R4

- R1 and BB2 peering session is authenticated used the password value of "CISCO".

Multicast

- IP Multicast routing configured on R1, R2, R3, R4, R5, SW1 and SW2.
- PIM sparse mode enabled on the following interfaces:

Device	Interface
R1	Fa0/0
R1	S0/0.1
R2	Fa0/0
R2	S0/0
R3	Fa0/0
R3	Fa0/1
R3	S1/0.1
R4	Fa0/0
R4	Fa0/1
R4	S0/0
R5	Fa0/0
R5	Fa0/1
R5	S0/0
SW1	Fa0/3
SW1	VL7
SW2	VL258

- Auto-RP is used to disseminate Group-to-RP mapping information through the network.
  - R2 is the RP for the multicast groups 225.0.0.0 through 225.255.255.255.
  - R5 is the RP for the multicast groups 239.0.0.0 through 239.255.255.255.
- SW2 is the Auto-RP Mapping Agent

Trouble Tickets

Note

Prior to loading the initial configurations, make sure you switched SW2 to dual IPv4/IPv6 mode using the command **`sdm prefer dual-ipv4-and-ipv6 routing`** and reloading the switch

Ticket 1: WWW Connectivity

- The user on a host on VLAN255 (simulated by SW3) has been reported that he is unable to reach the Web server on VLAN37
- The user told you that he was able to reach the same host until today. He suspects that the recent software updates on the Web server are to blame.
- Assuming you cannot make any changes to the user's PC isolate and fix this issue.

3 Points

Ticket 2: Suboptimal Routing

- Your NMS reports high link utilization for the Frame-Relay connection between R4 and R5.
- However, per the normal configuration the fast link connecting R4 and R5 should be used.
- Find the problem and ensure optimal routing while introducing minimal changes to the network.

3 Points

Ticket 3: Connectivity Issue

- Users on VLAN 12 report that they cannot reach any of VLAN43 resources.
- You suspect that the latest security policy changes in AS 400 are to blame.
- Fix this issue and make sure you can ping and telnet from VLAN12 to VLAN43

3 Points

Ticket 4: RIPv2

- After a recent change in the routing policy users behind BB1 started complaining they cannot reach VLAN7 and VLAN77 subnets.
- Figure out what the issue is and fix the problem.

3 Points

Ticket 5: Traceroute

- The admin in charge of the networks behind BB3 reported that he cannot reach R5 via the **`traceroute`** command.
- He suspects you implementing some sort of filtering blocking the **`traceroute`** command from working properly.
- Find what might be causing such behavior and fix the problem.

3 Points

Ticket 6: NAT

- There is a server on VLAN45 that has been just moved in. Apparently, the server admin did not configure the default gateway properly.
- A user with the IP address of BB3 needs to access the server immediately, and you cannot find the server admin to fix the configuration problem.
- You tasked a subordinate of yours to configure NAT in R4 so that BB3 should see the server as 204.12.X.100 and the server should communicate to the BB3 using the IP 141.X.145.254.
- However, the configuration made by your subordinate is not working for some reason.
- Using your knowledge of Cisco NAT make sure the configuration works.

3 Points

Ticket 7: BGP

Note: This ticket requires you to resolve Ticket 3 prior to starting.

- Recently, the administrator of AS 254 told you that they cannot reach AS 400 networks anymore.
- As the admin of AS400 you have been given access to AS300 and AS200 BGP routers to investigate this issue. .
- With minimal configuration changes, provide a solution to this problem.

2 Points

Ticket 8: DHCP

- A new host has been deployed on VLAN45 configured to obtain an IP address via DHCP.
- The address to be allocated should be 141.X.145.100.
- R5 is pre-configured as DHCP relay inserting information option with the subscriber-id value of “VLAN45”.
- The DHCP server is located in R3 and you asked your subordinate to configure it for DHCP address allocation based on Option 82.
- However, when you came back from lunch you found that the host still cannot auto-configure the IP address.
- Fix the issue and make sure the allocation based on Option 82 works.

4 Points

Ticket 9: IPv6

- R1 is not being able to reach the Loobpack100 subnet of R5 but it can reach the Loopback100 subnet of SW2.
- Modifying only the IPv6 relevant configuration, resolve this issue.
- Apply minimal configuration changes to accomplish this task.

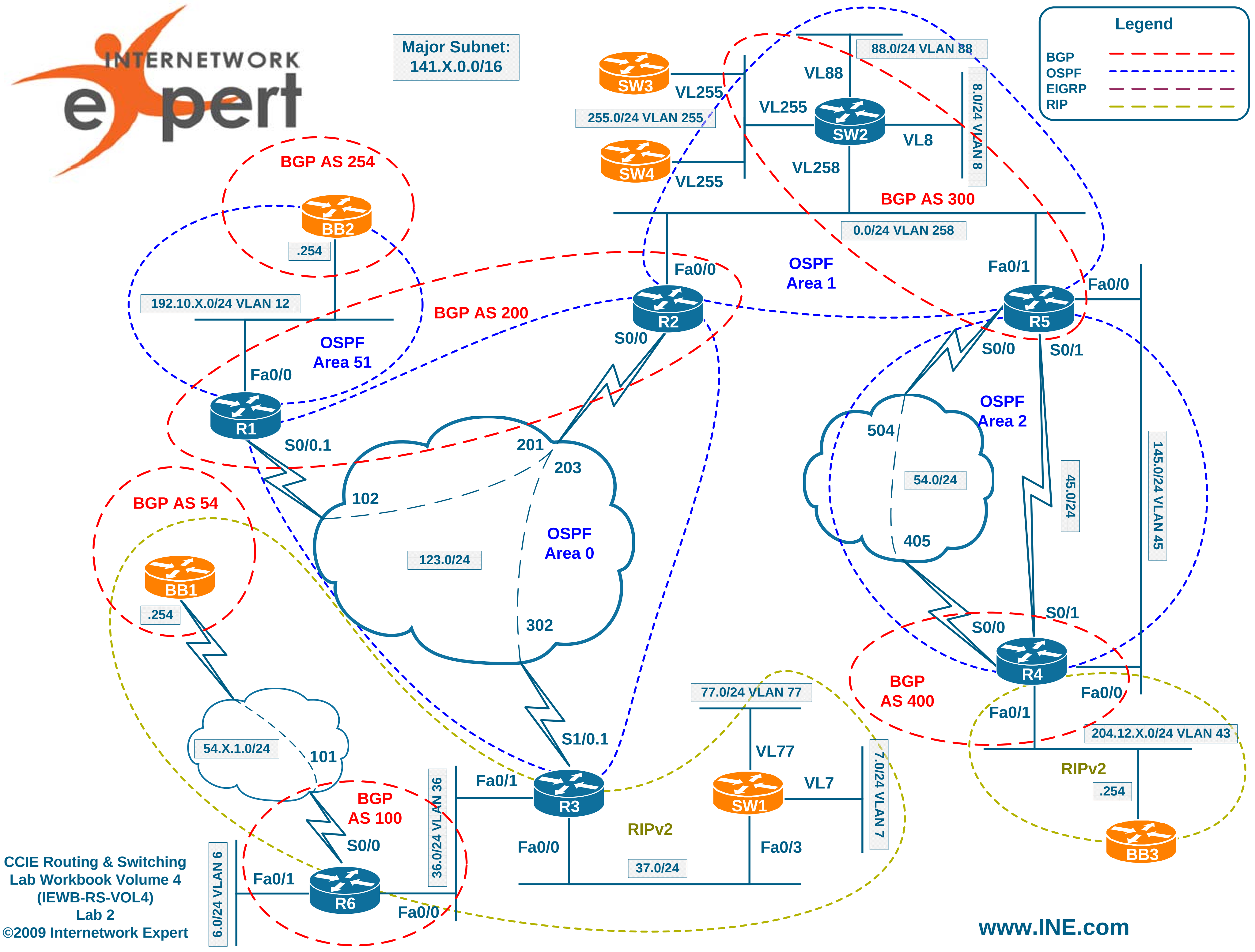
3 Points

Ticket 10: Multicast

Note: This ticket requires you to resolve Tickets 2 and 3 prior to starting.

- Multicast listeners on VLAN37 cannot receive feeds sourced in VLAN12 and VLAN43.
- The group used to source the multicast feed is 239.X.X.X where X is your rack number.
- Resolve this issue without using any static multicast routes and make sure the feed flows through the network.

3 Points


INTERNETWORK expert
Major Subnet: 141.X.0.0/16
Legend
BGP
OSPF
EIGRP
RIP
SW3
SW4
SW2
VL255
VL255
VL255
VL88
VL8
VL258
255.0/24 VLAN 255
88.0/24 VLAN 88
8.0/24 VLAN 8
0.0/24 VLAN 258
BGP AS 300
OSPF Area 1
BGP AS 254
BB2
.254
192.10.X.0/24 VLAN 12
OSPF Area 51
BGP AS 200
R1
Fa0/0
S0/0.1
R2
Fa0/0
S0/0
R5
Fa0/1
Fa0/0
S0/0
S0/1
201
203
102
302
123.0/24
OSPF Area 0
OSPF Area 2
504
405
54.0/24
45.0/24
145.0/24 VLAN 45
BGP AS 54
BB1
.254
54.X.1.0/24
101
R4
S0/0
S0/1
Fa0/0
Fa0/1
BGP AS 400
204.12.X.0/24 VLAN 43
RIPv2
.254
BB3
R3
S1/0.1
Fa0/1
Fa0/0
36.0/24 VLAN 36
77.0/24 VLAN 77
VL77
VL7
SW1
Fa0/3
7.0/24 VLAN 7
RIPv2
37.0/24
BGP AS 100
R6
S0/0
Fa0/1
Fa0/0
6.0/24 VLAN 6
CCIE Routing & Switching
Lab Workbook Volume 4
(IEWB-RS-VOL4)
Lab 2
©2009 Internetwork Expert
www.INE.com

IEWB-RS VOL4 Lab 2 Solutions

Table of Contents

IEWB-RS VOL4 Lab 2 Solutions 1
- Table of Contents 1

Build and Analyze the Baseline 3
- Layer 2 Diagram 4
- BGP Diagram 5
- Multicast and Redistribution 6
  - Redistribution Loops Analysis 6
  - Multicast Propagation Analysis 7
- IPv6 Diagram 8
- Read over the Lab 8

Solutions 9
- Ticket 1 9
  - Analyze the Symptoms 9
  - Isolate the Issue 9
  - Fix the issue 11
  - Verify 11
- Ticket 2 12
  - Analyze the Symptoms 12
  - Isolate the Issue 12
  - Fix the Issue 13
  - Verify 13
- Ticket 3 15
  - Analyze the Symptoms 15
  - Isolate the Issue 16
  - Fix the Issue 19
  - Verify 20
- Ticket 4 22
  - Isolate the Issue 23
  - Fix the Issue 25
  - Verify 28
- Ticket 5 30
  - Analyze the Symptoms 30
  - Isolate the Issue 31
  - Fix the Issue 35
  - Verify 35
- Ticket 6 36
  - Analyze the Symptoms 36
  - Isolate the Issue 36
  - Fix the Issue 39

Verify....39
Ticket 7....40
Analyze the Symptoms....40
Isolate the Issue....41
Fix the Issue....42
Verify....42
Ticket 8....44
Analyze the Symptoms....44
Isolate the Issue....44
Fix the Issue....48
Verify....48
Ticket 9....49
Analyze the Symptoms....49
Isolate the Issue....50
Fix the Issue....54
Verify....54
Ticket 10....55
Analyze the Symptoms....55
Isolate the Issue....55
Fix the Issue....63
Verify....64

Build and Analyze the Baseline

When you start with the scenario, all you have is the diagram and some textual information on the network baseline. Your goal at this moment is structuring the available information and making additional diagrams. We recommend extra diagrams to outline the following: L2 connection, BGP Peerings, Multicast & Redistribution and IPv6 Topology. Notice that some of these could be combined in a single diagram – for example you may put the Multicast and Redistribution outlines on the initial L3 diagram. This is probably the best way to save your time during the analysis stage. However, we are going to use separate diagrams for the ease of explanation here.

Layer 2 Diagram

This is where our Ethernet L2 connections are outlined (notice, no L3 Etherchannels). We outline the Root Bridge, VTP domains and roles, port numbers and trunk encapsulations. This diagram would helps us finding any L2 mis-configurations. Notice that we only put the routers that have configured Ethernet connections on this diagram.

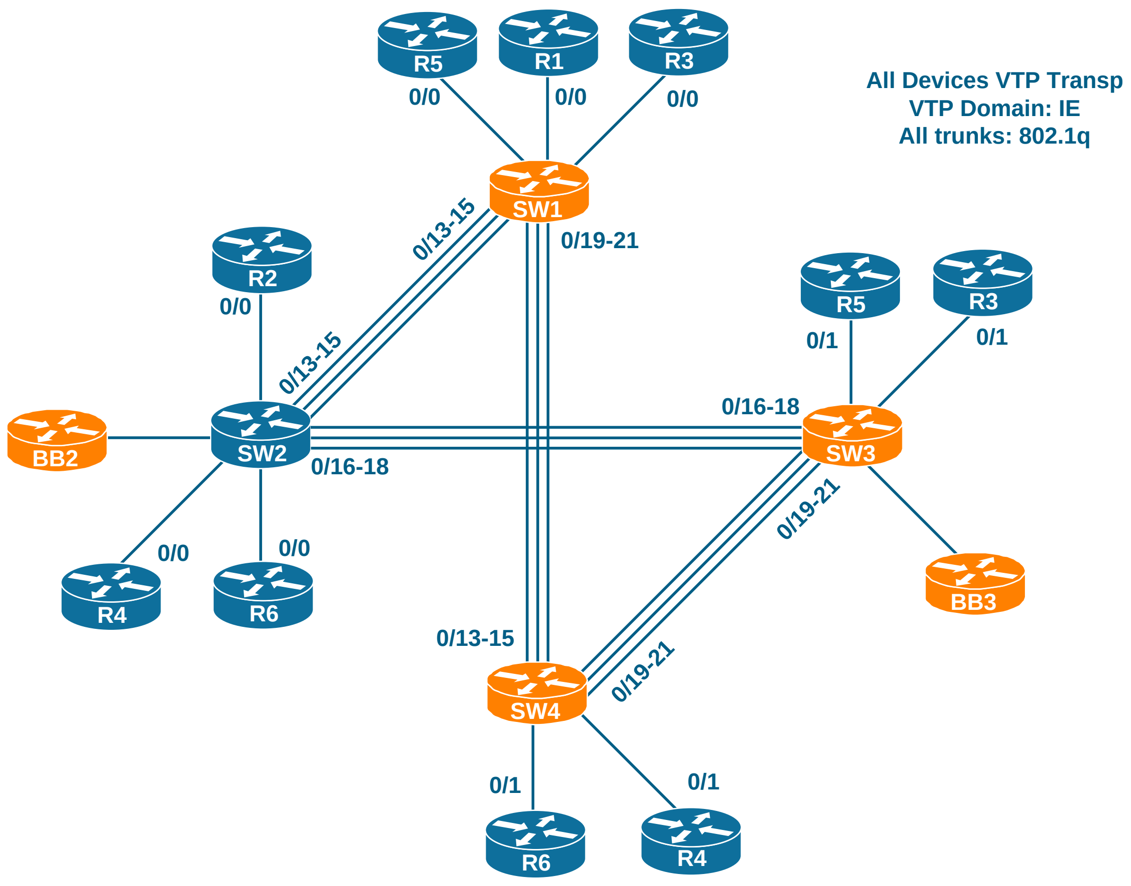


Fig 1

The use of VTP transparent makes the topology a bit simpler, as there is no VTP pruning and VTP synchronization issues. The connections form a loop, and thus there should be at least one switch blocking uplinks to one of its peers. The blocking ports may differ for every VLAN, and there is not enough information in the baseline to tell what are the logical topologies for every VLAN.

BGP Diagram

Using the information found in the Baseline configuration outline, we may draw a BGP peering diagram. The red arrows mark eBGP peering sessions while blue arrows mark iBGP sessions. Notice that we omit the link IP addressing and use simplified drawing for Ethernet/Serial/FR links.

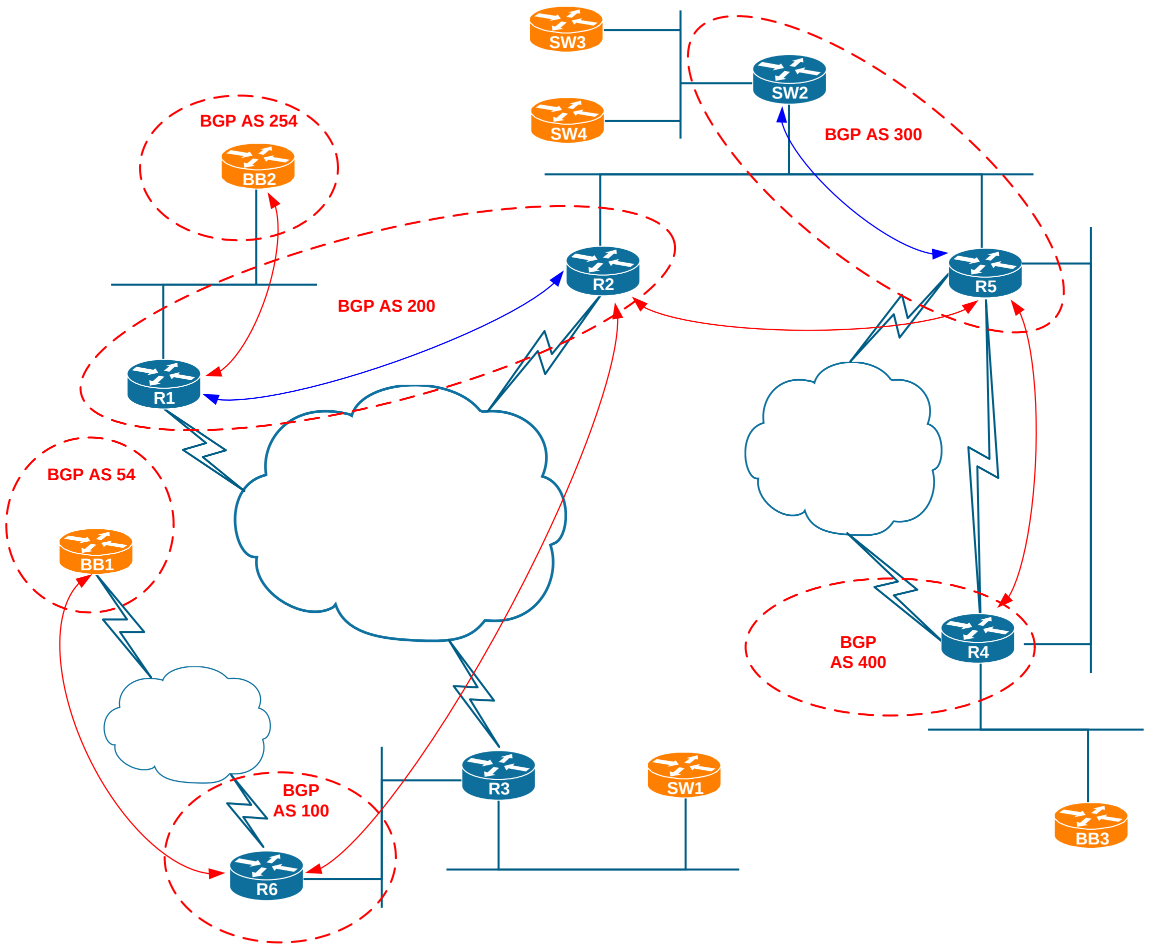


Fig 2

No extra information on BGP policies is provided, so we just leave the diagram for further reference.

Multicast and Redistribution

We usually group these two technologies on the same diagram because they both depend on IGP protocols. Thus, the diagram should have IGPs outlined in addition to the Multicast and redistribution marking. We mark the links that have PIM enabled using the **RED** color. This makes it easy to see where the multicast traffic may actually flow. Notice that we copied some interface names from the main L3 diagram so we know the multicast-enabled interfaces by name.

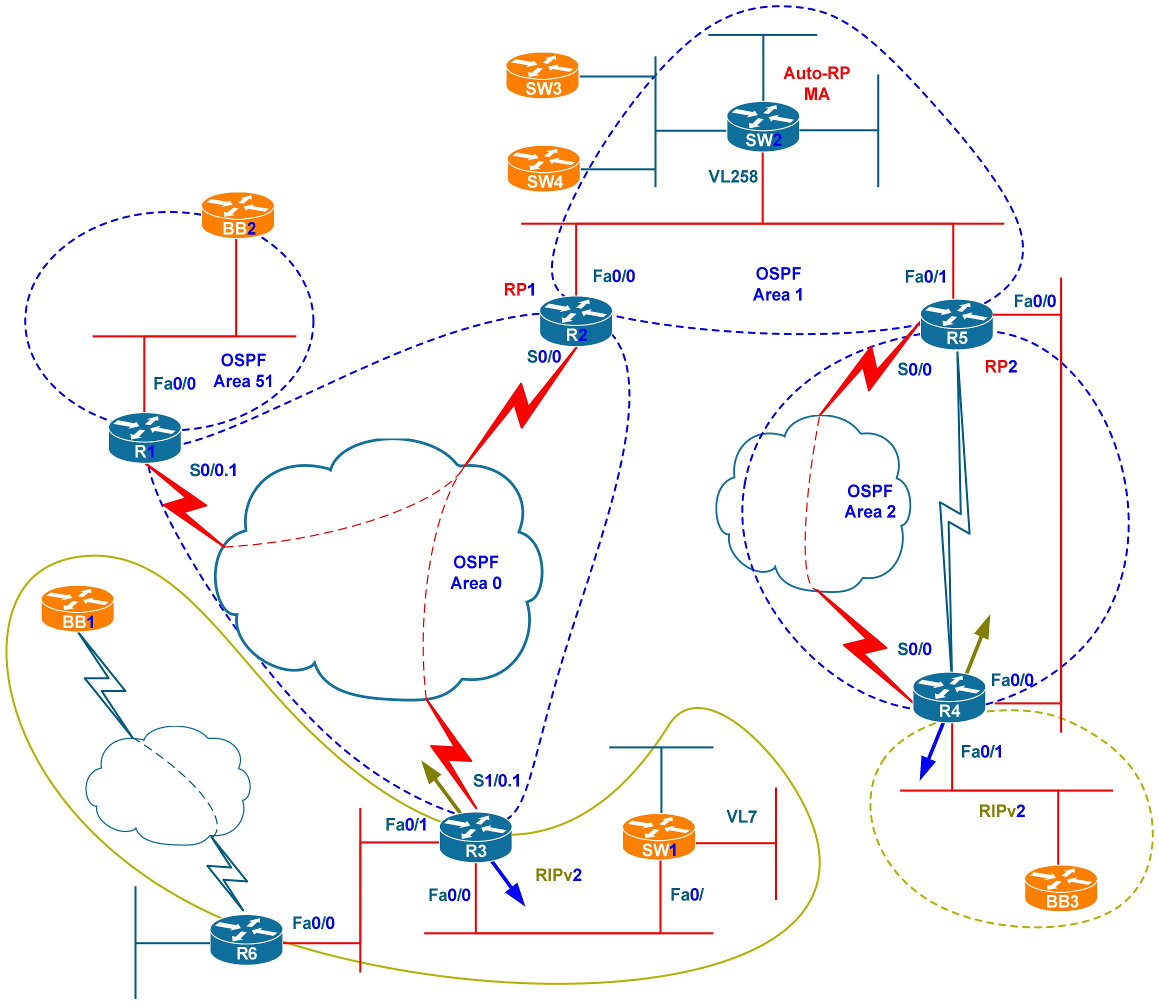


Fig 3

You can also see the PIM SM RPs outlined on this diagram. The detailed groups to RP mappings are not provided on the diagram.

Redistribution Loops Analysis

Luckily in this topology there are no loops, and thus we could skip the time-consuming redistribution loop analysis.

Multicast Propagation Analysis

Since the multicast domain spans just the area covered by one IGP, we should not expect any RPF failures due to redistribution.

Looking at this diagram we may notice some potential issues already. For example, there could be an OIL (Output Interface List) issue when sources behind R1 send traffic to receivers behind R3, as the interface of R2 is multipoint. Next, there are multiple links enabled for multicast and connecting R4 and R5. If for some reason the multicast traffic would take the path not preferred by IGP, there could be an RPF issue on R4 or R5.

IPv6 Diagram

IPv6 spans 5 devices in this topology. There is just one IGP, OSPFv3 with two areas. We may notice that there could be some potential problems with FR mappings on R2, due to the multipoint interface nature.

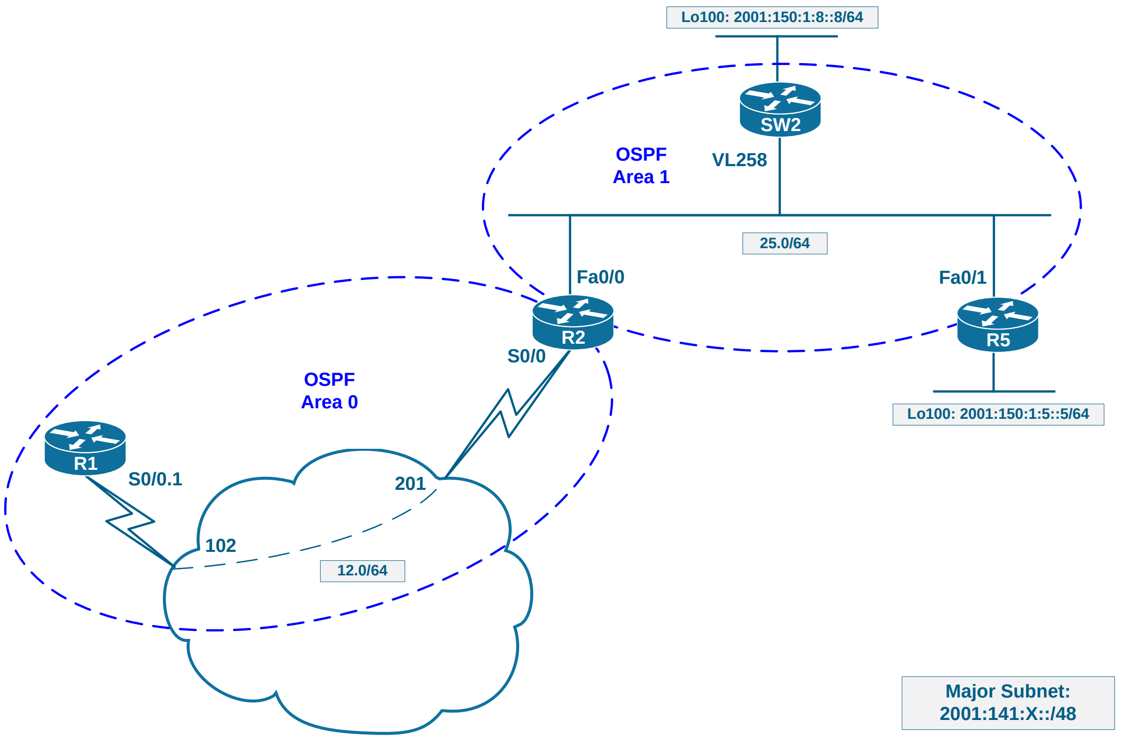

Lo100: 2001:150:1:8::8/64
SW2
OSPF Area 1
VL258
25.0/64
Fa0/0
Fa0/1
R2
R5
S0/0
Lo100: 2001:150:1:5::5/64
OSPF Area 0
R1
S0/0.1
201
102
12.0/64
Major Subnet: 2001:141:X::/48

Fig 4

Read over the Lab

Our last step is looking through the tickets and seeing if we can find any potentially useful information prior to starting the troubleshooting process. First we can see Ticket 10 having dependency on tickets 2 and 3. We may want to consider this ticket last in series, as it requires so much work. The same logic applies to Ticket 7, which depends on ticket 3.

Ticket 8 is somewhat special because it's about "application" related issue. You may not be familiar with DHCP Option 82 so you may want to skip it. However, this leaves just one more ticket to fail! Remember you need around 8 tickets to pass the section (actually it is only about getting 24 points).

There is also something special about Ticket 9 – it states that you only have to deal with IPv6 settings. This sets the ticket apart from other ones, and prompts that there should NOT be any L1/L2 issues on the FR cloud and VLAN258 segment! This is a nice piece of information we get just by reading the tickets!

Solutions

Ticket 1

Analyze the Symptoms

First, we have the suggesting that the recent software changes might have caused the issue. However, we cannot trust the user's opinion in such delicate question and need to suspect every link/node on the path between the two endpoints. Using the divide-and-conquer approach let's try finding the initial scope of the problem. We use the traceroute command to check end-to-end connecitivity.

```
Rack1R3#traceroute 141.1.255.9 source fastEthernet 0/0

Type escape sequence to abort.
Tracing the route to 141.1.255.9

  1 141.1.123.2 28 msec 28 msec 28 msec
  2 141.1.0.8 32 msec 28 msec 32 msec
  3  *  *  *
  4  *  *
```

We trace to another host on the segment and the trace succeeds:

```
Rack1R3#traceroute 141.1.255.10 source fastEthernet 0/0

Type escape sequence to abort.
Tracing the route to 141.1.255.10

  1 141.1.123.2 32 msec 29 msec 32 msec
  2 141.1.0.8 28 msec 32 msec 28 msec
```

Alright, so now we can may a guess – there is something wrong between the host and SW2, which is supposed to be the host's default router. The initial problem scope is limited to SW2 and SW3 as well.

Isolate the Issue

Let's start bottom-up checking the physical connectivity between SW2 and SW3:

```
Rack1SW2#ping 141.1.255.9

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 141.1.255.9, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 1/2/8 ms
```

This simple test rules out any link issues and allows us to suppose that the problem may is somewhere in Layer 3 configuration. One of the most common issues is misconfiguring the default gateway – you may quickly see whether this holds true by pinging the host off the directly attached subnet and then using different source for the ping operation.

```
Rack1SW2#ping 141.1.255.9 source vlan 258

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 141.1.255.9, timeout is 2 seconds:
Packet sent with a source address of 141.1.0.8
.....
Success rate is 0 percent (0/5)
```

Well it looks like a default gateway issue… however this may be something else, like a misconfigured access-list. Where could that list apply? To the SW2's SVI, then to any transit switch on the path between SW2 and SW4 SVIs, which may take any way around other switches. But luckily we only have access to SW2, so we just check if there are any access-lists on there:

```
Rack1SW2#sh ip interface vlan 255 | inc acces
  Outgoing access list is not set
  Inbound  access list is not set
  IP access violation accounting is disabled
```

So apparently is the default gateway misconfiguration. We have two options here: either the host has incorrect default gateway set OR it does not have any default gateway at all. In the first case, the issue could be fixed by configuring the appropriate IP address on SW2's SVI. In the second case, we may try using Proxy ARP if the host supports it. Let's investigate how the host generates ARPs requests.

```
Rack1SW2#ping 141.1.255.9 source vlan 258

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 141.1.255.9, timeout is 2 seconds:
Packet sent with a source address of 141.1.0.8
.....
Success rate is 0 percent (0/5)

Rack1SW2#
IP ARP: rcvd req src 141.1.255.9 000f.f76d.ac80, dst 141.1.0.8 Vlan255
IP ARP: rcvd req src 141.1.255.9 000f.f76d.ac80, dst 141.1.0.8 Vlan255
IP ARP: rcvd req src 141.1.255.9 000f.f76d.ac80, dst 141.1.0.8 Vlan255
IP ARP: rcvd req src 141.1.255.9 000f.f76d.ac80, dst 141.1.0.8 Vlan255
IP ARP: rcvd req src 141.1.255.9 000f.f76d.ac80, dst 141.1.0.8 Vlan255

Rack1SW2#show ip interface brief  | include Vlan258
Vlan258                141.1.0.8       YES manual up
up
```

As you can see, the host attempts to ARP for the IP address we used as source for the ping packets. It did not ARP for the default gateway IP, which means proxy ARP is in use. Now we may guess that SW2 is not responding to the Proxy RP requests, while it should be ON by default. Let's check the interface settings:

```
Rack1SW2#show ip interface vlan 255 | inc Proxy
  Proxy ARP is disabled
  Local Proxy ARP is disabled
```

As we guessed, proxy ARP is off.

`Conclusion:` Don't trust user's guesses blindly – always do an end-to-end connectivity check. Also, default gateway misconfiguration is a common problem when you configure hosts manually.

Fix the issue

```
SW2:
interface Vlan 255
 ip proxy-arp
```

Verify

Try to ping SW3 off a non-directly connected segment next:

```
Rack1SW2#ping 141.1.255.9 source vlan 258

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 141.1.255.9, timeout is 2 seconds:
Packet sent with a source address of 141.1.0.8
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 1/4/9 ms
```

And now get back to R3 and repeat the initial traceroute:

```
Rack1R3#traceroute 141.1.255.9 source fastEthernet 0/0

Type escape sequence to abort.
Tracing the route to 141.1.255.9

  1 141.1.123.2 32 msec 28 msec 32 msec
  2 141.1.0.8 28 msec 32 msec 32 msec
  3  *
    141.1.255.9 28 msec *
```

Fine, looks like we're done with this ticket!

Ticket 2

Analyze the Symptoms

First two things that come to mind when looking at this ticket are probably: there is something wrong with the link, or the metrics are set incorrectly. At least we know the problem should be someone between R4 and R5, as nothing else should probably affect traffic flows between these two.

```
Rack1R5#ping 141.1.145.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 141.1.145.4, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 1/3/4 ms
```

The link appears to be in order. So we limit the problem scope to R4 and R5 OSPF misconfiguration.

Isolate the Issue

First, let's check the health of OSPF neighbors adjacency:

```
Rack1R5#show ip ospf neighbor fastEthernet 0/0

Neighbor ID     Pri   State           Dead Time   Address
Interface
150.1.4.4         0   EXSTART/  -     00:00:07    141.1.145.4
FastEthernet0/0
```

So we spotted the problem – OSPF adjacency is not coming up. Also, we can tell that the link type is most likely point-to-point as there is no DB/BDR in the "State" column. We know how to deal with the OSPF adjacency troubleshooting – there is a debug command specifically for that.

```
Rack1R5#
OSPF: Rcv DBD from 150.1.4.4 on FastEthernet0/0 seq 0x1CA9 opt 0x52
flag 0x7 len 32  mtu 1440 state EXSTART
OSPF: First DBD and we are not SLAVE
OSPF: Send DBD to 150.1.4.4 on FastEthernet0/0 seq 0x2311 opt 0x52 flag
0x7 len 32
OSPF: Retransmitting DBD to 150.1.4.4 on FastEthernet0/0 [16]

OSPF: Rcv DBD from 150.1.4.4 on FastEthernet0/0 seq 0x1CA9 opt 0x52
flag 0x7 len 32  mtu 1440 state EXSTART
OSPF: First DBD and we are not SLAVE
OSPF: Send DBD to 150.1.4.4 on FastEthernet0/0 seq 0x2311 opt 0x52 flag
0x7 len 32
OSPF: Retransmitting DBD to 150.1.4.4 on FastEthernet0/0 [17]
```

It appears that R5 constantly retransmits DBDs to R4. The other side announces MTU of 1440 in DBD packets, so we may suspect that there is MTU mismatch between the two routers. Let's check the same commands output on the other side of the connection:

```
OSPF: Rcv DBD from 150.1.5.5 on FastEthernet0/0 seq 0x23E3 opt 0x52
flag 0x7 len 32  mtu 1500 state EXSTART
OSPF: Nbr 150.1.5.5 has larger interface MTU
OSPF: Send DBD to 150.1.5.5 on FastEthernet0/0 seq 0x9F9 opt 0x52 flag
0x7 len 32
OSPF: Retransmitting DBD to 150.1.5.5 on FastEthernet0/0 [20]
```

Now R4 clearly states that the other side advertises higher interface MTU, and thus it ignores the DBD packets.

