Copyright Information

Disclaimer

Table of Contents

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 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.

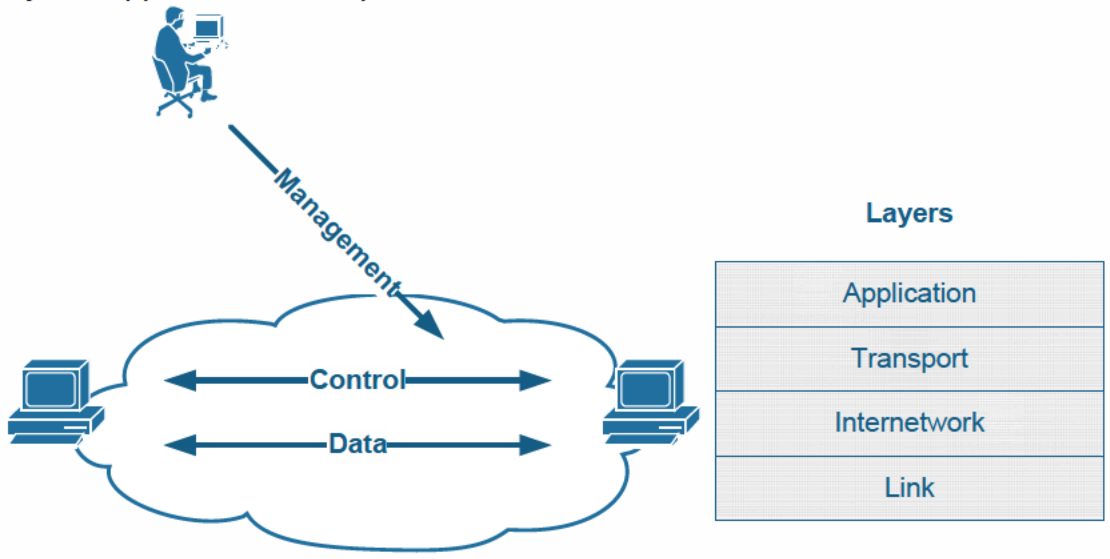
You may enhance your diagrams with any extra information provided, e.g. hints on the network pre-configuration and applications deployed, such as WWW, FTP, SMTP, VoIP and so on. This will help you analyzing symptoms later. You goal at this stage is to get clear picture of the network and discover any 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.

Lab Solutions Structure

Solution documents for Labs 1-6 are 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.

Lab 1 Solutions

Build and Analyze the Baseline

Starting the troubleshooting section, all you have is a diagram and textual information on the network baseline. The objective at this point is to structure the given information, and diagram all the aspects of the network to ease the troubleshooting process. Diagramming may seem somewhat trivial, but in many instances solves the problem directly. The common diagrams that will help outline the network are: L2, BGP Peering, Multicast & Redistribution, and the IPV6 diagrams. Combining several diagrams into one or keeping them separate is a personal preference; in our case we will use separate diagrams for ease of explanation.

Layer 2 Diagram

Fig 1 shows the L2 diagram consisting of the Ethernet connections between the routers & switches, as well the trunk links connecting the switches. The port numbers, Root Bridge, VTP information, and encapsulation types are the kind of information to be listed that will prevent any obvious misconfiguration and help isolate the problem within the layer 2 domain.

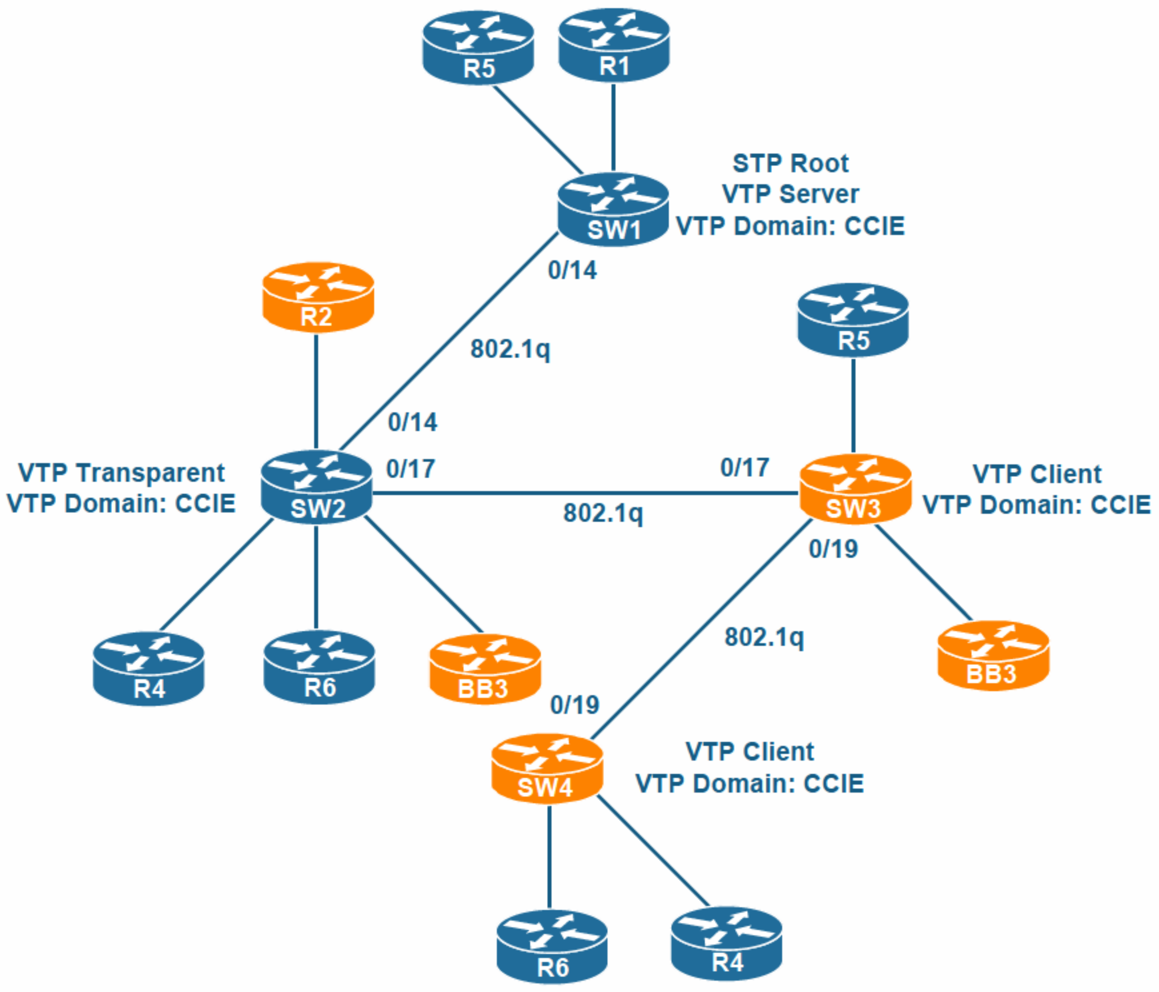


Fig 1

The topology has no physical loops, and thus STP will not engage to block any port. The diagram will be referenced in the coming sections.

BGP Diagram

Based on the information provided, we have constructed the BGP peering diagram as shown in Fig 2. The red arrows mark EBGP peerings while the blue arrows mark IBGP peerings.

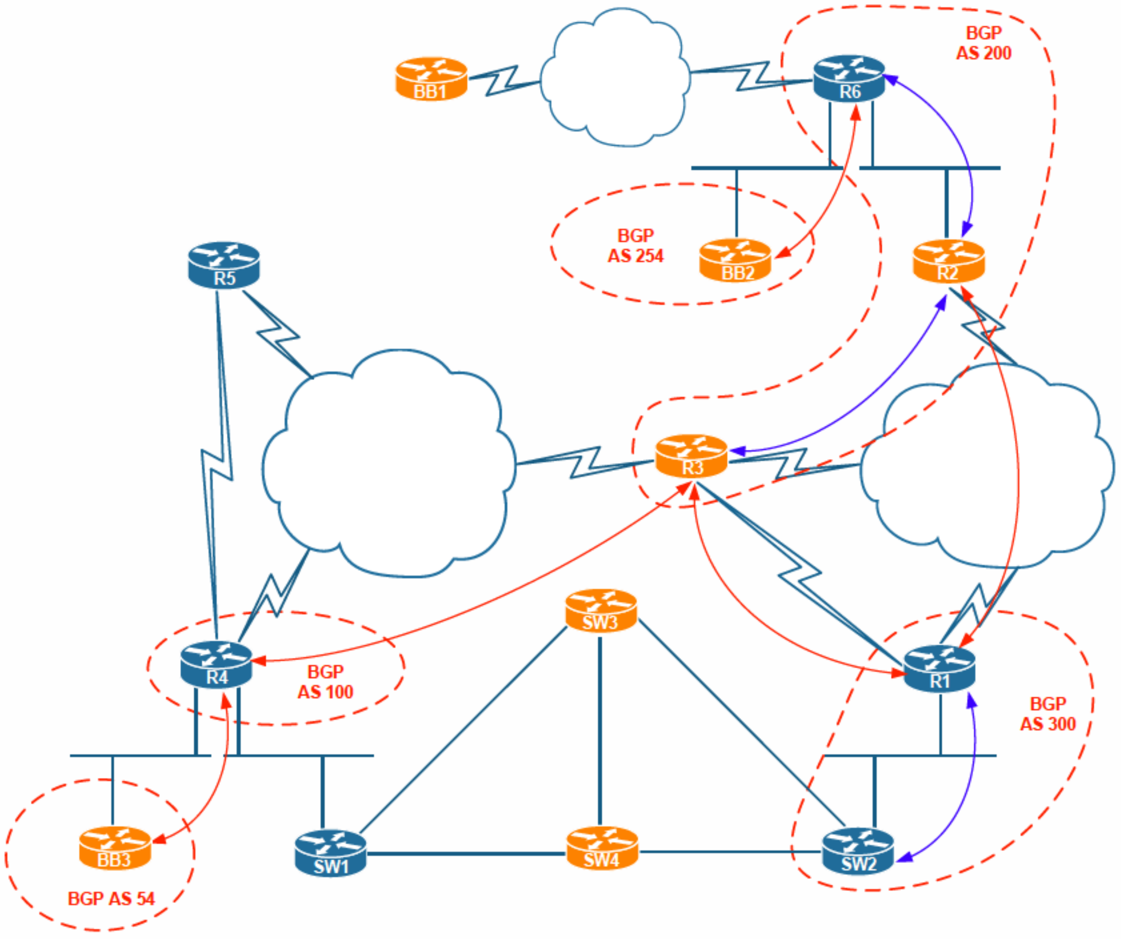


Fig 2

Note 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.

Multicast and Redistribution

Multicast & Redistribution technologies are grouped together in a single diagram because of the dependency PIM has on routing protocols (RPF checks). Fig 3 outlines the IGPs, the PIM enabled interfaces (marked with **RED**) and the redistribution points (arrows).

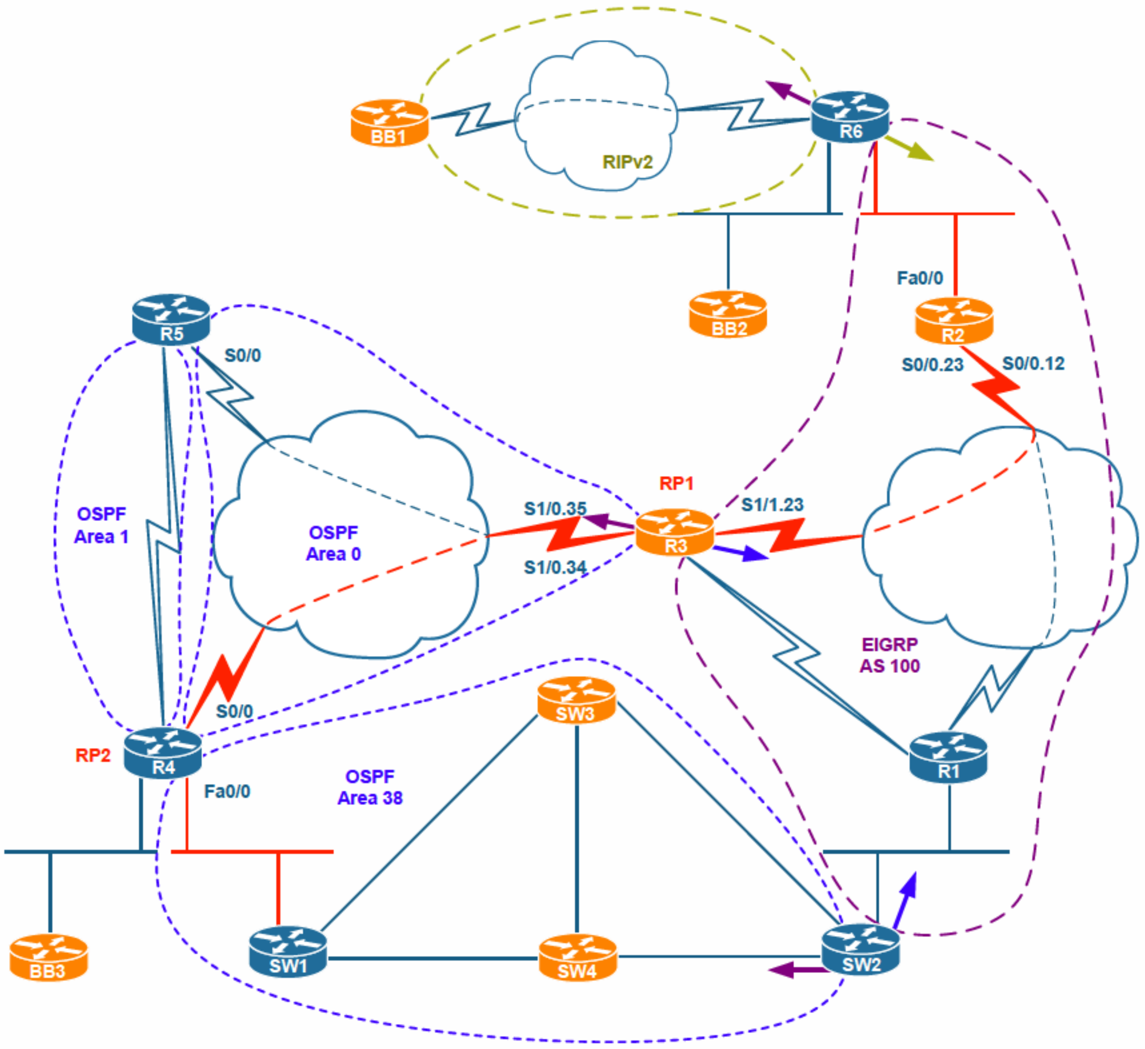


Fig 3

You can also see the RPs of the PIM sparse mode scenario outlined in Fig 3. The detailed group to RP mappings are not shown on the diagram due to the large number of groups.