Conclusion: when you see OSPF stuck in EXSTART it's mostly likely due to the MTU mismatch.

Fix the Issue

There are multiple was to fix the problem. First, you may adjust R5's IP MTU using the command **ip mtu <N>.** You may also configure the **ip ospf mtu-ignore** command on the router with lower MTU settings (i.e. R4 in this case). Since the task calls for minimal changes to the network, simply ignoring the larger MTU would be preferable, as it does not affect any other network components. Thus the solution is:

```
R4:
interface FastEthernet 0/0
 ip ospf mtu-ignore
```

Verify

Make sure all non-directly connected OSPF routes are preferred via the VLAN45 interface:

```
Rack1R4#sh ip route | excl direc
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS
level-2
       ia - IS-IS inter area, * - candidate default, U - per-user
static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     54.0.0.0/24 is subnetted, 1 subnets
O E2    54.1.1.0 [110/20] via 141.1.145.5, 00:01:20, FastEthernet0/0
     141.1.0.0/16 is variably subnetted, 13 subnets, 2 masks
```

```
O IA    141.1.255.0/24 [110/11114] via 141.1.145.5, 00:01:20,
FastEthernet0/0
O IA    141.1.8.0/24 [110/11114] via 141.1.145.5, 00:01:20,
FastEthernet0/0
O E2    141.1.6.0/24 [110/20] via 141.1.145.5, 00:01:20,
FastEthernet0/0
O IA    141.1.0.0/24 [110/2] via 141.1.145.5, 00:01:20, FastEthernet0/0
O IA    141.1.25.0/24 [110/11112] via 141.1.145.5, 00:01:20,
FastEthernet0/0
O E2    141.1.36.0/24 [110/20] via 141.1.145.5, 00:01:22,
FastEthernet0/0
O E2    141.1.37.0/24 [110/20] via 141.1.145.5, 00:01:22,
FastEthernet0/0
O IA    141.1.88.0/24 [110/11114] via 141.1.145.5, 00:01:22,
FastEthernet0/0
O IA    141.1.123.0/24 [110/11176] via 141.1.145.5, 00:01:22,
FastEthernet0/0
     150.1.0.0/16 is variably subnetted, 6 subnets, 2 masks
O E2    150.1.6.0/24 [110/20] via 141.1.145.5, 00:01:23,
FastEthernet0/0
O       150.1.5.5/32 [110/2] via 141.1.145.5, 00:01:23, FastEthernet0/0
O IA    150.1.3.3/32 [110/11177] via 141.1.145.5, 00:01:23,
FastEthernet0/0
O IA    150.1.2.2/32 [110/11113] via 141.1.145.5, 00:01:23,
FastEthernet0/0
O E2    150.1.8.0/24 [110/20] via 141.1.145.5, 00:01:23,
FastEthernet0/0
…
```

Ticket 3

Analyze the Symptoms

The ticket claims end-to-end issue between VLAN12 and VLAN43. This could be due to the missing routes, filtering, etc. We start with our favorite divide-and-conquer approach, tracing the route from VLAN12 to VLAN43: Notice that we trace to R4, as BB3 may not learn out routes for “some” reasons.

```
Rack1R1#traceroute 204.12.1.4

Type escape sequence to abort.
Tracing the route to 204.12.1.254

  1  *  *
Rack1R1#
```

The traceroute halts immediately, prompting that the destination prefix is not in the routing table. We check this hypothesis:

```
Rack1R1#sh ip route 204.12.1.4
% Network not in table
```

This must be because we’re having OSPF issues:

```
Rack1R1#sh ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address
Interface
192.10.1.254      1   FULL/BDR        00:00:30    192.10.1.254
FastEthernet0/0
```

OK, we’re missing the neighbor on the serial interface. The problem lies between R1 and R2. Per our initial analysis in the beginning of the lab, there should be no physical issues between R1 and R2. It never hurts to check though!

```
Rack1R1#ping 141.1.123.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 141.1.123.2, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

That’s interesting… was our initial analysis wrong?

Hypothesis: Issues with the link between R1 and R2.
Problem scope: link between R1 and R2.

Isolate the Issue

Start by checking the Frame-Relay link's health:

```
Rack1R1#show interfaces serial 0/0
Serial0/0 is up, line protocol is up
  Hardware is PowerQUICC Serial
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation FRAME-RELAY, loopback not set
  Keepalive set (10 sec)
  LMI enq sent  2977, LMI stat recvd 2978, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent  0, LMI upd sent  0
  LMI DLCI 1023  LMI type is CISCO  frame relay DTE
  FR SVC disabled, LAPF state down
  Broadcast queue 0/64, broadcasts sent/dropped 4574/0, interface
broadcasts 4204
  Last input 00:00:05, output 00:00:05, output hang never
  Last clearing of "show interface" counters 08:16:20
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
     Conversations  0/1/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     8849 packets input, 475063 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     8533 packets output, 602460 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
     0 output buffer failures, 0 output buffers swapped out
     2 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

We quickly spot that LMI is enabled, and LMI queries are being sent and replies received. Plus, the input/output counters increment, and the error counters are all zero. Lastly, we check for FR DLCIs learned by the local router:

```
Rack1R1#show frame-relay pvc | inc ST
DLCI = 102, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0.1
DLCI = 103, DLCI USAGE = UNUSED, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0
DLCI = 104, DLCI USAGE = UNUSED, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0
DLCI = 105, DLCI USAGE = UNUSED, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0
DLCI = 113, DLCI USAGE = UNUSED, PVC STATUS = INACTIVE, INTERFACE =
Serial0/0
```

Everything related to Frame-Relay appears to be in order. Thus, it's not a physical link issue, nor Frame-Relay misconfiguration. So we got to check why we cannot ping the other side. We would check the Frame-Relay mappings here first, but the link should be point-to-point and does not need any mappings. Or is it point-to-point?

```
Rack1R1#show frame-relay map
Serial0/0.1 (up): point-to-point dlci, dlci 102(0x66,0x1860), broadcast
          status defined, active
```

Yes it is. So what's next? Check the route for the IP address of R2:

```
Rack1R1#show ip route 141.1.123.2
% Subnet not in table
Rack1R1#
```

Not in table, while it should be connected! This probably means some IP address misconfiguration! Let's check the IP address of the local Serial interface:

```
Rack1R1#show ip interface serial 0/0.1
Serial0/0.1 is up, line protocol is up
  Internet address is 141.1.123.13/30
  Broadcast address is 255.255.255.255
  Address determined by setup command
<snip>
```

Now we found it. The address should be 141.1.123.1/24 per the baseline! Let's quickly fix the issue and see if OSPF comes back up:

```
R1:
interface Serial 0/0.1
 ip address 141.1.123.1 255.255.255.0
```

Rushing we check if the ping command now works:

```
Rack1R1#ping 141.1.123.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 141.1.123.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/58/60 ms
```

And if OSPF is back up:

```
Rack1R1#show ip ospf ne

Neighbor ID     Pri   State           Dead Time   Address
Interface
192.10.1.254      1   FULL/BDR        00:00:39    192.10.1.254
FastEthernet0/0
```

Which is not true, we don't see R1 among the neighbors. So something still prevents the OSPF adjacency from coming up. There could be a bunch of guesses, starting with mismatched OSPF networks, wrong subnet masks, or Frame-Relay mapping missing the broadcast keyword. The first thing to start with is compare OSPF settings on both sides of the connection:

```
Rack1R1#show ip ospf interface serial 0/0.1
Serial0/0.1 is up, line protocol is up
  Internet Address 141.1.123.1/24, Area 0
  Process ID 1, Router ID 150.1.1.1, Network Type NON_BROADCAST, Cost:
64
  Transmit Delay is 1 sec, State DROTHER, Priority 0
  No designated router on this network
  No backup designated router on this network
  Timer intervals configured, Hello 30, Dead 120, Wait 120, Retransmit
5
    oob-resync timeout 120
    Hello due in 00:00:26
  Supports Link-local Signaling (LLS)
  Index 1/1, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 2, maximum is 2
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 0, Adjacent neighbor count is 0
  Suppress hello for 0 neighbor(s)

Rack1R2#sho ip ospf interface serial 0/0
Serial0/0 is up, line protocol is up
  Internet Address 141.1.123.2/24, Area 0
  Process ID 1, Router ID 141.1.2.2, Network Type NON_BROADCAST, Cost:
64
  Transmit Delay is 1 sec, State DR, Priority 1
  Designated Router (ID) 141.1.2.2, Interface address 141.1.123.2
  No backup designated router on this network
  Timer intervals configured, Hello 30, Dead 120, Wait 120, Retransmit
5
    oob-resync timeout 120
    Hello due in 00:00:12
  Supports Link-local Signaling (LLS)
  Index 2/3, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 5, maximum is 9
  Last flood scan time is 0 msec, maximum is 4 msec
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 150.1.3.3
  Suppress hello for 0 neighbor(s)
```

Compare the above commands output. First, the subnet/masks do match. Next, the network type is nonbroadcast at both ends. R1 is DROTHER dues to its priority of zero, while R2 is the DR. Hello/Dead timers match, so this not an issue. However, R2 only has R3 as OSPF neighbor.

Since the network is non-broadcast, R2 should be configured for neighbors statically, as the DR. Let's see if that's true:

```
Rack1R2#sh ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address
Interface
150.1.3.3         0   FULL/DROTHER    00:01:58    141.1.123.3
Serial0/0
150.1.5.5         0   FULL/  -        00:00:30    141.1.25.5
Tunnel0
150.1.8.8         1   FULL/DR         00:00:38    141.1.0.8
FastEthernet0/0
```

There is only one neighbor configured on the Frame-Relay interface, which is R3. So this is the next issues – R1 is not configured as static neighbor in R2. Fix that:

```
R2:
router ospf 1
 neighbor 141.1.123.1
```

And that fixes our OSPF issue:

```
Rack1R2#sh ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address
Interface
150.1.3.3         0   FULL/DROTHER    00:01:58    141.1.123.3
Serial0/0
150.1.5.5         0   FULL/  -        00:00:30    141.1.25.5
Tunnel0
150.1.8.8         1   FULL/DR         00:00:38    141.1.0.8
FastEthernet0/0
```

There could be more problems on the way to VLAN43… We'll when performing the verifications for end-to-end connectivity.

Conclusion: Don't doubt your initial guesses! At least sometimes they are correct.

Fix the Issue

We put the things we fixed during the isolation stage together here:

```
R1:
interface Serial 0/0.1
 ip address 141.1.123.1 255.255.255.0

R2:
router ospf 1
 neighbor 141.1.123.1
```

Verify

Repeat the same traceroute test again:

```
Rack1R1#traceroute 204.12.1.4

Type escape sequence to abort.
Tracing the route to 204.12.1.4

  1 141.1.123.2 28 msec 29 msec 28 msec
  2 141.1.25.5 32 msec 28 msec 28 msec
  3 141.1.145.4 28 msec *  28 msec
```

Now ping and telnet:

```
Rack1R1#traceroute 204.12.1.4

Type escape sequence to abort.
Tracing the route to 204.12.1.4

  1 141.1.123.2 28 msec 29 msec 28 msec
  2 141.1.25.5 32 msec 28 msec 28 msec
  3 141.1.145.4 28 msec *  28 msec

Rack1R1#ping 204.12.1.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 204.12.1.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 60/60/61 ms

Rack1R1#telnet 204.12.1.4
Trying 204.12.1.4 ... Open


User Access Verification

Password:
Rack1R4>exit

[Connection to 204.12.1.4 closed by foreign host]
```

Appears to be good now…. oh, not quite – we forgot to source the pings and telnet operation properly!

```
Rack1R1#ping 204.12.1.4 source fastEthernet 0/0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 204.12.1.4, timeout is 2 seconds:
Packet sent with a source address of 192.10.1.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/59/60 ms

Rack1R1#telnet 204.12.1.4 /source-interface fastEthernet 0/0
Trying 204.12.1.4 ... Open


User Access Verification

Password:
Rack1R4>exit

[Connection to 204.12.1.4 closed by foreign host]
```

Well it seems to be good now!

Ticket 4

Analyze the Symptoms

Let's see what could cause the breaking in the end-to-end connectivity. Using our favorite divide-and-conquer approach we start by tracing the route from R6 to SW1 VLAN7:

```
Rack1R6#traceroute 141.1.7.7

Type escape sequence to abort.
Tracing the route to 141.1.7.7

  1  *  *  *
  2
```

Appears like we have no route to it, just like in the previous ticket! Let's try and see if we can reach SW1 at all:

```
Rack1R6#traceroute 141.1.37.7

Type escape sequence to abort.
Tracing the route to 141.1.37.7

  1 141.1.36.3 0 msec 0 msec 4 msec
  2  *
```

At least R6 can reach R3. We check our initial hypothesis, verifying if R6 has the route for VLAN7:

```
Rack1R6#show ip route 141.1.7.0
% Subnet not in table
```

Check if R3 has this route:

```
Rack1R3#sh ip route 141.1.7.0
% Subnet not in table
```

Nope. Thus we may assume there is some problem preventing RIP updates from SW1 from reaching R3. This could be anything from RIP filters down to multicast blocked in the switches.

Isolate the Issue

Starting from the network level, we try to see if there are any connectivity problems between R3 and SW1:

```
Rack1R3#ping 141.1.37.7

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 141.1.37.7, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

So the problem lies above. Let’s check if we have any obvious RIP misconfiguraitons:

```
Rack1R3#show ip protocols | beg \"rip\"
Routing Protocol is "rip"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Sending updates every 30 seconds, next due in 18 seconds
  Invalid after 180 seconds, hold down 180, flushed after 240
  Redistributing: ospf 1, rip
  Default version control: send version 2, receive version 2
    Interface             Send  Recv  Triggered RIP  Key-chain
    FastEthernet0/0       2     2
    FastEthernet0/1       2     2
  Automatic network summarization is not in effect
  Maximum path: 4
  Routing for Networks:
    141.1.0.0
  Passive Interface(s):
    Serial1/0
    Serial1/0.1
    Serial1/1
    Serial1/2
    Serial1/3
    Loopback0
    VoIP-Null0
  Routing Information Sources:
    Gateway         Distance      Last Update
    141.1.36.6           120      00:00:22
  Distance: (default is 120)
```

We can see that RIPv2 is enabled on both Ethernet interfaces; but the only gateway sending us routing information is R6. There are no routing filters configured, and VLAN37 interface is not passive for RIP. Thus we move to debugging RIP routing updates, trying to see if this could reveal any useful information:

```
Rack1R3#debug interface fastEthernet 0/0
Condition 1 set

Rack1R3#debug ip rip
```

```
RIP protocol debugging is on
```

Notice the conditional debugging statement, which limits the debugging output only to the prefixes received/sent via VLAN37 interface.

```
RIP: sending v2 update to 224.0.0.9 via FastEthernet0/0 (141.1.37.3)
RIP: build update entries
        54.1.1.0/24 via 0.0.0.0, metric 2, tag 0
        141.1.0.0/24 via 0.0.0.0, metric 1, tag 0
        141.1.6.0/24 via 0.0.0.0, metric 2, tag 0
        141.1.8.0/24 via 0.0.0.0, metric 1, tag 0
        141.1.25.0/24 via 0.0.0.0, metric 1, tag 0
        141.1.36.0/24 via 0.0.0.0, metric 1, tag 0
        141.1.45.0/24 via 0.0.0.0, metric 1, tag 0
        141.1.54.0/24 via 0.0.0.0, metric 1, tag 0
        141.1.88.0/24 via 0.0.0.0, metric 1, tag 0
        141.1.123.0/24 via 0.0.0.0, metric 1, tag 0
        141.1.145.0/24 via 0.0.0.0, metric 1, tag 0
        141.1.255.0/24 via 0.0.0.0, metric 1, tag 0
        150.1.2.2/32 via 0.0.0.0, metric 1, tag 0
        150.1.3.0/24 via 0.0.0.0, metric 1, tag 0
        150.1.4.0/24 via 0.0.0.0, metric 1, tag 0
        150.1.4.4/32 via 0.0.0.0, metric 1, tag 0
        150.1.5.5/32 via 0.0.0.0, metric 1, tag 0
        150.1.6.0/24 via 0.0.0.0, metric 2, tag 0
        150.1.8.0/24 via 0.0.0.0, metric 1, tag 0
        204.12.1.0/24 via 0.0.0.0, metric 1, tag 0
Rack1R3#
RIP: ignored v2 packet from 141.1.37.7 (invalid authentication)
```

So what we can is that R3 sending RIP updates to SW1, but complains on invalid authentication when receiving the updates from SW1.

We found the issue, but now we have the problem recovering from it. It’s not possible to access the other router, so we have to figure out the solution using just the information we can obtain at R3.

Oh, and remember that if you used the “undebug all command” it does NOT automatically disable the debugging condition. You need to disable the condition using the command “undebug condition”, for example

```
Rack1R3#undebug interface fastEthernet 0/0
This condition is the last interface condition set.
Removing all conditions may cause a flood of debugging
messages to result, unless specific debugging flags
are first removed.

Proceed with removal? [yes/no]: yes
Condition 1 has been removed

Rack1R3#show debugging condition

% No conditions found
```

```
Rack1R3#
```

If you were to leave the condition in place, it may affect your further debugging commands enabled in the same router.

Fix the Issue

We may only assume that the other side uses plain-text authentication. Otherwise, it’s impossible to recover the password without actually accessing the other router. In order to recover the password, we could use some of the “hidden” debugging commands, which dump the packet contents. We create an access-list to intercept just the RIP update packets and enable the debugging command:

```
R3:
access-list 100 permit udp host 141.1.37.7 any eq 520

Rack1R3#debug ip packet detail 100 dump

IP packet debugging is on (detailed) (dump) for access list 100
Rack1R3#s
IP: s=141.1.37.7 (FastEthernet0/0), d=224.0.0.9, len 112, rcvd 2
    UDP src=520, dst=520
07C3CAC0:                        0100 5E000009            ..^...
07C3CAD0: 001AE2C8 3EC10800 45C00070 00000000  ..bH>A..E@.p....
07C3CAE0: 011126AC 8D012507 E0000009 02080208  ..&,..%.`.......
07C3CAF0: 005C14EC 02020000 FFFF0002 43495343  .\.l........CISC
07C3CB00: 4F343332 31200000 00000000 00020000  O4321 ..........
07C3CB10: 8D010700 FFFFFF00 00000000 00000001  ................
07C3CB20: 00020000 8D014D00 FFFFFF00 00000000  ......M.........
07C3CB30: 00000001 00020000 96010700 FFFFFF00  ................
07C3CB40: 00000000 00000001                    ........
```

Even if you’re not familiar with RIPv2 packet format you can quickly spot the password in the packet body: “CISCO4321”. So we go ahead and configure the key chain:

```
R3:
key chain RIP
 key 1
   key-string CISCO4321
!
interface FastEthernet 0/0
 ip rip authentication key-chain RIP
```

Wait some time to receive RIP updates from SW1 and check the RIP process status:

```
Rack1R3#sh ip protocols | beg \"rip\"
Routing Protocol is "rip"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Sending updates every 30 seconds, next due in 6 seconds
  Invalid after 180 seconds, hold down 180, flushed after 240
  Redistributing: ospf 1, rip
  Default version control: send version 2, receive version 2
    Interface             Send  Recv  Triggered RIP  Key-chain
    FastEthernet0/0       2     2                    RIP
    FastEthernet0/1       2     2
  Automatic network summarization is not in effect
  Maximum path: 4
  Routing for Networks:
    141.1.0.0
  Passive Interface(s):
    Serial1/0
    Serial1/0.1
    Serial1/1
    Serial1/2
    Serial1/3
    Loopback0
    VoIP-Null0
  Routing Information Sources:
    Gateway         Distance      Last Update
    141.1.36.6           120      00:00:07
  Distance: (default is 120)
```

Key chain has been configured but there is still just one route information source! Means there is something wrong with our configuration. Back to RIP updates debugging:

```
Rack1R3#debug ip rip
RIP protocol debugging is on
Rack1R3#
RIP: sending v2 update to 224.0.0.9 via FastEthernet0/0 (141.1.37.3)
RIP: build update entries
        54.1.1.0/24 via 0.0.0.0, metric 2, tag 0
        141.1.0.0/24 via 0.0.0.0, metric 1, tag 0
        141.1.6.0/24 via 0.0.0.0, metric 2, tag 0
        141.1.8.0/24 via 0.0.0.0, metric 1, tag 0
        141.1.25.0/24 via 0.0.0.0, metric 1, tag 0
        141.1.36.0/24 via 0.0.0.0, metric 1, tag 0
        141.1.45.0/24 via 0.0.0.0, metric 1, tag 0
        141.1.54.0/24 via 0.0.0.0, metric 1, tag 0
        141.1.88.0/24 via 0.0.0.0, metric 1, tag 0
        141.1.123.0/24 via 0.0.0.0, metric 1, tag 0
        141.1.145.0/24 via 0.0.0.0, metric 1, tag 0
        141.1.255.0/24 via 0.0.0.0, metric 1, tag 0
        150.1.2.2/32 via 0.0.0.0, metric 1, tag 0
        150.1.3.0/24 via 0.0.0.0, metric 1, tag 0
        150.1.4.0/24 via 0.0.0.0, metric 1, tag 0
        150.1.4.4/32 via 0.0.0.0, metric 1, tag 0
        150.1.5.5/32 via 0.0.0.0, metric 1, tag 0
        150.1.6.0/24 via 0.0.0.0, metric 2, tag 0
        150.1.8.0/24 via 0.0.0.0, metric 1, tag 0
        204.12.1.0/24 via 0.0.0.0, metric 1, tag 0
Rack1R3#
RIP: received packet with text authentication CISCO4321
RIP: ignored v2 packet from 141.1.37.7 (invalid authentication)
```

This is odd, now the debugging shows the clear-text password. By the way, this is another way to learn it without using the packet dumps. But the local router still says authentication is invalid! Let's get back to the packet dump and try looking at the password in details. Maybe the is some hidden character we're missing?

```
IP packet debugging is on (detailed) (dump) for access list 100
IP: s=141.1.37.7 (FastEthernet0/0), d=224.0.0.9, len 112, rcvd 2
    UDP src=520, dst=520
07C3CAC0:                          0100 5E000009            ..^...
07C3CAD0: 001AE2C8 3EC10800 45C00070 00000000  ..bH>A..E@.p....
07C3CAE0: 011126AC 8D012507 E0000009 02080208  ..&,..%.`.......
07C3CAF0: 005C14EC 02020000 FFFF0002 43495343  .\.l........CISC
07C3CB00: 4F343332 31200000 00000000 00020000  O4321 ..........
07C3CB10: 8D010700 FFFFFF00 00000000 00000001  ................
```

Look at the hex-dump corresponding to the password. Per the convenient rules, the strings of text are usually ASCIIZ, that is, terminated with zero. Looking at the final bytes of the password string we see “31|20|00” where 0x31 is “1” and 0x20 is the… space character! So there is a space character in the password, which we cannot see using convenient debugging output. That's a mean way of configuring authentication. Now we change our key chain configuration:

```
R3:
!
! Space character at the end of the password!
!
key chain RIP
  key 1
   key-string CISCO4321
```

Now we check the RIP protocol status again:

```
Rack1R3#show ip protocols | beg \"rip\"
Routing Protocol is "rip"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Sending updates every 30 seconds, next due in 13 seconds
  Invalid after 180 seconds, hold down 180, flushed after 240
  Redistributing: ospf 1, rip
  Default version control: send version 2, receive version 2
    Interface             Send  Recv  Triggered RIP  Key-chain
    FastEthernet0/0       2     2                    RIP
    FastEthernet0/1       2     2
  Automatic network summarization is not in effect
  Maximum path: 4
  Routing for Networks:
    141.1.0.0
  Passive Interface(s):
    Serial1/0
    Serial1/0.1
    Serial1/1
    Serial1/2
    Serial1/3
    Loopback0
    VoIP-Null0
  Routing Information Sources:
    Gateway         Distance      Last Update
    141.1.37.7           120      00:00:08
    141.1.36.6           120      00:00:26
  Distance: (default is 120)
```

And we finally see SW1 as the route information source!

Verify

Now let's make sure R6 can reach VLAN7 and that R6 actually passes RIP updates to BB1.

```
Rack1R6#traceroute 141.1.7.7

Type escape sequence to abort.
Tracing the route to 141.1.7.7

  1 141.1.36.3 4 msec 0 msec 0 msec
  2 141.1.37.7 4 msec *  0 msec
```

And just to confirm the updates go to BB1:

```
Rack1R6#debug interface serial 0/0
Condition 1 set

Rack1R6#debug ip rip
RIP protocol debugging is on

RIP: sending v2 update to 224.0.0.9 via Serial0/0 (54.1.1.6)
RIP: build update entries
        54.1.1.0/24 via 0.0.0.0, metric 1, tag 0
        141.1.0.0/24 via 0.0.0.0, metric 2, tag 0
        141.1.6.0/24 via 0.0.0.0, metric 1, tag 0
        141.1.7.0/24 via 0.0.0.0, metric 3, tag 0
        141.1.8.0/24 via 0.0.0.0, metric 2, tag 0
        141.1.25.0/24 via 0.0.0.0, metric 2, tag 0
        141.1.36.0/24 via 0.0.0.0, metric 1, tag 0
        141.1.37.0/24 via 0.0.0.0, metric 2, tag 0
        141.1.45.0/24 via 0.0.0.0, metric 2, tag 0
        141.1.54.0/24 via 0.0.0.0, metric 2, tag 0
        141.1.77.0/24 via 0.0.0.0, metric 3, tag 0
        141.1.88.0/24 via 0.0.0.0, metric 2, tag 0
        141.1.123.0/24 via 0.0.0.0, metric 2, tag 0
        141.1.145.0/24 via 0.0.0.0, metric 2, tag 0
        141.1.255.0/24 via 0.0.0.0, metric 2, tag 0
        150.1.2.2/32 via 0.0.0.0, metric 2, tag 0
        150.1.3.0/24 via 0.0.0.0, metric 2, tag 0
        150.1.4.4/32 via 0.0.0.0, metric 2, tag 0
        150.1.5.5/32 via 0.0.0.0, metric 2, tag 0
        150.1.6.0/24 via 0.0.0.0, metric 1, tag 0
        150.1.7.0/24 via 0.0.0.0, metric 3, tag 0
        150.1.8.0/24 via 0.0.0.0, metric 2, tag 0
        204.12.1.0/24 via 0.0.0.0, metric 2, tag 0
```

And make sure R6 receives the routes from BB1 as well – just in case!

```
Rack1R6#sh ip route rip | inc Serial0/0
R    212.18.1.0/24 [120/1] via 54.1.1.254, 00:00:05, Serial0/0
R    212.18.0.0/24 [120/10] via 54.1.1.254, 00:00:05, Serial0/0
R    212.18.3.0/24 [120/1] via 54.1.1.254, 00:00:05, Serial0/0
R    212.18.2.0/24 [120/10] via 54.1.1.254, 00:00:05, Serial0/0
```

OK this should be enough to be sure the guys behind BB1 can now reach VLAN7 and VLAN77.

Ticket 5

Analyze the Symptoms

Let's see if we can replicate this problem by tracing to R5.

```
Rack1R3#traceroute 150.1.5.5

Type escape sequence to abort.
Tracing the route to 150.1.5.5

  1 141.1.123.2 28 msec 32 msec 28 msec
  2  *  *  *
  3  *  *  *
  4  *  *  *
  5  *  *  *
  6  *  *  *
```

OK, so we do have a problem tracing to R5. Let's see if the same happens when tracing from any other router:

```
Rack1SW2#traceroute 150.1.5.5

Type escape sequence to abort.
Tracing the route to 150.1.5.5

  1 141.1.0.2 0 msec *  0 msec
  2  *  *  *
  3
Rack1SW2#
```

Finally let's see if we can trace across R5:

```
Rack1R3#traceroute 150.1.4.4

Type escape sequence to abort.
Tracing the route to R4 (150.1.4.4)

  1 141.1.123.2 28 msec 32 msec 28 msec
  2 141.1.25.5 32 msec 28 msec 28 msec
  3 141.1.25.5 32 msec 32 msec 32 msec
  4 141.1.54.4 60 msec *  56 msec

Rack1R4#traceroute 150.1.3.3

Type escape sequence to abort.
Tracing the route to 150.1.3.3

  1 141.1.54.5 28 msec 32 msec 20 msec
  2 141.1.25.2 16 msec 32 msec 32 msec
  3 141.1.123.3 48 msec *  56 msec
```

So that works. What this means is that there probably no filtering on the way to R5, but there is probably something with R5's configuration.

Isolate the Issue

Before isolating the issue, we need to recall how **`traceroute`** works. The general idea is sending UDP packets in batches of 3 packets towards the ultimate destination, with increasing TTL starting at 1. The destination ports are set to the range that is most likely not used by any application, and every next packet has destination port incremented by one. When a router on the path to the destination catches a packet with TTL or 1, it drops it and generates an ICMP time-exceeded message. This allows for identifying nodes on the path to the destination as they send the ICMP messages off their local interfaces.

The ultimate node should send a different ICMP message type when dropping these packets – ICMP port unreachable. This will identify the ultimate destination based on the source address. The number of hops could always be calculated based on the port-number found in the UDP packet, since ICMP error messages include the header in the payload.

Thus we see two major components of a typical **`traceroute`** utility: probes (UDP packets sent to a reserved UDP range) and ICMP replies, which are Time-Exceeded for transit nodes and Port-Unreachable for ultimate destinations.

So we need to find out what exactly happens at R5: do the UDP packets get silently dropped or does the ICMP messages are not being sent. To figure that, we use our best "low-level" tool for IP packet debugging – **`debug ip packet detail`**. We create a special access-list to catch the UDP probes. We also enable the command **`debug ip icmp`** to trace the response ICMP packets from R3:

```
R5:
access-list 100 permit udp any any

Rack1R5#debug ip packet detail 100
Rack1R5#debug ip icmp
```

We'll be logging to the internal buffer while tracing off R3:

```
R5:
logging buffered debugging
logging buffered 65000
```

Now we can get back to R3 and start tracing to R5:

```
Rack1R3#traceroute 150.1.5.5

Type escape sequence to abort.
Tracing the route to 150.1.5.5
```

```
  1 141.1.123.2 28 msec 28 msec 32 msec
  2  *  *  *
  3  *  *  *
  4  *  *  *
  5  *  *  *
  6  *  *  *
  7  *  *  *
```

```
R5:
IP: tableid=0, s=141.1.123.3 (Tunnel0), d=150.1.5.5 (Loopback0), routed
via RIB
IP: s=141.1.123.3 (Tunnel0), d=150.1.5.5, len 28, rcvd 4
    UDP src=49394, dst=33486
ICMP: dst (150.1.5.5) port unreachable sent to 141.1.123.3
```

From the above debugging output we can see that R5 indeed sends back the ICMP port unreachable packet. We just need to make sure why it's not reaching R3. The first thing would be putting an logging access-list in R2 and seeing if any ICMP unreachable get there:

```
R2:
ip access-list extended LOG
 permit icmp any any unreachable log
 permit ip any any
!
interface FastEthernet0/0
 ip access-group LOG in
```

Generate the traceroute from R3 again:

```
Rack1R3#traceroute 150.1.5.5

Type escape sequence to abort.
Tracing the route to 150.1.5.5

  1 141.1.123.2 28 msec 32 msec 28 msec
  2  *

Rack1R2#show ip access-lists LOG
Extended IP access list LOG
    10 permit icmp any any unreachable log
    20 permit ip any any (98 matches)
```

No matches for ICMP unreachables. This means they are being dropped by either R5 or any transit switch between R2 and R5. However, the only switch under our control is SW2. Let's check if there are any access-groups applied there, or VLAN filters configured.

```
Rack1SW2#show ip interface | inc access
  Outgoing access list is not set
  Inbound  access list is not set
  IP access violation accounting is disabled
  Outgoing access list is not set
  Inbound  access list is not set
  IP access violation accounting is disabled
  Outgoing access list is not set
  Inbound  access list is not set
  IP access violation accounting is disabled
  Outgoing access list is not set
  Inbound  access list is not set
  IP access violation accounting is disabled
  Inbound  access list is not set
  Inbound  access list is not set
  Inbound  access list is not set
  Inbound  access list is not set
  Inbound  access list is not set
  Inbound  access list is not set
  Inbound  access list is not set
  Inbound  access list is not set
  Inbound  access list is not set
  Inbound  access list is not set
  Inbound  access list is not set
  Inbound  access list is not set
  Inbound  access list is not set
  Inbound  access list is not set
  Inbound  access list is not set
  Inbound  access list is not set
  Inbound  access list is not set
  Inbound  access list is not set
  Inbound  access list is not set
  Inbound  access list is not set
  Inbound  access list is not set
  Inbound  access list is not set
  Inbound  access list is not set
  Inbound  access list is not set
  Inbound  access list is not set
  Outgoing access list is not set
  Inbound  access list is not set
  IP access violation accounting is disabled

Rack1SW2#show vlan filter

Rack1SW2#
```

There are other ways to filter traffic, like using policy-maps and such. Make sure they are not applied as well:

```
Rack1SW2#show policy-map interface

Rack1SW2#
```

There are still more ways to do traffic filtering, but they don’t allow for granularity needed to filter just ICMP unreachables. So our next guess is that some sort of filtering is implemented in R5. What could that be? Outbound access-lists do not affect router-generated traffic. QoS configuration does, so we check for policy-maps applied to the interfaces:

```
Rack1R5#show policy-map interface
 FastEthernet0/1

  Service-policy output: QoS

    Class-map: SMTP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: protocol smtp
      Queueing
        Output Queue: Conversation 265
        Bandwidth 1500 (kbps)Max Threshold 64 (packets)
        (pkts matched/bytes matched) 0/0
        (depth/total drops/no-buffer drops) 0/0/0

    Class-map: ABOVE_1250_BYTES (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: packet length min 1251
      police:
          cir 2500000 bps, bc 78125 bytes
        conformed 0 packets, 0 bytes; actions:
          transmit
        exceeded 0 packets, 0 bytes; actions:
          drop
        conformed 0 bps, exceed 0 bps

    Class-map: class-default (match-any)
      11663 packets, 1150252 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
```

So there is a policy here. However, it seems to affect only SMTP traffic and large packets. Plus, the first class just had the CBWFQ weight applied and the second only policies the traffic. And there are not drops in the statistics. So our next guess would probably be CBAC filtering configured to inspect local traffic. Check if we have CBAC enabled:

```
Rack1R5#show ip inspect all

Rack1R5#
```

Nothing… so the last thing is probably local policy routing. This type of filtering applies to all locally originated traffic. Let's see if we have anything configured.