Redistribution Loops Analysis

By referring to Fig 3, we note that the arrows are used on the redistribution points to signify the direction of redistribution, and the color is used to specify the protocol being redistributed. For instance, on R3 we see EIGRP being injected into OSPF and OSPF being injected into EIGRP based on the corresponding arrow colors. Redistribution loops can only occur in the presence of multiple redistribution points.

Prior to starting our analysis, note that there are three IGP domains in the topology: OSPF, EIGRP AS 100 and RIP. A routing loop will occur when routing information propagates from one domain to another and finally leaks back to the domain it was originated in. It is this re-entering that causes the routing loop. Routing loops are a function of the Administrative distance or the so called "believability of the protocol". We will use the tracing procedure to check for routing loops in the network.

The tracing procedure is to traverse the path that routing information takes from one IGP domain to another as a result of redistribution and checking if the path "loops" i.e. the routing information comes back to the original domain it originated in. Here is the tracing algorithm, consisting of two nested loops, exhausting all routing domains and information flows.

Loop 1: is the bigger (outer loop) which is going to encompass all the routing domains (RIP/EIGRP/OSPF) in our case.

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:** For a single protocol (routing domain) we are going to check the redistribution points available, and follow the path that routing information flows through continuously checking for a looped path of routing information.
>
> > **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 loops 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 applies to our scenario. We will start by first examining the EIGRP AS 100 domain. The EIGRP AS 100 domain 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.

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 the IGP domains and redistribution points as in Fig 4. You can see the "RIP" being "witnessed" passing twice through the EIGRP AS 100 domain thus causing a routing loop.

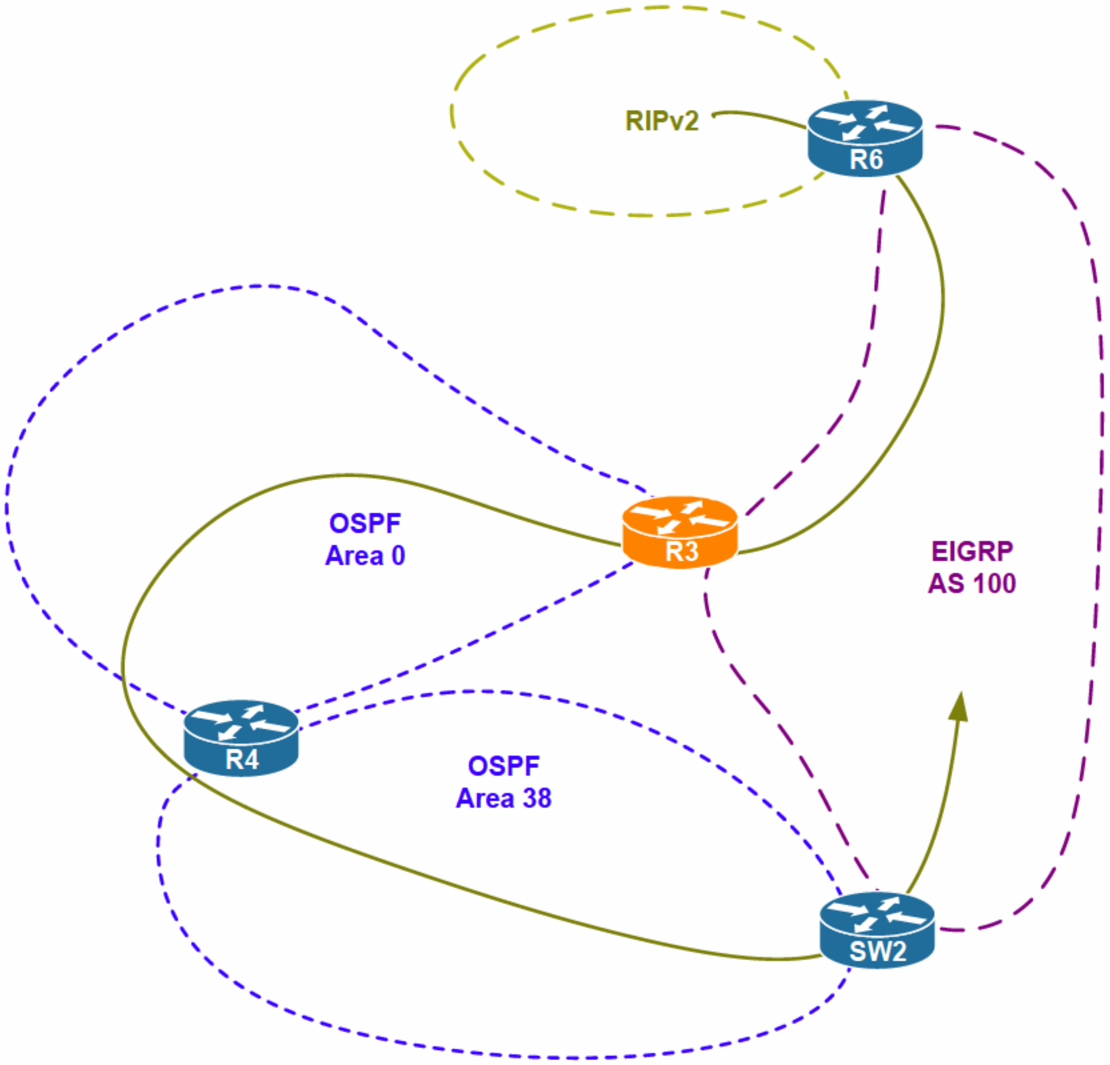


Fig 4

Based on the analysis performed, we conclude that some sort of filtering should be implemented at R3 and SW2 to prevent the routing loop. Methods to achieve this are: tags, access-lists, or AD manipulation. In the simplest case we could just increase OSPF External AD over 170 on R3 and SW2, thus preventing the looped flows.

We should be mostly concerned with SW2 as R3 is not under our control.

Multicast Propagation Analysis

Refer to **Fig 3** when analyzing the multicast section. Multicast is spanning the OSPF and EIGRP routing domains. This is a potential for suboptimal routing & RPF failures. Multicast will always flow through R3 regardless of the source as it is the only interface running PIM between the two domains. One thing to verify is that the OSPF domain will tend to reach the EIGRP domain via R3 to avoid RPF failures. Static RPs are used in the network, thus there will be no trouble propagating the RP information and dealing with RPF checks against Candidate RP and Mapping Agent (Auto-RP case).

IPv6 Diagram

The IPv6 part of this scenario shown in Fig 5 appears to be simple. RIPng is the only IGP running, and IPv6 is only running on three routers. Recall that R1 is the only router under our control so we may expect not to have any major issues with IPv6.

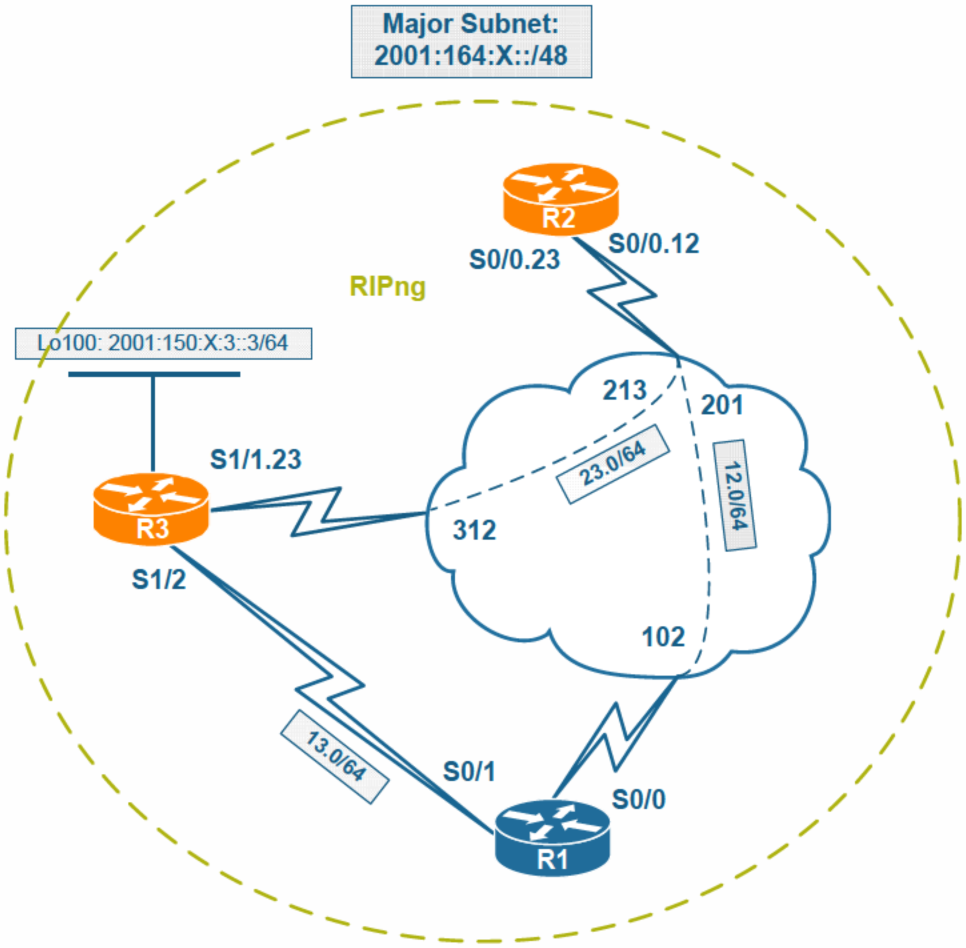


Fig 5

Read over the Lab

Our last step is looking through the tickets and seeing if we can extract any potentially useful information. Looking at tickets 1 and 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 that we should be careful when performing isolation for Ticket 1, and don’t take VLAN 7 into consideration.

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

Tickets 3 and 8 warn us that there could be some problems in getting (R1, R3) on one hand and (R1, SW2) on the other hand to communicate properly. Thus, IGP routing may be affected.

Finally, Ticket 10 clearly states its dependency on two other tickets.

Solutions

Ticket 1

Analyze the Symptoms

Looking at the case, it is definitely not an end-to-end connectivity problem. Probably one of the things that might be causing this is asymmetric behavior (VLAN 7→VLAN 5) versus (VLAN 5 →VLAN 7). Routing from R4 and R5 may be taking the slow ISDN link or vice-versa. This encompasses the first hypothesis for troubleshooting this ticket. Another possibility is that the QOS policy is not applied uniformly throughout the network i.e. the policy at R4 might be giving preferential treatment to the Voice traffic than R5's policy or vice-versa. This is the second hypothesis that needs to be checked.

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

Isolate the Issue

To check for asymmetric routing (Hypothesis 1), network layer testing will be used. 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 policies on 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. We see traffic being fragmented according to the fragment size of 320 bytes. Let's check the serialization delay for such a fragment.

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

To verify fragmentation is actually working, check that the number of fragments increments when you ping across the FR cloud with a packet size greater than 320 bytes. Let us clear the FR counters 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
```

The output verifies that frame-relay fragmentation is working properly.

```
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 verify fragmentation is working properly.

```
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. Take another look at the policies:

R4:

```
  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
```

R5:

```
  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
```

It is obvious that the policy from R4 to R5 uses simple bandwidth reservation, and doesn't use LLQ. Thus R4 is not providing voice traffic the level of service it requires by deploying this policy. Note that the interleaving of VoIP packets as well as data packets is disabled. Our second hypothesis, which states that the QOS policy hasn't been applied uniformly throughout the network, is proven to be true!

Conclusion: Pay due diligence and look through the various outputs 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 SW2 is supposed to load balance across the two links (SW2-SW3) and (SW2-SW4) to reach VLAN 9.Let see the route SW2 takes to reach VLAN 9:

```
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, SW2 is using the path via SW4 to reach it. The problem scope revolves around three devices: SW2, SW3 and SW4. First, we might need to check the link between SW2 and SW3 at a basic level. Another thing to check is to verify that the two paths (SW2-SW3-VLAN 9) & (SW2-SW4-SW3-VLAN 9) have the same metric to achieve equal cost load balancing.