```
Rack1R5#show ip local policy
Local policy routing is enabled, using route map LOCAL
route-map LOCAL, permit, sequence 10
  Match clauses:
    ip address (access-lists): UNREACH
  Set clauses:
    interface Null0
  Policy routing matches: 250 packets, 17707 bytes
Rack1R5#show ip acce UNREACH
```

```
Extended IP access list UNREACH
    10 permit icmp any any unreachable (250 matches)
```

We hit the bulleye, the local policy is exactly the thing that affects our ICMP messages flow. It appears we can easily disable this and let our ICMP flow.

Conclusion: Exhaustive and systematic search always give good results!

Fix the Issue

Since the policy appears to be just an “evil trick” we just remove it:

R3:
```
no ip local-policy LOCAL
```

Verify

Now we trace to R5 to make sure everything works:

```
Rack1R3#traceroute 150.1.5.5

Type escape sequence to abort.
Tracing the route to 150.1.5.5

  1 141.1.123.2 28 msec 32 msec 28 msec
  2 141.1.25.5 33 msec 32 msec 32 msec
Rack1R3#

Rack1R4#traceroute 150.1.5.5

Type escape sequence to abort.
Tracing the route to 150.1.5.5

  1 141.1.54.5 28 msec 20 msec
    141.1.45.5 12 msec
```

And it seems to be working just fine!

Ticket 6

Analyze the Symptoms

The good thing is that we already know it's the NAT issue. So we don't have to narrow our problem scope, it is limited to R4. Let's check the NAT configuration in R4 to get the initial feeling of the problem:

```
Rack1R4#show ip nat statistics
Total active translations: 2 (2 static, 0 dynamic; 0 extended)
Outside interfaces:
  FastEthernet0/1
Inside interfaces:
  FastEthernet0/0
Hits: 0  Misses: 0
CEF Translated packets: 0, CEF Punted packets: 1478
Expired translations: 0
Dynamic mappings:
Queued Packets: 0

Rack1R4#show ip nat translations
Pro Inside global      Inside local       Outside local      Outside global
--- ---                ---                141.1.145.254      204.12.1.254
--- 204.12.1.100       141.1.145.100      ---                ---
```

The interface connected to BB3 is the outside and the one connected to VLAN45 is inside. Makes sense so far. The static mappings look alright too: 204.12.1.254 maps bidirectionally to 141.1.145.254 and 141.1.145.100 maps bidirectionally to 204.12.1.100. So what might be wrong in this configuration? In order to figure that out, we need to have a host on VLAN45. Let's use SW2 and configure an SVI interface there. Notice that in real life you will probably have to use a different IP and change the NAT mappings accordingly.

```
SW2:
interface Vlan 45
 ip address 141.1.145.100 255.255.255.0
!
ip route 204.12.1.254 255.255.255.255 141.1.145.4
```

Isolate the Issue

Now we have a testbed in place and may continue with NAT debugging. What we need to debug is the NAT events and the IP packet switching.

```
R4:
access-list 100 permit icmp any any
logging buffered debugging

Rack1R4#debug ip packet detail 100
IP packet debugging is on (detailed) for access list 100

Rack1R4#debug ip nat detailed
IP NAT detailed debugging is on
```

```
Rack1SW2#ping 141.1.145.254

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 141.1.145.254, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/5/9 ms
```

Hmm.. so is there a problem?! Let's look at out debugging output:

```
IP: tableid=0, s=141.1.145.100 (FastEthernet0/0), d=141.1.145.254
(FastEthernet0/0), routed via RIB

IP: s=141.1.145.100 (FastEthernet0/0), d=141.1.145.254
(FastEthernet0/0), len 100, rcvd 3
    ICMP type=8, code=0

IP: tableid=0, s=141.1.145.254 (local), d=141.1.145.100
(FastEthernet0/0), routed via FIB
IP: s=141.1.145.254 (local), d=141.1.145.100 (FastEthernet0/0), len
100, sending
    ICMP type=0, code=0

IP: tableid=0, s=141.1.145.100 (FastEthernet0/0), d=141.1.145.254
(FastEthernet0/0), routed via RIB
IP: s=141.1.145.100 (FastEthernet0/0), d=141.1.145.254
(FastEthernet0/0), len 100, rcvd 3
    ICMP type=8, code=0

IP: tableid=0, s=141.1.145.254 (local), d=141.1.145.100
(FastEthernet0/0), routed via FIB
IP: s=141.1.145.254 (local), d=141.1.145.100 (FastEthernet0/0), len
100, sending
<snip>
```

There is not mentioning on NAT in the output. Plus we can see ICMP replies to 141.X.141.100 sent from the address 141.X.145.254. This probably means that the local router responds to the ICMP messages itself! To confirm that, use the following debugging command:

```
Rack1R4#debug ip icmp
ICMP packet debugging is on

Rack1R4#clear logging
Clear logging buffer [confirm]
```

And ping from SW2 again:

```
R4:
IP: tableid=0, s=141.1.145.100 (FastEthernet0/0), d=141.1.145.254
(FastEthernet0/0), routed via RIB
IP: s=141.1.145.100 (FastEthernet0/0), d=141.1.145.254
(FastEthernet0/0), len 100, rcvd 3
    ICMP type=8, code=0
```

```
ICMP: echo reply sent, src 141.1.145.254, dst 141.1.145.100
IP: tableid=0, s=141.1.145.254 (local), d=141.1.145.100
(FastEthernet0/0), routed via FIB
IP: s=141.1.145.254 (local), d=141.1.145.100 (FastEthernet0/0), len
100, sending
    ICMP type=0, code=0
```

Yes, this is the local router responding to the ICMP echo requests! So why does that happen? The answer is that IOS creates a local alias in the system and install an ARP entry for the mapped IP address. Look at the output below and notice that the MAC address of the local interface is the same as the MAC for the ARP entry of the IP address 141.X.145.254.

```
Rack1R4#show ip aliases
Address Type             IP Address      Port
Interface                141.1.145.4
Interface                150.1.4.4
Interface                141.1.45.4
Interface                141.1.54.4
Interface                204.12.1.4
Dynamic                  141.1.145.254

Rack1R4#show ip arp 141.1.145.254
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  141.1.145.254           -   000f.34df.74a0  ARPA
FastEthernet0/0

Rack1R4#show int fa 0/0
FastEthernet0/0 is up, line protocol is up
  Hardware is AmdFE, address is 000f.34df.74a0 (bia 000f.34df.74a0)
```

To make sure the connectivity is not really working, try out the following command:

```
Rack1SW2#telnet 141.1.145.254
Trying 141.1.145.254 ...
% Connection refused by remote host
```

So the connection does not really reach the interface of BB3 and most likely get dropped by R4. What should be our next guess? Let’s recall how NAT works. When a packet hits the inside interface, it is first routed and than translated. This applies to both source and destination IP addresses. Thus in our case, when a packet targeted to 141.X.145.254 and sourced off 141.X.141.100 hits the inside interface, the router attempts to use the RIB and finds itself to be the target! Thus, the NAT translation rule to redirect IP 141.X.145.254 to 204.12.X.254 is never triggered.

Notice that in opposite direction, when a packet hits the outside interface, it is being un-translated first, and only then routed. Thus is BB3 would send a packet to 204.12.X.100 it will be rewritten to 141.X.145.100 and then routed. So how would we fix that for the inside direction? One obvious solution is using a static route to redirect packets to 141.X.145.254 to 204.12.X.254. You may think that static routing is now allowed, but if you look carefully into the Lab introduction section you will notice that there is not requirement disallowing the use of static routes! We should have probably noticed that before, when doing the initial lab analysis, but still it's not bad.

`Conclusion:` If you can ping something, it does not necessary means the thing works! Also, remember the order of operations!

Fix the Issue

We create a static route in R4:

```
R4:
ip route 141.1.145.254 255.255.255.255 204.12.1.254
```

And this should assist in NAT being applied correctly.

Verify

Now we ping from SW2 again, leaving NAT/IP packet debugging on in R4:

```
Rack1SW2#ping 141.1.145.254

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 141.1.145.254, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 8/10/16 ms
```

Let's see our logs. Below you can see that the packet from SW2 is being routed and than translated. "NAT: i" means the translation was performed on the inside interface. You can see the source AND destination address being modified. After that, the modified packet is routed to BB3.

```
IP: tableid=0, s=141.1.145.100 (FastEthernet0/0), d=141.1.145.254
(FastEthernet0/1), routed via RIB

NAT: i: icmp (141.1.145.100, 6) -> (141.1.145.254, 6) [124]
NAT: s=141.1.145.100->204.12.1.100, d=141.1.145.254 [124]
NAT: s=204.12.1.100, d=141.1.145.254->204.12.1.254 [124]

IP: s=204.12.1.100 (FastEthernet0/0), d=204.12.1.254 (FastEthernet0/1),
g=204.12.1.254, len 100, forward
    ICMP type=8, code=0
```

Now the returning packet is first untranslated and THEN routed. You can see both the source and destination addresses being changed by the NAT rules.

```
NAT*: o: icmp (204.12.1.254, 6) -> (204.12.1.100, 6) [125]
NAT*: s=204.12.1.254->141.1.145.254, d=204.12.1.100 [125]
NAT*: s=141.1.145.254, d=204.12.1.100->141.1.145.100 [125]
```

```
IP: tableid=0, s=141.1.145.100 (FastEthernet0/0), d=141.1.145.254
(FastEthernet0/1), routed via RIB
```

Now we can literally see how the NAT process works and that the server may now reach BB3 router.

Ticket 7

Analyze the Symptoms

We start by checking if R1 actually does not advertise any information to BB2. First, check BGP peering status:

```
Rack1R1#show ip bgp summary
BGP router identifier 150.1.1.1, local AS number 200
BGP table version is 4, main routing table version 4
12 network entries using 1404 bytes of memory
12 path entries using 624 bytes of memory
4/1 BGP path/bestpath attribute entries using 496 bytes of memory
3 BGP AS-PATH entries using 72 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 2596 total bytes of memory
BGP activity 12/0 prefixes, 12/0 paths, scan interval 60 secs

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
141.1.123.2     4   200      20      19        4    0    0 00:14:59
9
192.10.1.254    4   254     374     373        4    0    0 06:09:10
3
```

Appears like R1 receives routes both from R2 and BB2. Let's see if any of them get advertised to BB2:

```
Rack1R1#show ip bgp neighbors 192.10.1.254 advertised-routes

Total number of prefixes 0
```

Nope, let's see if we have the AS400 routes in the local table:

```
Rack1R1#show ip bgp regexp 400$
BGP table version is 4, local router ID is 150.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
* i150.1.4.0/24     141.1.0.5                0    100      0 300 400 i
```

OK so we got the route locally, but it is not marked as best. This is why it's not being advertised to BB2! Now we only has to find what might be causing the route lose it's status.

Isolate the Issue

To find the reason of the route not being marked as "best", check the prefix details in the BGP table:

```
Rack1R1#show ip bgp 150.1.4.0
BGP routing table entry for 150.1.4.0/24, version 0
Paths: (1 available, no best path)
  Not advertised to any peer
  300 400
    141.1.0.5 (metric 65) from 141.1.123.2 (150.1.2.2)
      Origin IGP, metric 0, localpref 100, valid, internal, not
synchronized
```

As we quickly find that the prefix is marked as "not synchronized". What this means is that there is not match in the IGP table for this iBGP prefix, and thus the BGP speaker does not announce it any further. Let's check the RIB:

```
Rack1R1#show ip route 150.1.4.0
Routing entry for 150.1.4.0/24
  Known via "ospf 1", distance 110, metric 1
  Tag 300, type extern 2, forward metric 65
  Last update from 141.1.123.2 on Serial0/0.1, 00:30:47 ago
  Routing Descriptor Blocks:
  * 141.1.123.2, from 141.1.2.2, 00:30:47 ago, via Serial0/0.1
      Route metric is 1, traffic share count is 1
      Route tag 300
```

Apparently, the prefix is in the IGP table. So what else may be causing this? There is one more problem that was introduced with RFC 1403 – BGP and OSPF interaction. The router-ID of the BGP process that injected the route into OSPF should match the router-ID of the OSPF process doing the redistribution. This was needed to ensure "sanity" and make sure the route source is the same router. However, this caused numerous issues for people unaware of this poorly documented feature and using BGP synchronization.

The source of the routing information is R2 in our case, since this is where the other BGP peer is. Let's go there and check the router-IDs:

```
Rack1R2#show ip ospf
 Routing Process "ospf 1" with ID 141.1.2.2
<snip>

Rack1R2#show ip bgp
BGP table version is 18, local router ID is 150.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
<snip>
```

So this is where the problem is – router-IDs are different.

Fix the Issue

We could change either OSPF or BGP router-ID. It makes sense to use the IP 150.X.2.2 for the router-ID as this the Loopback0 IP address. So we go ahead and reconfigure the OSPF process. Don't forget to reset it after the changes!

```
R2:
router ospf 1
 router-id 150.1.2.2

Rack1R2#clear ip ospf process
Reset ALL OSPF processes? [no]: yes
```

Not that older IOS releases would probably require you to reload the whole router in order for the router-ID changes to take effect.

Verify

Check the BGP table entry for this prefix again:

```
Rack1R1#show ip bgp 150.1.4.0
BGP routing table entry for 150.1.4.0/24, version 6
Paths: (1 available, best #1, table Default-IP-Routing-Table, RIB-
failure(17))
  Advertised to update-groups:
     1
  300 400
    141.1.0.5 (metric 65) from 141.1.123.2 (150.1.2.2)
      Origin IGP, metric 0, localpref 100, valid, internal,
synchronized, best
```

Now it is marked as synchronized. Make sure the prefix is advertised to BB2:

```
Rack1R1#show ip bgp neighbors 192.10.1.254 advertised-routes
BGP table version is 22, local router ID is 150.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
r>i28.119.16.0/24   141.1.36.6               0    100      0 100 54 i
r>i28.119.17.0/24   141.1.36.6               0    100      0 100 54 i
r>i114.0.0.0        141.1.36.6               0    100      0 100 54 i
r>i115.0.0.0        141.1.36.6               0    100      0 100 54 i
r>i116.0.0.0        141.1.36.6               0    100      0 100 54 i
r>i117.0.0.0        141.1.36.6               0    100      0 100 54 i
```

```
r>i118.0.0.0        141.1.36.6               0    100      0 100 54 i
r>i119.0.0.0        141.1.36.6               0    100      0 100 54 i
r>i150.1.4.0/24     141.1.0.5                0    100      0 300 400 i
```

Lastly, notice that the “r” sign is normal when using BGP synchronization. This is because the routes in the RIB (learned via IGP) are preferred over iBGP prefixes. Now trace the route to the AS400 prefix:

```
Rack1R1#traceroute 150.1.4.4 source fastEthernet 0/0

Type escape sequence to abort.
Tracing the route to 150.1.4.4

  1 141.1.123.2 32 msec 28 msec 28 msec
  2 141.1.25.5 28 msec 32 msec 28 msec
  3 141.1.25.5 29 msec 28 msec 32 msec
  4 141.1.45.4 56 msec
    141.1.54.4 64 msec *
```

To make sure there are no any path errors to prevent communications. Even though this is not a truly end-to-end test, it still helps ensuring end-to-end communications.

Ticket 8

Analyze the Symptoms

First, we classify this issue as related to the application configuration. The only process involved here is DHCP. There could be a bunch of issues:

1) DHCP relay cannot reach DHCP server, i.e. if the DHCP server IP is misconfigured or something blocks the communications.
2) DHCP server cannot respond back to DHCP relay, for example if it does not know the route back to the relay's IP address
3) DHCP settings are not configured properly in the server.

To investigate all issues we will "spoof" a DHCP client on VLAN45 and see if it can obtain an IP address. We are going to use SW2 for this purpose again:

```
SW2:
interface VLAN45
 ip address DHCP
```

Initially, we suspect all devices in the path for the DHCP failure, but mostly focus on R5 (the relay) and R3 (the server). It's time to use some debugging commands starting with the DHCP server.

Isolate the Issue

First we enable DHCP debugging in the server to check if it receives any DHCP queries. Also, we enable DHCP Option 82 related events debugging. Notice that we use the buffer to store logging messages:

```
Rack1R3#debug ip dhcp server events
Rack1R3#debug ip dhcp server packet
Rack1R3#debug ip dhcp server class
Rack1R3#sh debugging

DHCP server packet debugging is on.
DHCP server event debugging is on.

Rack1R3(config)#logging buffered debugging
Rack1R3#clear logging
Clear logging buffer [confirm]
```

Now shutdown and bring back the SVI in SW2 (we'll be using that continually during the troubleshooting process):

```
Rack1SW2(config)#interface vlan 45
Rack1SW2(config-if)#shutdown
Rack1SW2(config-if)#no shutdown
```

And check the debugging logs in R3:

```
Rack1R3#show logging
Syslog logging: enabled (11 messages dropped, 2 messages rate-limited,
                0 flushes, 0 overruns, xml disabled, filtering
disabled)
    Console logging: level debugging, 1143 messages logged, xml
disabled,
                     filtering disabled
    Monitor logging: level debugging, 0 messages logged, xml disabled,
                     filtering disabled
    Buffer logging: level debugging, 2 messages logged, xml disabled,
                    filtering disabled
    Logging Exception size (4096 bytes)
    Count and timestamp logging messages: disabled

No active filter modules.

    Trap logging: level informational, 114 message lines logged

Log Buffer (4096 bytes):

DHCPD: checking for expired leases.
```

Nothing there about DHCP requests – so they are not making it to the server. Let's go back to the relay and see how it's configured:

```
Rack1R5#show ip interface fastEthernet 0/0 | inc [Hh]elper
  Helper address is 141.1.123.1
```

Great, this is not R3's IP address at all. Let's go ahead and change it:

```
R5:
interface FastEthernet 0/0
 no ip helper-address 141.1.123.1
 ip helper-address 141.1.123.3
```

But now the server tells us that there is no Option 82 in the incoming packets!

```
R3:
HCPD: Sending notification of DISCOVER:
  DHCPD: htype 1 chaddr 0015.6348.0745
  DHCPD: circuit id 012e0000
DHCPD: DHCPDISCOVER received from client
0063.6973.636f.2d30.3031.352e.3633.3438.2e30.3734.352d.566c.3435
through relay 141.1.145.5.
DHCPD: Seeing if there is an internally specified pool class:
  DHCPD: htype 1 chaddr 0015.6348.0745
  DHCPD: circuit id 012e0000
DHCPD: input does not contain option 82
```

Per the requirements, the relay should be inserting Option 82 in the packets. Let's get back to R5 and check this. Here we have to break one of our rules – we'll have to use a **`show run`** command, as there is not much information you can get on DHCP relay using the normal show commands. So we check the interface configuration on R5:

```
R5:
interface FastEthernet0/0
 ip dhcp relay information option subscriber-id VLAN45
 ip address 141.1.145.5 255.255.255.0
 ip helper-address 141.1.123.3
 ip pim sparse-mode
 ip ospf network point-to-point
 ip ospf hello-interval 2
 ip ospf mtu-ignore
<snip>
```

OK, we can see that the interface is configured for the insertion of subscriber-id “VLAN45”. So the other thing that might be missing is the command that enables the insertion of the information option. Let’s forcefully enable it:

```
R5:
ip dhcp relay information option
```

And now back to R3 to see the debugging output:

```
R3:
DHCPD: Sending notification of DISCOVER:
  DHCPD: htype 1 chaddr 0015.6348.0745
  DHCPD: circuit id 012e0000
DHCPD: DHCPDISCOVER received from client
0063.6973.636f.2d30.3031.352e.3633.3438.2e30.3734.352d.566c.3435
through relay 141.1.145.5.
DHCPD: Seeing if there is an internally specified pool class:
  DHCPD: htype 1 chaddr 0015.6348.0745
  DHCPD: circuit id 012e0000
DHCPD: Class 'OPTION82' matched by default
DHCPD: Searching for a match to 'relay-information
020c020a00008d019105000000000606564c414e3435' in class OPTION82
```

Great, now we see that the system matches a pool and then matches the Option 92 strings against the configured class OPTION82. Since there is no address allocation, the match obviously does not succeed. This means that we need to go ahead and edit the class settings, assigning the option we just found.

Now we go in the configuration mode and set the relay-information value for the class OPTION82

```
R3:
no ip dhcp class OPTION82
ip dhcp class OPTION82
   relay agent information
      relay-information hex
020c020A00008D019105000000000606564c414e3435
```

```
!
ip dhcp pool VLAN45
   network 141.1.145.0 /24
   class OPTION82
     address range 141.1.145.100 141.1.145.100
```

Notice that we deleted the class prior to configuring it, in order to avoid entering multiple Option 82 strings. This also requires us reconfiguring the range associated with the option, which should be the single IP address to be allocated to SW2.

Now look at the debugging output in R3. The first fragment says we match the OPTION82 class, on the reception of DHCPDISCOVER message.

```
DHCPD: Sending notification of DISCOVER:
  DHCPD: htype 1 chaddr 0015.6348.0745
  DHCPD: circuit id 012e0000
DHCPD: DHCPDISCOVER received from client
0063.6973.636f.2d30.3031.352e.3633.3438.2e30.3734.352d.566c.3435
through relay 141.1.145.5.
DHCPD: Seeing if there is an internally specified pool class:
  DHCPD: htype 1 chaddr 0015.6348.0745
  DHCPD: circuit id 012e0000

DHCPD: Class 'OPTION82' matched by default
DHCPD: Searching for a match to 'relay-information
020c020a00008d0191050000000000606564c414e3435' in class OPTION82
DHCPD: input pattern 'relay-information
020c020a00008d0191050000000000606564c414e3435' matches class OPTION82
DHCPD: input matches class OPTION82
```

Now the IP address is being allocated to the client, and BOOTREPLY, BOOTREQUEST and DHCPACK messages are exchanged, finalizing the configuration phase.

```
DHCPD: Adding binding to radix tree (141.1.145.100)
DHCPD: Adding binding to hash tree
DHCPD: assigned IP address 141.1.145.100 to client
0063.6973.636f.2d30.3031.352e.3633.3438.2e30.3734.352d.566c.3435.

DHCPD: Sending DHCPOFFER to client
0063.6973.636f.2d30.3031.352e.3633.3438.2e30.3734.352d.566c.3435
(141.1.145.100).
DHCPD: unicasting BOOTREPLY for client 0015.6348.0745 to relay
141.1.145.5.

DHCPD: DHCPREQUEST received from client
0063.6973.636f.2d30.3031.352e.3633.3438.2e30.3734.352d.566c.3435.
DHCPD: Sending notification of ASSIGNMENT:
 DHCPD: address 141.1.145.100 mask 255.255.255.0
  DHCPD: htype 1 chaddr 0015.6348.0745
  DHCPD: lease time remaining (secs) = 86400
DHCPD: No default domain to append - abort update
```

```
DHCPD: Sending DHCPACK to client
0063.6973.636f.2d30.3031.352e.3633.3438.2e30.3734.352d.566c.3435
(141.1.145.100).
DHCPD: unicasting BOOTREPLY for client 0015.6348.0745 to relay
141.1.145.5.
```

Conclusion: Troubleshooting application could be much harder than a simple network issue, as there are so many factors involved! Plus, it never hurts to use the **show run** is you know what exactly you are looking for.

Fix the Issue

The fixing process took quite a while. We are going to output the new configuration applied to the devices here:

```
R3:
no ip dhcp class OPTION82
ip dhcp class OPTION82
   relay agent information
      relay-information hex
020c020A00008D019105000000000606564c414e3435
!
ip dhcp pool VLAN45
   network 141.1.145.0 /24
   class OPTION82
     address range 141.1.145.100 141.1.145.100

R5:
ip dhcp relay information option
!
interface FastEthernet 0/0
 no ip helper-address 141.1.123.1
 ip helper-address 141.1.123.3
```

Verify

First, we make sure that the DHCP binding is now in R3's database:

```
Rack1R3#show ip dhcp binding
Bindings from all pools not associated with VRF:
IP address          Client-ID/              Lease expiration
Type
                    Hardware address/
                    User name
141.1.145.100       0063.6973.636f.2d30.    Jul 15 2009 07:44 PM
Automatic
                    3031.352e.3633.3438.
                    2e30.3734.352d.566c.
                    3435
Rack1R3#
```

Now check SW2 and make sure it has got the new IP address assigned via DHCP:

```
SW2:
Interface Vlan45 assigned DHCP address 141.1.145.100, mask
255.255.255.0

Rack1SW2#show ip interface vlan 45
Vlan45 is up, line protocol is up
  Internet address is 141.1.145.100/24
  Broadcast address is 255.255.255.255
```

Ticket 9

Analyze the Symptoms

As we mentioned previously during the initial analysis, the requirement to adjust only the IPv6 settings allows us to assume that there are no physical link issues. Thus we may go ahead and check where L3 connectivity breaks. Look at R1's IPv6 routing table:

```
Rack1R1#show ipv6 route
IPv6 Routing Table - 6 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS
summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF
ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
C   2001:141:1:12::/64 [0/0]
     via ::, Serial0/0.1
L   2001:141:1:12::1/128 [0/0]
     via ::, Serial0/0.1
OI  2001:141:1:25::/64 [110/65]
     via FE80::2, Serial0/0.1
OI  2001:150:1::/60 [110/65]
     via FE80::2, Serial0/0.1
L   FE80::/10 [0/0]
     via ::, Null0
L   FF00::/8 [0/0]
     via ::, Null0
```

OK, so there is a route that encompasses the Loopback100 subnets of SW2 and R5. We do traceroutes to the respective Loopbacks addresses:

```
Rack1R1#traceroute 2001:150:1:8::8

Type escape sequence to abort.
Tracing the route to 2001:150:1:8::8

  1 2001:141:1:12::2 40 msec 40 msec 40 msec
  2 2001:141:1:25::A 44 msec 44 msec 44 msec

Rack1R1#traceroute 2001:150:1:5::5
```

```
Type escape sequence to abort.
Tracing the route to 2001:150:1:5::5

  1 2001:141:1:12::2 44 msec 44 msec 44 msec
  2 2001:141:1:12::2 !H  !H  !H
```

That clears a lot. First, we can see that R2 is the host doing the summarization – it's the only ABR plus it reports host-unreachables for R5's Loopback100. Thus, there must be some issue between R2 and R5 that prevents R2 from learning the Loopback100 subnet of R5. This is our working hypothesis for now.

Isolate the Issue

Now we start working with R2. First we check if there is OSPF adjacency with R5:

```
Rack1R2#show ipv6 ospf neighbor

Neighbor ID     Pri   State           Dead Time   Interface ID
Interface
150.1.1.1         1   FULL/  -        00:00:32    10
Serial0/0
150.1.8.8         1   FULL/DR         00:00:38    2326
FastEthernet0/0
150.1.5.5         1   INIT/DROTHER    00:00:39    5
FastEthernet0/0
```

OK, the adjacency is stuck in the INIT state. This means the two routers do not even recognize each other in their HELLO packets. Let's use the IPv6 OSPF adjacency debugging command:

```
Rack1R2#debug ipv6 ospf adj
OSPFv3 adjacency events debugging is on

%OSPFv3-5-ADJCHG: Process 1, Nbr 150.1.5.5 on FastEthernet0/0 from
LOADING to FULL, Loading Done

OSPFv3: Cannot see ourself in hello from 150.1.5.5 on FastEthernet0/0,
state INIT

OSPFv3: Neighbor change Event on interface FastEthernet0/0
OSPFv3: DR/BDR election on FastEthernet0/0
OSPFv3: Elect BDR 150.1.2.2
OSPFv3: Elect DR 150.1.8.8
       DR: 150.1.8.8 (Id)   BDR: 150.1.2.2 (Id)

OSPFv3: Rcv DBD from 150.1.5.5 on FastEthernet0/0 seq 0xE74 opt 0x0013
flag 0x2 len 268  mtu 1500 state INIT
OSPFv3: 2 Way Communication to 150.1.5.5 on FastEthernet0/0, state 2WAY
OSPFv3: Neighbor change Event on interface FastEthernet0/0
OSPFv3: DR/BDR election on FastEthernet0/0
OSPFv3: Elect BDR 150.1.2.2
OSPFv3: Elect DR 150.1.8.8
```

```
       DR: 150.1.8.8 (Id)   BDR: 150.1.2.2 (Id)
OSPFv3: Send DBD to 150.1.5.5 on FastEthernet0/0 seq 0xA34 opt 0x0013
flag 0x7 len 28
OSPFv3: Unrecognized dbd for EXSTART
OSPFv3: Rcv DBD from 150.1.5.5 on FastEthernet0/0 seq 0xE75 opt 0x0013
flag 0x0 len 28  mtu 1500 state EXSTART
OSPFv3: Unrecognized dbd for EXSTART
OSPFv3: Rcv DBD from 150.1.5.5 on FastEthernet0/0 seq 0xE76 opt 0x0013
flag 0x0 len 28  mtu 1500 state EXSTART
OSPFv3: Unrecognized dbd for EXSTART
OSPFv3: Rcv DBD from 150.1.5.5 on FastEthernet0/0 seq 0x12F2 opt 0x0013
flag 0x7 len 28  mtu 1500 state EXSTART
OSPFv3: NBR Negotiation Done. We are the SLAVE
OSPFv3: Send DBD to 150.1.5.5 on FastEthernet0/0 seq 0x12F2 opt 0x0013
flag 0x2 len 268
OSPFv3: Rcv DBD from 150.1.5.5 on FastEthernet0/0 seq 0x2ED opt 0x0013
flag 0x7 len 28  mtu 1500 state EXCHANGE
OSPFv3: EXCHANGE - OPTIONS/INIT not match
OSPFv3: Bad seq received from 150.1.5.5 on FastEthernet0/0
OSPFv3: Send DBD to 150.1.5.5 on FastEthernet0/0 seq 0x1220 opt 0x0013
flag 0x7 len 28
```

As we can see, R2 is constantly changing through all OSPF states with R5. The reason it drops the adjacency and starts over appears to be the fact that it sees some mismatch in OSPF options or other OSPF header fields. Plus there is a mismatch in DBD packets received from the peer. Let's go back to R5 and see how the problem looks like there. When you enable logging you may notice the following messages:

```
%OSPFv3-5-ADJCHG: Process 1, Nbr 150.1.2.2 on FastEthernet0/1 from
LOADING to FULL, Loading Done
%OSPFv3-5-ADJCHG: Process 1, Nbr 150.1.8.8 on FastEthernet0/1 from
LOADING to FULL, Loading Done
%OSPFv3-5-ADJCHG: Process 1, Nbr 150.1.2.2 on FastEthernet0/1 from
LOADING to FULL, Loading Done
```

Which means R5 keeps synchronizing with SW2 and R2 in sequence. If we look at the debugging output above, we'll see that SW2 is elected as DR and R2 as the BDR.

```
OSPFv3: DR/BDR election on FastEthernet0/0
OSPFv3: Elect BDR 150.1.2.2
OSPFv3: Elect DR 150.1.8.8
```

On a broadcast network, R5 should have been synchronized with both of the routers simultaneously, sending packets to 224.0.0.6. However we see the router cycling between those two constantly. Maybe the problem is that R5 does not have an idea that there is DR/BDR. Let's check the network type on R5's interface:

```
Rack1R5#show ipv6 ospf interface fastEthernet 0/1
FastEthernet0/1 is up, line protocol is up
  Link Local Address FE80::20F:90FF:FEFB:A21, Interface ID 5
  Area 1, Process ID 1, Instance ID 0, Router ID 150.1.5.5
  Network Type POINT_TO_POINT, Cost: 1
  Transmit Delay is 1 sec, State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:02
  Index 1/1/2, flood queue length 0
  Next 0x0(0)/0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 4
  Last flood scan time is 0 msec, maximum is 4 msec
  Neighbor Count is 2, Adjacent neighbor count is 1
    Adjacent with neighbor 150.1.2.2
  Suppress hello for 0 neighbor(s)
```

This explains a lot. R5 hears hello packets from R2 and SW2 and attempts synchronizing with every new router in sequence. As you can see the neighbor count is 2, while the number of adjacent neighbors is one. Let’s change the network type:

```
R5:
interface FastEthernet 0/1
 ipv6 ospf network broadcast
```

Let’s see if that helps:

```
Rack1R5#show ipv6 ospf neighbor

Neighbor ID     Pri   State           Dead Time   Interface ID
Interface
150.1.8.8         1   FULL/DR         00:00:30    2326
FastEthernet0/1
150.1.2.2         1   FULL/BDR        00:00:37    4
FastEthernet0/1
```

Now check if R2 has the route for R5’s Loopback100:

```
Rack1R2#show ipv6 route ospf
IPv6 Routing Table - 8 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS
summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF
ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
O   2001:150:1::/60 [110/0]
     via ::, Null0
O   2001:150:1:8::8/128 [110/1]
     via FE80::215:63FF:FE48:744, FastEthernet0/0
```

Nope, just the summary “discard-route” and SW2’s Loopback100. Let’s get to R5 and see if there are any issues advertising this prefix.

```
Rack1R5#show ipv6 interface loopback 100
Loopback100 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::20F:90FF:FEFB:A20
  Global unicast address(es):
    2001:150:1:5::5, subnet is 2001:150:1:5::/64
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::5
    FF02::1:FF00:5
    FF02::1:FFFB:A20
  MTU is 1514 bytes
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
  ND DAD is not supported
  ND reachable time is 30000 milliseconds
  Hosts use stateless autoconfig for addresses.

Rack1R5#show ipv6 ospf interface loopback 100
Loopback100 is up, line protocol is up
  Link Local Address FE80::20F:90FF:FEFB:A20, Interface ID 11
  Area 0, Process ID 1, Instance ID 0, Router ID 150.1.5.5
  Network Type LOOPBACK, Cost: 1
  Loopback interface is treated as a stub Host
```

The IP address and prefix length appear to be correct. As for the OSPFv3 settings, we can see that Loopback100 is in Area0. However, R5, R2 and SW2 share the Area 1. This means we need either a virtual link, or area changed. However, per the baseline the Loopback100 should be under Area 1, and this is what we are going to ensure:

```
R5:
Interface Loopback100
 ipv6 ospf 1 area 1
```

See if that helps:

```
Rack1R2#show ipv6 route ospf
IPv6 Routing Table - 9 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS
summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF
ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
O   2001:150:1::/60 [110/0]
     via ::, Null0
O   2001:150:1:5::5/128 [110/1]
     via FE80::20F:90FF:FEFB:A21, FastEthernet0/0
```

```
O   2001:150:1:8::8/128 [110/1]
     via FE80::215:63FF:FE48:744, FastEthernet0/0
```

Now R2 has the route to R5's Loopback100 interface. It seems like we fixed this issue finally.

`Conclusion:` Weird things happen when you mix point-to-point and broadcast networks on the same segment!

Fix the Issue

We summarize the fixing steps in this section:

```
R5:
interface FastEthernet0/1
 ipv6 ospf network broadcast
!
interface Loopback100
 ipv6 ospf 1 area 1
```

Verify

Confirm that R1 can now reach R5's Loopback100:

```
Rack1R1#traceroute 2001:150:1:5::5

Type escape sequence to abort.
Tracing the route to 2001:150:1:5::5

  1 2001:141:1:12::2 40 msec 40 msec 40 msec
  2 2001:150:1:5::5 44 msec 45 msec 48 msec
```

Ticket 10

Analyze the Symptoms

Like we previously suggested, there could me issues caused by this multicast topology: OIL problems at R2's multipoint interface and RPF problems at R2 and R4. Per the ticket requirements, we have multicast sources exactly behind the "problem" areas. What we could do now, is reproduce the failure. We configure R1 and R4 to source multicast traffic to the group 239.1.1.1, as we're working with Rack 1. Configure R3 to join this group on its VLAN37 interface and R1, R4 to source multicast traffic to this group:

```
R3:
interface FastEthernet 0/0
 ip igmp join-group 239.1.1.1

Rack1R1#ping 239.1.1.1 repeat 1000

Rack1R4#ping 239.1.1.1 repeat 1000

Rack1R4#ping 239.1.1.1 repeat 10000

Type escape sequence to abort.
Sending 10000, 100-byte ICMP Echos to 239.1.1.1, timeout is 2 seconds:
….