The two hypotheses concerning this ticket are:

Hypothesis 1: Issues with the link between SW2 and SW3.
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 164.1.32.9

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 164.1.32.9, 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 neighbors appear to be UP thus breaking our first hypothesis. 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
```

The total cost for SW2 to reach VLAN 9 taking the path (SW2-SW4-SW3-VLAN9) is 21.

Now let's check what path SW4 chooses to reach VLAN 9:

```
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 msec
```

Apparently SW4 uses the direct link connecting it to SW3 to reach VLAN 9.

Now let us calculate the total cost for SW2 to reach VLAN 9 using the path (SW2-SW3-VLAN 9). This will be the sum of the cost that SW2 needs to reach SW3 (Port-channel 23) and the cost that SW3 needs to reach VLAN 9.

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

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>
```

Looking at the highlighted output we can see that SW3's metric to reach VLAN 9 is 1. Thus the total cost to reach VLAN 9 using the path (SW2-SW3-VLAN 9) is also 21.

The two paths used to reach VLAN 9 from SW2 have an equal cost of 21. Thus, our two hypotheses appear to be invalid and the current situation is:

- 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 an unknown reason.

The next step will involve looking at the OSPF topology and trying to figure out why SW2 doesn't use the direct path to SW3. We will look at the Router LSAs generated by (SW2, SW3, and SW4). Note that in the following output only the relevant links are shown.

```
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
```

The two transit links connecting (SW2-SW3) and (SW2-SW4) are shown. Note the link type of transit implying that a neighbor exists on each of those links.

Now it’s time to check the relevant entries in SW4’s database. Note that since SW3 and SW4 are out of our control, we will check the entries in the database of SW2 that are advertised by SW3, and SW4.

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>
```

All networks appear to be in order – both are marked as transit, thus the OSPF Shortest Path First algorithm (SPF) should account for these links as valid entries in the topology.

Now let us check the Router LSAs generated by SW3:

```
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 output shows the links connecting SW3 to SW2 and SW4. The link to SW4 is a regular transit network, and is a valid link in the OSPF database. However note that the output shows that SW3's link to SW2 is defined as a point-to-point link (SW3's side), and a stub network i.e. SW3 is not able to detect a neighbor on the link even though the adjacency seems to be UP. SW2 treats the link to SW3 as a transit network, but SW3 treats the link to SW2 as a point-to-point network. Even though the adjacency seems to be UP, it is the mismatch in Network types between SW2 and SW3 that leads to the problem.

This is the danger concerning OSPF adjacencies – they come up as soon as timer values match. However, if the topological views of the link aren't the same, OSPF will never be able to build a proper uniform map of the network.

An additional check to perform on SW2 is the OSPF state of Port-channel 23. Notice that in the following output, the state of the local interface is DROTHER; however its priority is 1 (non-zero value). SW3 is acting as the DR & BDR which is not the regular behavior. This proves again that the OSPF adjacency between SW2 & SW3 has a problem due to the mismatch in network types.

```
Port-channel23 is up, line protocol is up (connected)
  Internet Address 164.1.32.8/24, Area 38
  Process ID 1, Router ID 150.1.8.8, Network Type BROADCAST, Cost: 20
  Transmit Delay is 1 sec, State DROTHER, Priority 1
  Designated Router (ID) 150.1.9.9, Interface address 164.1.32.9
  Backup Designated router (ID) 150.1.9.9, Interface address 164.1.32.9
  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)
  Cisco NSF helper support enabled
  IETF NSF helper support enabled
  Index 2/2, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 3
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 150.1.9.9  (Designated Router)
  Suppress hello for 0 neighbor(s)
```

Conclusion: OSPF adjacencies are not to be fully trusted! You always need to be skeptical about why one path is not being used throughout the network.

Fix the Issue

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

Verify

Now that the neighbor relationship between (SW2-SW3) is in the proper state, we can check SW2 routing towards VLAN 9 and verify load balancing across the two paths:

```
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 is clearly tackling a BGP peering issue. The scope of the problem is related to the link connecting R1 and R3. A very basic approach would be to test reachability between the two routers:

```
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
```

Network layer testing is successful. Let us now move to troubleshooting the transport layer. Let us recall some basic rules regarding BGP adjacency:

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

All the events that happen during the transport session establishment are reflected in BGP notification messages – e.g. mismatched BGP AS numbers.

Recall from RFC 1771

“After a transport protocol connection is established, the first message sent by each side is an OPEN message. If the OPEN message is acceptable, a KEEPALIVE message confirming the OPEN is sent back. Once the OPEN is confirmed, UPDATE, KEEPALIVE, and NOTIFICATION messages may be exchanged.”

The following output shows that the BGP state is “OpenSent” thus BGP has established the TCP session, and is currently negotiating the required parameters to establish the peering.

```
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
```

Telnet is one way to simulate the initial 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]
```

This proves that the TCP connection on port 179 is successful, thus no traffic filters (access-lists…etc) are blocking the TCP connection. Recall that the BGP state is “OpenSent”, and this means initial TCP connection is established, and that probably there is a parameter mismatch between the two routers leading to the problem.

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 transactions and BGP messages.

```
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

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

Looking at the output, it seems that R3 is continuously sending the same OPEN message to R1, as if it is not receiving the OPEN CONFIRM message from R1. Thus a BGP parameter mismatch is supposed to exist, preventing R3 from receiving the messages R1 is sending. R1 considers the messages coming from R3 as bad segments.

```
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)
```

The facts that we have accumulated up to this point are:

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

Despite using heavy artillery, the cause of the problem is still not obvious. 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 the OpenSent state. The other side considers the transport connection open, but never sees an open message. The hold time and keepalive interval are set to their default.

```
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

As for the transport connection the connection is established and later dropped. The very important fact to note in the following output is the incoming and outgoing TTL values. Minimum incoming TTL is 254 and outgoing TTL is 255.

```
  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
```

As mentioned previously, BGP by default uses a TTL of 1 when establishing an EBGP peering. However, the above output shows that the remote party (R3) is expecting the incoming BGP messages to have a TTL of 254 and sets outgoing TTL to 255. Looking at the output on R1 shows:

```
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
```

R1 on the other hand is expecting an incoming TTL of 0 from R3 and sets its outgoing TTL to 1. Finally we were able to spot out the configuration mismatch! We should have read carefully the information in the scenario first before spending so much time on heavy debugging. The reason why the initial connection from R3 to R1 succeeds is because R1 responds to the initial handshake using the SAME TTL value in the initial SYN packet. Thus, it becomes apparent that the security feature R3 is using is General TTL Security Mechanism.

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. The OPEN sent and confirm messages of R1 have a TTL of 1, whereas R3 is expecting a minimum TTL of 254. This is the reason behind R3 rejecting the OPEN sent & confirm messages sent by R1. The telnet session test was successful due to the fact that by default outgoing TCP connections uses TTL of 255, which perfectly matched the setting of R3 (Minimum TTL 254 Expected).

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 R1 to accept incoming BGP packets from R3 with a minimum TTL of 254 and set the outgoing TTL to 255. Thus the two sides match!

Verify

The connection has moved to the established state!

```
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

The first thing to do is test connectivity from R5 to VLAN 41:

```
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  *  *
```

The problem scope seems to be limited to R4 and SW1.

Isolate the Issue

To justify the scope of the problem, an attempt to ping SW1's VLAN 41 interface from R4.

```
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)
```

The ping fails, this justifies our initial hypothesis. OSPF adjacency between R4 and SW1 is obviously down as per the following output.

```
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
```

Let's investigate the problem at the physical as well as the data link level.

Referring back to the initial layer 2 diagram, we note that R4 connects to SW2 and it is SW2 that connects to SW1. CDP is a very basic and helpful way to discover the physical connections between the devices.

```
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
```

Looking at the above CDP output, the physical connections connecting R4 and SW2 and those connecting SW1 and SW2 are fine. Our guess is that the problem exists at Layer 2 probably related to STP or VTP.

Prior to checking STP & VTP issues, let us check no filtering is configured on 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 a time-saving standard check showing that no filtering is applied.

```
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 required ports and VLAN configuration is correct. The problem gets restricted to being a spanning-tree problem dealing with VLAN 41. Recalling that SW1 is the root bridge of the spanning-tree domain, we go ahead to check the spanning-tree domain:

```
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
```

The output clearly shows that something is wrong. SW1 is supposed to be the root, however we see SW2 declaring the root port as Fa0/17 in the direction of SW3 and declaring Fa0/14(towards SW1) as a downstream designated port. However this shouldn’t prevent communication at a basic level.

The output on SW1 is as follows:

```
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
```

Interface Fa0/14 on SW1 is in the root inconsistent state and is therefore blocked. The most probable reason is that root guard is configured for this interface.

```
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
```

Root guard is enabled on the port, thus when SW1 receives a superior BPDU on Fa0/14 interface, it puts Fa0/14 in the root inconsistent state and thus blocks the port. The problem is that a bridge other than SW1 claims to be the root for VLAN 41, whereas the given clearly states that SW1 is the root of the spanning-tree domain.

```
Rack1SW2#show spanning-tree vlan 41

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

The claimed root has the priority set to 4096+41, which means that the root bridge priority is 4096. The additional 41 is due to the “extended system-ID” feature, which takes some bits from the priority field to guarantee a unique bridge ID by concatenating it with the VLAN number.

Conclusion: For problems relating to L1/L2 issues, troubleshoot the problem using the bottom-up approach (starting with the physical connections...etc). Remember that even in a topology free of loops, STP problems can still occur!

Fix the Issue

Configure SW1 as the root bridge for VLAN 41 by setting its priority lower than any other bridge priority (our case we set it to 0). Another solution would be to remove the root guard; however this will break the lab requirement which states that SW1 is the root bridge.

```
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
```

Recall that the initial problem was the connectivity between VLAN 5 & VLAN 41.The end-to-end connectivity test passes as shown below & our ticket is solved successfully!

```
Rack24R5#ping 164.1.47.7 source fa0/0
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 164.1.47.7, timeout is 2 seconds:
Packet sent with a source address of 164.1.5.5
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 116/116/120
ms
```

Ticket 5

Analyze the Symptoms

R6’s power supply fails with no recent backup. The objective is to recover the router to the original state. Having no idea about the level of damage, we start the troubleshooting process using the bottom-up approach, checking interface state and per-link communication. Complete analysis and troubleshooting will be performed in a methodological manner to lead to a feasible solution for the ticket.

Isolate the Issue, Fix the Issue & Verify

First, let us check the status of all interfaces:

```
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
```

The above output allows us to detect any interfaces that are in the down state. R6’s connection to BB2 is administratively down. The IP addressing matches the diagram, so we start by fixing this issue:

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

After bringing all the interfaces to the UP state, a per link connectivity test needs to be done.

```
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
```

The connection between R6 and BB1 across the frame-relay link appears to have a problem. We tend to 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 per the above output, LMI exchange is occurring properly between R6 and the frame-relay switch; we are also being notified about the local DLCI. Since this is a physical interface, (multipoint) mappings are required to reach the other side. Let’s check 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 self-mappings is not a good sign but should not affect connectivity as they are mapped to unused DLCIs. Our interest is in the last static mapping which is configured incorrectly as the remote peer ip address is 54.1.2.254. Based on this, we fix the 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
```

Let ‘sverify that pings to BB1 are now successful:

```
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
```

Having solved the per-link connectivity issue to BB1, we now move to check our IGPs.

```
Rack1R6#show ip route rip

Rack1R6#
```

This indicates that no routes are being received via RIP. One thing to check for is the reception of RIP updates through debugging.

```
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 problem; R6 is considering the V2-updates coming from BB1 to be illegal. Thus, the logical conclusion is that R6 is running RIP Version 1. To verify this:

```
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)
```

This above output proves that the RIP process is set to send & receive version 1 on the frame-relay interface. To solve the problem, we configure R6 to run RIP version 2 globally.

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

The message indicating the illegal version didn't stop. The most probable reason is that R6's Frame-Relay interface is configured to receive RIP version 1 at the interface level, thus overriding the global process command.

```
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)
```

To verify this we check the send & receive version of R6's Frame-Relay interface:

```
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)
```

To solve this issue, we need to configure R6 s0/0 interface to receive version 2 to ensure RIP-V2 updates can be received on R6's s0/0 interface.

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

Now we should be able to receive the V2-updates 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 process, 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)
```

The output references a BGP authentication problem (TCP Port 179). Recall that the authentication password is “CISCO”. Hence, we configure R6 to use the MD5 hash of CISCO causing each segment on the TCP connection to be verified.

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

After configuring R6's BGP authentication, the BGP peering is properly established.

```
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
```

Having fixed the RIP & BGP, let us move to troubleshooting EIGRP.

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

The output indicates that R6 has not formed an EIGRP neighbor relationship. Let us check the global settings of the protocol.

```
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
```

All the interfaces of R6 are set to “passive”. To restore the EIGRP neighbor relationship between R6 and R2 we need to un-passive the Fastethernet0/1 interface.

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

The neighbor relationship between R6 & R2 is restored as shown in the output below.

```
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 R1, a router in the EIGRP domain, sees the prefixes R6 redistributed from RIP into EIGRP.

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

R1 is not learning the prefixes R6 redistributed from RIP into EIGRP as EIGRP routes. Let us check if R1 has the route information from any routing protocol.

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

Checking for the possibility of filtered updates, use the following command:

```
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 us check if R6 has these routes in the EIGRP topology table as expected.

```
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 has actually performed the redistribution from RIP into EIGRP. Now, we need to know if R2 & R3 are getting the redistributed RIP routes, and these routers are out of our control. One simple trick, is to create static route to reach one of the redistributed rip routes (212.18.1.0/24) pointing towards R2, and attempt to “traceroute” from R1.

```
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 *
```

Since the traceroute reaches R6, we can conclude that both R2 and R3 have a route to the redistributed RIP network, else they would have dropped the packet. Don’t forget to remove the static route after the test!

To check if R1 is receiving the EIGRP updates from R6, debugging can be an excellent tool. After running the debug, 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
```

The above output verifies that R1 is indeed receiving the update packet pertaining to the redistributed rip networks. 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 point is worth the check, as the routes coming from R6 are external routes. If R6 & R1 happen to have the same router-id, the external routes coming from R6 will be filtered as they arrive to R1.