Rack1R1#ping 239.1.1.1 repeat 1000

Type escape sequence to abort.
Sending 1000, 100-byte ICMP Echos to 239.1.1.1, timeout is 2 seconds:
………
```

Both pings fail, and we start the issue isolation process.

Isolate the Issue

First we go to R3 – it's always better to track the issue from receiver, climbing up the multicast delivery tree.

```
Rack1R3#show ip mroute 239.1.1.1
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C -
Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry,
       X - Proxy Join Timer Running, A - Candidate for MSDP
Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode
```

```
(*, 239.1.1.1), 00:03:43/00:02:39, RP 0.0.0.0, flags: SJPL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list: Null
```

First, we only see (*,G) state for the group; next, the RP is set to 0.0.0.0 meaning R3 has no information on the RP for the group! This prompts us checking if there are any Auto-RP issues, as we know this protocol is used for RP information dissemination.

```
Rack1R3#show ip pim rp mapping
PIM Group-to-RP Mappings

Rack1R3#
```

We go upstream to R2 – this is the router that should be forwarding down Auto-RP information – and check the RP cache there.

```
Rack1R2#show ip pim rp mapping
PIM Group-to-RP Mappings
This system is an RP (Auto-RP)

Group(s) 225.0.0.0/8
  RP 150.1.2.2 (?), v2v1
    Info source: 150.1.8.8 (?), elected via Auto-RP
         Uptime: 04:07:57, expires: 00:02:26
Group(s) 239.0.0.0/8
  RP 150.1.5.5 (?), v2v1
    Info source: 150.1.8.8 (?), elected via Auto-RP
         Uptime: 01:47:54, expires: 00:02:25
```

Great, at least R2 has the information. Let's track what's going on between R2 and R3, by debugging Auto-RP messages:

```
Rack1R3#debug ip pim auto-rp
PIM Auto-RP debugging is on
```

We spend enough time (a couple of minutes) to see at least one message, but don't see any. Must be something at R2 preventing the Auto-RP messages from going through. Let's get back and enable debugging there:

```
Auto-RP(0): Build RP-Announce for 150.1.2.2, PIMv2/v1, ttl 16, ht 181
Auto-RP(0):  Build announce entry for (225.0.0.0/8)
Auto-RP(0): Send RP-Announce packet on Tunnel0
Auto-RP(0): Send RP-Announce packet on FastEthernet0/0
Auto-RP(0): Send RP-Announce packet on Serial0/0
Auto-RP: Send RP-Announce packet on Loopback0
Auto-RP(0): Received RP-discovery, from 150.1.8.8, RP_cnt 2, ht 181
Auto-RP(0): Update (225.0.0.0/8, RP:150.1.2.2), PIMv2 v1
Auto-RP(0): Update (239.0.0.0/8, RP:150.1.5.5), PIMv2 v1
Auto-RP(0): Build RP-Announce for 150.1.2.2, PIMv2/v1, ttl 16, ht 181
Auto-RP(0):  Build announce entry for (225.0.0.0/8)
```

```
Auto-RP(0): Send RP-Announce packet on Tunnel0
Auto-RP(0): Send RP-Announce packet on FastEthernet0/0
Auto-RP(0): Send RP-Announce packet on Serial0/0
Auto-RP: Send RP-Announce packet on Loopback0
```

It appears that R2 is actually sending the packets! Maybe there is some filtering configured? Let's check for multicast route states. The one we're interested in is the Auto-RP discovery group, 224.0.1.40:

```
Rack1R2#show ip mroute 224.0.1.40
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C -
Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry,
       X - Proxy Join Timer Running, A - Candidate for MSDP
Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 224.0.1.40), 04:38:42/stopped, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Loopback0, Forward/Sparse, 04:38:42/00:02:28

(150.1.8.8, 224.0.1.40), 04:29:37/00:02:41, flags: LT
  Incoming interface: FastEthernet0/0, RPF nbr 141.1.0.8
  Outgoing interface list:
    Loopback0, Forward/Sparse, 04:29:37/00:02:28
```

Just as expected, the group is dense, but it's not being forwarded down the Frame-Relay interface. As we know, Auto-RP works with either PIM SM/DM mode or PIM SM + AutoRP listener. Let's check the mode enabled for all interfaces:

```
Rack1R2#show ip pim interface

Address          Interface                Ver/   Nbr    Query  DR
DR
                                          Mode   Count  Intvl  Prior
150.1.2.2        Loopback0                v2/S   0      30     1
150.1.2.2
141.1.0.2        FastEthernet0/0          v2/S   2      30     1
141.1.0.8
141.1.123.2      Serial0/0                v2/S   2      30     1
141.1.123.3
```

It's all sparse. So the issue may be that AutoRP listener is not enabled. Let's enabled it and see what changes:

```
R2:
ip pim autorp listener

Rack1R2#show ip mroute 224.0.1.40
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C -
Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry,
       X - Proxy Join Timer Running, A - Candidate for MSDP
Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 224.0.1.40), 00:00:23/00:02:51, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/0, Forward/Sparse, 00:00:23/00:00:00
    FastEthernet0/0, Forward/Sparse, 00:00:23/00:00:00
    Tunnel0, Forward/Sparse, 00:00:23/00:00:00
    Loopback0, Forward/Sparse, 00:00:23/00:00:00
```

Now we have the group flooded out of all interfaces. Let’s get down to R3 and see if it has the RP information now:

```
Rack1R3#show ip pim rp mapping
PIM Group-to-RP Mappings

Group(s) 225.0.0.0/8
  RP 150.1.2.2 (?), v2v1
    Info source: 150.1.8.8 (?), elected via Auto-RP
         Uptime: 00:08:13, expires: 00:02:42
Group(s) 239.0.0.0/8
  RP 150.1.5.5 (?), v2v1
    Info source: 150.1.8.8 (?), elected via Auto-RP
         Uptime: 00:08:13, expires: 00:02:43
Good, apparently it does. So back to our multicast tree, let’s see if
it makes it:
```

Now look for the multicast tree for the group 239.1.1.1:

```
Rack1R3#show ip mroute 239.1.1.1
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C -
Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry,
       X - Proxy Join Timer Running, A - Candidate for MSDP
Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.1.1.1), 00:00:56/00:02:25, RP 150.1.5.5, flags: SJPL
  Incoming interface: Serial1/0.1, RPF nbr 141.1.123.2
  Outgoing interface list: Null
```

But the pings at R1 and R4 still fail. Let's deal with every source separately, starting with the source at R4. Go up one level and check the multicast tree in R2:

```
Rack1R2#show ip mroute 239.1.1.1
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C -
Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry,
       X - Proxy Join Timer Running, A - Candidate for MSDP
Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.1.1.1), 00:14:46/00:03:22, RP 150.1.5.5, flags: S
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/0, Forward/Sparse, 00:14:46/00:03:22
```

The incoming interface is Null, while the outgoing interface list includes Serial0/0. Plus this is an (*,G) tree. The fact that Incoming interface is Null means there is no proper PIM enabled interface to reach the RP address of 150.1.5.5. This in turn means that the shortest route to 150.1.5.5 is not via a PIM enabled interface. Let's check this:

```
Rack1R2#show ip route 150.1.5.5
Routing entry for 150.1.5.5/32
  Known via "ospf 1", distance 110, metric 11112, type inter area
  Last update from 141.1.25.5 on Tunnel0, 03:33:55 ago
  Routing Descriptor Blocks:
  * 141.1.25.5, from 150.1.5.5, 03:33:55 ago, via Tunnel0
      Route metric is 11112, traffic share count is 1
```

Ah, there is a tunnel, and apparently it leads R5! Nice surprise, not mentioned in the baseline configuration. Apparently the unicast routing prefers this tunnel, while multicast is transported via some other means. Let's check where PIM is enabled between R2 and R5:

```
Rack1R2#show ip pim interface fastEthernet 0/0

Address          Interface                Ver/   Nbr    Query  DR
DR
                                          Mode   Count  Intvl  Prior
141.1.0.2        FastEthernet0/0          v2/S   2      30     1
141.1.0.8

Rack1R2#show ip pim interface tunnel 0

Address          Interface                Ver/   Nbr    Query  DR
DR
                                          Mode   Count  Intvl  Prior
Rack1R2#

Rack1R5#show ip pim interface fastEthernet 0/0

Address          Interface                Ver/   Nbr    Query  DR
DR
                                          Mode   Count  Intvl  Prior
141.1.145.5      FastEthernet0/0          v2/S   1      30     1
141.1.145.5

Rack1R5#show ip pim interface tunnel 0

Address          Interface                Ver/   Nbr    Query  DR
DR
                                          Mode   Count  Intvl  Prior
```

OK, we don’t know the exact reason for this tunnel and have no time to figure it right now. But multicast routing is not enabled on the tunnel which is preferred for unicast routing. And we have to fix the problem as soon as possible. So the easiest thing would be configuring the Tunnel for multicast routing on both ends.

```
R2:
interface Tunnel 0
 ip pim sparse-mode

R5:
interface Tunnel0
 ip pim sparse-mode
```

So does anything changes after this? It looks like now R4 can ping VLAN37:

```
Rack1R4#ping 239.1.1.1 repeat 10000

Type escape sequence to abort.
Sending 10000, 100-byte ICMP Echos to 239.1.1.1, timeout is 2 seconds:

Reply to request 0 from 141.1.123.3, 60 ms
Reply to request 0 from 141.1.123.3, 120 ms
Reply to request 1 from 141.1.123.3, 60 ms
Reply to request 1 from 141.1.123.3, 120 ms
Reply to request 2 from 141.1.123.3, 60 ms
Reply to request 2 from 141.1.123.3, 124 ms
```

But let's check the multicast delivery tree once again, to ensure VLAN43 IP address is among the sources:

```
Rack1R5#show ip mroute 239.1.1.1
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C -
Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry,
       X - Proxy Join Timer Running, A - Candidate for MSDP
Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.1.1.1), 03:26:02/stopped, RP 150.1.5.5, flags: S
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Tunnel0, Forward/Sparse, 00:16:25/00:02:48

(141.1.54.4, 239.1.1.1), 00:02:21/00:03:28, flags: T
  Incoming interface: Serial0/0, RPF nbr 0.0.0.0
  Outgoing interface list:
    Tunnel0, Forward/Sparse, 00:02:21/00:02:48

(141.1.145.4, 239.1.1.1), 00:02:22/00:03:27, flags: T
  Incoming interface: FastEthernet0/0, RPF nbr 0.0.0.0
  Outgoing interface list:
    Tunnel0, Forward/Sparse, 00:02:22/00:02:47

(204.12.1.4, 239.1.1.1), 00:00:07/00:02:57, flags: T
  Incoming interface: FastEthernet0/0, RPF nbr 141.1.145.4
  Outgoing interface list:
    Tunnel0, Forward/Sparse, 00:00:07/00:02:52
```

Good, this makes us done with the source on VLAN43. As for the other source, we may already know the issue. Like we guessed in the initial analysis, let's check if PIM NMBA mode is enabled for R2's Frame-Relay interface:

```
Rack1R2#show ip pim interface serial 0/0 detail
Serial0/0 is up, line protocol is up
  Internet address is 141.1.123.2/24
  Multicast switching: fast
  Multicast packets in/out: 2009/351
  Multicast TTL threshold: 0
  PIM: enabled
    PIM version: 2, mode: sparse
    PIM DR: 141.1.123.3
    PIM neighbor count: 2
    PIM Hello/Query interval: 30 seconds
    PIM Hello packets in/out: 1407/735
    PIM State-Refresh processing: enabled
    PIM State-Refresh origination: disabled
    PIM NBMA mode: disabled
    PIM ATM multipoint signalling: disabled
    PIM domain border: disabled
  Multicast Tagswitching: disabled
```

OK, so it's disabled. We are going to enable it and try pinging from R1 again:

```
Rack1R1#ping 239.1.1.1 repeat 1000

Type escape sequence to abort.
Sending 1000, 100-byte ICMP Echos to 239.1.1.1, timeout is 2 seconds:

Reply to request 0 from 141.1.123.3, 172 ms
Reply to request 1 from 141.1.123.3, 160 ms
Reply to request 2 from 141.1.123.3, 141 ms
Reply to request 3 from 141.1.123.3, 144 ms
Reply to request 4 from 141.1.123.3, 144 ms
Reply to request 5 from 141.1.123.3, 144 ms
Reply to request 6 from 141.1.123.3, 148 ms
Reply to request 7 from 141.1.123.3, 148 ms
Reply to request 8 from 141.1.123.3, 148 ms
```

Appears to be working – notice that it may take some time for PIM to signal the new states on the Frame-Relay interface and everything to start working. Check the multicast routes in R3 to confirm VLAN12 among sources:

```
Rack1R3#show ip mroute 239.1.1.1
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C -
Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry,
       X - Proxy Join Timer Running, A - Candidate for MSDP
Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.1.1.1), 01:09:30/stopped, RP 150.1.5.5, flags: SJPLF
  Incoming interface: Serial1/0.1, RPF nbr 141.1.123.2
  Outgoing interface list: Null

(141.1.54.4, 239.1.1.1), 00:09:44/00:02:59, flags: PLT
  Incoming interface: Serial1/0.1, RPF nbr 141.1.123.2
  Outgoing interface list: Null
```

```
(141.1.123.1, 239.1.1.1), 00:02:47/00:02:58, flags: PLFT
  Incoming interface: Serial1/0.1, RPF nbr 0.0.0.0
  Outgoing interface list: Null

(141.1.145.4, 239.1.1.1), 00:09:45/00:02:58, flags: PLT
  Incoming interface: Serial1/0.1, RPF nbr 141.1.123.2
  Outgoing interface list: Null

(192.10.1.1, 239.1.1.1), 00:00:08/00:02:55, flags: PLT
  Incoming interface: Serial1/0.1, RPF nbr 141.1.123.2
  Outgoing interface list: Null

(204.12.1.4, 239.1.1.1), 00:01:25/00:01:44, flags: PLT
  Incoming interface: Serial1/0.1, RPF nbr 141.1.123.2
  Outgoing interface list: Null
```

Just to see the effect of PIM NMBA mode, check the multicast routing table in R2. The below output shows the SPT built to join the source in R4:

```
Rack1R2#show ip mroute 239.1.1.1
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C -
Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry,
       X - Proxy Join Timer Running, A - Candidate for MSDP
Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

<snip>

(192.10.1.1, 239.1.1.1), 00:00:05/00:03:29, flags: T
  Incoming interface: Serial0/0, RPF nbr 141.1.123.1
  Outgoing interface list:
    Tunnel0, Forward/Sparse, 00:00:05/00:03:26
    Serial0/0, 141.1.123.3, Forward/Sparse, 00:00:05/00:02:54
```

Now we're done with fixing the multicast issues for real.

Conclusion: The best way to troubleshoot unknown multicast issues is by digging bottom-up from the leaves of the multicast delivery tree.

Fix the Issue

Summary of the changes been made:

```
R2:
ip pim autorp listener
!
```

```
interface Serial 0/0
 ip pim nbma-mode
!
interface Tunnel0
 ip pim sparse-mode
```

R5:

```
interface Tunnel0
 ip pim sparse-mode
```

Verify

We already verified everything while isolating the issues, so there is nothing more left to verify!

IEWB-RS VOL4 Lab 3

Lab Overview:

The following scenario is a practice lab exam designed to test your skills at troubleshooting Cisco networking devices. Specifically, this scenario is designed to assist you in your preparation for Cisco Systems' CCIE Routing & Switching Lab exam Troubleshooting Section. However, remember that in addition to being designed as a simulation of the actual CCIE lab exam, this practice lab should be used as a learning tool. Instead of rushing through the lab in order to resolve all issues, take the time to apply the structured troubleshooting methodology and improve your strategy.

Lab Instructions:

Prior to starting, ensure that the initial configuration scripts for this lab have been applied. For a current copy of these scripts, see the Internetwork Expert members' site at http://members.INE.com

Refer to the attached diagrams for interface and protocol assignments. Any reference to X in an IP address refers to your rack number, while any reference to Y in an IP address refers to your router number. When not explicitly mentioned, a router's IP address on the segment is based off the router number, Y. For example, R1 will have the IP address of 150.X.100.1 on the subnet 150.X.100.0/24, and SW3 will have the IP address 150.X.100.9 on the same subnet.

Use the name **`cisco`** along with the password of **`cisco`** to access the console line of any device used in the topology.

Lab Do's and Don'ts:

- Do not access the routers that are marked as restricted for your access.
- Do not use the **`show running-config`** or **`show startup-config`** commands or their equivalents when performing troubleshooting.
- Do not change or add any IP addresses from the initial configuration unless required for troubleshooting.
- Do not change any interface encapsulations unless required for troubleshooting.
- Do not change the console, AUX, and VTY passwords or access methods unless otherwise specified.
- Do not use static route, default routes, default networks, or policy routing unless otherwise specified.
- Save your configurations often.

Grading:

This practice lab consists of 10 trouble tickets totaling 30 points. A score of 24 points is required to achieve a passing grade. A trouble ticket must be fixed 100% with the requirements given in order to be awarded the points for that ticket. No partial credit is awarded. If a ticket has multiple possible resolutions, choose the solution that best meets the requirements and requires minimal changes. Per the CCIE R&S lab exam requirements, you are required to finish this lab in **`two`** hours.

The tickets generally have no dependencies, unless explicitly stated in the ticket outline, so you may work through them in any order you like. It's up to you to select the tickets that you feel most easy to deal with and manage your time accordingly.

GOOD LUCK!

Baseline

All network devices are configured according to the diagram provided with this scenario. The diagram reflects the proper network configuration, including IP address, IGP protocol settings and BGP AS numbers and serves as your primary source of the information. This section provides the scenario-specific configuration information that you may need during troubleshooting process. Notice that not all of the information may be useful during the troubleshooting process; however, all statement made below reflect the correct network configuration.

Devices under your Control

For this lab, you may only access and modify the configuration of all “core” devices in the network. Backbone devices BB1, BB2 and BB3 are out of your control per the initial topology configuration. If you refer to the diagram provided, the devices colored in **ORANGE** are out of your control.

Bridging and Switching

- The following is the list of the trunk links interconnecting the switches:
  - SW1 interface Fa0/16 and SW3 interface Fa0/13.
  - SW1 interface Fa0/19 and SW4 interface Fa0/13.
- Additionally, here is the outline of the trunk links configuration between SW1 and SW2:

```
#show etherchannel 13 port-channel
                Port-channels in the group:
                ---------------------------

Port-channel: Po13
------------

Age of the Port-channel   = 00d:00h:02m:38s
Logical slot/port   = 2/13          Number of ports = 3
GC                  = 0x00000000      HotStandBy port = null
Port state          = Port-channel Ag-Inuse
Protocol            =    -

Ports in the Port-channel:

Index   Load   Port     EC state        No of bits
------+------+------+------------------+-----------
  0     00     Fa0/13   On/FEC             0
  0     00     Fa0/14   On/FEC             0
  0     00     Fa0/15   On/FEC             0

Time since last port bundled:    00d:00h:01m:34s    Fa0/15
```

```
#show interfaces po13 trunk

Port        Mode         Encapsulation  Status        Native vlan
Po13        on           802.1q         trunking      1

Port        Vlans allowed on trunk
Po13        1-6,8-4094

Port        Vlans allowed and active in management domain
Po13        1,3-6,42,55,57,263

Port        Vlans in spanning tree forwarding state and not pruned
Po13        1,3-6,42,55,57,263
```

- The Frame-Relay sub-interfaces of R5 are configured as point-to-point.

IGP

- IGP protocols are configured per the diagram supplied with the scenario.

BGP

- Using the Layer 3 connectivity diagram as your reference, recover the BGP peering sessions mesh.
- Do not break the requirements of the lab scenario when performing this operation.
- R1 and BB2 peering session is authenticated used the password value of “CISCO”.

Trouble Tickets

Ticket 1: RIPv2

- Fix the issue preventing R6 from learning RIPv2 prefixes from BB1.
- Do not change the IP addressing on R6 to resolve this problem.

2 Points

Ticket 2: BGP Peering

- The BGP session between R6 and SW2 is not coming up for some reason.
- Find what might be causing this and fix the problem applying minimal changes.

3 Points

Ticket 3: Backup Link

- Users on VLAN42 report terrible performance when connecting the servers on VLAN5.
- You suspect that the problem is that packets are taking suboptimal path across the backup ISDN link between the two sites.
- Restore optimal performance and ensure the primary path over Frame-Relay cloud is being used.

3 Points

Note

You need to resolve Ticket 4 prior to continuing with any other ticket that follows.

Ticket 4: Network Optimization

This ticket requires Tickets 1 and 2 resolved prior to starting.

- After a recent merge process, some networks have been joined together.
- In order to provide end-to-end connectivity, mutual redistribution has been configured between RIP and OSPF on R1, R2, R3, R4 and SW1.
- However, the straight-forward configuration approach resulted in network instabilities and sub-optimal routing.
- Fix the network configuration to ensure optimal routing, which means using high-speed links (Ethernet and Frame-Relay) for primary data paths.
- The routers behind R5 should receive a single default route to reach the rest of the network.
- R4 may still use the Serial link to reach the networks behind R5.

4 Points

Ticket 5: Server Farm

- Two new servers have been recently connected to SW1 port 0/18 and SW2 port 0/19 respectively.
- For the lab purpose, the servers are emulated by L3 ports in SW3 and SW4 respectively.
- The following is the configuration policy for the server block:
  - The new servers are assigned to the subnet 192.10.X.0/24
  - To ensure security, the servers should not be able to communicate directly.
  - The servers should be able to reach and communicate via R4.
- However, the configuration appears to be incomplete, as servers cannot reach to each other.
- Ensure the connectivity via R4 and verify it by using IP addresses assigned to the switchports of SW3 and SW4.

3 Points

Ticket 6: BGP

- AS254 and AS54 cannot exchange routes across AS 100, AS 200 and AS 300.
- Applying changes to BGP configurations only, restore the end-to-end connectivity.
- Do not apply any changes to AS 300 routers to accomplish this.
- The ticked will be resolved when AS 54 and AS 254 could see each other’s routes.

3 Points

Ticket 7: L2VPN

- You have tasked a junior system administrator to configure an L2VPN tunnel between VLAN64 and VLAN46 interfaces of R6 and R4.
- Some time after this, he reported that the configuration has been done, but there are some issues preventing it from working properly.
- Find and resolve the issues preventing the Layer 2 VPN tunnel from coming up.

3 Points

Ticket 8: L2VPN Issues

- The Frame-Relay connection between R1 and R2 has been recently replaced with L2 VPN. Soon after this, users on VLAN263 started complaining that they cannot download files from the servers on VLAN42.
- They still could use **`telnet`** and **`ssh`** to access the servers though.
- Suspecting that this may be caused by Path MTU discovery failures, you configured a workaround by clearing DF bit on all TCP packets received by R2 on its VLAN263 interface.
- However, this does not seem to help and the problem persists. Find the solution to this problem that works for all types of network traffic.

3 Points

Ticket 9: NTP

- You have recently re-loaded the configuration for R3 and R4 using the backup copies and soon after that discovered the logging timestamps went wrong.
- You know that NTP has been used for time synchronization on R3 and R4.
- R5 was supposed to be the NTP master for R3 and R4.
- You may assume that R5 is configured correctly and fix the issues in R3 and R4.

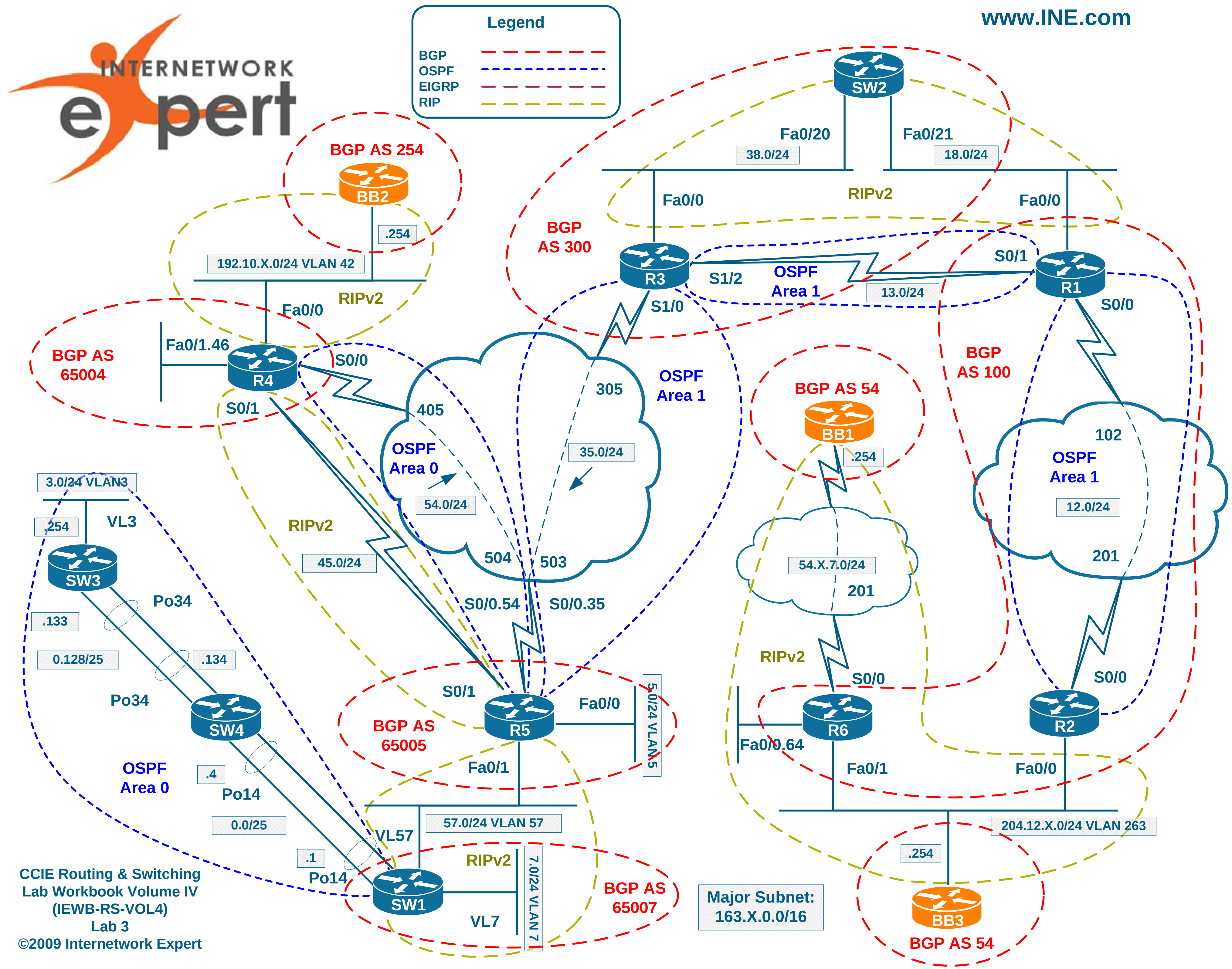
3 Points

Ticket 10: Internet Access

- After the recent security policy changes users on VLAN5 started complaining they cannot reach YouTube videos anymore.
- Asking a few questions you found that uses cannot access any other WWW sites as well.
- After a short investigation you found that the uplink used to reach the YouTube website is via BB1.
- Talking to the users you found that only HTTP is being affected, FTP downloads work well.
- Applying minimal changes to the network configuration resolve this problem and restore WWW connectivity..

3 Points

www.INE.com



CCIE Routing & Switching
Lab Workbook Volume IV
(IEWB-RS-VOL4)
Lab 3
©2009 Internetwork Expert

IEWB-RS VOL4 Lab 3 Solutions

Table of Contents

IEWB-RS VOL4 Lab 3 Solutions 1
Table of Contents 1
Build and Analyze the Baseline 3
Layer 2 Diagram 4
BGP Diagram 5
Multicast and Redistribution 7
IPv6 Diagram 7
Read over the Lab 7
Solutions 8
Ticket 1 8
Analyze the Symptoms 8
Isolate the Issue 8
Fix the issue 13
Verify 13
Ticket 2 14
Analyze the Symptoms 14
Isolate the Issue 15
Fix the Issue 22
Verify 22
Ticket 3 23
Analyze the Symptoms 23
Isolate the Issue 25
Fix the Issue 31
Verify 32
Ticket 4 33
Analyze the Symptoms 33
Isolate the Issue 36
Fix the Issue 41
Verify 42
Ticket 5 43
Analyze the Symptoms 43
Isolate the Issue 43
Fix the Issue 50
Verify 50
Ticket 6 53
Analyze the Symptoms 53
Isolate the Issue 53
Fix the Issue 62
Verify 62

Ticket 7 63
Analyze the Symptoms 63
Isolate the Issue 63
Fix the Issue 67
Verify 67
Ticket 8 69
Analyze the Symptoms 69
Isolate the Issue 70
Fix the Issue 73
Verify 73
Ticket 9 75
Analyze the Symptoms 75
Isolate the Issue 75
Fix the Issue 79
Verify 79
Ticket 10 80
Analyze the Symptoms 80
Isolate the Issue 80
Fix the Issue 83
Verify 83

Build and Analyze the Baseline

When you start with the scenario, all you have is the diagram and some textual information on the network baseline. Your goal at this moment is structuring the available information and making additional diagrams. We recommend extra diagrams to outline the following: L2 connection, BGP Peerings, Multicast & Redistribution and IPv6 Topology. Notice that some of these could be combined in a single diagram – for example you may put the Multicast and Redistribution outlines on the initial L3 diagram. This is probably the best way to save your time during the analysis stage. However, we are going to use separate diagrams for the ease of explanation.

Layer 2 Diagram

This is where our Ethernet L2 connections are outlined (notice, no L3 Etherchannels). This diagram would helps us finding any L2 mis-configurations. Notice that we only put the routers that have Ethernet connections on the initial diagram.

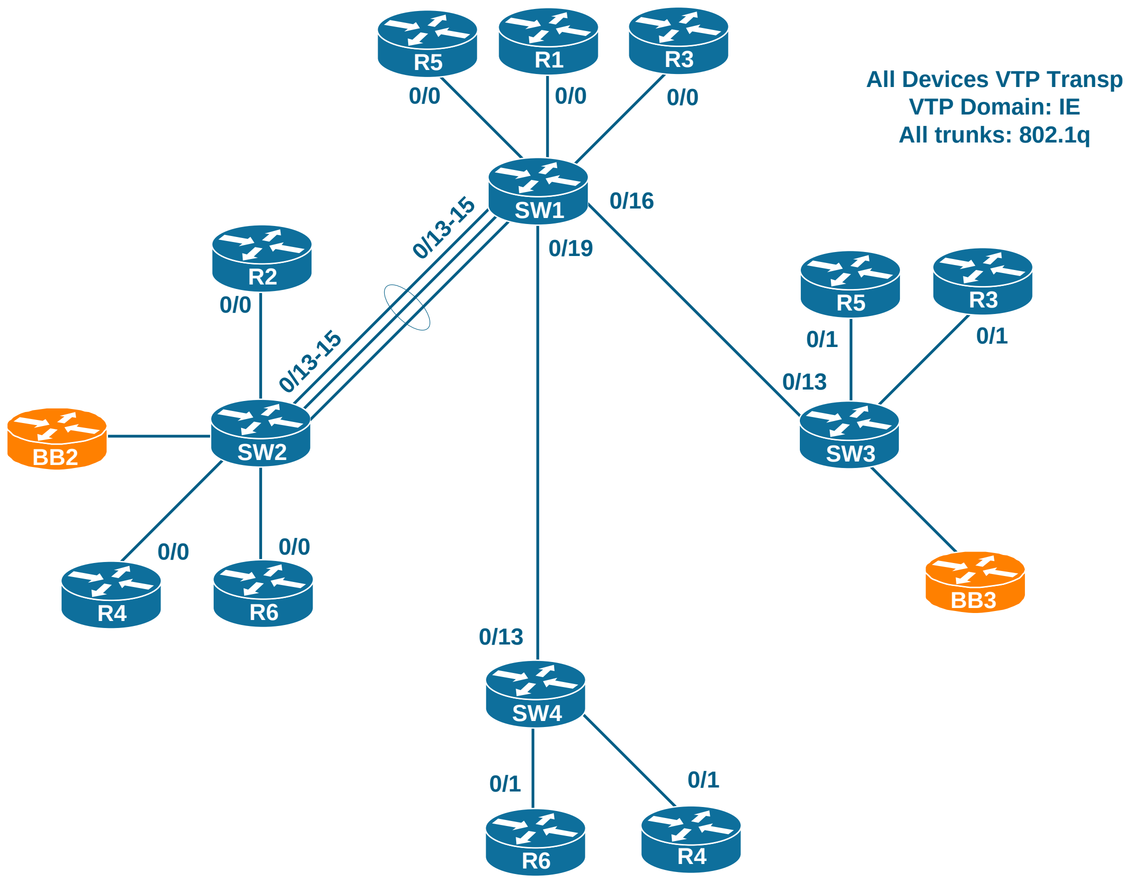


Fig 1

The scenario only tells us about the trunks between SW1 and SW3/SW4. We have to recover the Layer2 port-channel based on the show command output presented in the baseline description. The resulting topology has no L2 loops, the topology is tree-like. This reduces the risk of any STP-related problems.

BGP Diagram

We don't have the description of BGP peerings/AS numbers in the baseline. This means we need to routinely proceed through all routers and use the **`show ip bgp summary`** command to recover the peering sessions. For example, start with R1:

```
Rack1R1#show ip bgp summary
BGP router identifier 150.1.1.1, local AS number 100
BGP table version is 1, main routing table version 1

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
163.1.12.2      4   100      21      21        1    0    0 00:18:22
0
163.1.13.3      4   200      20      20        1    0    0 00:17:48
0
163.1.18.8      4   200       0       0        0    0    0 never
Active
```

You may notice that some peering sessions are displayed as "Active" which already signals an issue. We omit the lengthy outputs and present just the resulting diagram. Notice that R4, R5 and SW1 constitute a BGP confederation.

When looking at the BGP summary output you will notice something similar to the following:

```
Rack1R5#show ip bgp summary
BGP router identifier 150.1.5.5, local AS number 65005
BGP table version is 1, main routing table version 1

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
150.1.4.4       4 65004       0       0        0    0    0 never
Active
163.1.35.3      4   300      25      24        1    0    0 00:21:29
0
163.1.57.7      4 65007      22      23        1    0    0 00:19:20
0
```

You may discover the real AS# used for external peering by looking at the external neighbor's status. For example if you look at R3 you will find the real AS# used for the confederation:

```
Rack1R3#show ip bgp summary
BGP router identifier 150.1.3.3, local AS number 300
BGP table version is 1, main routing table version 1