Let us check R6's router-ID:

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

The router-id of R1 and R6 are identical, causing R1 to filter the external routes. This is an additional loop-prevention mechanism in EIGRP to make sure external routes do not enter the domain twice. This also happens due to the fact that external EIGRP updates carry the originating router field within the update. Thus R1 will not accept the external route update from R6, since it "mistakenly assumes" it originated it!

To solve the problem, we set the EIGRP router-id of R6 to be unique.

```
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 sessions.

```
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
```

One final test would be to traceroute from R1 to one of the prefixes learned from BB1 to ensure 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

The prefixes of AS 54 are not reachable by AS 254. An initial check would be to verify the peering relationships between R4 (AS 100) & BB3 (AS 54) on one hand, and R4 (AS 100) and R3(AS 200) on the other hand.

```
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
```

As per the output, R4's peering relationships with R3 and BB3 is properly established.

Having verified the peering relationships, let us check that R4 advertises AS 54 prefixes to R3.

```
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 check if R6 receives the AS 54 prefixes from R2:

```
Rack1R6#show ip bgp neighbors 164.1.26.2 routes

Total number of prefixes 0
```

R6 is not receiving these prefixes even though R3 does. So our troubleshooting area is narrowed to involve AS 200 routers (R2, R3 & 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

Let us check if R6 is configured to filter any prefixes it receives 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 us check the per-neighbor statistic on R6, i.e. the number of prefixes exchanged with R2, as well as local policies leading to the denial of prefixes.

```
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>
```

Look closely at the above output. It states that the total number of received prefixes is 40, and that a bunch of them are being denied due to CLUSTER_LIST loop, as well as ORIGINATOR loop. This means that R6 is receiving prefixes having its BGP router-id in the CLUSTER_LIST or ORIGINATOR attribute. The route-reflector appends its CLUSTER-ID in the CLUSTER_LIST field as it reflects routes from client to non-client peers or vice-versa. The ORIGINATOR attribute is the BGP router-id of the router that originated this route within the local AS and is also used as a loop prevention mechanism. Thus a logical hypothesis is that another router is using the same BGP router-id.

A debug can help us verify our hypothesis:

```
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 the output, it becomes clear that R2 is using the cluster-ID of 150.1.6.6 which happens to be the local router ID of R6. Note that by default, in a cluster formed of a route-reflector and its clients, the Cluster-ID is identical to the Route-reflector ID. As R6 is receiving the update, it sees its own Cluster-ID (Router-ID) in the Cluster list field (150.1.6.6), thus it rejects the update.

To solve the problem, we need to change the BGP Cluster-ID of R6 to a non-conflicting ID.

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, be careful when setting the BGP cluster-ID in non RRs!

Note that duplicate router-ids, cluster-ids...etc are a main potential for loops in many protocols. In Ticket 5, we saw the same idea used with EIGRP. In BGP, it is used more frequently & extensively especially when checking 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 BB2 (AS 254) has 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
```

For further clarification, let us go through a detailed check of a single route for instance 28.119.16.0/24.

```
Rack24R6#show ip bgp 28.119.16.0
BGP routing table entry for 28.119.16.0/24, version 2
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Advertised to update-groups:
        1
  100 54
    150.1.4.4 (metric 2428416) from 164.1.26.2 (150.1.2.2)
      Origin IGP, metric 0, localpref 100, valid, internal, best
      Originator: 150.1.3.3, Cluster list: 150.1.6.6
```

The Originator field points to R3 because it originated the route within AS 200. The route-reflector appends the Cluster list field when reflecting the routes, thus R2 is configured to have the Cluster id of 150.1.6.6. The fact that R6's Cluster ID (Router-ID) was already present in the Cluster list field of the updates coming from R2 resulted in R6 rejecting the updates.

Ticket 7

Analyze the Symptoms

Let us recollect the facts given in the ticket to start troubleshooting. VLAN55 users are no longer able to get an IP address due to a “security” tuning on SW3. A DHCP server is configured on R5.

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
```

Configure SW2 as a DHCP client itself:

```
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
```

The above output shows that R5 is indeed receiving the DHCP request. Note that the “giaddr” is a field in the DHCP request that provides information about the relay agent that forwarded the packet. The giaddr field should be nonzero per the RFC standards. Based on the fact that the security tuning on SW3 caused the problem, one can obviously identify that the implemented feature is DHCP snooping. DHCP snooping is supposed to insert the information option, and the “giaddr” field of zero, as the switch no longer “relays” the DHCP packet. SW3 is out of our control, so R5 should be configured in a manner to accept packets with a “giaddr” of zero.

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

Fix the Issue

R5’s DHCP server must be configured to accept packets even those having the giaddr field of zero.

```
R5:
ip dhcp relay information trust-all
```

Verify

Verify that SW2's VLAN 55 has taken an IP address:

```
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
```

Confirm that 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

Let us start by checking the BGP peering 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
```

SW2 peering relationship with R1 is in the "ACTIVE" state. To get more detailed information check the following:

```
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
```

The output clearly indicates that no active TCP connection is taking place.
Let us check the connectivity between SW2 & R1:

```
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)
```

Thus the problem appears to be a link problem. The problematic area is limited to R1, SW2 & SW1 as SW1 is connecting R1 & SW2 together (Refer to L2 diagram).

Isolate the Issue

Let us check the link status of SW2, R1, and SW1.

```
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
```

The line protocol of R1’s Fa0/0 is down. This indicates a configuration problem between R1 & SW1. Let us check SW1’s fast Ethernet interface.

```
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 in the err-disabled state. The cause of this “err-disabled” state could be found by checking the logging messages.

```
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
```

Buffer & console logging have been disabled as per the above output. Let us enable them and get SW1’s interface UP again to check for the problem.

```
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
```

The problem becomes evident; SW1's port is in the err-disabled state due to a port-security violation. We check the port-security configuration on SW1's Fa0/1 port.

```
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
```

One MAC address is configured to communicate with the port. Note that looking at the offending MAC address obviously this is of the form 0000.0c07.acxx (HSRP virtual MAC address). Thus R1 is communicating with SW1, using its own burned-in address, as well as the HSRP MAC address resulting in a port-security violation, and an err-disabled port on SW1.

Conclusion: Sometimes irrelevant configuration may cause cascading effect and result in communication problems!

Fix the Issue

The problem can be solved by either configuring the switch to allow more MAC addresses, or configuring the router to use the hardware (BIA) for HSRP communication. Note that the ticket mentioned that configuration changes of ONLY R1 & SW2 are allowed, this leads us to tune R1's configuration.

```
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 shut and un-shut SW1's Fast Ethernet 0/1 interface.

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

Recall that using the BIA has some disadvantages. When the active forwarder changes, a gratuitous ARP packet needs to be sent to update the ARP caches of all nodes, else some clients will keep using the old MAC address.

Verify

Confirm that the BGP peering relationship is 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

Looking at the IPV6 diagram, we may safely assume that there is something wrong with R1's configuration as the other routers are out of our control. First, let us check that the reported problem truly 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)
```

R1 can’t ping R3’s loopback100 interface. The ticket states that no link issues exist, however it never hurts to 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]
```

The above output ensures that no physical problems exist. We go ahead and check the IPV4 connectivity to isolate the problem.

```
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
```

The IPV4 domain tests are successful. We proceed to limit the problem scope to the IPV6 domain.

Isolate the Issue

On R1, let us test the IPV6 connectivity to R2 & R3.

```
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
```

Connectivity Tests are successful. Let's check the routing table of R1 for Loopback 100's subnet.

```
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
```

The route to loopback100 exists, however the initial PING test failed. One thing we may need 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 us 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. It can be inferred that either R1 has no mapping to reach this next-hop address, or the remote side (R2) has no mapping back. However, note that R2 is using a point-to-point sub-interface which requires no mapping. Thus the logical reasoning is that R1 has no mapping to reach the next-hop address. Let us check the local mappings on R1.

```
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
```

There is no mapping to FE80::202! This appears to be the root cause of the problem.

Conclusion: Be careful when running IPV6 routing protocols on IPV6 NBMA interfaces, and recall that the next-hop of routing updates always points to the link-local addresses. IPV6 frame-relay doesn’t support inverse-ARP.

Fix the Issue

We simply add the mapping to R2’s link-local address.

```
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

Refer to the multicast distribution diagram. Recall that we discussed RPF issues that may happen during redistribution. RPF occurs due to the fact that a multicast stream is received on an interface which is not the best interface leading to the source (Routing Protocols). R4 is a possible RPF failure point. In order to check this let us verify if R4 sees the route to the source (VLAN 26) through 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
```

RPF check passes on R4! RPF checks can occur anywhere along the path between the source & receiver. However, the only router other than the source under our control is R4.

The best method to troubleshoot multicast is to simulate the multicast flow from server (source) to client (receiver), checking for potential issues at every step. Note that the RP for 226.37.1.1 is R4 as per the baseline information. Let us configure R6 as a multicast source & SW1 as a multicast sink.

Isolate the Issue

R6 is configured to flood all multicast traffic out of VLAN 26 interface. Recall that the DR is elected on every segment, and its purpose here is to register the source with the RP. Since the priorities happen to be the same, R6 will be elected the DR (Highest IP address), thus it needs to know about the RP to be able to send the register messages. SW1 is just configured to accept multicast packets for the desired group.

```
R6:
ip multicast-routing
!
access-list 3 permit 225.10.0.0 0.48.255.255
access-list 4 permit 226.37.0.0 1.8.255.255
!
ip pim rp-address 150.1.3.3 3
ip pim rp-address 150.1.4.4 4
!
interface FastEthernet 0/1
 ip pim sparse-dense-mode
```

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

The multicast routing table of R4 is checked. 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 output indicates that R4 has the outgoing interface set to Fast Ethernet 0/0, as this is where the join came in, and the incoming interface to null due to the fact that R6 didn't start streaming.

We 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. Note that the 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
```

The RP still has no clue about the traffic source. As we verified that no RPF failures are taking place, one good method to check the problem is through debugging the PIM messages.

```
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
```

The above debugging clearly shows that the PIM Register messages from R6 are being dropped, probably due to a policy configured on R6's serial0/0 interface. Let us 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 first note that there are no adjacent neighbors. Secondly, we notice that there is a neighbor filter applied to the interface, which most probably is the cause of 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)
```

Apparently, everything is being denied. Let's go ahead and remove it.

Conclusion: In some situations, having fewer devices under your control would ease the troubleshooting process.

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

Finally, it is time to check if the stream from R6 reaches SW1.

```
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
```

Finally, let us check R4’s multicast routing table.

```
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
```

Note that the above output references the behavior of sparse-mode which starts by a shared-tree where everything passes by the RP, and then shifts to a source based tree looking at the best path between the source & client (Irrelevant of RP). Note that this behavior may be tuned by the command ip pim spt-threshold.

Lab 2 Solutions

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.

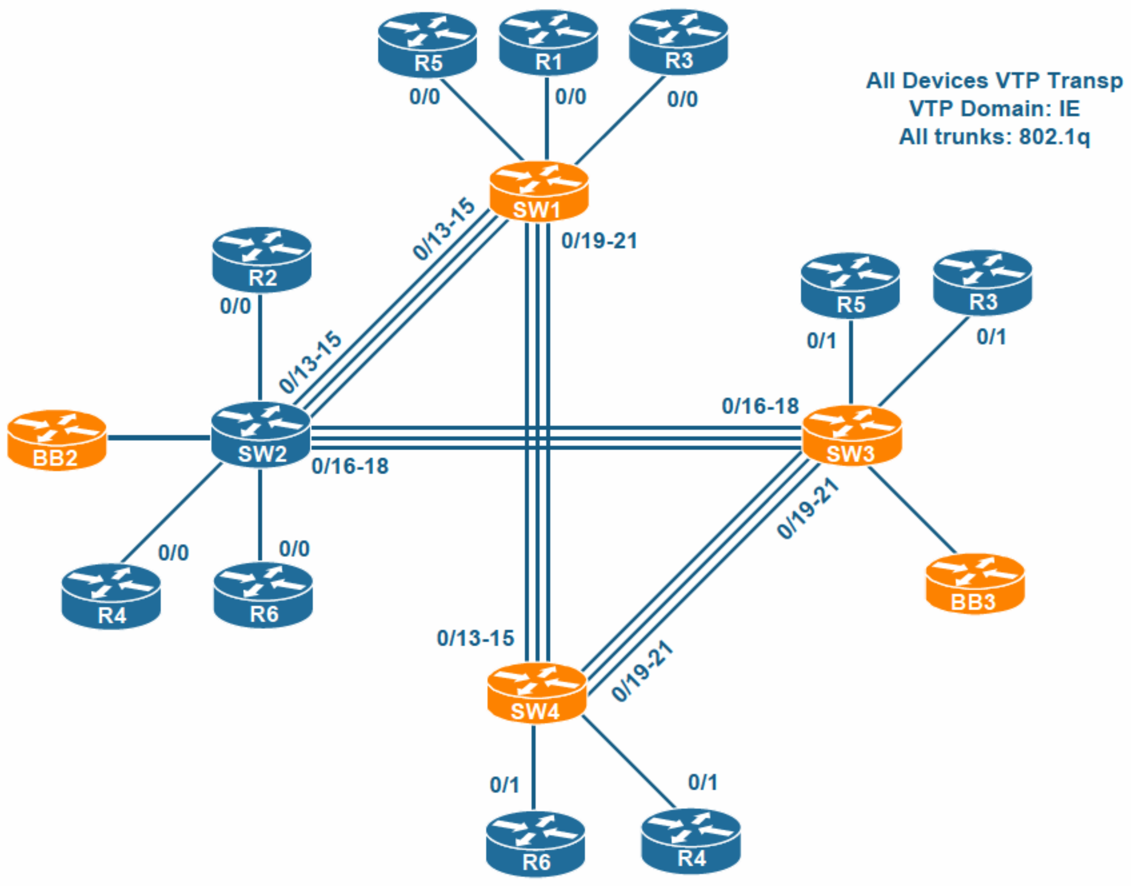


Fig 1

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.