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
```

```
163.1.13.1      4    100       32       32          1     0     0 00:29:32
0
163.1.35.5      4    200       30       31          1     0     0 00:27:46
0
163.1.38.8      4    300        0        0          0     0     0 never
Active
```

Here is the final diagram that you should get after you finished with BGP session discovery.

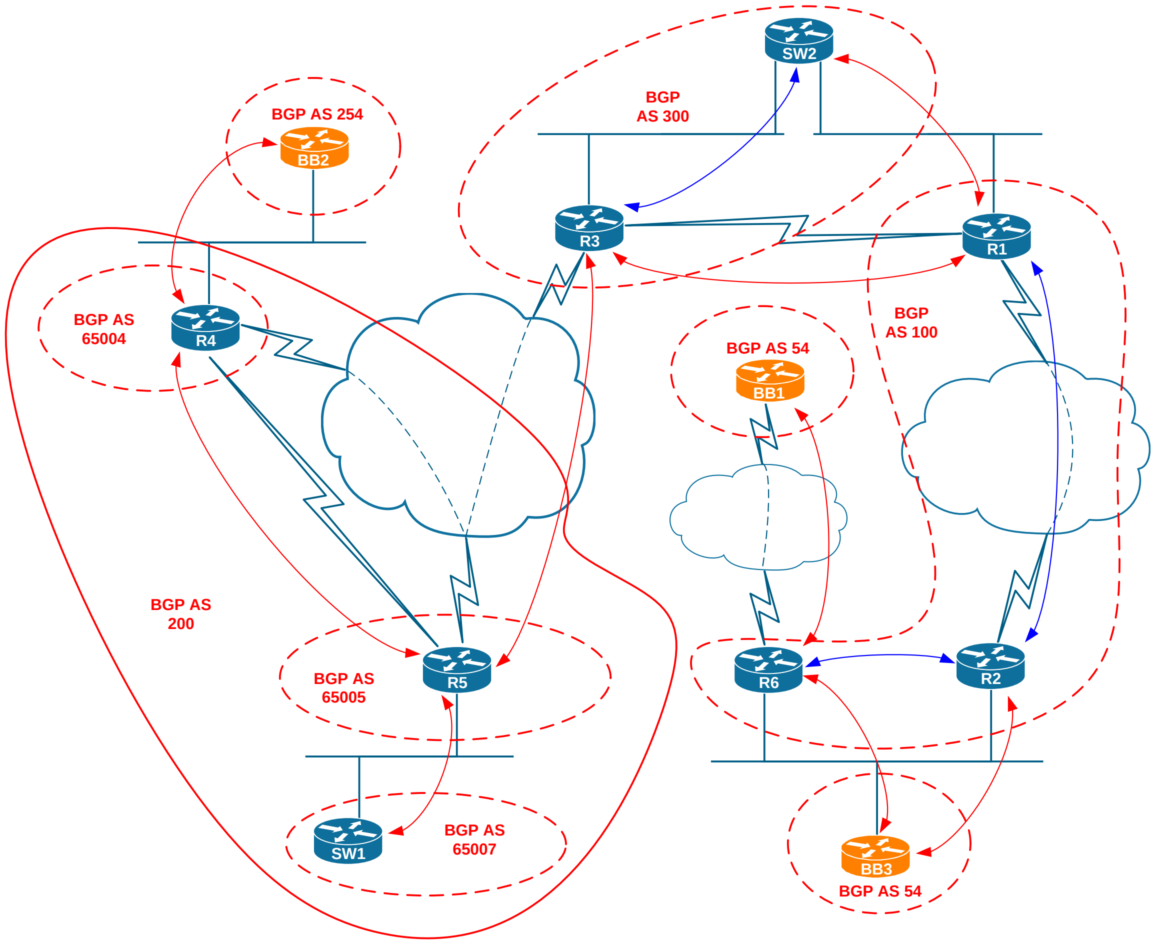


Fig 2

Very little information is provided on BGP, so we only have to collect the basic facts and save the diagram for further reference.

Multicast and Redistribution

There are no multicast configurations mentioned in the scenario tickets or the baseline, and so we don't spend our time dealing with multicast propagation analysis. As for redistribution, there is a ticket (Ticket 4) that explicitly requires us to analyze and deal with redistribution problems. We'll postpone the redistribution analysis till we get to this ticket.

IPv6 Diagram

There is nothing mentioned about IPv6 in this scenario, nor there is anything about it in the baseline, so we can simply skip IPv6 diagrams and analysis.

Read over the Lab

Our last step is looking through the tickets. We quickly notice that Tickets 1, 2 and 4 are a must to complete in order to succeed in this lab. What makes it especially dangerous, is that ticket 4 worth 4 points, which probably means is really is hard. However, the good news is that ticket 1 clearly limits its problem scope to R6 only.

Ticket 6 seems to be relatively attractive, as it only requires you to change BGP settings. As for Ticket 8, it clearly specifies the problem scope to be limited to R3 and R4.

Solutions

Ticket 1

Analyze the Symptoms

At least we know the scope of the problem which is limited to R6. We don't know the way that R6 connects to BB1, but it should be Frame-Relay for sure. Let's check the current situation:

```
Rack1R6#show ip interface brief
Interface                  IP-Address      OK? Method Status
Protocol
FastEthernet0/0            163.1.6.6       YES manual up
up
Serial0/0                  unassigned      YES manual up
up
FastEthernet0/1            204.12.1.6      YES manual up
up
Virtual-Access1            unassigned      YES unset  down
down
Virtual-Template1          54.1.7.6        YES manual down
down
Virtual-Access2            unassigned      YES TFTP   down
down
Virtual-Access3            unassigned      YES TFTP   down
down
Loopback0                  150.1.6.6       YES manual up
up
```

This gives us an issue right away. We see that have a virtual-template with the addressing of the link connecting to R6. This means that PPPoFR is used to establish connectivity. As you can see, all virtual access interfaces cloned from the template are down, so we suspect link failure. This is our primary hypothesis for now.

Isolate the Issue

The first thing we do is check the physical connectivity. This is the bottom-up approach and it suits the case perfectly, as we only have one interface to check.

```
Rack1R6#show interfaces serial 0/0
Serial0/0 is up, line protocol is up
  Hardware is PowerQUICC Serial
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation FRAME-RELAY, loopback not set
  Keepalive set (10 sec)
  LMI enq sent  1052, LMI stat recvd 1053, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent  0, LMI upd sent  0
  LMI DLCI 1023  LMI type is CISCO  frame relay DTE
<snip>
```

The interface and the protocol is up, we see LMI messages being sent and received. This rules out any physical issue. So now we consider some misconfiguration. As we know, PPPoFR is configured via a virtual template bound to a PVC. Per the diagram the DLCI used is 201. Let’s see if everything is OK with that:

```
Rack1R6#show frame-relay pvc 201

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 201, DLCI USAGE = UNUSED, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0

  input pkts 2346          output pkts 0            in bytes 108135
  out bytes 0              dropped pkts 0           in pkts dropped 0
  out pkts dropped 0                out bytes dropped 0
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0          in DE pkts 0             out DE pkts 0
  out bcast pkts 0         out bcast bytes 0
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
  pvc create time 03:00:58, last time pvc status changed 03:00:38
```

The DLCI shows up as unused. That means it is not associated with any Virtual-Template like it should be with PPPoFR. So we associate the PVC with the virtual interface:

```
R6:
interface Serial 0/0
 frame-relay interface-dlci 201 ppp virtual-Template 1
```

Let’s see if that does the trick:

```
Rack1R6#show ip interface brief
Interface                  IP-Address      OK? Method Status
Protocol
FastEthernet0/0            163.1.6.6       YES manual up
up
Serial0/0                  unassigned      YES manual up
up
FastEthernet0/1            204.12.1.6      YES manual up
up
Virtual-Access1            unassigned      YES unset  down
down
Virtual-Template1          54.1.7.6        YES manual down
down
Virtual-Access2            unassigned      YES TFTP   down
down
Virtual-Access3            54.1.7.6        YES TFTP   up
down
```

```
Loopback0                  150.1.6.6       YES manual up
up
```

We see that Virtual-Access 3 cloned from the virtual-template interface is in up/down state. This means there is something preventing PPP from coming up. Let's check the detailed interface statistics.

```
Rack1R6#show interface Virtual-Access 3
Virtual-Access3 is up, line protocol is down
  Hardware is Virtual Access interface
  Internet address is 54.1.7.6/24
  MTU 1500 bytes, BW 100000 Kbit, DLY 100000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation PPP, LCP Listen
  PPPoFR vaccess, cloned from Virtual-Template1
  Vaccess status 0x44
  Bound to Serial0/0 DLCI 201, Cloned from Virtual-Template1, loopback
not set
  Keepalive set (10 sec)
  DTR is pulsed for 5 seconds on reset
  Last input 00:00:00, output never, output hang never
  Last clearing of "show interface" counters 17:55:44
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 0 bits/sec, 2 packets/sec
  5 minute output rate 0 bits/sec, 2 packets/sec
     85 packets input, 1734 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     106 packets output, 2015 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
```

As we can see PPP is stuck in LCP phase, and there is intense packet exchange. Most likely this is related to some authentication procedure, as this is what prevents LCP from finishing correctly most often. Let's do some debugging. However prior to starting, remember that PPP now runs over a "leased" line, which means the negotiation process goes on constantly. This may result in a flood of debugging message. For this purpose, we disable console logging temporarily and stick to logging into the local buffer.

```
R6:
no logging console
logging buffered debugging
```

Collect the debugging output:

```
Rack1R6#debug ppp negotiation
PPP protocol negotiation debugging is on

Rack1R6#show logging
```

```
Syslog logging: enabled (11 messages dropped, 0 messages rate-limited,
                0 flushes, 0 overruns, xml disabled, filtering
disabled)
    Console logging: disabled
    Monitor logging: level debugging, 0 messages logged, xml disabled,
                     filtering disabled
    Buffer logging: level debugging, 32 messages logged, xml disabled,
                    filtering disabled
    Logging Exception size (4096 bytes)
    Count and timestamp logging messages: disabled

No active filter modules.

    Trap logging: level informational, 68 message lines logged
        Logging to 163.1.5.100 (udp port 514, audit disabled, link up),
13 message lines logged, xml disabled,
               filtering disabled
        Logging to 163.1.6.100 (udp port 514, audit disabled, link up),
13 message lines logged, xml disabled,
               filtering disabled

Log Buffer (4096 bytes):

Vi3 LCP: Timeout: State Listen
Vi3 LCP: O CONFREQ [Listen] id 32 len 15
Vi3 LCP:    AuthProto CHAP (0x0305C22305)
Vi3 LCP:    MagicNumber 0x13052D58 (0x050613052D58)
Vi3 LCP: I CONFREQ [REQsent] id 144 len 15
Vi3 LCP:    AuthProto CHAP (0x0305C22305)
Vi3 LCP:    MagicNumber 0x16F4C47E (0x050616F4C47E)
Vi3 LCP: O CONFACK [REQsent] id 144 len 15
Vi3 LCP:    AuthProto CHAP (0x0305C22305)
Vi3 LCP:    MagicNumber 0x16F4C47E (0x050616F4C47E)
Vi3 LCP: I CONFACK [ACKsent] id 32 len 15
Vi3 LCP:    AuthProto CHAP (0x0305C22305)
Vi3 LCP:    MagicNumber 0x13052D58 (0x050613052D58)
Vi3 LCP: State is Open
Vi3 PPP: Phase is AUTHENTICATING, by both
Vi3 CHAP: O CHALLENGE id 119 len 28 from "ROUTER6"
Vi3 CHAP: I CHALLENGE id 126 len 24 from "BB1"
Vi3 CHAP: Using hostname from interface CHAP
Vi3 CHAP: Using password from interface CHAP
Vi3 CHAP: O RESPONSE id 126 len 28 from "ROUTER6"
Vi3 CHAP: I RESPONSE id 119 len 24 from "BB1"
Vi3 PPP: Phase is FORWARDING, Attempting Forward
Vi3 PPP: Phase is AUTHENTICATING, Unauthenticated User
Vi3 CHAP: O FAILURE id 119 len 25 msg is "Authentication failed"
Vi3 PPP: Sending Acct Event[Down] id[80]
Vi3 PPP: Phase is TERMINATING
Vi3 LCP: O TERMREQ [Open] id 33 len 4
Vi3 LCP: I TERMACK [TERMsent] id 33 len 4
Vi3 LCP: State is Closed
Vi3 PPP: Phase is DOWN
```

Notice some interesting things in the output. First we see CHAP being used as the authentication protocol, both sides agree on that. Next we see that both R6 and BB1 challenge each other. However, the credentials presented by BB1 are not valid, and therefore R6 sends an outbound failure message and closes the connection. At the same time, we don't see any failure messages from BB1, which means only the local side cannot authenticate the remote router. Thus, we're going to go ahead and configure R6 not to request credentials from BB1.

```
R6:
interface Virtual-Template 1
 no ppp authentication chap
```

The above command ensures that R6 will NOT ask for CHAP authentication from BB1. Check the cloned interface state again:

```
Rack1R6#show interfaces virtual-access 3
Virtual-Access3 is up, line protocol is up
  Hardware is Virtual Access interface
  Internet address is 54.1.7.6/24
  MTU 1500 bytes, BW 100000 Kbit, DLY 100000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation PPP, LCP Open
  Open: IPCP
  PPPoFR vaccess, cloned from Virtual-Template1
```

You can also check the PPP negotiation logs on R6 to see that the issue has been resolved:

```
Vi3 LCP: O CONFREQ [Listen] id 188 len 10
Vi3 LCP:    MagicNumber 0x13162BA6 (0x050613162BA6)
Vi3 LCP: I CONFREQ [REQsent] id 218 len 15
Vi3 LCP:    AuthProto CHAP (0x0305C22305)
Vi3 LCP:    MagicNumber 0x1705C34D (0x05061705C34D)
Vi3 LCP: O CONFACK [REQsent] id 218 len 15
Vi3 LCP:    AuthProto CHAP (0x0305C22305)
Vi3 LCP:    MagicNumber 0x1705C34D (0x05061705C34D)
Vi3 LCP: I CONFACK [ACKsent] id 188 len 10
Vi3 LCP:    MagicNumber 0x13162BA6 (0x050613162BA6)
Vi3 LCP: State is Open
Vi3 PPP: Phase is AUTHENTICATING, by the peer
Vi3 CHAP: I CHALLENGE id 176 len 24 from "BB1"
Vi3 CHAP: Using hostname from interface CHAP
Vi3 CHAP: Using password from interface CHAP
Vi3 CHAP: O RESPONSE id 176 len 28 from "ROUTER6"
Vi3 CHAP: I SUCCESS id 176 len 4
Vi3 PPP: Phase is FORWARDING, Attempting Forward
Vi3 PPP: Phase is ESTABLISHING, Finish LCP
Vi3 PPP: Phase is UP
Vi3 IPCP: O CONFREQ [Closed] id 1 len 10
Vi3 IPCP:    Address 54.1.7.6 (0x030636010706)
Vi3 PPP: Process pending ncp packets
Vi3 IPCP: I CONFREQ [REQsent] id 1 len 10
Vi3 IPCP:    Address 54.1.7.254 (0x0306360107FE)
Vi3 IPCP: O CONFACK [REQsent] id 1 len 10
Vi3 IPCP:    Address 54.1.7.254 (0x0306360107FE)
```

```
Vi3 IPCP: I CONFACK [ACKsent] id 1 len 10
Vi3 IPCP:    Address 54.1.7.6 (0x030636010706)
Vi3 IPCP: State is Open
Vi3 IPCP: Install route to 54.1.7.254
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access3,
changed state to up
```

Fix the issue

We summarize the configuration changes applied to R6 in this section:

```
R6:
interface Serial 0/0
 frame-relay interface-dlci 201 ppp virtual-template 1
!
interface Virtual-Template 1
 no ppp authentication chap
```

Verify

Check that we have all RIP routes from BB1:

```
Rack1R6#show ip route rip
R    212.18.1.0/24 [120/1] via 54.1.7.254, 00:00:06, Virtual-Access3
R    212.18.0.0/24 [120/1] via 54.1.7.254, 00:00:06, Virtual-Access3
R    212.18.3.0/24 [120/1] via 54.1.7.254, 00:00:06, Virtual-Access3
R    212.18.2.0/24 [120/1] via 54.1.7.254, 00:00:06, Virtual-Access3
```

Ticket 2

Analyze the Symptoms

We’ve got another issue related to R6. We have seen the BGP session between R6 and R2 in “Active” state already when performing BGP session discovery cycle. Let’s see what might be causing this. Looks at the status of BGP peering once again:

```
Rack1R6#show ip bgp summary
BGP router identifier 150.1.6.6, local AS number 100
BGP table version is 11, main routing table version 11
10 network entries using 1170 bytes of memory
10 path entries using 520 bytes of memory
8/4 BGP path/bestpath attribute entries using 992 bytes of memory
2 BGP AS-PATH entries using 48 bytes of memory
1 BGP community entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 2754 total bytes of memory
BGP activity 10/0 prefixes, 10/0 paths, scan interval 60 secs

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
54.1.7.254      4    54      23      22       11    0    0 00:02:33
10
204.12.1.2      4   100       0       0        0    0    0 never
Active
204.12.1.254    4    54       0       0        0    0    0 never
Active
```

Starting with divide-and-conquer approach, check the link connectivity:

```
Rack1R6#ping 204.12.1.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 204.12.1.2, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)

Rack1R6#ping 204.12.1.254

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 204.12.1.254, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

Which makes us suspect of a link failure somewhere between routers. However, this could also be an access list of VLAN filter issue, you never know! So we’ll have to use the bottom-up approach for this issue.

Isolate the Issue

First, check the interface states in R6 and R2:

```
Rack1R6#show ip interface brief
Interface                  IP-Address      OK? Method Status
Protocol
FastEthernet0/0            163.1.6.6       YES manual up
up
Serial0/0                  unassigned      YES manual up
up
FastEthernet0/1            204.12.1.6      YES manual up
up
Virtual-Access1            unassigned      YES unset  down
down
Virtual-Template1          54.1.7.6        YES manual down
down
Virtual-Access2            unassigned      YES TFTP   down
down
Virtual-Access3            54.1.7.6        YES TFTP   up
up
Loopback0                  150.1.6.6       YES manual up
up

Rack1R2#show ip interface brief
Interface                  IP-Address      OK? Method Status
Protocol
FastEthernet0/0            204.12.1.2      YES manual up
up
Serial0/0                  163.1.12.2      YES manual up
up
Serial0/1                  unassigned      YES manual administratively
down down
Loopback0                  150.1.2.2       YES manual up
up
```

So there seems to be everything OK with the interfaces. Let's trace the path from R6 to R2 across the switches and see if everything is all-right there as well. Per our Layer 2 diagram, R6 connects to SW4 and SW4 connects to SW1 which in turn connects to SW2 that has R2 connected. So we have to check three switches on the path between the two devices. We start with the physical layer:

```
Rack1SW4#show cdp neighbors fastEthernet 0/13
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater, P -
Phone

Device ID        Local Intrfce     Holdtme    Capability  Platform
Port ID
Rack1SW1         Fas 0/13          147          R S I     WS-C3560- Fas
0/19

Rack1SW1#show cdp neighbors fastEthernet 0/19
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
```

```
                  S - Switch, H - Host, I - IGMP, r - Repeater, P -
Phone

Device ID        Local Intrfce     Holdtme    Capability  Platform
Port ID
Rack1SW4         Fas 0/19          138          R S I     WS-C3550- Fas
0/13
```

Check SW1's links to SW2 now:

```
Rack1SW1#show cdp neighbors fastEthernet 0/13
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater, P -
Phone

Device ID        Local Intrfce     Holdtme    Capability  Platform
Port ID
Rack1SW2         Fas 0/13          131          R S I     WS-C3560- Fas
0/13

Rack1SW1#show cdp neighbors fastEthernet 0/14
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater, P -
Phone

Device ID        Local Intrfce     Holdtme    Capability  Platform
Port ID
Rack1SW2         Fas 0/14          130          R S I     WS-C3560- Fas
0/14

Rack1SW1#show cdp neighbors fastEthernet 0/15
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater, P -
Phone

Device ID        Local Intrfce     Holdtme    Capability  Platform
Port ID
Rack1SW2         Fas 0/15          128          R S I     WS-C3560- Fas
0/15
```

So this almost rules out the physical issues factor. Now we have to go up one level and check the logical topology for the VLAN connecting R2 and R6. We'll do that by checking the spanning-tree port states. We start from SW2, and then proceed to SW1 and SW4.

```
Rack1SW2#show spanning-tree vlan 263

VLAN0263
  Spanning tree enabled protocol ieee
  Root ID    Priority    33031
             Address     000f.f76d.ac80
             Cost        28
             Port        152 (Port-channel13)
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
  Bridge ID  Priority    33031  (priority 32768 sys-id-ext 263)
             Address     001f.2711.d580
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time 300

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- -----------------------
Fa0/2               Desg FWD 19        128.4    P2p
Po13                Root FWD 9         128.152  P2p

Rack1SW2#
```

From the above output we could see that STP spans to SW1 without any obvious problems.

```
Rack1SW1#show spanning-tree vlan 263

VLAN0263
  Spanning tree enabled protocol ieee
  Root ID    Priority    33031
             Address     000f.f76d.ac80
             Cost        19
             Port        18 (FastEthernet0/16)
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    33031  (priority 32768 sys-id-ext 263)
             Address     001f.6d94.7b80
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time 300

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- -----------------------
Fa0/16              Root FWD 19        128.18   P2p
Po13                Desg FWD 9         128.152  P2p
```

Now when we look at SW1, we only see the STP spanning to SW2 and SW3, but not on the port to SW4. We are going to check if SW4 has the same issue:

```
Rack1SW4#show spanning-tree vlan 263

VLAN0263
  Spanning tree enabled protocol ieee
  Root ID    Priority    33031
             Address     0011.21c4.5d00
             This bridge is the root
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    33031  (priority 32768 sys-id-ext 263)
             Address     0011.21c4.5d00
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time 300

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- -----------------------
Fa0/6               Desg FWD 19        128.6    P2p
```

Correct, SW4 only has the STP instance active on the connection to R6, not on the link to SW1. This may mean some misconfiguration on the link between SW1 and SW4. The link should be trunk, let’s see how it works:

```
Rack1SW4#show interfaces fastEthernet 0/13 trunk

Port        Mode             Encapsulation  Status        Native vlan
Fa0/13      auto             802.1q         not-trunking  1

Port        Vlans allowed on trunk
Fa0/13      1

Port        Vlans allowed and active in management domain
Fa0/13      1

Port        Vlans in spanning tree forwarding state and not pruned
Fa0/13      1
```

So here is a problem, the port is non-trunking. Let’s see what might be causing this:

```
Rack1SW4#show interfaces fastEthernet 0/13 switchport
Name: Fa0/13
Switchport: Enabled
Administrative Mode: dynamic auto
Operational Mode: static access
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: native
Negotiation of Trunking: On
Access Mode VLAN: 1 (default)
Trunking Native Mode VLAN: 1 (default)
Administrative Native VLAN tagging: enabled
Voice VLAN: none
```

This command shows the switch-port configuration. We can see that DTP is enabled, as the mode is “dynamic auto”. The trunking encapsulation is set to 802.1q. Let’s check if these settings match the other side’s:

```
Rack1SW1#show interfaces fastEthernet 0/19 switchport
Name: Fa0/19
Switchport: Enabled
Administrative Mode: dynamic auto
Operational Mode: static access
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: native
Negotiation of Trunking: On
Access Mode VLAN: 1 (default)
```

We see that the other side uses the same DTP auto mode. This is not going to work, as DTP auto/auto does not negotiate a trunk. Let’s make one side of the trunk operating in “dynamic desirable” mode and see what happens:

```
SW1:
interface FastEthernet 0/19
 switchport mode dynamic desirable
```

Check the trunking status after this:

```
Rack1SW1#show interfaces fastEthernet 0/19 trunk

Port        Mode             Encapsulation  Status        Native vlan
Fa0/19      desirable        802.1q         not-trunking  1

Port        Vlans allowed on trunk
Fa0/19      1

Port        Vlans allowed and active in management domain
Fa0/19      1

Port        Vlans in spanning tree forwarding state and not pruned
Fa0/19      none
```

The port is still non-trunking. We may want to disable DTP at all and see if the trunk works this way:

```
SW1:
Interface FastEthernet 0/19
 switchport mode trunk

SW4:
interface FastEthernet 0/13
 switchport mode trunk

Rack1SW1#show interfaces fastEthernet 0/19 trunk

Port        Mode             Encapsulation  Status        Native vlan
Fa0/19      on               802.1q         trunking      1

Port        Vlans allowed on trunk
Fa0/19      1-4094

Port        Vlans allowed and active in management domain
Fa0/19      1,3-7,42,57,100-101,263,500

Port        Vlans in spanning tree forwarding state and not pruned
Fa0/19      1,3-7,42,57,100-101,263,500

Rack1R6#ping 204.12.1.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 204.12.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/4 ms
```

But we have to apply minimal changes why resolving the issue. So let's roll back to DTP and see if there are any misconfigurations preventing it from working properly. Let's see if we have some logging messages, which may give us a hint. It is always a good idea to have logging to the memory buffer enabled and check it from time to time:

```
SW1:
logging buffered debugging
!
interface FastEthernet 0/19
 shutdown
 no shutdown
```

The above command will trigger the DTP negotiation process once again. When you issue the show logging command you may see the following message:

```
%DTP-5-DOMAINMISMATCH: Unable to perform trunk negotiation on port
Fa0/19 because of VTP domain mismatch.
```

Which gives us the exact clue to the DTP problem – VTP domains configured on the switches do no match. Confirm that with the following "show" commands:

```
Rack1SW1#show vtp status
VTP Version                     : running VTP1 (VTP2 capable)
Configuration Revision          : 0
Maximum VLANs supported locally : 1005
Number of existing VLANs        : 16
VTP Operating Mode              : Transparent
VTP Domain Name                 : csico
VTP Pruning Mode                : Disabled
VTP V2 Mode                     : Disabled
VTP Traps Generation            : Disabled
MD5 digest                      : 0x47 0x64 0xC8 0xC1 0x85 0x1D 0x8E
0x21
Configuration last modified by 0.0.0.0 at 0-0-00 00:00:00

Rack1SW4#show vtp status
VTP Version                     : running VTP1 (VTP2 capable)
Configuration Revision          : 0
Maximum VLANs supported locally : 1005
Number of existing VLANs        : 14
VTP Operating Mode              : Transparent
VTP Domain Name                 : cisco
VTP Pruning Mode                : Disabled
VTP V2 Mode                     : Disabled
VTP Traps Generation            : Disabled
MD5 digest                      : 0x28 0xD3 0x99 0x65 0xAE 0x01 0xC0
0xCA
Configuration last modified by 163.1.0.4 at 3-1-93 14:57:06
```

Let's go ahead and fix the domain in SW1 (which looks wrong):

```
SW1:
vtp domain cisco
```

After this, shutdown/unshutdown SW1's interface and see what happens:

```
Rack1SW1#show interfaces fastEthernet 0/19 trunk

Port        Mode             Encapsulation  Status        Native vlan
Fa0/19      desirable        802.1q         trunking      1

Port        Vlans allowed on trunk
Fa0/19      1-4094

Port        Vlans allowed and active in management domain
Fa0/19      1,3-7,42,57,100-101,263,500

Port        Vlans in spanning tree forwarding state and not pruned
Fa0/19      none
```

Now the trunking works. Let's check connectivity between R2 and R6 once again:

```
Rack1R6#ping 204.12.1.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 204.12.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

Conclusion: Observing system logs is critical for proper troubleshooting. Always make sure you have logging configured to the system buffer, or the console line. If you log to the console, make sure to rate-limit the system messages.

Fix the Issue

The following is the summary of the changes that we have applied to resolve the issue:

```
SW1:
vtp domain cisco
!
interface FastEthernet 0/19
 switchport mode dynamic desirable
```

Effectively we corrected the VTP domain name and changed DTP mode to desirable at one side. The VTP domain names must match when using DTP for trunk negotiation.

Verify

Check the status of the BGP sessions to confirm that everything is OK now:

```
Rack1R6#show ip bgp summary
BGP router identifier 150.1.6.6, local AS number 100
BGP table version is 31, main routing table version 31
12 network entries using 1404 bytes of memory
22 path entries using 1144 bytes of memory
12/5 BGP path/bestpath attribute entries using 1488 bytes of memory
1 BGP rrinfo entries using 24 bytes of memory
2 BGP AS-PATH entries using 48 bytes of memory
1 BGP community entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 4132 total bytes of memory
BGP activity 16/4 prefixes, 76/54 paths, scan interval 60 secs

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
54.1.7.254      4    54    1408    1417       31    0    0 03:53:23
10
204.12.1.2      4   100      47      48       31    0    0 00:10:25
2
204.12.1.254    4    54      41      53       31    0    0 00:10:44
10

All of them seems to be in normal state now.
```

Ticket 3

Analyze the Symptoms

The ticket already suggests the possible cause for the problem. We can't fully trust the initial information, so we run the **traceroute** tool to check the path taken by the packets from VLAN5 to VLAN42:

```
Rack1R5#traceroute 192.10.1.4 source fastEthernet 0/0

Type escape sequence to abort.
Tracing the route to 192.10.1.4

  1 163.1.45.4 16 msec *  12 msec
```

The next hop is R4 with the IP address on the Serial link. Based on the diagrams we may have guesses that R5 should prefer reaching VLAN42 using OSPF for information source. Let's check the routing table on R5:

```
Rack1R5#show ip route 192.10.1.0
Routing entry for 192.10.1.0/24
  Known via "rip", distance 120, metric 1
  Redistributing via rip
  Last update from 163.1.45.4 on Serial0/1, 00:00:21 ago
  Routing Descriptor Blocks:
  * 163.1.45.4, from 163.1.45.4, 00:00:21 ago, via Serial0/1
      Route metric is 1, traffic share count is 1
```

The route is preferred via RIP. Now let's see the routing protocols running on R5 and the routing information sources.

```
Rack1R5#show ip protocols
Routing Protocol is "ospf 1"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Router ID 150.1.5.5
  It is an area border router
  Number of areas in this router is 2. 2 normal 0 stub 0 nssa
  Maximum path: 4
  Routing for Networks:
    150.1.5.5 0.0.0.0 area 0
    163.1.5.5 0.0.0.0 area 0
    163.1.15.5 0.0.0.0 area 0
    163.1.35.5 0.0.0.0 area 1
    163.1.54.5 0.0.0.0 area 0
 Reference bandwidth unit is 100 mbps
  Routing Information Sources:
    Gateway         Distance      Last Update
    (this router)        110      00:09:16
    150.1.3.3            110      00:09:16
    150.1.2.2            110      00:09:16
    150.1.1.1            110      00:09:16
  Distance: (default is 110)
```

```
Routing Protocol is "rip"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Sending updates every 30 seconds, next due in 23 seconds
  Invalid after 180 seconds, hold down 180, flushed after 240
  Redistributing: rip
  Default version control: send version 2, receive version 2
    Interface             Send  Recv  Triggered RIP  Key-chain
    FastEthernet0/1       2     2
    Serial0/1             2     2
  Automatic network summarization is not in effect
  Maximum path: 4
  Routing for Networks:
    163.1.0.0
  Passive Interface(s):
    FastEthernet0/0
    Serial0/0
    Serial0/0.35
    Serial0/0.54
    Loopback0
    Tunnel0
    VoIP-Null0
  Routing Information Sources:
    Gateway         Distance      Last Update
    Gateway         Distance      Last Update
    163.1.45.4           120      00:00:01
    163.1.57.7           120      00:00:01
  Distance: (default is 120).
```

As you can easily see, R5 does not see R4 as routing information source. This probably means that R4 is not R5'S OSPF neighbor:

```
Rack1R5#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address
Interface
150.1.1.1         0   FULL/  -        00:00:32    163.1.15.1
Tunnel0
150.1.3.3         0   FULL/  -        00:00:39    163.1.35.3
Serial0/0.35
```

Which is true. Thus, we're troubleshooting the OSPF adjacency issue.

Isolate the Issue

Using divide-and-conquer approach we start by checking if L3 is healthy between R4 and R5:

```
Rack1R5#ping 163.1.54.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 163.1.54.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/32 ms
```

The link appears to be healthy. Let's see why we can't establish any OSPF adjacency. Start by activating OSPF adjacency debugging:

```
Rack1R5#debug interface serial 0/0.54
Condition 1 set

Rack1R5#debug ip ospf adj
OSPF adjacency events debugging is on
```

Time passes and we don't see any messages. Let's see if OSPF is configured on the Frame-Relay interface at all:

```
Rack1R5#show ip ospf interface serial 0/0.54
Serial0/0.54 is down, line protocol is down
  Internet Address 163.1.54.5/24, Area 0
  Process ID 1, Router ID 150.1.5.5, Network Type POINT_TO_POINT, Cost:
64
  Transmit Delay is 1 sec, State DOWN,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    oob-resync timeout 40
```

Wait a minute, the interface appears to be down! But we just pinged across with no problems. Let's ping again, maybe something has changed:

```
Rack1R5#ping 163.1.54.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 163.1.54.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/29/32 ms
```

This does not make any sense. Let's check the route for the other end:

```
Rack1R5#show ip route 163.1.54.4
Routing entry for 163.1.54.0/24
  Known via "rip", distance 120, metric 1
  Redistributing via rip
  Last update from 163.1.45.4 on Serial0/1, 00:00:18 ago
  Routing Descriptor Blocks:
```

```
  * 163.1.45.4, from 163.1.45.4, 00:00:18 ago, via Serial0/1
      Route metric is 1, traffic share count is 1
```

This clears something. We see that the network is reachable via the backup link to R4. This is why pings succeeded, even though the interface is down. Let's find out what the problem with the Frame-Relay interface is. Check the physical interface first:

```
Rack1R5#show interfaces serial 0/0
Serial0/0 is up, line protocol is up
  Hardware is PowerQUICC Serial
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation FRAME-RELAY, loopback not set
  Keepalive set (10 sec)
  LMI enq sent  285, LMI stat recvd 286, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent  0, LMI upd sent  0
  LMI DLCI 1023  LMI type is CISCO  frame relay DTE
  FR SVC disabled, LAPF state down
<snip>
```

The interface appears to be healthy, that means we don't have problems between the local router and the FR switch. Let's check for the DLCI mapped to the selected subinterface:

```
Rack1R5#show frame-relay pvc 504

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 504, DLCI USAGE = UNUSED, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0

  input pkts 79            output pkts 0            in bytes 869
  out bytes 0              dropped pkts 0           in pkts dropped 0
  out pkts dropped 0                out bytes dropped 0
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0          in DE pkts 79            out DE pkts 0
```

The DLCI appears to be unused, that means it is not associated with any local interface. Let's see what other DLCIs are configured:

```
Rack1R5#show frame-relay pvc | inc ST
DLCI = 405, DLCI USAGE = LOCAL, PVC STATUS = DELETED, INTERFACE =
Serial0/0.54
DLCI = 501, DLCI USAGE = UNUSED, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0
DLCI = 502, DLCI USAGE = UNUSED, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0
DLCI = 503, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0.35
DLCI = 504, DLCI USAGE = UNUSED, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0
```

```
DLCI = 513, DLCI USAGE = UNUSED, PVC STATUS = INACTIVE, INTERFACE =
Serial0/0
```

The DLCI mapped to the 0/0.54 subinterface is 405, not 504 and the Frame-Relay switch reports it as DELETED. Let' go ahead and configure the proper DLCI.