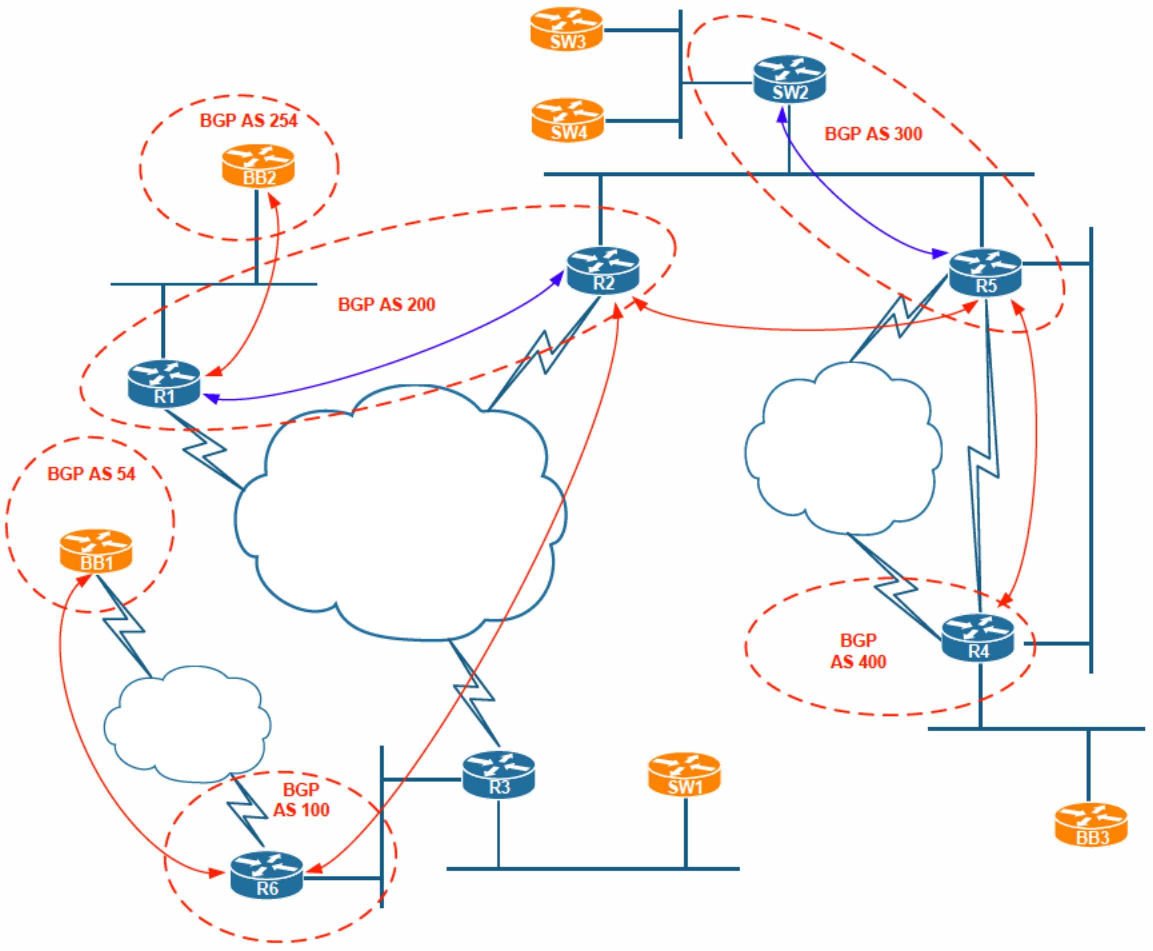


Fig 2

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.

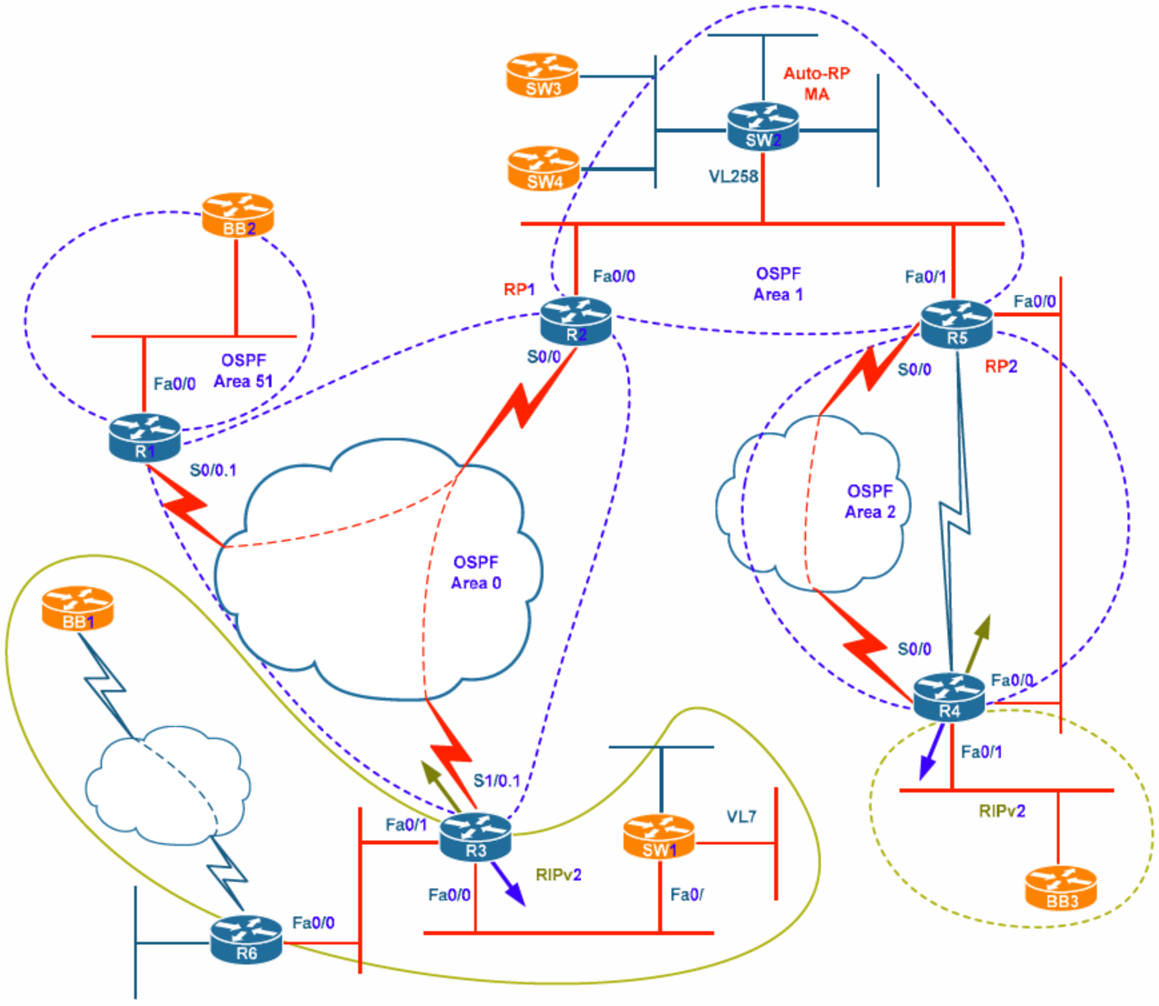


Fig 3

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.

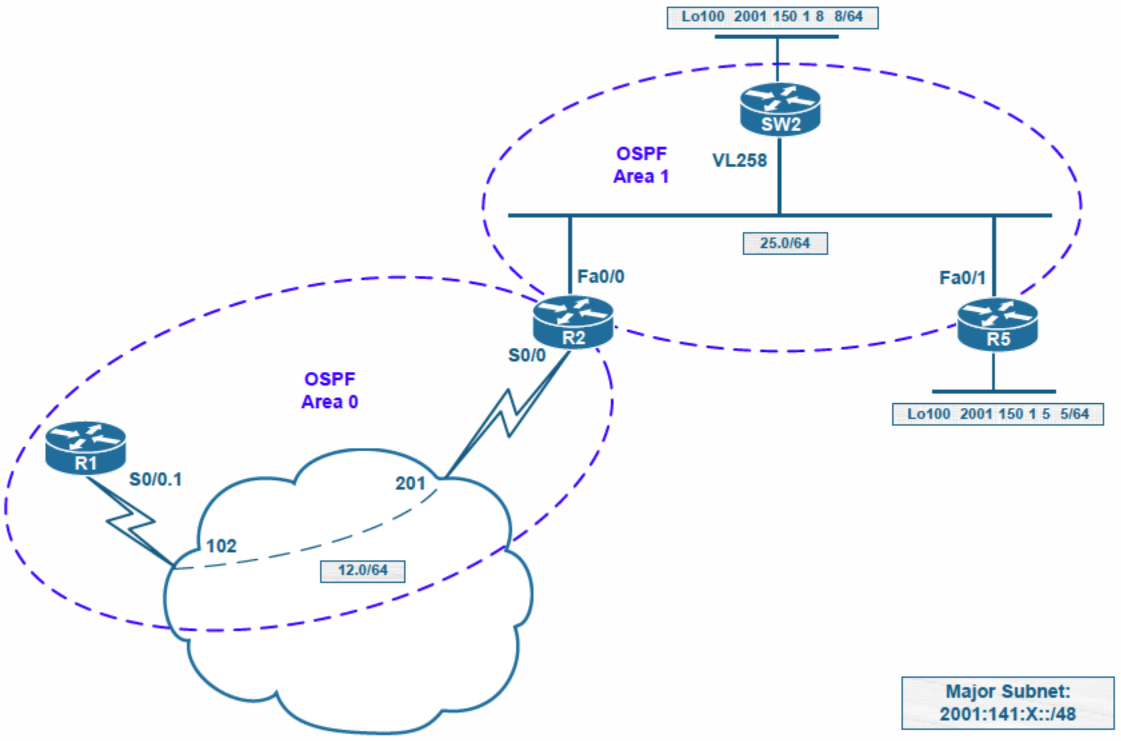


Fig 4

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 suggestion 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 make 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 ARP 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 something 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#  <MISSING THE DEBUG COMMAND>
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” reason.

```
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 have 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 due 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 issue – R1 is not configured as a 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 see 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 31000000 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

```
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 see SW1 as the route information source, which means authentication is working now!

Note

In more recent versions of IOS software, the debugging commands will actually revent the plain-text password:

```
Rack1R3#debug ip rip
RIP protocol debugging is on

RIP: received packet with text authentication CISCO4321
RIP: ignored v2 packet from 141.28.37.7 (invalid authentication)
```

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
```

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
  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 no 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 bullseye, 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:

```
R5:
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 no mention of 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 have to find what might be causing the route to lose its 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 no 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
```

Note 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 not 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 82 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
020c020a00008d019105000000000606564c414e3435' in class OPTION82
DHCPD: input pattern 'relay-information
020c020a00008d019105000000000606564c414e3435' 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 what 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 steps for fixing issues found 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 be 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 out 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:
ip multicast-routing
!
interface Tunnel0
 ip pim sparse-mode
```

As you noticed, even though PIM was enabled on some R5's interfaces, multicast routing was not configured! Luckily, the router will give you an explicit warning when you try enabling PIM without multicast routing enabled. 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 that have been made:

```
R2:
ip pim autorp listener
!
interface Serial 0/0
 ip pim nbma-mode
!
interface Tunnel0
 ip pim sparse-mode

R5:
ip multicast-routing
!
interface Tunnel0
 ip pim sparse-mode
```

Verify

We already verified everything while isolating the issues, so else is nothing more left to verify!

Lab 3 Solutions

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.

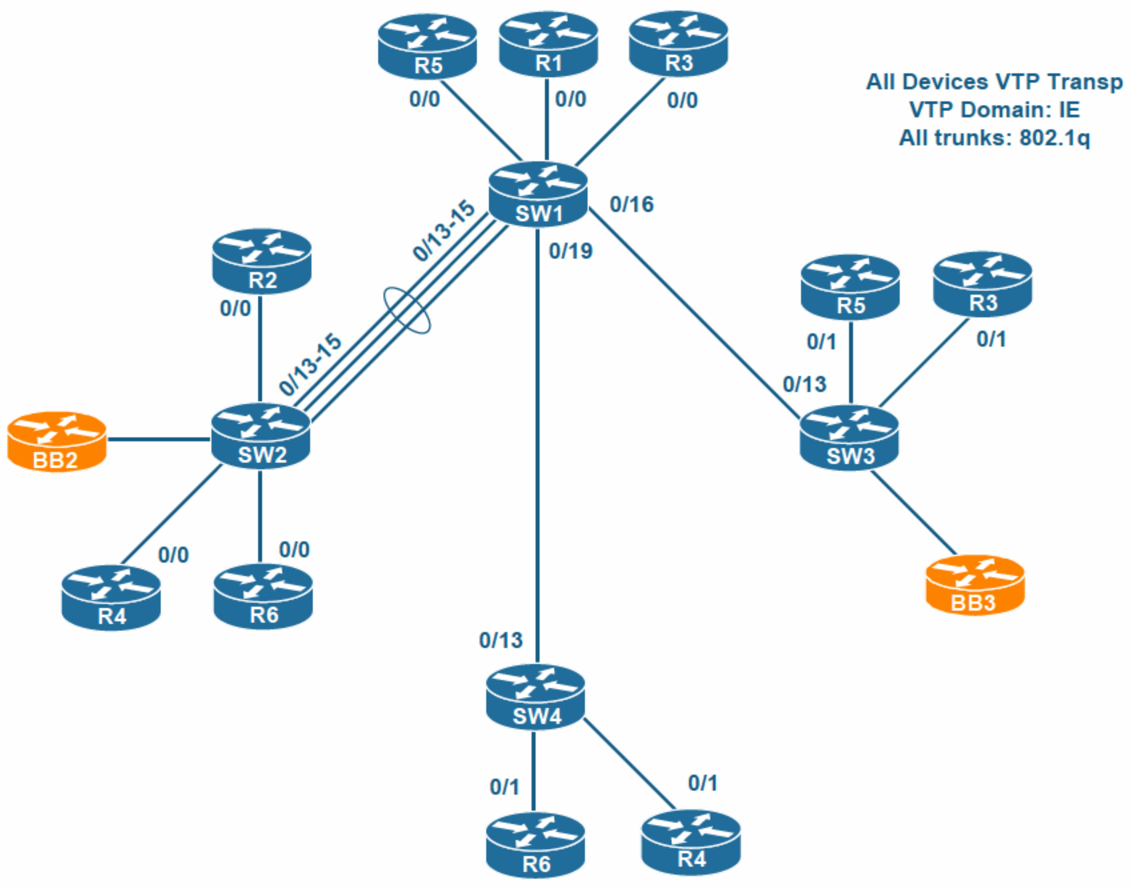


Fig 1

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.

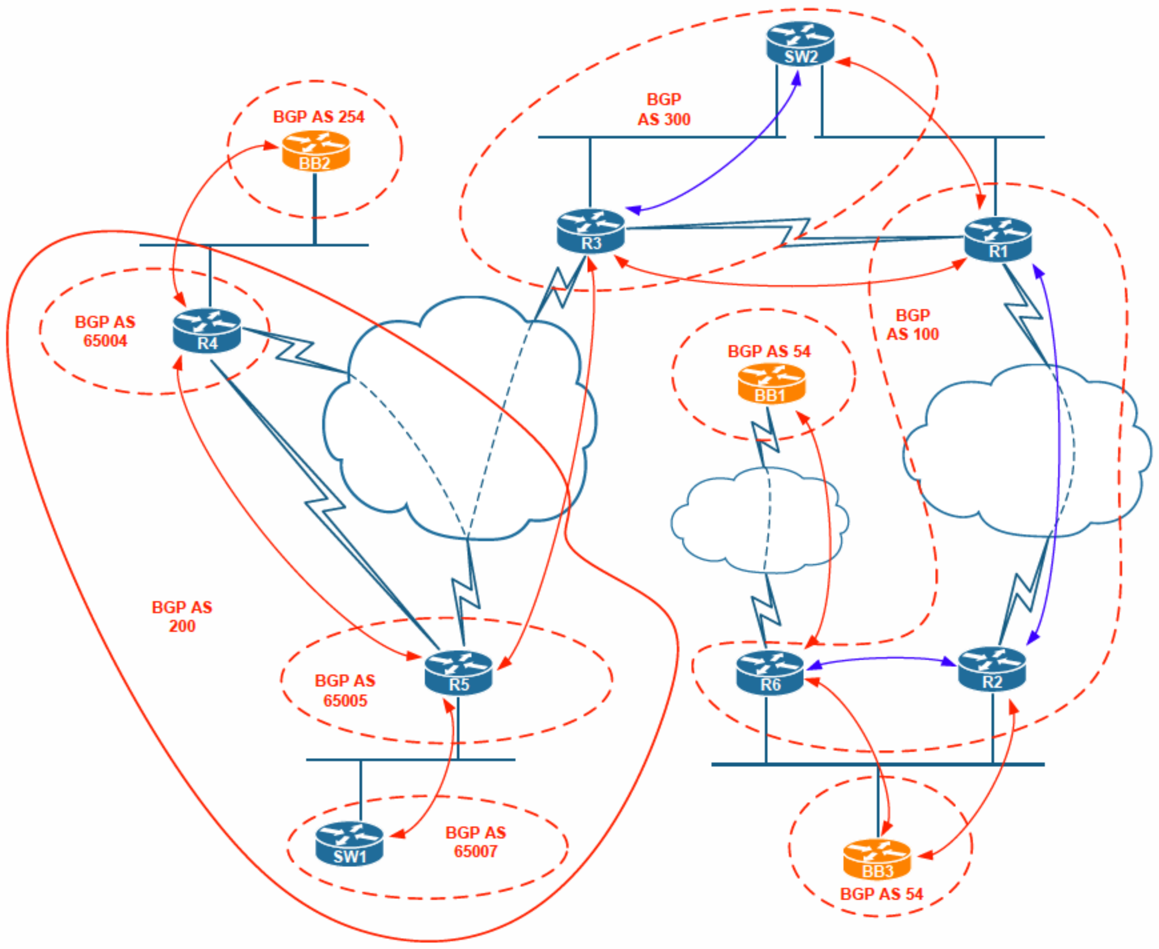


Fig 2

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 is there 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 is worth 4 points, which probably means it is really 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 0/0/0
 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 or 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 on 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 it seems to be 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
```

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
Fa0/6               Desg FWD 19        128.8    P2p
Po13                Desg 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 when 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 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 on 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 0/0/0.54
 no frame-relay interface 405
 frame-relay interface-dlci 504
  class DLCI_504

R4:
interface Serial 0/0/0
 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.

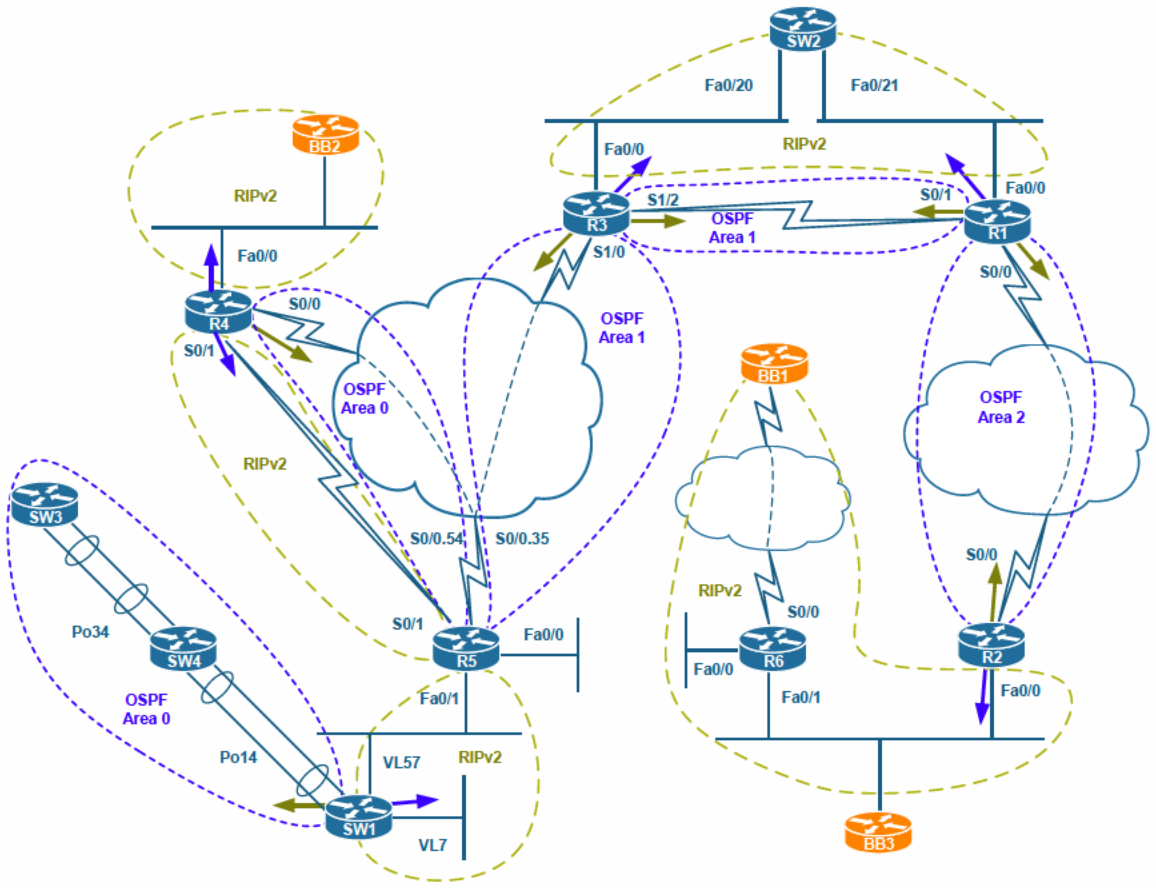


Fig 3

Apparently, the topology has routing domains with multiple redistribution points, so this opens the 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.

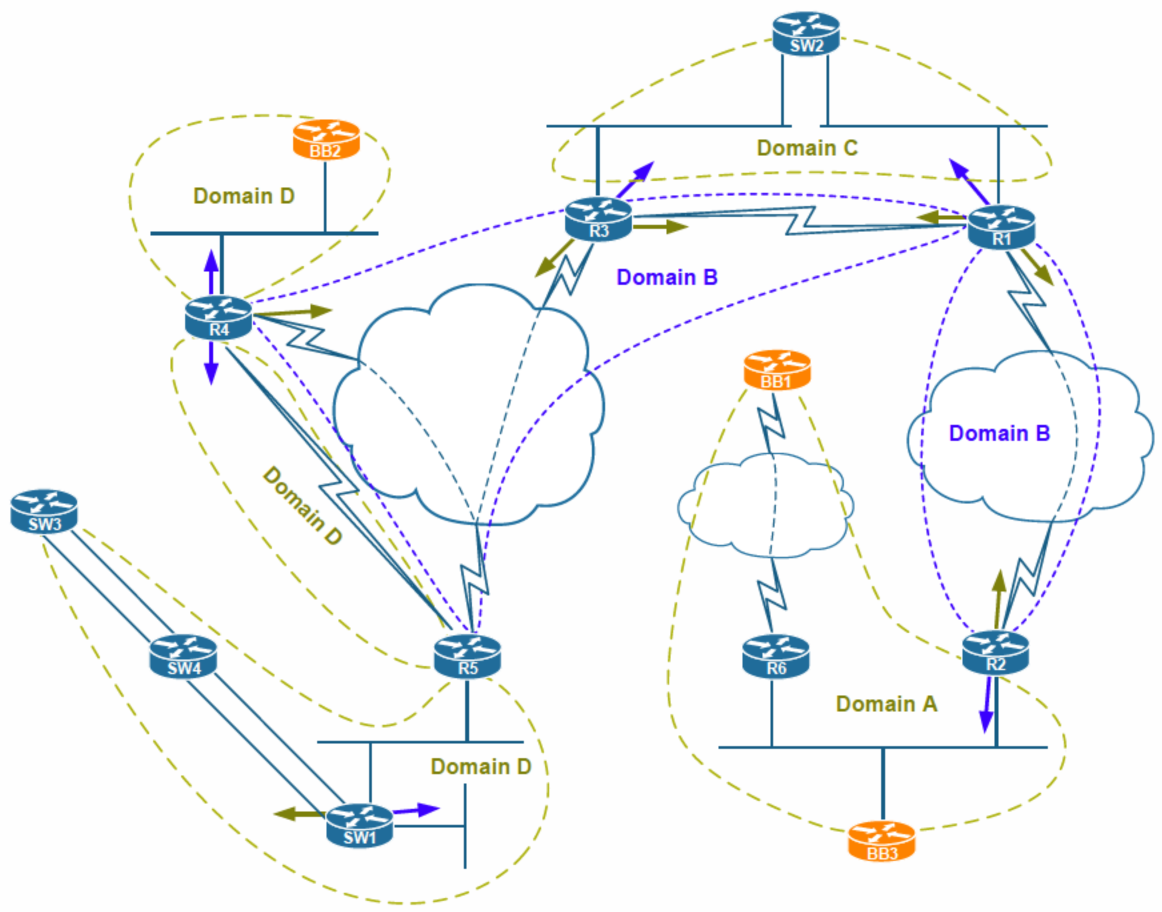


Fig 4

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 routes 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 to configure the 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 them 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 its 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 AD of 121.

```
R1, R3:
router ospf 1
 distance 121
```

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 still 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 121

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 121

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:

```
Rack1SW3#show interfaces fastEthernet 0/15
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 in turn, we are going to check the physical link between SW1 and SW2 and the link connecting SW2’s Fa 0/19 (another server port) to SW4. There are three links grouped in a port-channel connecting SW1 and SW2. 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

<snip>

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)
```

Summarize our findings: the only issue that looks related to physical/L2 problems is due to Private VLANs misconfiguration. We may now continue with troubleshooting the private VLANs. When working with private VLANs, you may want to follow the checklist below:

- Make sure primary and secondary VLANs have been created and their types have been assigned properly.
- Verify associations between the primary and secondary VLANs.
- Verify host and promiscuous ports configuration.

The first thing we need to do is check 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” or rather seems to be working. 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 should 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
```

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 no prefixes from AS54 on R1. Now let’s get back to R1 and see if these prefixes are actually advertised 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 neither 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
```

Next, we’re going to R3 and checking 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;
```

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

Total number of prefixes 2
```

If we stop and think about it, it becomes 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 163.1.12.1 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 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.
```

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.