```
R5:
Interface Serial 0/0.54
 frame-relay interface-dlci 504
```

We check L3 again after this configuration and:

```
Rack1R5#ping 163.1.54.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 163.1.54.4, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

It's not working. So what may be causing this? Let's see if the network is directly connected now:

```
Rack1R5#show ip route 163.1.54.4
Routing entry for 163.1.54.0/24
  Known via "connected", distance 0, metric 0 (connected, via
interface)
  Redistributing via rip
  Advertised by rip
  Routing Descriptor Blocks:
  * directly connected, via Serial0/0.54
      Route metric is 0, traffic share count is 1
```

It is. Now we're getting to R4 and trying to find the issue there.

```
Rack1R4#show interfaces serial 0/0
Serial0/0 is up, line protocol is up
  Hardware is PowerQUICC Serial
  Internet address is 163.1.54.4/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation FRAME-RELAY, loopback not set
  Keepalive set (10 sec)
  LMI enq sent  350, LMI stat recvd 351, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent  0, LMI upd sent  0
  LMI DLCI 1023  LMI type is CISCO  frame relay DTE
<snip>
```

The physical interface and the LMI are in normal condition. Checking the PVCs next:

```
Rack1R4#show frame-relay pvc | inc ST
```

```
DLCI = 401, DLCI USAGE = UNUSED, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0
DLCI = 402, DLCI USAGE = UNUSED, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0
DLCI = 403, DLCI USAGE = UNUSED, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0
DLCI = 405, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE (EEK DOWN),
INTERFACE = Serial0/0
DLCI = 413, DLCI USAGE = UNUSED, PVC STATUS = INACTIVE, INTERFACE =
Serial0/0
```

And we can see that 405 shows EEK DOWN state. What this means is that R4 uses End-to-End keepalives and R5 is not responding. Apparently, we may have erased the EEK configuration on R5 when removing the old DLCI! To restore the configuration, we may look at the EEK parameters in R4:

```
Rack1R4#show frame-relay end-to-end keepalive

End-to-end Keepalive Statistics for Interface Serial0/0 (Frame Relay
DTE)

DLCI = 405, DLCI USAGE = LOCAL, VC STATUS = ACTIVE (EEK DOWN)

SEND SIDE STATISTICS

Send Sequence Number: 190,      Receive Sequence Number: 1
Configured Event Window: 3,     Configured Error Threshold: 2
Total Observed Events: 192,     Total Observed Errors: 189
Monitored Events: 3,            Monitored Errors: 3
Successive Successes: 0,        End-to-end VC Status: DOWN
```

We may try configuring the settings on R5 to match this output. However, there is a shorter way. We may look for the frame-relay maps in R5's running configuration! While this is against the "rules", we are in an emergency condition as we may have de-associated the proper map-class in R5. Remember, even though the scenario prohibits the use of "show run" command, the sole purpose of this is to prevent you from peeking the configuration changes and forcing to use more effective approaches.

```
Rack1R5#show running-config map-class
Building configuration...

Current configuration:
!
map-class frame-relay DLCI_503
 frame-relay cir 768000
!
map-class frame-relay DLCI_504
 frame-relay end-to-end keepalive mode reply
 frame-relay cir 768000
 frame-relay bc 7680
 service-policy output FR_QOS
end
```

This worked pretty well as we can quickly spot the map-class corresponding to DLCI 504. This is good in lab environment since we don't have hundreds of map-classes configured. In real life, you should be careful when erasing something from the interface, and always keep a copy of the original configuration.

We re-apply the map-class to the DLCI on R5's subinterface and check the PVC status once again:

```
R5:
interface Serial 0/0.54
 frame-relay interface-dlci 504
   class DLCI_504

Rack1R4#show frame-relay pvc | inc ST
DLCI = 401, DLCI USAGE = UNUSED, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0
DLCI = 402, DLCI USAGE = UNUSED, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0
DLCI = 403, DLCI USAGE = UNUSED, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0
DLCI = 405, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE (EEK UP), INTERFACE
= Serial0/0
DLCI = 413, DLCI USAGE = UNUSED, PVC STATUS = INACTIVE, INTERFACE =
Serial0/0
```

Now EEK status shows as up and we can ping the other end!

```
Rack1R5#ping 163.1.54.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 163.1.54.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/58/60 ms
```

But the OSPF adjacency is still not up.

```
Rack1R5#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address
Interface
150.1.1.1         0   FULL/  -        00:00:32    163.1.15.1
Tunnel0
150.1.3.3         0   FULL/  -        00:00:38    163.1.35.3
Serial0/0.35
```

What might be the reason for this? Let's start by comparing OSPF settings at both ends:

```
Rack1R5#show ip ospf interface serial 0/0.54
Serial0/0.54 is up, line protocol is up
  Internet Address 163.1.54.5/24, Area 0
  Process ID 1, Router ID 150.1.5.5, Network Type POINT_TO_POINT, Cost:
64
  Transmit Delay is 1 sec, State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    oob-resync timeout 40
    Hello due in 00:00:08
  Supports Link-local Signaling (LLS)
  Index 4/5, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 0, maximum is 0
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 0, Adjacent neighbor count is 0
  Suppress hello for 0 neighbor(s)

Rack1R4#show ip ospf interface serial 0/0
Serial0/0 is up, line protocol is up
  Internet Address 163.1.54.4/24, Area 0
  Process ID 1, Router ID 150.1.4.4, Network Type POINT_TO_POINT, Cost:
64
  Transmit Delay is 1 sec, State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    oob-resync timeout 40
    Hello due in 00:00:09
  Supports Link-local Signaling (LLS)
  Index 3/3, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 0, maximum is 0
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 1, Adjacent neighbor count is 0
  Suppress hello for 0 neighbor(s)
```

Subnet addresses match, timers match and network types match as well. The only difference is that R4 sees R5, but not vice versa. Since we tested unicast connectivity, this may be caused by unidirectional multicast traffic. That is, R4 cannot send multicast to R5. Let's see if the Frame-Relay mappings are configured for broadcast relaying:

```
Rack1R4#show frame-relay map
Serial0/0 (up): ipv6 FEC0:CC1E:1:54::5 dlci 405(0x195,0x6450), static,
              broadcast,
              CISCO, status defined, active
Serial0/0 (up): ipv6 FE80::5 dlci 405(0x195,0x6450), static,
              CISCO, status defined, active
Serial0/0 (up): ip 163.1.54.5 dlci 405(0x195,0x6450), static,
              CISCO, status defined, active
```

Right in the spot, there is no broadcast keyword next to the IP address of R5 So we go ahead and change the Frame-Relay mapping in R4:

R4:
```
interface Serial 0/0
 frame-relay map ip 163.1.54.5 405 broadcast
```

And now we check for OSPF neighbors:

```
Rack1R4#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address
Interface
150.1.5.5         0   FULL/  -        00:00:35    163.1.54.5
Serial0/0
```

Finally getting them up. Now we check routing to VLAN42 subnet:

```
Rack1R5#show ip route 192.10.1.0
Routing entry for 192.10.1.0/24
  Known via "ospf 1", distance 110, metric 20, type extern 2, forward
metric 64
  Last update from 163.1.54.4 on Serial0/0.54, 00:02:02 ago
  Routing Descriptor Blocks:
  * 163.1.54.4, from 150.1.4.4, 00:02:02 ago, via Serial0/0.54
      Route metric is 20, traffic share count is 1
```

Everything goes across the Frame-Relay cloud.

Fix the Issue

We put the things we fixed during the isolation stage together here. We corrected the wrong DLCI number and applied the missing broadcast parameter at R4.

```
R5:
interface Serial 0/0.54
 no frame-relay interface 405
 frame-relay interface-dlci 504
  class DLCI_504

R4:
interface Serial 0/0
 frame-relay map ip 163.1.54.5 405 broadcast
```

Verify

Make sure we reach VLAN42 across the Frame-Relay cloud now:

```
Rack1R5#traceroute 192.10.1.4 source fastEthernet 0/0

Type escape sequence to abort.
Tracing the route to 192.10.1.4

  1 163.1.54.4 28 msec *  48 msec
Rack1R5#õ
```

In addition to that, make sure R4 prefers reaching VLAN5 across the Frame-Relay interface as well:

```
Rack1R4#show ip route 163.1.5.0
Routing entry for 163.1.5.0/24
  Known via "ospf 1", distance 110, metric 65, type intra area
  Redistributing via rip
  Advertised by rip metric 1
  Last update from 163.1.54.5 on Serial0/0, 00:08:31 ago
  Routing Descriptor Blocks:
  * 163.1.54.5, from 150.1.5.5, 00:08:31 ago, via Serial0/0
      Route metric is 65, traffic share count is 1
```

Ticket 4

Analyze the Symptoms

This is the ticket that actually specifies the IGP redistribution settings and requires us performing redistribution loop analysis. We start by taking a simplified IGP diagram and converting it into the IGP redistribution diagram. As usual, we use arrows to outline the flow of IGP routing information.

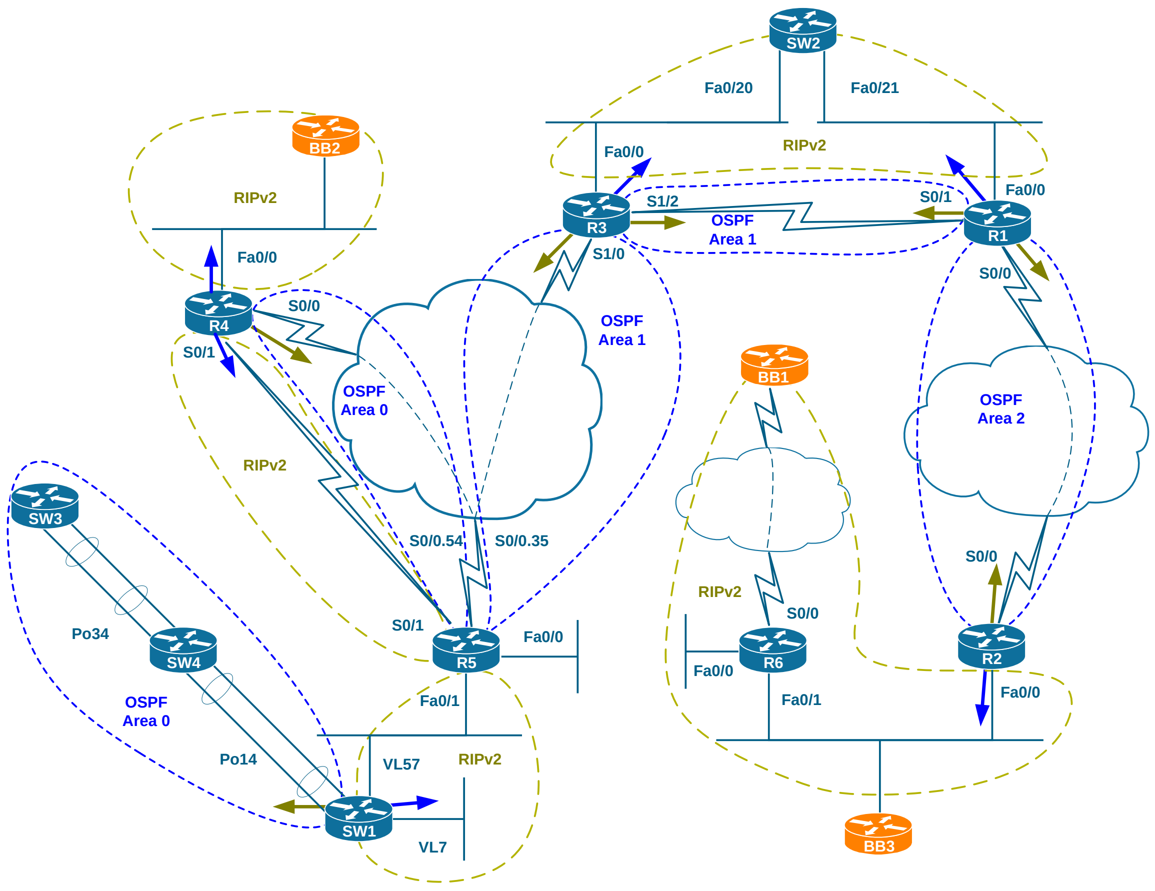


Fig 3

Apparently, the topology has routing domains with multiple redistribution points, so this opens possibility for routing loops. Let's apply our tracing procedure outlined in **Lab 1** solution to every domain found in the topology. We start by simplifying the diagram, and outlining four major IGP routing domains: A, B, C and D (see the figure below). Notice that Domain D actually covers both OSPF and RIPv2 protocols. This is possible because OSPF has only one connection to the RIPv2 domain on SW1, and all OSPF routes are visible to other domains via RIPv2.

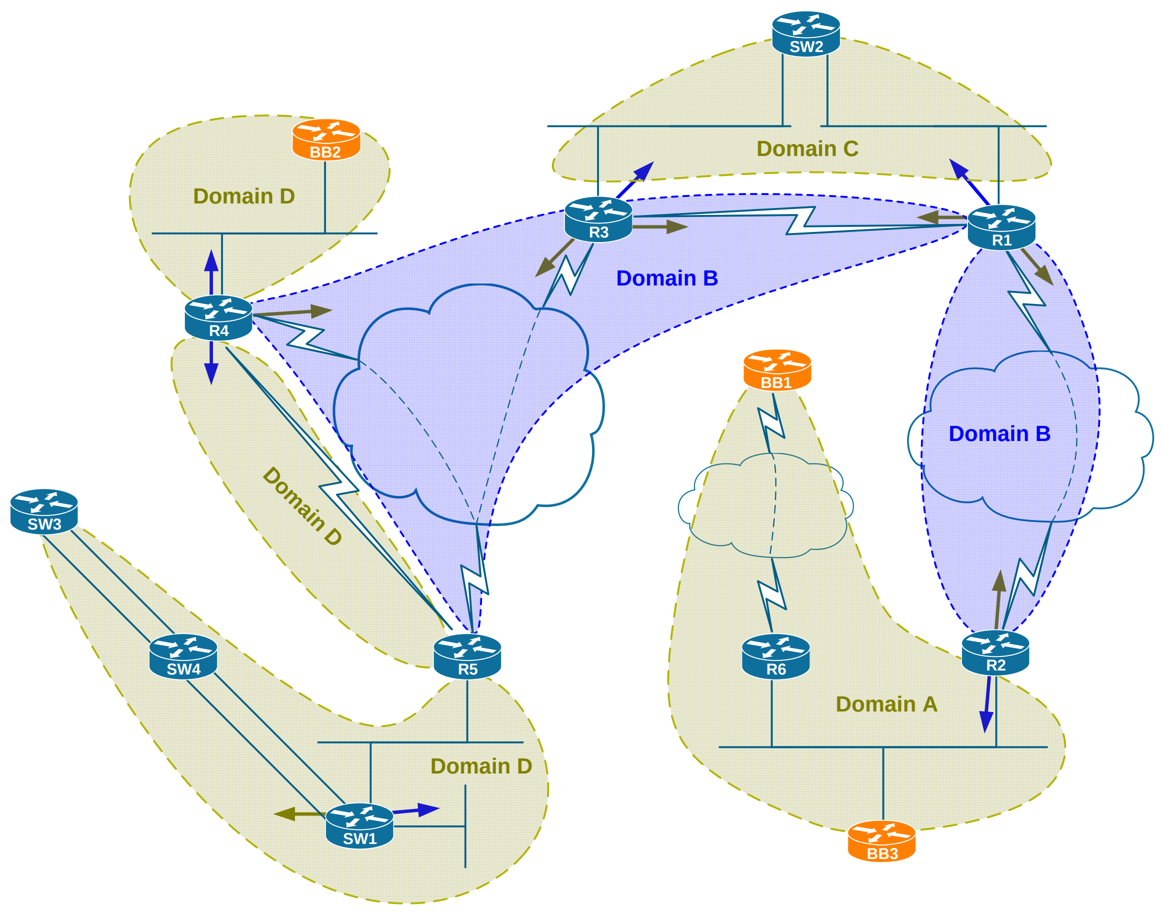


Fig 4

Now we need to trace the information flows for all four domains. Let's start with Domain A. The information from this domain enters Domain B via R2 and then splits at R1. Domain A routes arrive as redistributed to SW2 via RIPv2 and reach R3 as well. However, the same prefixes reach R3 via OSPF, and so R3 would prefer reaching Domain A via R1. R3 will not redistribute Domain A route back to OSPF because information from RIPv2 has higher admin distance. Finally, Domain A information arrives to R4 and R5. R4 will successfully propagate it to BB2 and advertise it down to R5. However, R5 will prefer Domain A routes learned via OSPF. Thus, Domain A information will not go down to SW1, as there is no redistribution configured on R5. This is the first issue we spotted: in order for Domain A routes to reach Domain D, R5 should be configured to prefer Domain A routes advertised via RIP from R4.

Another issue with Domain A is that RIP routes may enter Domain B and the wind back through Domain C re-entering Domain B. Due to the better AD of OSPF, RIP prefixes at R2 will be wiped out by the same prefixes learned via OSPF. This means we need configuring RIP process in R2 so that the native prefixes are now pre-empted by external OSPF routes. Usually it is enough to set RIPv2 AD to a value lower than that of OSPF.

Looking at Domain B routes, we can easily see that those will reach Domain A without any problems, traversing R1 and R2 where needed. When being advertised to Domain C via R1 an R3 Domain B routes will never come back, as OSPF has better administrative distance than RIPv2. However, Domain B routes will never reach to SW1 due to the same issue that prevents Domain A routes from getting there. R4 will inject Domain B prefixes into RIP and R5 will ignore those and not pass down to SW1. R5 should be additionally configured to honor these prefixes and prefer then via RIP. Obviously, this issue will also apply to Domain C routes injected into Domain B.

Next for Domain C we can quickly see that it prefixes will reach Domain A. However, we may quickly see that both R1 and R3 will prefer Domain C prefixes via OSPF after they have been redistributed there. This will form a routing loop, as they both attempt to advertise Domain C information back to the same domain. The simplest way to fix it is configure R1 and R3 to prefer Domain C routes via RIPv2.

Finally, for Domain D we may see the following. It will propagate to Domain B via R4, and reach R5 as OSPF prefixes. R5 will immediately prefer OSPF learned Domain B prefixes via RIPv2 learned and stop advertising them to R4 via RIP. This will break redistribution and lead to a constant oscillation of Domain D prefixes on R4 and R5. To prevent this, we should make sure Domain D prefixes are more preferred via RIPv2 than via OSPF on R5. Another issue is that Domain D prefixes may loop back to OSPF thanks to the mutual redistribution configured on R1 and R3. This will effectively wipe RIPv2 routes from R4 in favor of the same prefixes learned via OSPF. Thus, R4 should be configured properly to “guard” RIPv2 prefixes.

Now let’s see how the above issues manifest themselves in the network and try fixing the redistribution step by step.

Isolate the Issue

From the previous step, we figured the routers that are should be properly configured to ensure optimal routing: R5, R1, R2 and R3. Let's start with R2, as we know RIP routes may have some problems there due to AD mismatch.

```
R2:
router rip
 distance 109
```

The above configuration ensures that RIP routes re-injected into OSPF will never preempt the native RIP prefixes learned from R6/BB3.

Now for R5. We know that due to OSPF having better administrative distance then RIP, the routers behind R5 cannot learn about Domain A, B and C routes. Check this by using the following command:

```
Rack1SW1#show ip route rip
     163.1.0.0/16 is variably subnetted, 11 subnets, 3 masks
R       163.1.35.0/24 [120/1] via 163.1.57.5, 00:00:14, Vlan57
R       163.1.45.4/32 [120/1] via 163.1.57.5, 00:00:14, Vlan57
R       163.1.45.0/24 [120/1] via 163.1.57.5, 00:00:14, Vlan57
R       163.1.54.0/24 [120/1] via 163.1.57.5, 00:00:14, Vlan57
R       163.1.5.0/24 [120/1] via 163.1.57.5, 00:00:14, Vlan57
R       163.1.15.0/24 [120/1] via 163.1.57.5, 00:00:14, Vlan57
     150.1.0.0/24 is subnetted, 2 subnets
R       150.1.4.0 [120/2] via 163.1.57.5, 00:00:14, Vlan57
```

As you can see, only the directly connected routes of R5 and those routes of R4 that are not advertised into OSPF are learned by SW1.

We cannot easily fix this issue without configuring redistribution in R5, which is prohibited. However, we may instruct R5 to originate a default route into RIP as the scenario permits this. The RIPv2 process will originate a default route without any matching local default route. Thus, we configure

```
R5:
router rip
 default-information originate

Rack1SW1#show ip route rip
     163.1.0.0/16 is variably subnetted, 11 subnets, 3 masks
R       163.1.35.0/24 [120/1] via 163.1.57.5, 00:00:20, Vlan57
R       163.1.45.4/32 [120/1] via 163.1.57.5, 00:00:20, Vlan57
R       163.1.45.0/24 [120/1] via 163.1.57.5, 00:00:20, Vlan57
R       163.1.54.0/24 [120/1] via 163.1.57.5, 00:00:20, Vlan57
R       163.1.5.0/24 [120/1] via 163.1.57.5, 00:00:20, Vlan57
R       163.1.15.0/24 [120/1] via 163.1.57.5, 00:00:20, Vlan57
     150.1.0.0/24 is subnetted, 2 subnets
R       150.1.4.0 [120/2] via 163.1.57.5, 00:00:20, Vlan57
R*   0.0.0.0/0 [120/1] via 163.1.57.5, 00:00:02, Vlan57
```

Good, now the second issue with R5 was that it cannot consistently see the routes learned from SW1 via RIP. This behavior has been outlined above and the problem is that RIP routes injected into OSPF on R4 preempts the same RIP routes learned by R5 from SW1 and resulting in oscillating behavior. This could be easily seen using the following debugging command:

```
Rack1R5#debug ip routing
IP routing debugging is on

RT: SET_LAST_RDB for 10.0.0.0/8
  NEW rdb: via 163.1.57.7

RT: add 10.0.0.0/8 via 163.1.57.7, rip metric [120/1]
RT: NET-RED 10.0.0.0/8
RT: SET_LAST_RDB for 150.1.7.0/24
  NEW rdb: via 163.1.57.7

RT: add 150.1.7.0/24 via 163.1.57.7, rip metric [120/1]
RT: NET-RED 150.1.7.0/24
RT: SET_LAST_RDB for 163.1.0.0/25
  NEW rdb: via 163.1.57.7

RT: add 163.1.0.0/25 via 163.1.57.7, rip metric [120/1]
RT: NET-RED 163.1.0.0/25
RT: SET_LAST_RDB for 163.1.0.128/25
  NEW rdb: via 163.1.57.7

RT: add 163.1.0.128/25 via 163.1.57.7, rip metric [120/1]
RT: NET-RED 163.1.0.128/25
RT: SET_LAST_RDB for 163.1.3.0/24
  NEW rdb: via 163.1.57.7

RT: add 163.1.3.0/24 via 163.1.57.7, rip metric [120/1]
RT: NET-RED 163.1.3.0/24
RT: SET_LAST_RDB for 163.1.7.0/24
  NEW rdb: via 163.1.57.7

RT: add 163.1.7.0/24 via 163.1.57.7, rip metric [120/1]
RT: NET-RED 163.1.7.0/24
RT: closer admin distance for 10.0.0.0, flushing 1 routes
RT: NET-RED 10.0.0.0/8
RT: SET_LAST_RDB for 10.0.0.0/8
  NEW rdb: via 163.1.54.4

RT: add 10.0.0.0/8 via 163.1.54.4, ospf metric [110/20]
RT: NET-RED 10.0.0.0/8
RT: closer admin distance for 150.1.7.0, flushing 1 routes
RT: NET-RED 150.1.7.0/24
RT: SET_LAST_RDB for 150.1.7.0/24
  NEW rdb: via 163.1.54.4

RT: add 150.1.7.0/24 via 163.1.54.4, ospf metric [110/20]
RT: NET-RED 150.1.7.0/24
RT: closer admin distance for 163.1.0.0, flushing 1 routes
```

```
RT: NET-RED 163.1.0.0/25
RT: SET_LAST_RDB for 163.1.0.0/25
  NEW rdb: via 163.1.54.4

RT: add 163.1.0.0/25 via 163.1.54.4, ospf metric [110/20]
RT: NET-RED 163.1.0.0/25
RT: closer admin distance for 163.1.0.128, flushing 1 routes
RT: NET-RED 163.1.0.128/25
RT: SET_LAST_RDB for 163.1.0.128/25
  NEW rdb: via 163.1.54.4

RT: add 163.1.0.128/25 via 163.1.54.4, ospf metric [110/20]
RT: NET-RED 163.1.0.128/25
RT: closer admin distance for 163.1.3.0, flushing 1 routes
RT: NET-RED 163.1.3.0/24
RT: SET_LAST_RDB for 163.1.3.0/24
  NEW rdb: via 163.1.54.4

RT: add 163.1.3.0/24 via 163.1.54.4, ospf metric [110/20]
RT: NET-RED 163.1.3.0/24
RT: closer admin distance for 163.1.7.0, flushing 1 routes
RT: NET-RED 163.1.7.0/24
RT: SET_LAST_RDB for 163.1.7.0/24
  NEW rdb: via 163.1.54.4
```

You can see the same routes being learned via RIP and OSPF in turns. In order to stop this behavior, we could make OSPF external distance higher than the AD of RIP. However, this will make R5 prefer route to VLAN42 via RIP. Thus, we will configure R5 to make RIP routes learned from SW1 having better distance than OSPF.

R5:
```
router rip
 distance 109 163.1.57.7 0.0.0.0
```

Use the following command to make sure the routing table is now stable:

```
Rack1R5#debug ip routing
IP routing debugging is on
Rack1R5#
```

After R5, we should go ahead and deal with R4. As we noticed, routes learned by R4 via RIP leak into OSPF and may loop back to R4 via two-way redistribution in R1 and R3. In order to prevent this most unwanted behavior we may do either of the following:

1) Configure R4’s OSPF process with external distance higher than that of RIP, e.g. 121
2) Configure R1 and R3 to prevent “route looping” by applying tag based filtering.

The first method appears to be simpler, and thus we configure R4:

```
R4:
router ospf 1
 distance ospf external 121
```

Now we need to deal with R1 and R3. Per the task requirements we need to make sure that R1 and R3 prefer the fast Ethernet links to reach each other. Basically, there are two ways to accomplish this

1) Configure RIP with better AD on both R1 and R3
2) Configure OSPF with worse AD on both R1 and R3

The second way is preferred, as we can selectively tune internal/external AD for OSPF routing process. Let’s see what happens if we configure both R1 and R3 with OSPF default AD of 121.

```
R1, R3:
router ospf 1
 distance ospf 120
```

1) OSPF prefixes learned by R1 will be injected into RIP and preempt OSPF prefixes in R3’s routing table. This will make R3 use RIP routes to reach the prefixes behind R5, which is unwanted.
2) OSPF prefixes learned by R3 will be injected into RIP and preferred by R1 over the same prefixes learned via OSPF. This is basically OK, as we want R1 to travel across RIP domain and use the Serial link for backup.

In order to fix the first problem, we configure R3 to prefer OSPF routes learned from R5 and R4 over the same routes learned via RIP as follows:

```
R3:
router ospf 1
 distance 110 150.1.5.5 0.0.0.0
 distance 110 150.1.4.4 0.0.0.0
```

As we implemented this fix, let’s quickly check if we can traceroute across the network in optimal manner:

```
Rack1R2#traceroute 150.1.7.7

Type escape sequence to abort.
Tracing the route to 150.1.7.7

  1 163.1.12.1 28 msec 28 msec 28 msec
  2 163.1.18.8 28 msec 28 msec 28 msec
  3 163.1.38.3 28 msec 28 msec 28 msec
  4 163.1.13.1 36 msec 32 msec 36 msec
  5 163.1.18.8 32 msec 36 msec 32 msec
  6 163.1.38.3 33 msec 36 msec 32 msec
  7 163.1.13.1 40 msec 40 msec 40 msec
  8 163.1.18.8 40 msec 64 msec 40 msec
  9 163.1.38.3 40 msec 40 msec 40 msec
 10 163.1.13.1 44 msec 44 msec 44 msec
 11 163.1.18.8 49 msec 44 msec 44 msec
 12 163.1.38.3 44 msec 48 msec 44 msec
 13 163.1.13.1 48 msec 52 msec 52 msec
```

Too bad, there is a loop between R1 and R3. Let's see why it may be happening:

```
Rack1R3#show ip route ospf
     163.1.0.0/16 is variably subnetted, 17 subnets, 3 masks
O E2    163.1.0.128/25 [121/20] via 163.1.13.1, 00:02:56, Serial1/2
O E2    163.1.45.5/32 [121/20] via 163.1.13.1, 00:02:56, Serial1/2
O E2    163.1.45.0/24 [121/20] via 163.1.13.1, 00:02:56, Serial1/2
O IA    163.1.54.0/24 [110/845] via 163.1.35.5, 00:02:56, Serial1/0
O E2    163.1.57.0/24 [121/20] via 163.1.13.1, 00:02:56, Serial1/2
O E2    163.1.3.0/24 [121/20] via 163.1.13.1, 00:02:56, Serial1/2
O E2    163.1.0.0/25 [121/20] via 163.1.13.1, 00:02:56, Serial1/2
O E2    163.1.7.0/24 [121/20] via 163.1.13.1, 00:02:56, Serial1/2
O IA    163.1.4.0/24 [110/846] via 163.1.35.5, 00:02:56, Serial1/0
O IA    163.1.5.0/24 [110/782] via 163.1.35.5, 00:02:56, Serial1/0
O IA    163.1.15.0/24 [110/10781] via 163.1.35.5, 00:02:56, Serial1/0
                      [110/10781] via 163.1.13.1, 00:02:56, Serial1/2
O E2 10.0.0.0/8 [121/20] via 163.1.13.1, 00:02:56, Serial1/2
O E2 192.10.1.0/24 [121/20] via 163.1.13.1, 00:02:56, Serial1/2
     150.1.0.0/16 is variably subnetted, 9 subnets, 3 masks
O E2    150.1.7.0/24 [121/20] via 163.1.13.1, 00:02:58, Serial1/2
O IA    150.1.4.0/23 [110/782] via 163.1.35.5, 00:02:58, Serial1/0
```

R3 prefers the routes behind R5 via R1. This is because these prefixes are now injected into RIP and then come back to OSPF via R3. The cost to reach these prefixes via R3 beats the cost of reaching via directly via R5, hence the problem. To fix the issue, we may adjust the redistribution on R3 so that external prefixes now have large external cost:

```
R1, R3:
router ospf 1
 redistribute rip metric 10000 subnets
```

It makes sense to apply the same on R3, as the redistribution scheme is symmetrical. After this, it looks like we have reached all the goals and made the redistribution working, using optimal paths.

Conclusion: When fixing redistribution issues, consider the following general rules:

1) Native prefixes for one protocol should be reached via this protocol
2) External prefixes should have AD that is worse than the AD of the native prefixes

3) When injecting external information into a protocol, assign it a high metric value to make it less preferred.

These rules do not guarantee loop-free topology, but they could be used as steps to fix the broken redistribution.

Fix the Issue

It took quite a while to figure all redistribution caveats and fix all of them. There are other solutions possible, such as using tag-based filtering on R1 and R3, but the ticket does not restrict us to any particular solution method. We summarize here all the changes that we had to apply:

```
R1:
router ospf 1
 redistribute rip metric 10000 subnets
 distance ospf 120

R2:
router rip
 distance 109

R3:
router ospf 1
 distance 110 150.1.5.5 0.0.0.0
 distance 110 150.1.4.4 0.0.0.0
 redistribute rip metric 10000 subnets
 distance ospf 120

R4:
router ospf 1
 distance ospf external 121

R5:
router rip
 default-information originate
 distance 109 163.1.57.7 0.0.0.0
```

Notice that we only used AD and metric manipulation to fix the issue.

Verify

It would be too much to perform full routing path verification, so we would just trace routes between the edge networks of the topology:

```
Rack1R4#traceroute 150.1.7.7

Type escape sequence to abort.
Tracing the route to 150.1.7.7

  1 163.1.45.5 16 msec 16 msec 16 msec
  2 163.1.57.7 16 msec *  16 msec

Rack1R4#traceroute 204.12.1.2

Type escape sequence to abort.
Tracing the route to 204.12.1.2

  1 163.1.54.5 28 msec 32 msec 28 msec
  2 163.1.35.3 88 msec 60 msec 60 msec
  3 163.1.38.8 56 msec 56 msec 60 msec
  4 163.1.18.1 60 msec 61 msec 56 msec
  5 163.1.12.2 84 msec *  84 msec
Rack1R4#

Rack1R2#traceroute 192.10.1.4

Type escape sequence to abort.
Tracing the route to 192.10.1.4

  1 163.1.12.1 28 msec 28 msec 28 msec
  2 163.1.18.8 32 msec 28 msec 28 msec
  3 163.1.38.3 32 msec 28 msec 28 msec
  4 163.1.35.5 56 msec 61 msec 56 msec
  5 163.1.54.4 84 msec *  84 msec

Rack1R2#traceroute 150.1.7.7

Type escape sequence to abort.
Tracing the route to 150.1.7.7

  1 163.1.12.1 28 msec 28 msec 28 msec
  2 163.1.18.8 28 msec 32 msec 32 msec
  3 163.1.38.3 28 msec 28 msec 32 msec
  4 163.1.35.5 56 msec 57 msec 60 msec
  5 163.1.57.7 56 msec *  56 msec
```

Ticket 5

Analyze the Symptoms

If we look at the Layer 2 diagram we made previously, we'll notice that "servers" connect to SW1 and SW2. These are SW3 and SW4 emulating the actual end devices using the IP addresses 192.10.X.9 and 192.10.X.10 respectively. Since only Layer 1 and Layer 2 technologies are involved, we may want to start with bottom-up approach and verify the physical layer first.

Isolate the Issue

We check the first port, SW1 Fa0/18 connecting to SW4's Fa0/16.

```
Rack1SW1#show interfaces fastEthernet 0/18
FastEthernet0/18 is up, line protocol is down (notconnect)
  Hardware is Fast Ethernet, address is 001f.6d94.7b94 (bia
001f.6d94.7b94)
  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Full-duplex, 100Mb/s, media type is 10/100BaseTX
<snip>
```

The interface appears to be down, at least the line protocol (which is odd, usually means failure of duplex negotiation). Let's look at the other side:

```
Rack1SW4#show interfaces fastEthernet 0/16
FastEthernet0/16 is up, line protocol is up (connected)
  Hardware is Fast Ethernet, address is 0011.21c4.5d00 (bia
0011.21c4.5d00)
  Internet address is 192.10.1.10/24
  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Full-duplex, 100Mb/s, media type is 10/100BaseTX
```

The other side shows as up/up. There could be issues related to physical cabling, but this is highly unlikely, as the exam does not test your ability to fix the physical cabling problem. So we may only suspect some layer 2 misconfiguration. Let's check the switchport's configuration:

```
Rack1SW1#show interfaces fastEthernet 0/18 switchport
Name: Fa0/18
Switchport: Enabled
Administrative Mode: private-vlan host
Operational Mode: down
Administrative Trunking Encapsulation: negotiate
Negotiation of Trunking: Off
Access Mode VLAN: 1 (default)
Trunking Native Mode VLAN: 1 (default)
```

```
Administrative Native VLAN tagging: enabled
Voice VLAN: none
Administrative private-vlan host-association: 42 (VLAN0042) 500
(VLAN0500)
Administrative private-vlan mapping: none
Administrative private-vlan trunk native VLAN: none
Administrative private-vlan trunk Native VLAN tagging: enabled
Administrative private-vlan trunk encapsulation: dot1q
Administrative private-vlan trunk normal VLANs: none
Administrative private-vlan trunk associations: none
Administrative private-vlan trunk mappings: none
Operational private-vlan: none
Trunking VLANs Enabled: ALL
Pruning VLANs Enabled: 2-1001
Capture Mode Disabled
Capture VLANs Allowed: ALL

Protected: false
Unknown unicast blocked: disabled
Unknown multicast blocked: disabled
Appliance trust: none
```

We can see that there are private VLANs assigned to the port, but the "Operational private-vlan" list shows as "none". So it appears the problem is most likely related to the Private VLAN misconfiguration. Before we start with Private VLANs we should continue our troubleshooting and look for other Layer 2 issues.

Next we'll check the physical link between SW1 and SW2 and the link connecting SW2's Fa 0/19 to SW4. There are three links grouped in a port-channel connecting SW1 and SW2. We will check the port-channel and the member links for any issues:

```
Rack1SW1#show interfaces trunk

Port        Mode             Encapsulation  Status        Native vlan
Fa0/16      desirable        802.1q         trunking      1
Po13        on               802.1q         trunking      1

Port        Vlans allowed on trunk
Fa0/16      1-4094
Po13        1-6,8-4094

Port        Vlans allowed and active in management domain
Fa0/16      1,3-7,42,57,100-101,263,500
Po13        1,3-6,42,57,100-101,263,500

Port        Vlans in spanning tree forwarding state and not pruned
Fa0/16      1,3-7,42,57,100-101,263,500
Po13        1,3-6,42,57,100-101,263,500

Rack1SW1#show int fa 0/13
FastEthernet0/13 is up, line protocol is up (connected)
<snip>
```

```
Rack1SW1#show int fa 0/14
FastEthernet0/14 is up, line protocol is up (connected)
<snip>

Rack1SW1#show int fa 0/15
FastEthernet0/15 is up, line protocol is up (connected)
<snip>
```

After this, we check SW2's connection to SW4:

```
Rack1SW2#show interfaces fastEthernet 0/19
FastEthernet0/19 is up, line protocol is up (connected)
```

OK, so we summarize our findings: the only issue that looks related to physical/L2 problems is due to Private VLANs misconfiguraiton. We may now continue with troubleshooting the private VLANs. When working with private VLANs, you may want to follow the checklist below:

1) Make sure primary and secondary VLANs have been created and their types have been assigned properly.
2) Verify associations between the primary and secondary VLANs.
3) Verify host and promiscuous ports configuration.

So the first thing we got to do it checking if the primary and secondary VLANs have been created. We already know the private VLAN numbers in SW1, those are 42 and 500, with 42 being the primary. Let's check the VLAN configuration in VLAN database:

```
Rack1SW1#show vlan id 42

VLAN Name                             Status    Ports
---- -------------------------------- --------- -----------------------
42   VLAN0042                         active    Fa0/16, Po13

VLAN Type  SAID       MTU   Parent RingNo BridgeNo Stp  BrdgMode Trans1
Trans2
---- ----- ---------- ----- ------ ------ -------- ---- -------- ------
42   enet  100042     1500  -      -      -        -    -        0
0

Remote SPAN VLAN
----------------
Disabled

Primary Secondary Type              Ports
------- --------- ----------------- -----------------------------------
42                primary

Rack1SW1#show vlan id 500

VLAN Name                             Status    Ports
---- -------------------------------- --------- -----------------------
```

```
500  VLAN0500                         active    Fa0/16, Po13

VLAN Type  SAID       MTU   Parent RingNo BridgeNo Stp  BrdgMode Trans1
Trans2
---- ----- ---------- ----- ------ ------ -------- ---- -------- ------
500  enet  100500     1500  -      -      -        -    -        0
0

Remote SPAN VLAN
----------------
Disabled

Primary Secondary Type              Ports
------- --------- ----------------- ----------------------------------
isolated
```

It is often helpful comparing the same output with the switch where the primary VLANs “work”. We may need to go to SW2 and check the private VLANs there.

```
Rack1SW2#show interfaces fastEthernet 0/19 switchport
Name: Fa0/19
Switchport: Enabled
Administrative Mode: private-vlan host
Operational Mode: private-vlan host
Administrative Trunking Encapsulation: negotiate
Operational Trunking Encapsulation: native
Negotiation of Trunking: Off
Access Mode VLAN: 1 (default)
Trunking Native Mode VLAN: 1 (default)
Administrative Native VLAN tagging: enabled
Voice VLAN: none
Administrative private-vlan host-association: 42 (VLAN0042) 500
(VLAN0500)
Administrative private-vlan mapping: none
Administrative private-vlan trunk native VLAN: none
Administrative private-vlan trunk Native VLAN tagging: enabled
Administrative private-vlan trunk encapsulation: dot1q
Administrative private-vlan trunk normal VLANs: none
Administrative private-vlan trunk associations: none
Administrative private-vlan trunk mappings: none
Operational private-vlan:
  42 (VLAN0042) 500 (VLAN0500)
Trunking VLANs Enabled: ALL
Pruning VLANs Enabled: 2-1001
Capture Mode Disabled
Capture VLANs Allowed: ALL

Protected: false
Unknown unicast blocked: disabled
Unknown multicast blocked: disabled
Appliance trust: none
```

As we can see, the private VLAN numbers are the same, but this time they are operational. Let’s see the VLAN database configuration in SW2:

```
Rack1SW2#show vlan id 42

VLAN Name                             Status    Ports
---- -------------------------------- --------- -----------------------
42   VLAN0042                         active    Fa0/6, Po13

VLAN Type  SAID       MTU   Parent RingNo BridgeNo Stp  BrdgMode Trans1
Trans2
---- ----- ---------- ----- ------ ------ -------- ---- -------- ------
42   enet  100042     1500  -      -      -        -    -        0
0

Remote SPAN VLAN
----------------
Disabled

Primary Secondary Type              Ports
------- --------- ----------------- -----------------------------------
42      500       isolated          Fa0/4, Fa0/19, Fa0/24

Rack1SW2#show vlan id 500

VLAN Name                             Status    Ports
---- -------------------------------- --------- -----------------------
500  VLAN0500                         active    Fa0/6, Po13

VLAN Type  SAID       MTU   Parent RingNo BridgeNo Stp  BrdgMode Trans1
Trans2
---- ----- ---------- ----- ------ ------ -------- ---- -------- ------
500  enet  100500     1500  -      -      -        -    -        0
0

Remote SPAN VLAN
----------------
Disabled

Primary Secondary Type              Ports
------- --------- ----------------- -----------------------------------
42      500       isolated          Fa0/4, Fa0/19, Fa0/24
```

You can see the difference here: primary VLAN in SW1 is not associated with the secondary in the database configuration. This means that in order to fix the things in SW1 we need to do the following:

```
SW1:
vlan 42
  private-vlan association add 500
```

And check the VLAN database contents again:

```
Rack1SW1#show vlan id 500

VLAN Name                             Status    Ports
---- -------------------------------- --------- -----------------------
500  VLAN0500                         active    Fa0/16, Po13
```

```
VLAN Type  SAID       MTU   Parent RingNo BridgeNo Stp  BrdgMode Trans1
Trans2
---- ----- ---------- ----- ------ ------ -------- ---- -------- ------
500  enet  100500     1500  -      -      -        -    -        0
0

Remote SPAN VLAN
----------------
Disabled

Primary Secondary Type              Ports
------- --------- ----------------- ------------------------------------
42      500       isolated          Fa0/18
```

Let’s see if that helped with the port being down:

```
Rack1SW1#show interfaces fastEthernet 0/18
FastEthernet0/18 is up, line protocol is up (connected)
<snip>

Rack1SW1#show interfaces fastEthernet 0/18 switchport
Name: Fa0/18
Switchport: Enabled
Administrative Mode: private-vlan host
Operational Mode: private-vlan host
Administrative Trunking Encapsulation: negotiate
Operational Trunking Encapsulation: native
Negotiation of Trunking: Off
Access Mode VLAN: 1 (default)
Trunking Native Mode VLAN: 1 (default)
Administrative Native VLAN tagging: enabled
Voice VLAN: none
Administrative private-vlan host-association: 42 (VLAN0042) 500
(VLAN0500)
Administrative private-vlan mapping: none
Administrative private-vlan trunk native VLAN: none
Administrative private-vlan trunk Native VLAN tagging: enabled
Administrative private-vlan trunk encapsulation: dot1q
Administrative private-vlan trunk normal VLANs: none
Administrative private-vlan trunk associations: none
Administrative private-vlan trunk mappings: none
Operational private-vlan:
  42 (VLAN0042) 500 (VLAN0500)
Trunking VLANs Enabled: ALL
Pruning VLANs Enabled: 2-1001
Capture Mode Disabled
Capture VLANs Allowed: ALL

Protected: false
Unknown unicast blocked: disabled
Unknown multicast blocked: disabled
Appliance trust: none
```

Let’s see if the servers may ping R4:

```
Rack1SW4#ping 192.10.1.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.10.1.4, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 1/2/4 ms

Rack1SW3#ping 192.10.1.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.10.1.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

Per the task requirements, the servers should be able to reach across R4. So now we can try pinging between the two “servers”.

```
Rack1SW4#ping 192.10.1.9

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.10.1.8, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

This is not working still, so there are other issues. But at least now we know that both servers may reach R4. The last thing we may think of is that R4 is not doing proxy-ARP to assist the servers in reaching each other. Let’s check this:

```
Rack1R4#show ip interface fastEthernet 0/0 | inc Proxy
  Proxy ARP is enabled
  Local Proxy ARP is disabled
```

```
R4:
interface FastEthernet 0/0
 ip local-proxy-arp
```

And check again is SW4 can ping SW3:

```
Rack1SW4#ping 192.10.1.9

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.10.1.9, timeout is 2 seconds:
..!!!
Success rate is 60 percent (3/5), round-trip min/avg/max = 1/3/4 ms
```

Now confirm that communications take places across R4:

```
Rack1SW4#traceroute 192.10.1.9

Type escape sequence to abort.
Tracing the route to 192.10.1.9

  1 192.10.1.4 4 msec 4 msec 0 msec
  2 192.10.1.9 4 msec *  0 msec
```

And this means we're done troubleshooting the private VLANs issue.

Fix the Issue

So there were two problems we had to fix: first, we added the primary/secondary VLAN mapping to the VLAN database in SW1 and next we've enable local proxy ARP function in R4.

```
R4:
interface FastEthernet 0/0
 ip local-proxy-arp

SW1:
vlan 42
  private-vlan association add 500
```

Verify

Here is a list of the verification commands that you may want to use in order to verify primary VLANs connectivity. They summarize the commands we were using at the isolation stage.

The first set verifies that private VLANs are associated with the host ports:

```
Rack1SW1#show interfaces fastEthernet 0/18 switchport
Name: Fa0/18
Switchport: Enabled
Administrative Mode: private-vlan host
Operational Mode: private-vlan host
Administrative Trunking Encapsulation: negotiate
Operational Trunking Encapsulation: native
Negotiation of Trunking: Off
Access Mode VLAN: 1 (default)
Trunking Native Mode VLAN: 1 (default)
Administrative Native VLAN tagging: enabled
Voice VLAN: none
Administrative private-vlan host-association: 42 (VLAN0042) 500
(VLAN0500)
Administrative private-vlan mapping: none
Administrative private-vlan trunk native VLAN: none
Administrative private-vlan trunk Native VLAN tagging: enabled
Administrative private-vlan trunk encapsulation: dot1q
Administrative private-vlan trunk normal VLANs: none
Administrative private-vlan trunk associations: none
Administrative private-vlan trunk mappings: none
Operational private-vlan:
  42 (VLAN0042) 500 (VLAN0500)
Trunking VLANs Enabled: ALL
Pruning VLANs Enabled: 2-1001
Capture Mode Disabled
Capture VLANs Allowed: ALL

Protected: false
Unknown unicast blocked: disabled
Unknown multicast blocked: disabled
Appliance trust: none
Rack1SW1#

Rack1SW2#show interfaces fastEthernet 0/19 switchport
Name: Fa0/19
Switchport: Enabled
Administrative Mode: private-vlan host
Operational Mode: private-vlan host
Administrative Trunking Encapsulation: negotiate
Operational Trunking Encapsulation: native
Negotiation of Trunking: Off
Access Mode VLAN: 1 (default)
Trunking Native Mode VLAN: 1 (default)
Administrative Native VLAN tagging: enabled
Voice VLAN: none
Administrative private-vlan host-association: 42 (VLAN0042) 500
(VLAN0500)
Administrative private-vlan mapping: none
Administrative private-vlan trunk native VLAN: none
Administrative private-vlan trunk Native VLAN tagging: enabled
Administrative private-vlan trunk encapsulation: dot1q
Administrative private-vlan trunk normal VLANs: none
Administrative private-vlan trunk associations: none
Administrative private-vlan trunk mappings: none
Operational private-vlan:
  42 (VLAN0042) 500 (VLAN0500)
Trunking VLANs Enabled: ALL
Pruning VLANs Enabled: 2-1001
Capture Mode Disabled
Capture VLANs Allowed: ALL

Protected: false
Unknown unicast blocked: disabled
Unknown multicast blocked: disabled
Appliance trust: none
```

The second command verifies the promiscuous port:

```
Rack1SW2#show interfaces fastEthernet 0/4 switchport
Name: Fa0/4
Switchport: Enabled
Administrative Mode: private-vlan promiscuous
Operational Mode: private-vlan promiscuous
Administrative Trunking Encapsulation: negotiate
Operational Trunking Encapsulation: native
Negotiation of Trunking: Off
Access Mode VLAN: 1 (default)
Trunking Native Mode VLAN: 1 (default)
Administrative Native VLAN tagging: enabled
Voice VLAN: none
Administrative private-vlan host-association: none
Administrative private-vlan mapping: 42 (VLAN0042) 500 (VLAN0500)
Administrative private-vlan trunk native VLAN: none
Administrative private-vlan trunk Native VLAN tagging: enabled
Administrative private-vlan trunk encapsulation: dot1q
Administrative private-vlan trunk normal VLANs: none
Administrative private-vlan trunk associations: none
Administrative private-vlan trunk mappings: none
Operational private-vlan:
  42 (VLAN0042) 500 (VLAN0500)
Trunking VLANs Enabled: ALL
Pruning VLANs Enabled: 2-1001
Capture Mode Disabled
Capture VLANs Allowed: ALL

Protected: false
Unknown unicast blocked: disabled
Unknown multicast blocked: disabled
Appliance trust: none
```

For all outputs listed, make sure the primary/secondary VLANs are in Operational list. If there are some configuration errors, the VLANs will not be operational.

Now when you're done with the ports, you can check the actual end to end connectivity using the ping and the **`traceroute`** commands to ensure the packets take path across R4:

```
Rack1SW4#ping 192.10.1.9

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.10.1.9, timeout is 2 seconds:
..!!!
Success rate is 60 percent (3/5), round-trip min/avg/max = 1/3/4 ms

Now confirm that communications take places across R4:

Rack1SW4#traceroute 192.10.1.9

Type escape sequence to abort.
Tracing the route to 192.10.1.9

  1 192.10.1.4 4 msec 4 msec 0 msec
  2 192.10.1.9 4 msec *  0 msec
```

Ticket 6

Analyze the Symptoms

Since we resolved Tickets 1-4, we can be sure that there are no Layer2/3 issues preventing BGP peering sessions from coming up, so most likely this is some BGP misconfiguration issue. Plus, we are only allowed to change BGP settings, which further confirm our guess.

Isolate the Issue

Let's start from one of the "edges" where our topology borders AS54 or AS254, for example at R6 and start "tracing" the prefixes across the topology.

```
Rack1R6#show ip bgp regexp _54_
BGP table version is 24, local router ID is 150.1.6.6
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 28.119.16.0/24   54.1.7.254                    200      0 54 i
*                   204.12.1.254             0             0 54 i
*> 28.119.17.0/24   54.1.7.254                    200      0 54 i
*                   204.12.1.254             0             0 54 i
*  112.0.0.0        54.1.7.254               0             0 54 50 60 i
*>                  204.12.1.254                  200      0 54 50 60 i
*  113.0.0.0        54.1.7.254               0             0 54 50 60 i
*>                  204.12.1.254                  200      0 54 50 60 i
*> 114.0.0.0        54.1.7.254               0    200      0 54 i
*                   204.12.1.254                           0 54 i
*> 115.0.0.0        54.1.7.254               0    200      0 54 i
*                   204.12.1.254                           0 54 i
*> 116.0.0.0        54.1.7.254               0    200      0 54 i
*                   204.12.1.254                           0 54 i
*> 117.0.0.0        54.1.7.254               0    200      0 54 i
*                   204.12.1.254                           0 54 i
*> 118.0.0.0        54.1.7.254               0    200      0 54 i
   Network          Next Hop            Metric LocPrf Weight Path
*                   204.12.1.254                           0 54 i
*> 119.0.0.0        54.1.7.254               0    200      0 54 i
*                   204.12.1.254                           0 54 i
```

OK, so R6 sees the prefixes learned from both BB1 and BB3. Next we move to R2 and check the prefixes there, then move to R1:

```
Rack1R2#show ip bgp regexp _54_
BGP table version is 14, local router ID is 150.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
```

```
   Network          Next Hop            Metric LocPrf Weight Path
*  28.119.16.0/24   204.12.1.254             0             0 54 i
*>i                 54.1.7.254               0    200      0 54 i
*  28.119.17.0/24   204.12.1.254             0             0 54 i
*>i                 54.1.7.254               0    200      0 54 i
*  112.0.0.0        204.12.1.254                           0 54 50 60 i
*>i                 204.12.1.254             0    200      0 54 50 60 i
*  113.0.0.0        204.12.1.254                           0 54 50 60 i
*>i                 204.12.1.254             0    200      0 54 50 60 i
*  114.0.0.0        204.12.1.254                           0 54 i
*>i                 54.1.7.254               0    200      0 54 i
*  115.0.0.0        204.12.1.254                           0 54 i
*>i                 54.1.7.254               0    200      0 54 i
*  116.0.0.0        204.12.1.254                           0 54 i
*>i                 54.1.7.254               0    200      0 54 i
*  117.0.0.0        204.12.1.254                           0 54 i
*>i                 54.1.7.254               0    200      0 54 i
*  118.0.0.0        204.12.1.254                           0 54 i
   Network          Next Hop            Metric LocPrf Weight Path
*>i                 54.1.7.254               0    200      0 54 i
*  119.0.0.0        204.12.1.254                           0 54 i
*>i                 54.1.7.254               0    200      0 54 i
Rack1R2#

Rack1R1#show ip bgp regexp _54_

Rack1R1#
```

And we see that there are not prefixes from AS54 on R1. Now let’s get back to R1 and see if these prefixes are actually advertise to R1, to rule out prefix filtering:

```
Rack1R2#show ip bgp neighbors 163.1.12.1 advertised-routes

Total number of prefixes 0
```

This means that either there is filtering configured on R2, or there is something else that prevents R2 from advertising these prefixes to R1. Let’s check the neighbor configuration settings:

```
Rack1R2#show ip bgp neighbors 163.1.12.1
BGP neighbor is 163.1.12.1,  remote AS 100, internal link
  BGP version 4, remote router ID 150.1.1.1
  BGP state = Established, up for 00:00:33
  Last read 00:00:33, last write 00:00:33, hold time is 180, keepalive
interval is 60 seconds
  Neighbor capabilities:
    Route refresh: advertised and received(old & new)
    Address family IPv4 Unicast: advertised and received
  Message statistics:
    InQ depth is 0
    OutQ depth is 0
                         Sent       Rcvd
    Opens:                  3          3
    Notifications:          0         0
    Updates:                2         5
    Keepalives:            24        24
    Route Refresh:          0         0
    Total:                 29        32
  Default minimum time between advertisement runs is 0 seconds

 For address family: IPv4 Unicast
  BGP table version 30, neighbor version 30/0
 Output queue size : 0
  Index 2, Offset 0, Mask 0x4
  2 update-group member
                                 Sent       Rcvd
  Prefix activity:               ----       ----
    Prefixes Current:               0          3 (Consumes 156 bytes)
    Prefixes Total:                 0          3
    Implicit Withdraw:              0          0
    Explicit Withdraw:              0          0
    Used as bestpath:             n/a          3
    Used as multipath:            n/a          0

                                   Outbound    Inbound
  Local Policy Denied Prefixes:    --------    -------
    Bestpath from this peer:              3        n/a
    Bestpath from iBGP peer:             10        n/a
    Total:                               13          0
  Number of NLRIs in the update sent: max 8, min 2

  Connections established 3; dropped 2
  Last reset 00:00:36, due to RR client config change
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Connection is ECN Disabled, Mininum incoming TTL 0, Outgoing TTL 255
Local host: 163.1.12.2, Local port: 56926
Foreign host: 163.1.12.1, Foreign port: 179
<snip>
```

As we can see the connection is in healthy state and IPv4 address family is enabled for this peer. The peer is a route-reflector client and there are no output filters configured. So there is something else preventing R2 from sending the updates to R1. To check what might be causing this, notice that the above peer is a member of update group “2”. BGP uses update groups for update optimization – all peers within the same group receive the same updates. Now let’s take any prefix from AS 54 and see what update group it belongs to.

```
Rack1R2#**show ip bgp update-group**
BGP version 4 update-group 1, external, Address Family: IPv4 Unicast
  BGP Update version : 30/0, messages 0
  Update messages formatted 9, replicated 0
  Number of NLRIs in the update sent: max 8, min 1
  Minimum time between advertisement runs is 30 seconds
  Has 1 member (* indicates the members currently being sent updates):
   204.12.1.254

BGP version 4 update-group 2, internal, Address Family: IPv4 Unicast
```

```
  BGP Update version : 30/0, messages 0
  Update messages formatted 4, replicated 0
  Number of NLRIs in the update sent: max 4, min 1
  Minimum time between advertisement runs is 0 seconds
  Has 2 members (* indicates the members currently being sent updates):
   163.1.12.1       204.12.1.6

Rack1R2#show ip bgp 112.0.0.0
BGP routing table entry for 112.0.0.0/8, version 4
Paths: (2 available, best #2, table Default-IP-Routing-Table)
  Advertised to update-groups:
     1
  54 50 60
    204.12.1.254 from 204.12.1.254 (31.3.0.1)
      Origin IGP, localpref 100, valid, external
  54 50 60
    204.12.1.254 from 204.12.1.6 (150.1.6.6)
      Origin IGP, metric 0, localpref 200, valid, internal, best
```

There are two update groups – one for the eBGP peer, and another for iBGP peers. The prefix 112.0.0.0 is preferred via an iBGP path, and only advertised to the eBGP peer. This most likely means that nor R1 nor R6 are configured as route-reflectors, which they should be per the diagram.

```
Rack1R2#show ip bgp neighbors 163.1.12.1 | inc refle
Rack1R2#show ip bgp neighbors 204.12.1.6 | inc refle
Rack1R2#
```

We go ahead and fix this problem:

```
R2:
router bgp 100
 neighbor 163.1.12.1 route-reflector-client
 neighbor 204.12.1.6 route-reflector-client
```

Now let’s go back to R1 and trace the prefixes from AS 54 futher.

```
Rack1R1#show ip bgp regexp _54
BGP table version is 122, local router ID is 150.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
s>i28.119.16.0/24   54.1.7.254               0    200      0 54 i
*> 28.119.16.0/23   0.0.0.0                       200  32768 54 i
s>i28.119.17.0/24   54.1.7.254               0    200      0 54 i
s>i112.0.0.0        204.12.1.254             0    200      0 54 50 60 i
s>i113.0.0.0        204.12.1.254             0    200      0 54 50 60 i
s>i114.0.0.0        54.1.7.254               0    200      0 54 i
s>i115.0.0.0        54.1.7.254               0    200      0 54 i
s>i116.0.0.0        54.1.7.254               0    200      0 54 i
s>i117.0.0.0        54.1.7.254               0    200      0 54 i
```

```
s>i118.0.0.0        54.1.7.254               0    200      0 54 i
s>i119.0.0.0        54.1.7.254               0    200      0 54 i
```

We can see the prefixes being summarized. Now we check that the AS54 prefixes are being advertised further to AS300:

```
Rack1R1#show ip bgp neighbors 163.1.18.8 advertised-routes
BGP table version is 122, local router ID is 150.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
s>i28.119.16.0/24   54.1.7.254               0    200      0 54 i
*> 28.119.16.0/23   0.0.0.0                       200  32768 54 i
s>i112.0.0.0        204.12.1.254             0    200      0 54 50 60 i
*> 112.0.0.0/5      0.0.0.0                       200  32768 {54,50,60}
i
s>i113.0.0.0        204.12.1.254             0    200      0 54 50 60 i
s>i114.0.0.0        54.1.7.254               0    200      0 54 i
s>i115.0.0.0        54.1.7.254               0    200      0 54 i
*> 205.90.31.0      163.1.13.3                             0 200 300
200 254 ?
*> 220.20.3.0       163.1.13.3                             0 200 300
200 254 ?
*> 222.22.2.0       163.1.13.3                             0 200 300
200 254 ?

Total number of prefixes 10

Rack1R1#show ip bgp neighbors 163.1.13.3 advertised-routes
BGP table version is 122, local router ID is 150.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 28.119.16.0/23   0.0.0.0                       200  32768 54 i
s>i28.119.17.0/24   54.1.7.254               0    200      0 54 i
*> 112.0.0.0/5      0.0.0.0                       200  32768 {54,50,60}
i
s>i116.0.0.0        54.1.7.254               0    200      0 54 i
s>i117.0.0.0        54.1.7.254               0    200      0 54 i
s>i118.0.0.0        54.1.7.254               0    200      0 54 i
s>i119.0.0.0        54.1.7.254               0    200      0 54 i

Total number of prefixes 7
```

And then go to R5 of AS 200 to check that it receives the prefixes from AS 300.

```
Rack1R5#show ip bgp regexp 54
BGP table version is 12, local router ID is 150.1.5.5
```

```
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 205.90.31.0      192.10.1.254             0    100      0 (65004)
254 ?
*> 220.20.3.0       192.10.1.254             0    100      0 (65004)
254 ?
*> 222.22.2.0       192.10.1.254             0    100      0 (65004)
254 ?

Rack1R5#show ip bgp neighbors 163.1.35.3 routes

Total number of prefixes 0
```

OK, so we’re going to go to R3 and check if there is any sort of outbound filtering applied there:

```
Rack1R3#show ip bgp neighbors 163.1.35.5 advertised-routes
BGP table version is 70, local router ID is 150.1.3.3
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 28.119.16.0/23   163.1.13.1               0             0 200 100 54
i
*> 112.0.0.0/5      163.1.13.1               0             0 200 100
{54,50,60} i

Total number of prefixes 2
```

We can see two summary prefixes being advertised to AS 200. Now let’s BGP updates debugging in R5 to see why these prefixes are blocked:

```
Rack1R5#debug ip bgp updates
BGP updates debugging is on for address family: IPv4 Unicast

Rack1R5#clear ip bgp 163.1.35.3 soft in

BGP(0): 163.1.35.3 rcv UPDATE w/ attr: nexthop 163.1.35.3, origin i,
aggregated by 100 150.1.1.1, originator 0.0.0.0, path 300 200 100 54,
community , extended community
BGP(0): 163.1.35.3 rcv UPDATE about 28.119.16.0/23 -- DENIED due to:
AS-PATH contains our own AS;
BGP(0): 163.1.35.3 rcv UPDATE w/ attr: nexthop 163.1.35.3, origin i,
aggregated by 100 150.1.1.1, originator 0.0.0.0, path 300 200 100
{54,50,60}, community , extended community
BGP(0): 163.1.35.3 rcv UPDATE about 112.0.0.0/5 -- DENIED due to: AS-
PATH contains our own AS;
```

OK, so we see the reason, AS 300 somehow prepends AS 200 to BGP updates sent to R5. We may go ahead and figure out what part of AS 300 configuration results in this behavior. However, we are not allowed to modify AS 300 BGP configurations, and therefore it makes sense to apply a fix in R5. There is a BGP command to allow the prefixes with the same BGP AS# inbound:

```
R5:
router bgp 65005
 neighbor 163.1.35.3 allowas-in
```

Now R5 receives these prefixes.

```
Rack1R5#show ip bgp neighbors 163.1.35.3 routes
BGP table version is 14, local router ID is 150.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 28.119.16.0/23   163.1.35.3                             0 300 200
100 54 i
*> 112.0.0.0/5      163.1.35.3                             0 300 200
100 {54,50,60} i

Total number of prefixes 2
```

Let’s jump to R4 and see if AS54 prefixes made it there:

```
Rack1R4#show ip bgp regex 54
BGP table version is 12, local router ID is 150.1.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 205.90.31.0      192.10.1.254             0             0 254 ?
*> 220.20.3.0       192.10.1.254             0             0 254 ?
*> 222.22.2.0       192.10.1.254             0             0 254 ?

Rack1R4#show ip bgp neighbors 150.1.5.5 routes

Total number of prefixes 0
```

Appears to be not! Let’s see if R5 advertises the prefixes to R4:

```
Rack1R5#show ip bgp neighbors 150.1.4.4 advertised-routes
BGP table version is 14, local router ID is 150.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
```

```
   Network          Next Hop            Metric LocPrf Weight Path
*> 28.119.16.0/23   163.1.35.3                             0 300 200
100 54 i
*> 112.0.0.0/5      163.1.35.3                             0 300 200
100 {54,50,60} i
*> 205.90.31.0      192.10.1.254             0    100      0 (65004)
254 ?
*> 220.20.3.0       192.10.1.254             0    100      0 (65004)
254 ?
*> 222.22.2.0       192.10.1.254             0    100      0 (65004)
254 ?

Total number of prefixes 5
```

If we stop and think about it, it become obvious that R4 experiences the same issue as R5. Since R4 and R5 are in different sub-confederations, the peering session between them is eBGP, and therefore loop prevention based on AS numbers applies here. We need to configure R4 in the same manner we configured R5:

```
R4:
router bgp 65004
 neighbor 150.1.5.5 allowas-in
```

And now R4 advertises AS54 prefixes to AS 254.

```
Rack1R4#show ip bgp neighbors 192.10.1.254 advertised-routes
BGP table version is 14, local router ID is 150.1.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 28.119.16.0/23   163.1.35.3               0    100      0 (65005)
300 200 100 54 i
*> 112.0.0.0/5      163.1.35.3               0    100      0 (65005)
300 200 100 {54,50,60} i

Total number of prefixes 2
```

This has been a long way, but we only traced the prefixes in one direction! That is, we need to make sure that AS 254 prefixes are being advertised to AS 54 speakers (BB1 and BB3). Let’s get back to R2 where it all started and see if this is true:

```
Rack1R2#show ip bgp regexp 254
BGP table version is 58, local router ID is 150.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
```

```
   Network          Next Hop            Metric LocPrf Weight Path
*>i205.90.31.0      163.1.13.3               0    100      0 200 300
200 254 ?
*>i220.20.3.0       163.1.13.3               0    100      0 200 300
200 254 ?
*>i222.22.2.0       163.1.13.3               0    100      0 200 300
200 254 ?
```

And now we make sure these prefixes are advertised to BB1 and BB3:

```
Rack1R2#show ip bgp neighbors 204.12.1.254 advertised-routes
BGP table version is 58, local router ID is 150.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*>i28.119.16.0/24   54.1.7.254               0    200      0 54 i
*>i28.119.16.0/23   163.1.12.1               0    200      0 54 i
*>i28.119.17.0/24   54.1.7.254               0    200      0 54 i
*>i112.0.0.0        204.12.1.254             0    200      0 54 50 60 i
*>i112.0.0.0/5      163.1.12.1               0    200      0 {54,50,60}
i
*>i113.0.0.0        204.12.1.254             0    200      0 54 50 60 i
*>i114.0.0.0        54.1.7.254               0    200      0 54 i
*>i115.0.0.0        54.1.7.254               0    200      0 54 i
*>i116.0.0.0        54.1.7.254               0    200      0 54 i
*>i117.0.0.0        54.1.7.254               0    200      0 54 i
*>i118.0.0.0        54.1.7.254               0    200      0 54 i
*>i119.0.0.0        54.1.7.254               0    200      0 54 i
*>i205.90.31.0      163.1.13.3               0    100      0 200 300
200 254 ?
*>i220.20.3.0       163.1.13.3               0    100      0 200 300
200 254 ?
*>i222.22.2.0       163.1.13.3               0    100      0 200 300
200 254 ?

Total number of prefixes 15

Rack1R6#show ip  bgp neighbors 54.1.7.254 advertised-routes
BGP table version is 54, local router ID is 150.1.6.6
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 28.119.16.0/24   54.1.7.254                    200      0 54 i
*>i28.119.16.0/23   163.1.12.1               0    200      0 54 i
*> 28.119.17.0/24   54.1.7.254                    200      0 54 i
*> 112.0.0.0        204.12.1.254                  200      0 54 50 60 i
*>i112.0.0.0/5      163.1.12.1               0    200      0 {54,50,60}
i
*> 113.0.0.0        204.12.1.254                  200      0 54 50 60 i
*> 114.0.0.0        54.1.7.254               0    200      0 54 i
*> 115.0.0.0        54.1.7.254               0    200      0 54 i
*> 116.0.0.0        54.1.7.254               0    200      0 54 i
*> 117.0.0.0        54.1.7.254               0    200      0 54 i
*> 118.0.0.0        54.1.7.254               0    200      0 54 i
*> 119.0.0.0        54.1.7.254               0    200      0 54 i
*>i205.90.31.0      163.1.13.3               0    100      0 200 300
200 254 ?
*>i220.20.3.0       163.1.13.3               0    100      0 200 300
200 254 ?
*>i222.22.2.0       163.1.13.3               0    100      0 200 300
200 254 ?

Total number of prefixes 15
```

This is the maximum we could do in order to ensure end-to-end connectivity between AS

Fix the Issue

Here is the summary of all changes we applied to R2, R4 and R5:

```
R2:
router bgp 100
 neighbor 163.1.12.2 route-reflector-client
 neighbor 204.12.1.6 route-relfector-client

R4:
router bgp 65004
 neighbor 150.1.5.5 allowas-in

R5:
router bgp 65005
 neighbor 163.1.35.3 allowas-in
```

Verify

The verification has already been performed during the isolation stage.

Ticket 7

Analyze the Symptoms

Apparently, the issue is related to L2VPN configuration, as we're pretty sure about end-to-end connectivity. Thus we can jump straight into isolating the L2VPN configuration issues.