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 of 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 long. This means there is an authentication key mismatch. We need to decipher R5’s key and apply it to R3. You may try applying the encrypted key directly, but due to a bug in older IOS versions it will always be interpreted as plain-txt.

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 now 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:

- Using an access-group applied to an interface.
- Using a policy-map that routes fragmented traffic to Null0.
- Using a service-policy that drops fragments.
- Configuring IP virtual-reassembly on interface.
- Using IOS IPS to drop fragmented traffic.
- 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 heart 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 start looking for potential filtering configuration. As was 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.

Lab 4 Solutions

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 find any L2 mis-configurations. Notice that we only put the routers that have Ethernet connections on the initial diagram.

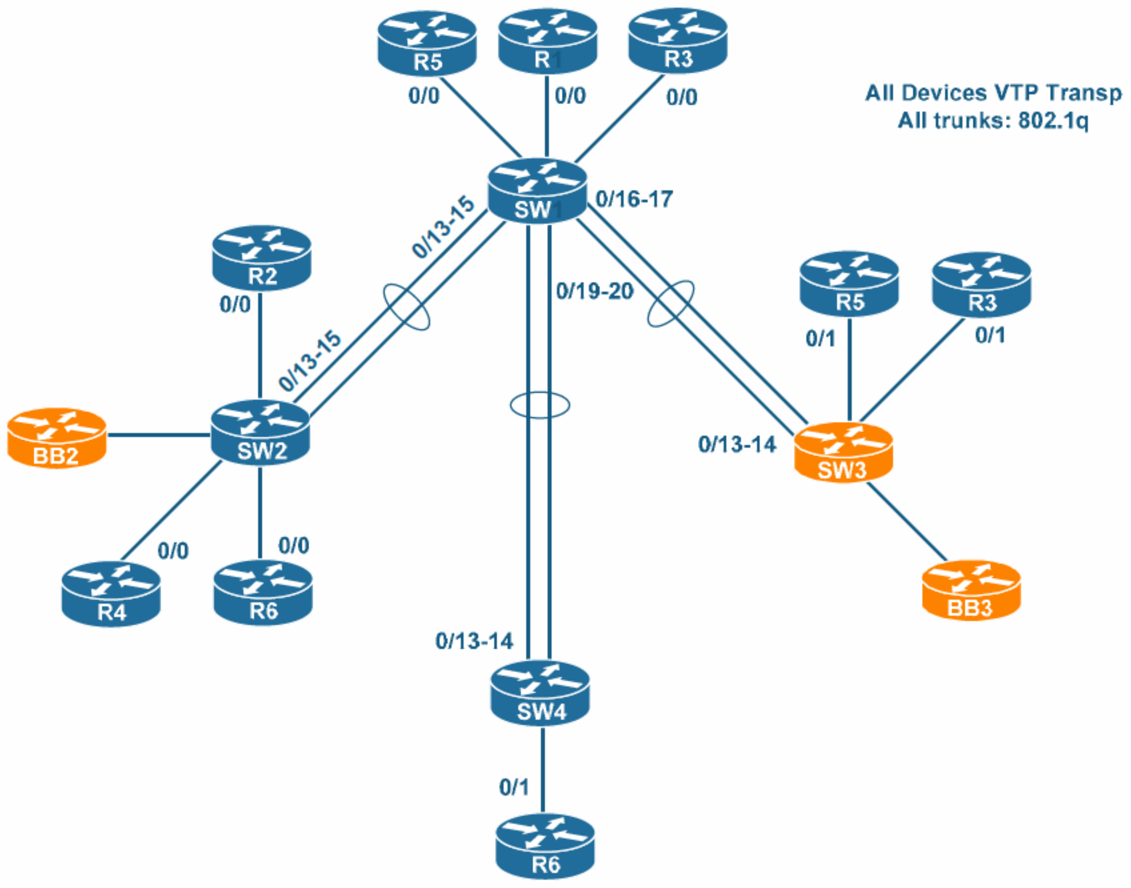


Fig 1

Based on the baseline information we build the above diagram, outlining free port-channels connecting the switches in star topology. Notice that the topology is loopless and thus there should be no STP blocked ports.

BGP Diagram

Based on the information provided with the scenario we can build the BGP peering diagram.

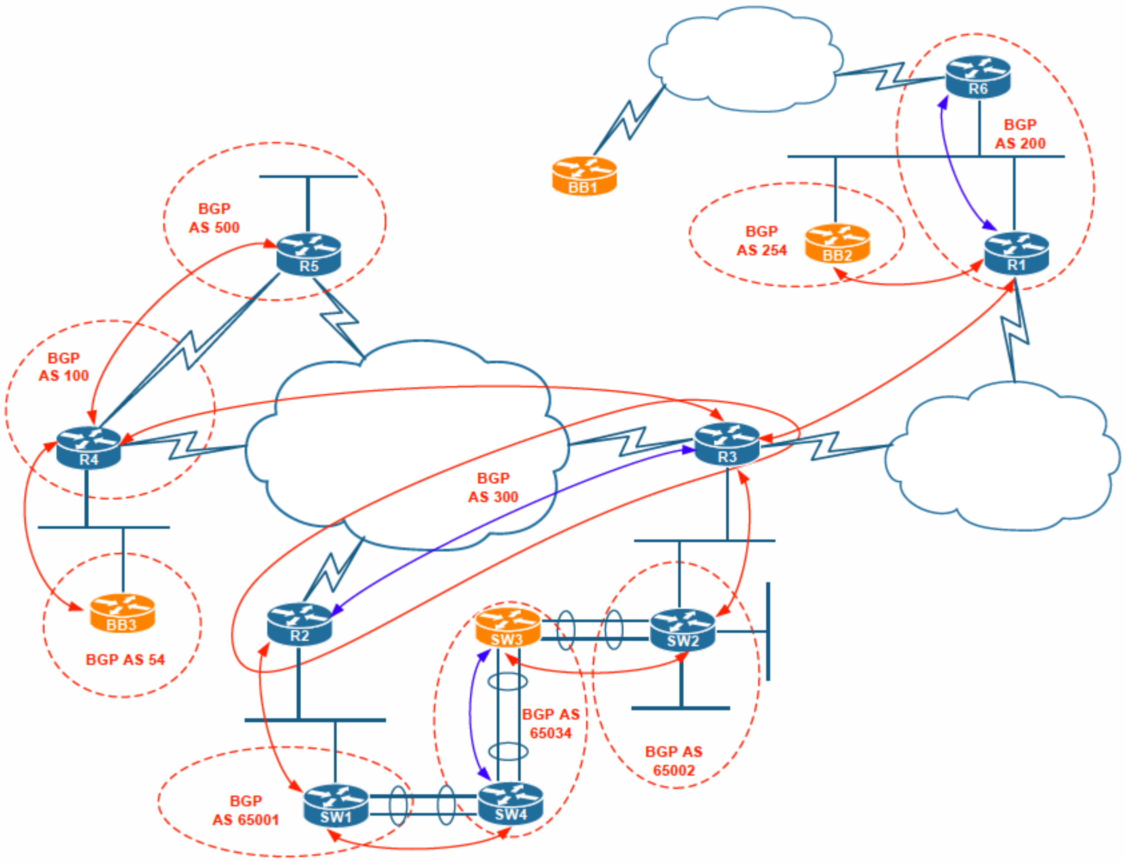


Fig 2

The scenario also mentions two routers – SW3 and SW4 advertising their loopbacks into BGP.

Multicast and Redistribution

There are no multicast scenarios in this lab and no multicast mentioned in the baseline so we skip this part. Also, there appear to be no routing loops in the topology and therefore we don’t have to worry about redistribution.

IPv6 Diagram

We are told to do IPv6 configuration discovery by hand. The first thing is learning IPv6 addressing, which is most effectively done using the command **`show ipv6 interface brief`**.

```
Rack1R1#show ipv6 interface brief
FastEthernet0/0            [up/up]
    FE80::213:80FF:FEAD:7A80
    2001:CC1E:1:1::1
Serial0/0                  [up/up]
    FE80::1
    2001:CC1E:1::1
Serial0/1                  [administratively down/down]
Loopback0                  [up/up]

Rack1R2#show ipv6 interface brief
FastEthernet0/0            [up/up]
    FE80::211:20FF:FEFD:6E00
    2001:CC1E:1:2::2
Serial0/0                  [up/up]
Serial0/0.1                [up/up]
    FE80::2
    FEC0:234::2
Serial0/1                  [administratively down/down]
Loopback0                  [up/up]
Rack1R2#

Rack1R3#show ipv6 interface brief
FastEthernet0/0            [up/up]
FastEthernet0/1            [up/up]
    FE80::211:20FF:FE93:E561
    2001:CC1E:1:3::3
Serial1/0                  [up/up]
    FE80::3
    FEC0:234::3
Serial1/1                  [up/up]
    FE80::3
    2001:CC1E:1::3
Serial1/2                  [administratively down/down]
Serial1/3                  [administratively down/down]
Loopback0                  [up/up]
Rack1R3#
```

```
Rack1R4#show ipv6 interface brief
FastEthernet0/0            [up/up]
    FE80::21E:7AFF:FE9E:417C
    2001:204:12:1::100
FastEthernet0/1            [administratively down/down]
    unassigned
Serial0/0/0                [up/up]
    FE80::4
    FEC0:234::4
Serial0/1/0                [up/up]
    unassigned
Loopback0                  [up/up]
    unassigned
Rack1R4#

Rack1R5#show ipv6 interface brief
FastEthernet0/0            [up/up]
    unassigned
FastEthernet0/1            [up/up]
    unassigned
Serial0/0/0                [up/up]
    unassigned
Serial0/1/0                [up/up]
    unassigned
Loopback0                  [up/up]
    unassigned
Loopback2                  [up/up]
    Unassigned

Rack1R6#show ipv6 interface brief
FastEthernet0/0            [up/up]
    unassigned
FastEthernet0/1            [administratively down/down]
    unassigned
Serial0/0/0                [up/up]
    unassigned
Serial0/0/0.1              [up/up]
    unassigned
SSLVPN-VIF0                [up/up]
    unassigned
Loopback0                  [up/up]
    Unassigned
```

We omitted the output from the switches due to its length. From the above information you may find that IPv6 is enabled on R1, R2, R3 and R4 and map the IPv6 addresses to the links. Following this, you may want to see what protocols are used to route IPv6.

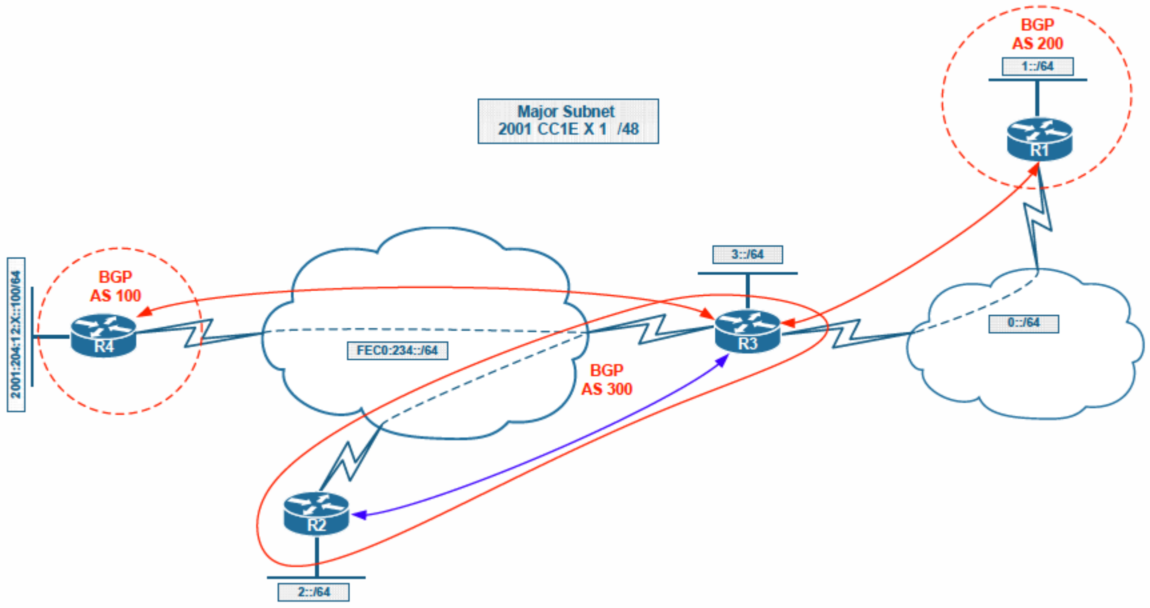
```
Rack1R1#show ipv6 protocols
IPv6 Routing Protocol is "connected"
IPv6 Routing Protocol is "static"
IPv6 Routing Protocol is "bgp 200"
  IGP synchronization is disabled
  Redistribution:
    None
  Neighbor(s):
    Address                    FiltIn FiltOut Weight RoutemapIn
RoutemapOut
    2001:CC1E:1::3
IPv6 Routing Protocol is "bgp multicast"
  IGP synchronization is disabled
  Redistribution:
    None
  Neighbor(s):
    Address                    FiltIn FiltOut Weight RoutemapIn
RoutemapOut
    2001:CC1E:1::3

Rack1R2#show ipv6 protocols
IPv6 Routing Protocol is "connected"
IPv6 Routing Protocol is "static"
IPv6 Routing Protocol is "bgp 300"
  IGP synchronization is disabled
  Redistribution:
    None
  Neighbor(s):
    Address                    FiltIn FiltOut Weight RoutemapIn
RoutemapOut
    FEC0:234::3
IPv6 Routing Protocol is "bgp multicast"
  IGP synchronization is disabled
  Redistribution:
    None
  Neighbor(s):
    Address                    FiltIn FiltOut Weight RoutemapIn
RoutemapOut
    FEC0:234::3
```

```
Rack1R3#show ipv6 protocols
IPv6 Routing Protocol is "connected"
IPv6 Routing Protocol is "static"
IPv6 Routing Protocol is "bgp 300"
  IGP synchronization is disabled
  Redistribution:
    None
  Neighbor(s):
    Address                    FiltIn FiltOut Weight RoutemapIn
RoutemapOut
    2001:CC1E:1::1
    FEC0:234::2
    FEC0:234::4
IPv6 Routing Protocol is "bgp multicast"
  IGP synchronization is disabled
  Redistribution:
    None
  Neighbor(s):
    Address                    FiltIn FiltOut Weight RoutemapIn
RoutemapOut
    2001:CC1E:1::1
    FEC0:234::2
    FEC0:234::4

Rack1R4#show ipv6 protocols
IPv6 Routing Protocol is "connected"
IPv6 Routing Protocol is "bgp 100"
  IGP synchronization is disabled
  Redistribution:
    None
  Neighbor(s):
    Address                    FiltIn FiltOut Weight RoutemapIn
RoutemapOut
    FEC0:234::3
IPv6 Routing Protocol is "bgp multicast"
  IGP synchronization is disabled
  Redistribution:
    None
```