Isolate the Issue

Our main research tool would be the following debugging commands **debug vpdn l2x-events** , **debug vpdn l2x-errors**:

```
Rack1R6#debug vpdn l2x-events
L2X protocol events debugging is on

Rack1R6#debug vpdn l2x-errors
L2X protocol errors debugging is on

Tnl/Sn 61281/1449 L2TP: Session state change from idle to wait-for-
tunnel
uid:11 Tnl/Sn 61281/1449 L2TP: Create session
 Tnl 61281 L2TP: SM State idle
 Tnl 61281 L2TP: O SCCRQ
 Tnl 61281 L2TP: Control channel retransmit delay set to 1 seconds
 Tnl 61281 L2TP: Tunnel state change from idle to wait-ctl-reply
 Tnl 61281 L2TP: SM State wait-ctl-reply
L2TP: I SCCRQ from Rack1R4 tnl 16167
 Tnl 26282 L2TP: O SCCRP  to Rack1R4 tnlid 16167
 Tnl 26282 L2TP: Control channel retransmit delay set to 1 seconds
 Tnl 26282 L2TP: Tunnel state change from idle to wait-ctl-reply
 Tnl 26282 L2TP: New tunnel created for remote Rack1R4, address
150.1.4.4
 Tnl 61281 L2TP: I SCCRP from Rack1R4
 Tnl 61281 L2TP: Tunnel state change from wait-ctl-reply to established
 Tnl 61281 L2TP: O SCCCN  to Rack1R4 tnlid 13478
 Tnl 61281 L2TP: Control channel retransmit delay set to 1 seconds
 Tnl 61281 L2TP: SM State established
uid:11 Tnl/Sn 61281/1449 L2TP: O ICRQ to Rack1R4 13478/0
uid:11 Tnl/Sn 61281/1449 L2TP: Session state change from wait-for-
tunnel to wait-reply
 Tnl 26282 L2TP: I SCCCN from Rack1R4 tnl 16167
 Tnl 26282 L2TP: Tunnel state change from wait-ctl-reply to established
 Tnl 26282 L2TP: SM State established
 Tnl 26282 L2TP: I ICRQ from Rack1R4 tnl 16167
 Tnl/Sn 26282/1450 L2TP: Session state change from idle to wait-connect
 Tnl/Sn 26282/1450 L2TP: Accepted ICRQ, new session created
uid:44 Tnl/Sn 26282/1450 L2TP: Xconnect VC ID 46 not provisioned
uid:44 Tnl/Sn 26282/1450 L2TP: O CDN to Rack1R4 16167/2801
 Tnl 26282 L2TP: Control channel retransmit delay set to 1 seconds
uid:44 Tnl/Sn 26282/1450 L2TP: Destroying session
uid:44 Tnl/Sn 26282/1450 L2TP: Session state change from wait-connect
to idle
uid:44 Tnl/Sn 26282/1450 L2TP: Accounting stop sent
```

```
 Tnl 26282 L2TP: Tunnel state change from established to no-sessions-
left
 Tnl 26282 L2TP: No more sessions in tunnel, shutdown (likely) in 15
seconds
uid:11 Tnl/Sn 61281/1449 L2TP: I CDN from Rack1R4 tnl 13478, cl 0
uid:11 Tnl/Sn 61281/1449 L2TP: disconnect (L2X) IETF: 9/nas-error
Ascend: 62/VPDN No Resources
uid:11 Tnl/Sn 61281/1449 L2TP: Destroying session
L2X: Sending L2TUN message <Connect Fail>
uid:11 Tnl/Sn 61281/1449 L2TP: Session state change from wait-reply to
idle
 Tnl 61281 L2TP: Tunnel state change from established to no-sessions-
left
 Tnl 61281 L2TP: No more sessions in tunnel, shutdown (likely) in 15
seconds
L2X: l2tun session [2228681864], event [server response], old state
[open], new state [open]
L2X: l2tun session [2228681864], event [client free], old state [open],
new state [open]
 Tnl 26282 L2TP: Control channel retransmit delay set to 1 seconds
```

Based on the debugging output above we can see that the problem is that the incoming call attempts to find the circuit ID 46 and apparently it is not provisioned. Let’s see how R6 is configured for L2VPN. We’ll have to peek at the interface configuration to discover that:

```
Rack1R6#sh running-config interface fastEthernet 0/0.64
Building configuration...

Current configuration : 141 bytes
!
interface FastEthernet0/0.64
 encapsulation dot1Q 64
 no cdp enable
 xconnect 150.1.4.4 64 encapsulation l2tpv3 pw-class PWC_R6_TO_R4
```

The local circuit ID is 64, which does not match 46. We’re changing this to 46 (we could have also edited R4’ configuration):

```
R6:
interface FastEthernet 0/0.64
 no xconnect 150.1.4.4 64 encapsulation l2tpv3 pw-class PWC_R6_TO_R4
 xconnect 150.1.4.4 46 encapsulation l2tpv3 pw-class PWC_R6_TO_R4
```

Now check it the tunnel is healthy:

```
Rack1R6#show l2tun session all

%No active L2F tunnels

L2TP Session Information Total tunnels 1 sessions 1

Session id 1469 is up, tunnel id 21123
Call serial number is 2859100048
Remote tunnel name is Rack1R4
```

```
  Internet address is 150.1.4.4
  Session is L2TP signalled
  Session state is established, time since change 00:01:32
    0 Packets sent, 0 received
    0 Bytes sent, 0 received
  Last clearing of "show vpdn" counters never
    Receive packets dropped:
      out-of-order:             0
      total:                    0
    Send packets dropped:
      exceeded session MTU:     0
      total:                    0
  Session vcid is 46
  Session Layer 2 circuit, type is Ethernet Vlan, name is
FastEthernet0/0.64:64
  Circuit state is UP
    Remote session id is 2821, remote tunnel id 30612
  Session PMTU enabled, path MTU is not known
  DF bit on, ToS reflect enabled, ToS value 0, TTL value 255
  No session cookie information available
  UDP checksums are disabled
  SSS switching enabled
  Sequencing is off
  Unique ID is 54

%No active PPTP tunnels
```

OK from the above output we can see that both the control and tunnel sessions are now up. Let's check if the end-to-end connectivity works. We'll configure SW2 and SW4 as virtual workstations connected to R4 and R6 respectively.

SW2:

```
interface FastEthernet0/6
 switchport trunk encapsulation dot1q
 switchport mode trunk
!
interface Vlan64
 ip address 163.1.64.10 255.255.255.0
```

SW4:

```
interface FastEthernet0/4
 switchport trunk encapsulation dot1q
 switchport mode trunk
!
interface Vlan46
 ip address 163.1.64.8 255.255.255.0
```

Wait some time for STP to converge and try pinging across the L2VPN tunnel:

```
Rack1SW2#ping 163.1.64.10

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 163.1.64.10, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

Oh great, another issue which we have to troubleshoot. Checking the port status following the bottom-up approach:

```
Rack1SW2#show interfaces fastEthernet 0/6
FastEthernet0/6 is up, line protocol is up (connected)
  Hardware is Fast Ethernet, address is 001f.2711.d588 (bia
001f.2711.d588)
  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
<snip>
```

Looking good at the first sight. Next we check the logical topology: that is the STP status:

```
Rack1SW2#show spanning-tree vlan 64

VLAN0064
  Spanning tree enabled protocol ieee
  Root ID    Priority    32832
             Address     001f.2711.d580
             This bridge is the root
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    32832  (priority 32768 sys-id-ext 64)
             Address     001f.2711.d580
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time 300

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- ----------------------
Fa0/6               Desg BKN*19        128.8    P2p *PVID_Inc
Po13                Desg FWD 9         128.152  P2p
```

And we see the port connected to R6 is blocking due to inconsistent native VLAN ID. Now this is pretty obvious, as PVST+ is being in use for both VLANs, and the VLAN numbers at every side do not match. This results in PVST+ inconsistency. To fix that, apply the BPDU filtering on the switch ports connected to SW2 and SW4:

```
SW2:
interface FastEthernet 0/6
 spanning-tree bpdufilter enable

SW4:
interface FastEthernet 0/4
 spanning-tree bpdufilter enable
```

Wait for STP to converge and ping again:

```
Rack1SW2#ping 163.1.64.10

Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 163.1.64.10, timeout is 2 seconds:
..!!!
Success rate is 60 percent (3/5), round-trip min/avg/max = 235/235/235
ms
```

Conclusion: Watch out of circuit IDs and beware of STP problems with mismatching VLANs in VLAN-connect mode.

Fix the Issue

What we have done to fix the issue is following: matching the circuit-IDs at both ends and filtering BPDU at the switch-ports to stop PVST+ from blocking on the inconsistent VLANs.

```
R6:
interface FastEthernet 0/0.64
 no xconnect 150.1.4.4 64 encapsulation l2tpv3 pw-class PWC_R6_TO_R4
 xconnect 150.1.4.4 46 encapsulation l2tpv3 pw-class PWC_R6_TO_R4

SW2:
interface FastEthernet 0/6
 spanning-tree bpdufilter enable

SW4:
interface FastEthernet 0/4
 spanning-tree bpdufilter enable
```

Verify

Let's check the tunnel status once again and confirm end-to-end connectivity:

```
Rack1R4#show l2tun session all

%No active L2F tunnels

L2TP Session Information Total tunnels 1 sessions 1

Session id 2821 is up, tunnel id 30612
Call serial number is 2859100048
Remote tunnel name is Rack1R6
  Internet address is 150.1.6.6
  Session is L2TP signalled
  Session state is established, time since change 01:16:29
    691 Packets sent, 353 received
    47724 Bytes sent, 64112 received
  Last clearing of "show vpdn" counters never
    Receive packets dropped:
      out-of-order:             0
      total:                    0
    Send packets dropped:
      exceeded session MTU:     0
      total:                    0
  Session vcid is 46
  Session Layer 2 circuit, type is Ethernet Vlan, name is
FastEthernet0/1.46:46
  Circuit state is UP
    Remote session id is 1469, remote tunnel id 21123
  Session PMTU enabled, path MTU is not known
  DF bit on, ToS reflect enabled, ToS value 0, TTL value 255
  No session cookie information available
  UDP checksums are disabled
  SSS switching enabled
  Sequencing is off
  Unique ID is 54

%No active PPTP tunnels
```

The above shows that both the control connection and the circuit tunnel are in up state. You may also use the following command to check the active tunnel parameters:

```
Rack1R4#show sss circuits

Current SSS Circuit Information: Total number of circuits 1

Unique ID 0                     Serial Num 4
-----------------------------------------------------------------------
   Status     Encapsulation
   UP  flg    len  dump
   Y  AES       0
   Y  AES      24  45000014 00004000 FF73456A 96010404 96010606
                   000005BD
```

This output provides the encapsulation header applied to all packets that are to be delivered across the L2 VPN tunnel. Finally an end-to-end test validates the connectivity:

```
Rack1SW2#ping 163.1.64.10

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 163.1.64.10, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 234/238/244
ms
```

Ticket 8

Analyze the Symptoms

The problem scope is limited to three routers – R3, R4 and R5 and we know there are no physical or Layer 2 issues involved. So we may only suspect Layer 3 filtering or NTP configuration issues. Let's get a quick snapshot of the current NTP relationships:

```
Rack1R5#show ntp status
Clock is synchronized, stratum 8, reference is 127.127.7.1
nominal freq is 249.5901 Hz, actual freq is 249.5842 Hz, precision is
2**18
reference time is CE0E4259.7576AD65 (01:09:45.458 UTC Mon Jul 20 2009)
clock offset is 0.0000 msec, root delay is 0.00 msec
root dispersion is 0.02 msec, peer dispersion is 0.02 msec

Rack1R5#show ntp associations detail
127.127.7.1 configured, our_master, sane, valid, stratum 7
ref ID 127.127.7.1, time CE0E4219.74D62340 (01:08:41.456 UTC Mon Jul 20
2009)
our mode active, peer mode passive, our poll intvl 64, peer poll intvl
64
<snip>
```

This shows us that the NTP master is in healthy status, synchronized with itself (127.127.7.1). Now check the NTP clients:

```
Rack1R3#show ntp status
Clock is unsynchronized, stratum 16, no reference clock
nominal freq is 249.5901 Hz, actual freq is 249.5837 Hz, precision is
2**18
reference time is CE0DFF1E.CCE933BC (20:22:54.800 UTC Sun Jul 19 2009)
clock offset is -1.0902 msec, root delay is 123.31 msec
root dispersion is 24.23 msec, peer dispersion is 4.58 msec

Rack1R3#show ntp associations detail
150.1.5.5 configured, authenticated, insane, invalid, unsynced, stratum
16
ref ID 0.0.0.0, time 00000000.00000000 (00:00:00.000 UTC Mon Jan 1
1900)
our mode client, peer mode unspec, our poll intvl 64, peer poll intvl
<snip>
```

This shows that R3 and R5 peering session is authenticated but something prevents it from being used for synchronization.

Now we look at R4's NTP session with R5:

```
Rack1R4#show ntp status
Clock is unsynchronized, stratum 16, no reference clock
nominal freq is 249.5901 Hz, actual freq is 249.5862 Hz, precision is
2**18
```

```
reference time is CE0DFF1E.D617F11D (20:22:54.836 UTC Sun Jul 19 2009)
clock offset is -2.8478 msec, root delay is 215.58 msec
root dispersion is 25.67 msec, peer dispersion is 4.26 msec

Rack1R4#show ntp associations detail
150.1.5.5 configured, authenticated, insane, invalid, unsynced, stratum
16
ref ID 0.0.0.0, time 00000000.00000000 (00:00:00.000 UTC Mon Jan 1
1900)
our mode client, peer mode unspec, our poll intvl 64, peer poll intvl
64
<snip>
```

Seems to be the same problem – the server is configured but the local peer cannot synchronize.

Isolate the Issue

To begin with, NTP is one part of IOS configuration that has very little information uncovered using the show commands. Basically, you need to use the command **`show running-config | inc ntp`** to learn about NTP configuration settings. In most cases this is OK, as the amount of NTP commands on a single router should be very limited. We start with learning the NTP configuration of R5:

```
Rack1R5#show running-config | inc ntp
ntp authentication-key 1 md5 014354560E592259791E165B40503245585D227B0A
7
ntp clock-period 17208484
ntp access-group peer 1
ntp access-group serve-only 2
ntp master
```

We see two access-lists being used for access-control. Let's check the contents of these access-lists:

```
Rack1R5#show ip access-lists 1
Standard IP access list 1
    10 permit 127.127.7.1 (49 matches)

Rack1R5#show ip access-lists 2
Standard IP access list 2
    10 permit 150.1.3.3 (48 matches)
    20 permit 150.1.4.4 (1 match)
```

The first access-list allows the server to synchronize with the local time source. No other routers are allowed to change the local clock in R5. The second access-list specifies the remote peers that are allowed to request time from the local server. Those are the loopback IP addresses of R3 and R4. This means that R3 and R4 are supposed to source NTP requests out of their Loopback0 interfaces. Lastly, we see an authentication key with index “1” configured in the server. The key is encrypted using reversible type “7” encryption technique. We don’t decrypt the key right now, but we may need to do this later.

Now let’s check R3’s configuration and compare it to R5’s.

```
Rack1R3#sh running-config | inc ntp
ntp authentication-key 1 md5 02252D682829 7
ntp authenticate
ntp clock-period 17208524
ntp source Loopback0
ntp server 150.1.5.5 key 1
```

Here we see that the server is configured to use the key with index 1 and the NTP packets are sourced off Loopback0. The client is configured to authenticate the NTP packets received from the server and there is a local key configured with the matching index (key indexes are carried in packets and must match). If we quickly compare the two encrypted keys we’ll notice that R3’s key is almost twice as longer. This means there is authentication key mismatch. We need to decipher R5’s key and apply it to R3. In order to decipher R5’s key we use the old “key-chain” trick: the encrypted key is configured under a key-chain as following:

```
R5:
key chain DECRYPT
 key 1
  key-string 7 014354560E592259791E165B40503245585D227B0A

Rack1R5#show key chain DECRYPT
Key-chain DECRYPT:
    key 1 -- text "02252D6828295E731F1A"
        accept lifetime (always valid) - (always valid) [valid now]
        send lifetime (always valid) - (always valid) [valid now]
```

This shows us the “cleartext” key which we may configure in R3 now. However, there is other thing still missing from the configuration – they key is not configured as trusted! Without that, the client will never consider the updates from the server, even though the key and the index may match.

```
R3:
ntp authentication-key 1 md5 02252D6828295E731F1A
ntp trusted-key 1
```

Everything appears to be in order now, let’s make sure this is how it is:

```
Rack1R3#show ntp status
Clock is synchronized, stratum 9, reference is 150.1.5.5
nominal freq is 249.5901 Hz, actual freq is 249.5837 Hz, precision is
2**18
reference time is CE0E4F81.4C472096 (02:05:53.297 UTC Mon Jul 20 2009)
clock offset is -0.7546 msec, root delay is 59.14 msec
root dispersion is 1.60 msec, peer dispersion is 0.81 msec
```

R3 has clocks synchronized via NTP. Now let's list the NTP associations:

```
Rack1R3#show ntp associations detail
150.1.5.5 configured, authenticated, our_master, sane, valid, stratum 8
ref ID 127.127.7.1, time CE0E4F59.897042C1 (02:05:13.536 UTC Mon Jul 20
2009)
our mode client, peer mode server, our poll intvl 256, peer poll intvl
128
<snip>
```

It is not time to deal with R4's configuration:

```
Rack1R4#show running-config | inc ntp
ntp authentication-key 2 md5 032772382520 7
ntp authenticate
ntp clock-period 17208348
ntp server 150.1.5.5 key 2
```

We see the same issue with the key here, it appears to be wrong. Plus, they key index used in R4 is "2" whereas the one configure in R5 has the index value of "1". We need to adjust the key index and change the key value:

```
R4:
no ntp authentication-key 2
ntp authentication-key 1 md5 02252D6828295E731F1A
ntp trusted-key 1
ntp server 150.1.5.5 key 1
```

Don't forget to change the key used for authentication with R5, as would be key 2 by default.

```
Rack1R4#show ntp status
Clock is unsynchronized, stratum 16, no reference clock
nominal freq is 249.5901 Hz, actual freq is 249.5862 Hz, precision is
2**18
reference time is CE0DFF1E.D617F11D (20:22:54.836 UTC Sun Jul 19 2009)
clock offset is -2.8478 msec, root delay is 215.58 msec
root dispersion is 25.67 msec, peer dispersion is 4.26 msec
```

That's not working still. If we look at R4's configuration and compare it to the working R3's config we'll notice that R4 misses the NTP update source interface. And as we remember R5 is configured to service only requests sourced off Loopback0 interfaces. We change R4's configuration:

```
R4:
ntp source Loopback0
```

Keep in mind that it may take some time for NTP to synchronize as the protocol is not fast. You may want to switch to another scenario while waiting for NTP to converge. The resulting state in R4 should display something like this:

```
Rack1R4#show ntp status
```

```
Clock is synchronized, stratum 9, reference is 150.1.5.5
nominal freq is 249.5901 Hz, actual freq is 249.5862 Hz, precision is
2**18
reference time is CE0E5704.4494DDE0 (02:37:56.267 UTC Mon Jul 20 2009)
clock offset is -0.2471 msec, root delay is 58.21 msec
root dispersion is 15875.31 msec, peer dispersion is 15875.02 msec

Rack1R4#show ntp associations detail
150.1.5.5 configured, authenticated, our_master, sane, valid, stratum 8
ref ID 127.127.7.1, time CE0E56D9.94D8B40D (02:37:13.581 UTC Mon Jul 20
2009)
our mode client, peer mode server, our poll intvl 64, peer poll intvl
64
root delay 0.00 msec, root disp 0.03, reach 1, sync dist 15904.144
delay 58.21 msec, offset -0.2471 msec, dispersion 15875.02
<snip>
```

Fix the Issue

We summarize the steps we applied to fix the issues here:

```
R3:
ntp authentication-key 1 md5 02252D6828295E731F1A
ntp trusted-key 1

R4:
ntp source Loopback0
no ntp authentication-key 2
ntp authentication-key 1 md5 02252D6828295E731F1A
ntp trusted-key 1
ntp server 150.1.5.5 key 1
```

Verify

Use the show ntp status command to quickly check all three routers. As mentioned previously, it may take some time to synchronize the clocks, so make sure you apply effective time management.

```
Rack1R4#show ntp status
Clock is synchronized, stratum 9, reference is 150.1.5.5
nominal freq is 249.5901 Hz, actual freq is 249.5862 Hz, precision is
2**18
reference time is CE0E5804.45C446F1 (02:42:12.272 UTC Mon Jul 20 2009)
clock offset is 0.7923 msec, root delay is 57.80 msec
root dispersion is 876.02 msec, peer dispersion is 875.21 msec
Rack1R4#

Rack1R3#show ntp status
Clock is synchronized, stratum 9, reference is 150.1.5.5
nominal freq is 249.5901 Hz, actual freq is 249.5837 Hz, precision is
2**18
reference time is CE0E57E3.591C06C5 (02:41:39.348 UTC Mon Jul 20 2009)
clock offset is -0.4015 msec, root delay is 58.20 msec
root dispersion is 0.64 msec, peer dispersion is 0.20 msec
Rack1R3#
```

```
Rack1R5#show ntp status
Clock is synchronized, stratum 8, reference is 127.127.7.1
nominal freq is 249.5901 Hz, actual freq is 249.5842 Hz, precision is
2**18
reference time is CE0E57D9.964E4136 (02:41:29.587 UTC Mon Jul 20 2009)
clock offset is 0.0000 msec, root delay is 0.00 msec
root dispersion is 0.02 msec, peer dispersion is 0.02 msec
```

Ticket 9

Analyze the Symptoms

Apparently, there is something in the network that does not like fragmented packets. Every time you implement a tunneling solution you should watch out for traffic fragmentation and Path MTU discovery issues. The process of Path MTU discovery was created to resolve the MTU mismatch issue and theoretically should make TCP protocol work with any network MTU. Unluckily, the discovery process is based on sending maximum MTU sized TCP packets with DF bit and listening to ICMP unreachable messages (packet too big). Some firewall, either network or end-host based, could block the ICMP messages and prevent Path MTU from working. One of the solutions to this problem would be clearing the DF bit from TCP packets and allowing the routers to fragment the large packets.

However, this results in another problem. Many network administrators consider fragmented packet to be dangerous, as it may hide a potential attack or cause network performance issues. Thus, some sites simply filter out fragmented packets effectively blocking fragmented TCP sessions. This seems to be the case in our situation.

There are two solutions to this problem: either remove the fragmented packets filtering policy or make sure packets are never fragmented. Unfortunately, it is only possible to ensure the second option if the traffic is TCP based. Using the command **`ip tcp adjust-mss`** you may instruct the router to modify the MSS filed in TCP packet headers. The configuration should be applied on the edge routers, affecting all TCP sessions coming through. The MSS should be selected to be 40 bytes less than the minimum network MTU. However, this solution obviously works only for TCP traffic, and has no effect on UDP-based applications. Still it works in most cases as TCP applications constitute the majority.

For our scenario, we have to find the solution that allows the fragmented traffic to pass through. This means we need to isolate the router dropping the fragmented traffic or re-configure it.

Isolate the Issue

So our first goal is finding the router on the path that is dropping the fragmented packets. What we are going to do is use pings with oversized packets, pinging every router on the path between R2 and R4 and seeing which one starts dropping those. In real life, the firewalls might be filtering ICMP, so this could not be used as a reliable detection mechanism, but in the lab environment this should work in most cases.

```
Rack1R2#ping 150.1.1.1 size 1500

Type escape sequence to abort.
```

```
Sending 5, 1500-byte ICMP Echos to 150.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 757/760/765
ms

Rack1R2#ping 150.1.3.3 size 1500

Type escape sequence to abort.
Sending 5, 1500-byte ICMP Echos to 150.1.3.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 761/761/762
ms

Rack1R2#ping 150.1.5.5 size 1500

Type escape sequence to abort.
Sending 5, 1500-byte ICMP Echos to 150.1.5.5, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)

Rack1R2#ping 150.1.1.1 size 1600

Type escape sequence to abort.
Sending 5, 1600-byte ICMP Echos to 150.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 781/784/786
ms

Rack1R2#ping 150.1.8.8 size 1600

Type escape sequence to abort.
Sending 5, 1600-byte ICMP Echos to 150.1.8.8, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 785/787/790
ms

Rack1R2#ping 150.1.3.3 size 1600

Type escape sequence to abort.
Sending 5, 1600-byte ICMP Echos to 150.1.3.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 789/789/790
ms

Rack1R2#ping 150.1.5.5 size 1600

Type escape sequence to abort.
Sending 5, 1600-byte ICMP Echos to 150.1.5.5, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

After the above output, we may suspect either R3 (outbound filtering) or R5 (inbound filtering). Let's ping some other hosts behind R5 to see if the fragmented packets are getting dropped.

```
Rack1R2#ping 150.1.4.4 size 1600
```

```
Type escape sequence to abort.
Sending 5, 1600-byte ICMP Echos to 150.1.4.4, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)

Rack1R2#ping 150.1.7.7 size 1600

Type escape sequence to abort.
Sending 5, 1600-byte ICMP Echos to 150.1.7.7, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

OK now we need to check both R3 and R5 for fragmented packets filtering. There are basically few ways to filter fragmented packets:

1) Using an access-group applied to an interface.
2) Using a policy-map that routes fragmented traffic to Null0.
3) Using a service-policy that drops fragments.
4) Configuring IP virtual-reassembly on interface.
5) Using IOS IPS to drop fragmented traffic.
6) Using CoPPr or Control-Plane policing to restrict packets going to the router.

We need to check these in sequence on R3 and R5 starting with R3. We know R3 is not configured for inbound filtering, otherwise it would drop packets destined for itself. So we only has to check the outgoing interface:

No access-lists applied:

```
Rack1R3#show ip interface serial 1/0 | inc access
  Outgoing access list is not set
  Inbound  access list is not set
  IP access violation accounting is disabled
```

No policy-map applied:

```
Rack1R3#show policy-map interface serial 1/0
```

No policy-routing configured:

```
Rack1R3#show ip policy
Interface      Route map
Rack1R3#
```

No virtual reassembly:

```
Rack1R3#show ip virtual-reassembly
Rack1R3#
```

And finally no IPS configured in R3:

```
Rack1R3#show ip ips all
Configured SDF Locations: none
Builtin signatures are enabled but not loaded
Last successful SDF load time: -none-
IPS fail closed is disabled
Fastpath ips is enabled
Quick run mode is enabled
Event notification through syslog is enabled
Event notification through SDEE is disabled
Total Active Signatures: 0
Total Inactive Signatures: 0
```

Good, now we should inspect R5's configuration.

```
Rack1R5#show ip interface serial 0/0.35 | inc acce
  Outgoing access list is not set
  Inbound  access list is not set
  IP access violation accounting is disabled
```

Next we see some QoS policy attached inbound to the interface, but it only marks packets and doe not drop anything.

```
Rack1R5#show policy-map interface serial 0/0.35

 Serial0/0.35

  Service-policy input: SET_DSCP_CS1

    Class-map: class-default (match-any)
      49794 packets, 4794222 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
      QoS Set
        dscp cs1
          Packets marked 49794
```

We check policy-routing next:

```
Rack1R5#show ip policy
Interface      Route map
```

After that comes virtual reassembly:

```
Rack1R5#show ip virtual-reassembly
Serial0/0.35:
   Virtual Fragment Reassembly (VFR) is ENABLED...
   Concurrent reassemblies (max-reassemblies): 16
   Fragments per reassembly (max-fragments): 1
   Reassembly timeout (timeout): 3 seconds
   Drop fragments: OFF

   Current reassembly count:0
   Current fragment count:0
```

```
    Total reassembly count:0
    Total reassembly timeout count:0
```

And we have something here. If you look closely, you will notice that the maximum amount of fragments per-packet is set to one. This means only unfragmented packets are allowed by the policy! So in order to fix the problem we only have to remove the VFR configuration.

Fix the Issue

The solution to this problem is remove the VFR configuration in R5

```
R5:
interface Serial 0/0.35
 no ip virtual-reassembly
```

Verify

Lets try pinging using large packets across R5:

```
Rack1R2#ping 150.1.7.7 size 1600

Type escape sequence to abort.
Sending 5, 1600-byte ICMP Echos to 150.1.7.7, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max =
1538/1540/1543 ms

Rack1R2#ping 150.1.4.4 size 1600 timeout 3

Type escape sequence to abort.
Sending 5, 1600-byte ICMP Echos to 150.1.4.4, timeout is 3 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max =
2167/2170/2180 ms
```

Notice the huge delay when you ping R4 – this is because the packets have to cross two slow Frame-Relay clouds, and one of the clouds is even crossed twice.

Ticket 10

Analyze the Symptoms

We already know that there are no physical or layer 2/3 problems that may prevent WWW access, as we got rid of those during previous troubleshooting steps. We may assume there are MTU issues, but the ticket explicitly states that bulky FTP transfers work. Thus, the issue is most likely related to application filtering configuration somewhere in the network. If you look at the path between R5 and BB1 you will notice that there are quite a lot of routers involved. Inspecting all of them for policy configuration could be could be time consuming. We need a way to locate the hearts of the issue without too much effort.

Isolate the Issue

What we are going to do in order to isolate the issue is configure all routers in the path as HTTP servers and emulate HTTP client requests off R5 to all routes in sequence:

```
R1, R2, R3, R6, SW2:
ip http server
ip http path flash:
```

And now we start the chain of HTTP requests. We request a file-name that is unlikely to be found in the flash memory of the router. Under “normal” conditions the server is supposed to return the “404 Not Found” code. If there is filtering configured, the request would time out.

```
Rack1R5#copy http://150.1.3.3/test null:
%Error opening http://150.1.3.3/test (No such file or directory)

Rack1R5#copy http://150.1.1.1/test null:
%Error opening http://150.1.1.1/test (No such file or directory)

Rack1R5#copy http://150.1.8.8/test null:
%Error opening http://150.1.8.8/test (No such file or directory)

Rack1R5#copy http://150.1.2.2/test null:
%Error opening http://150.1.2.2/test (No such file or directory)

Rack1R5#copy http://150.1.6.6/test null:
%Error opening http://150.1.6.6/test (I/O error)
```

From the above sequence we see that the issue is certainly related to R2’s configuration. We get to R2 and star looking for potential filtering configuration. As with the case with the previous ticket involving the fragmented traffic filtering we have the following options:

1) An access-group applied to an interface. Inbound access-list is highly unlikely as R2 responds to HTTP requests. However, the ACL may simply exclude R2 IP addresses.

2) Using a service-policy that matches HTTP traffic and drops it. It is possible to use either NBAR or access-lists for classification
3) IOS IPS is being used to drop HTTP traffic. This is highly unlikely, as it is impossible tuning IPS signatures in IOS.
4) Using some sort of web-filtering such as external Web-Sense server. This requires configuring ZFW or CBAC features.

First we check for access-group applied to the interfaces of R2:

```
Rack1R2#show ip interface serial 0/0 | inc access
  Outgoing access list is not set
  Inbound  access list is not set
  IP access violation accounting is disabled

Rack1R2#show ip interface fa 0/0 | inc access
  Outgoing access list is not set
  Inbound  access list is not set
  IP access violation accounting is disabled
```

Nothing here. Next goes a policy map attached to any of the interfaces. Notice that it’s possible applying the policy map to any interface in any direction and block the traffic in either inbound or outbound direction.

```
Rack1R2#show policy-map interface serial 0/0

 Serial0/0

  Service-policy input: MARK

    Class-map: HTTP (match-all)
      18 packets, 2326 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: protocol http url "*"
      Match: packet length min 1 max 1600
      Match: input-interface Serial0/0
      QoS Set
        precedence 5
          Packets marked 18

    Class-map: class-default (match-any)
      531 packets, 39354 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
      QoS Set
        precedence 0
          Packets marked 531
Rack1R2#show policy-map interface fa 0/0
```

Now there is something suspicious. There is a policy map matching ALL URLs, packet length and the input interface. Plus, there are matches to this class, which signals that the policy-map has been actively used. However, there is one problem – it only marks the packets, and does not drop them. However, there could be something else that drops the packets based on the marking. We continue our filtering inspection and look for IPS rules and or URL filtering.

```
Rack1R2#show ip inspect all

Rack1R2#show ip ips all
Configured SDF Locations: none
Builtin signatures are enabled but not loaded
Last successful SDF load time: -none-
IPS fail closed is disabled
Fastpath ips is enabled
Quick run mode is enabled
Event notification through syslog is enabled
Event notification through SDEE is disabled
Total Active Signatures: 0
Total Inactive Signatures: 0
```

There is nothing here. So we're out of choices from the above list. Let's get back to the suspicious policy-map applied to R2's Serial interface. What could be done with the marked packets? They could be matched by an access-group, another policy-map or policy-routing rules. Since there are no other access-group or policy-maps applied, let's looks for policy-routing configuration:

```
Rack1R2#show ip policy
Interface      Route map
Fa0/0          CLEAR_DF
Serial0/0      REDIREC
```

There are two route-map applied to different interfaces. The first one seems to be related to one of the previous tickets and is used to clear the DF bit on TCP packets:

```
Rack1R2#show route-map CLEAR_DF
route-map CLEAR_DF, permit, sequence 10
  Match clauses:
    ip address (access-lists): TCP_ONLY
  Set clauses:
    ip df 0
  Policy routing matches: 1508 packets, 493784 bytes
```

The other one looks like the one causing us the issue. It matches the IP precedence 5 packets and the redirects them to Null0. Thus all HTTP requests classified by NBAR previously are getting dropped at R2.

```
Rack1R2#show route-map REDIRECT
route-map REDIRECT, permit, sequence 10
  Match clauses:
    ip address (access-lists): PREC5
  Set clauses:
    interface Null0
  Policy routing matches: 14 packets, 2032 bytes
```

This type of configuration was popular in old IOS versions, until Cisco introduced the “drop” actions in MQC syntax. Notice that policy routing does not affect packets going through the process-switching path, and thus all HTTP request to R2 succeed.

Fix the Issue

There are multiple ways to fix the issue. We could either remove the service-policy or remove the policy routing configuration. We’ll go with the second option, as it prevents other unwanted side-effects such as dropping ALL IP precedence 5 marked traffic.

```
R2:
interface Serial 0/0
 no ip policy route-map REDIRECT
```

Verify

We may now repeat our HTTP request test from R5 and this time R6 should return 404.

```
Rack1R5#copy http://150.1.6.6/test null:
%Error opening http://150.1.6.6/test (No such file or directory)
```

This is fine, but what if there is some additional filtering configured in R6? This could be true, and in real life we should probably go ahead and check R6’s configuration. However, in the real life we would probably be able to connect to port 80 across R6 as well. In our scenario backbone routers do not support HTTP servers, so we are going to go ahead and give a quick check to R6. We start with access-lists and then continue to policy-maps, policy routing and CBAC/IPS configurations. Notice that we check the virtual-access interface cloned from the virtual-template in R6.

```
Rack1R6#show ip interface fastEthernet 0/1 | inc access
  Outgoing access list is not set
  Inbound  access list is not set
  IP access violation accounting is disabled

Rack1R6#show ip interface virtual-access 2 | inc access
  Outgoing access list is not set
  Inbound  access list is not set
  IP access violation accounting is disabled

Rack1R6#show policy-map interface

Rack1R6#show ip policy
Interface      Route map

Rack1R6#show ip ips all
Configured SDF Locations: none
Builtin signatures are enabled but not loaded
Last successful SDF load time: -none-
```

```
IPS fail closed is disabled
Fastpath ips is enabled
Quick run mode is enabled
Event notification through syslog is enabled
Event notification through SDEE is disabled
Total Active Signatures: 0
Total Inactive Signatures: 0
```

This assures us that no additional filtering is configured in R6.