From the above commands you may notice that the only protocol used to route IPv6 is BGP. You can quickly recover the BGP AS boundaries from the BGP diagram and the BGP peering sessions from the above output. The final IPv6 diagram would look like this:



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 is worth 4 points, which probably means it 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

Start with divide-and-conquer approach. Trace from R3 to a network behind BB1. To do this, first find out networks that R6 learns from BB1:

```
Rack1R6#show ip route rip | inc Serial0/0/0
R    212.18.1.0/24 [120/1] via 54.1.1.254, 00:00:19, Serial0/0/0.1
R    212.18.0.0/24 [120/1] via 54.1.1.254, 00:00:19, Serial0/0/0.1
R    212.18.3.0/24 [120/1] via 54.1.1.254, 00:00:19, Serial0/0/0.1
R    212.18.2.0/24 [120/1] via 54.1.1.254, 00:00:19, Serial0/0/0.1
```

Now trace to any of these from R3:

```
Rack1R3#traceroute 212.18.1.1

Type escape sequence to abort.
Tracing the route to 212.18.1.1

  1 162.1.13.1 28 msec 32 msec 28 msec
  2  *  *  *
  3  *
```

As we can see the traceroute stops at R1. Most likely the problem is localized between R1 and R6. This involves four devices in the troubleshooting process: R1, SW1, SW2 and R6.

Initial hypothesis: Problem in the link between R1 and R6
Problem Scope: R1, SW1, SW2 and R6.

Isolate the Issue

We start isolating the issue using divide-and-conquer approach. First, we ping between R1 and R6 hoping that there are no access-lists to prevent that type of probing.

```
Rack1R1#ping 192.10.1.6

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.10.1.6, timeout is 2 seconds:
 ...
Success rate is 0 percent (0/5)
```

The ping operation fails, which makes us think of some physical issue. We continue by checking the physical interface states. Since those are Ethernet interfaces, check for errors and/or duplex mismatch issues.

```
Rack1R1#show interfaces fastEthernet 0/0
FastEthernet0/0 is up, line protocol is up
  Hardware is AmdFE, address is 0013.80ad.7a80 (bia 0013.80ad.7a80)
  Internet address is 192.10.1.1/24
  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Full-duplex, 100Mb/s, 100BaseTX/FX
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
  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 1000 bits/sec, 1 packets/sec
  5 minute output rate 1000 bits/sec, 1 packets/sec
     14615 packets input, 1628928 bytes
     Received 12720 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
     0 watchdog
     0 input packets with dribble condition detected
     7280 packets output, 1163947 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
     0 babbles, 0 late collision, 0 deferred
     3 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
```

```
Rack1R6#show interfaces fastEthernet 0/0
FastEthernet0/0 is up, line protocol is up
  Hardware is MV96340 Ethernet, address is 001a.2f78.4678 (bia
001a.2f78.4678)
  Internet address is 192.10.1.6/24
  MTU 1500 bytes, BW 100000 Kbit/sec, 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, 100BaseTX/FX
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:03, output 00:00:05, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/37/0 (size/max/drops/flushes); Total output drops:
0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     15301 packets input, 1982011 bytes
     Received 12900 broadcasts, 0 runts, 0 giants, 1 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
     0 watchdog
     0 input packets with dribble condition detected
     5440 packets output, 578554 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
     767 unknown protocol drops
     767 unknown protocol drops
     0 babbles, 0 late collision, 0 deferred
     0 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
```

Right after this we can check the “transit” path across SW1 and SW2. We can use CDP for this purpose, as see if adjacent devices see each other.

```
Rack1SW1#show cdp neighbors  | inc SW2|R1
Rack1SW2         Fas 0/14         127          R S I     WS-C3560- Fas
0/14
Rack1SW2         Fas 0/13         126          R S I     WS-C3560- Fas
0/13
Rack1SW2         Fas 0/15         124          R S I     WS-C3560- Fas
0/15
Rack1R1          Fas 0/1          131          R S I     2610XM    Fas
0/0

Rack1SW2#show cdp neighbors | include SW1|R6
Rack1SW1         Fas 0/14         131          R S I     WS-C3560- Fas
0/14
Rack1SW1         Fas 0/13         131          R S I     WS-C3560- Fas
0/13
Rack1SW1         Fas 0/15         166          R S I     WS-C3560- Fas
0/15
Rack1R6          Fas 0/6          166          R S I     2811      Fas
0/0
```

So it appears there are no physical or layer 2 problems on the path between R1 and R6. We may confirm that by checking spanning-tree status and making sure the ports are not blocked:

```
Rack1SW1#show spanning-tree vlan 162

VLAN0162
  Spanning tree enabled protocol ieee
  Root ID    Priority    32930
             Address     0011.bb0a.c880
             This bridge is the root
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    32930  (priority 32768 sys-id-ext 162)
             Address     0011.bb0a.c880
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time 300

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- ----------------------
Fa0/1               Desg FWD 19        128.3    P2p Edge
Po12                Desg FWD 12        128.144  P2p
Po13                Desg FWD 12        128.152  P2p
Po14                Desg FWD 12        128.160  P2p

Rack1SW1#

Rack1SW2#show spanning-tree vlan 162

VLAN0162
  Spanning tree enabled protocol ieee
  Root ID    Priority    32930
             Address     0011.bb0a.c880
             Cost        12
             Port        144 (Port-channel12)
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    32930  (priority 32768 sys-id-ext 162)
             Address     0018.738d.9e00
             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        128.8    P2p
Fa0/24              Desg FWD 19        128.26   P2p
Po12                Root FWD 12        128.144  P2p
```

This shows that the logical topology is in order. Thus, our next hypothesis is that something may be filtering traffic between R1 and R6. There are two major options: filtering in switches and filtering in routers. We may want to isolate the device where filtering is being performed. We may assign IP addresses to SW1 and SW2 and see if we can ping between R1/SW1, SW1/SW2 and then SW2/R6.

```
SW1:
interface Vlan 162
 ip address 192.10.1.7 255.255.255.0

SW2:
interface Vlan 162
 ip address 192.10.1.8 255.255.255.0
```

Now we ping from R6 to SW2 and then SW1 (based on the physical layout):

```
Rack1R6#ping 192.10.1.8

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.10.1.8, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 1/2/4 ms

Rack1R6#ping 192.10.1.7

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.10.1.7, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 1/1/4 ms
```

OK, so the problem does not show up on the segment R6-SW2-SW1. What about SW1-R1?

```
Rack1R1#ping 192.10.1.7

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.10.1.7, timeout is 2 seconds:
 ...
Success rate is 0 percent (0/5)
```

So we have isolated our issue down to SW1 and R1 and the link between those two. Plus we know it's not a layer 2 / physical issue. We may only guess that this could be some sort of configuration issue, such as filtering. So we need to perform a routine check of R1 and SW1 for any traces of filtering. We know filtering could be performed by:

1) Access-lists
2) Policy routing
3) Traffic Policing
4) uRPF
5) VLAN Filters
6) Flexible Packet Matching

In order to narrow down this list, we may want to investigate the direction of the filter. For that purpose, we enable ICMP packet debugging in SW1 and R1 and ping from R1 and SW1 respectively, seeing if any of them will receive the ICMP echoes. First, we ping SW1 from R1:

```
Rack1SW1(config)#logging buffered debugging
Rack1SW1#debug ip icmp
ICMP packet debugging is on
Rack1SW1#
RSRack99AS>1
[Resuming connection 1 to R1 ... ]

Rack1R1#ping 192.10.1.7

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.10.1.7, timeout is 2 seconds:

Rack1SW1#
```

No logging messages output means pings are being filtered inbound by SW1 or outbound by R1. If R1 performs outbound filtering, this is performed by either QoS settings or local policy routing. If SW1 filters packets, it does that either via inbound access-group, interface policing settings or VLAN filters. Let’s see if packets from SW1 get to R1 to finally identify the filter direction.

```
Rack1R1#debug ip icmp
ICMP packet debugging is on
Rack1R1#
..
Rack1SW1#ping 192.10.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.10.1.1, timeout is 2 seconds:

RSRack99AS>1
[Resuming connection 1 to R1 ... ]

Rack1R1#
ICMP: echo reply sent, src 192.10.1.1, dst 192.10.1.7
Rack1R1#
ICMP: echo reply sent, src 192.10.1.1, dst 192.10.1.7
Rack1R1#
ICMP: echo reply sent, src 192.10.1.1, dst 192.10.1.7
Rack1R1#
ICMP: echo reply sent, src 192.10.1.1, dst 192.10.1.7
Rack1R1#
ICMP: echo reply sent, src 192.10.1.1, dst 192.10.1.7
```

This gives us an idea that filtering is most likely applied to the packets from R1 to SW1. We may now proceed with checking the filtering settings in R1: local PBR and QoS policies:

```
Rack1R1#show ip local policy
Local policy routing is disabled

Rack1R1#show policy-map interface fastEthernet 0/0
Rack1R1#
```

R1 looks clean so we move on to SW1 and keep working there. We need to check for access-list applied to the port connected to R1, VLAN filters and QoS configurations.

```
Rack1SW1#show ip interface fastEthernet 0/1
FastEthernet0/1 is up, line protocol is up
  Inbound  access list is not set

So no access-lists set, now checking VLAN filter:

Rack1SW1#show vlan filter
VLAN Map ICMP_FILTER is filtering VLANs:
  162
```

Oh this looks suspicious; let’s see how it’s configured:

```
Rack1SW1#show vlan access-map
Vlan access-map "ICMP_FILTER"  10
  Match clauses:
    ip  address: 100
  Action:
    drop
Vlan access-map "ICMP_FILTER"  20
  Match clauses:
  Action:
    Forward

Check the access-list:

Rack1SW1#show ip access-list 100
Extended IP access list 100
    10 permit icmp 205.90.31.0 0.0.0.255 any echo
```

Nothing about our network, which means the filter is innocent. Now we're going to check QoS settings. First let's see if MLS QoS is enabled at all:

```
Rack1SW1#show mls qos
QoS is disabled
QoS ip packet dscp rewrite is enabled

Rack1SW1#
```

So the Catalyst QoS features could not affect the traffic. What else could filter traffic? The setting similar to policing is storm-control. Let's see if this blocks our packets in SW1:

```
Rack1SW1#show storm-control multicast
Interface  Filter State   Upper        Lower        Current
---------  -------------  -----------  -----------  ----------

Rack1SW1#show storm-control broadcast
Interface  Filter State   Upper        Lower        Current
---------  -------------  -----------  -----------  ----------

Rack1SW1#show storm-control unicast
Interface  Filter State   Upper        Lower        Current
---------  -------------  -----------  -----------  ----------
Fa0/1      Blocking         0.00%        0.00%        0.00%
```

What's interesting, is that if you execute just show storm-control you won't see anything. For unicast filtering, you need the specific command with the unicast keyword! And we found our issue – SW1 blocks packets

Conclusion: Prior to investigating packet filtering configurations it is useful to narrow down the problem scope as much as possible and determine the direction of the packet filter.

Fix the issue

The solution was to remove the improperly configured unicast storm-control settings on SW1's connection to R1.

```
SW1:
interface fastEthernet 0/1
no storm-control unicast level 0
```

Verify

Make sure you may reach the routes learned from BB1 now:

```
Rack1R3#traceroute 212.18.1.1

Type escape sequence to abort.
Tracing the route to 212.18.1.1

  1 162.1.13.1 32 msec 32 msec 28 msec
  2 192.10.1.6 32 msec 28 msec 32 msec
  3 54.1.1.254 44 msec *  40 msec
```

Ticket 2

Analyze the Symptoms

The ticket is already very clear on the symptoms, so there is no nothing much to analyze here. We may proceed directly to the issue isolation step.

Isolate the Issue

From the output we collected from R5, we may conclude that R4 and R5 peer eBGP using their Loopback0 interface. Let’s see if we can ping between the loopbacks first:

```
Rack1R5#ping 150.1.4.4 source loopback 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.4.4, timeout is 2 seconds:
Packet sent with a source address of 150.1.5.5
 ...
Success rate is 0 percent (0/5)
```

The ping fails. There are two paths between R4 and R5 for redundancy. If both paths fail, this means that we either have no routing or the links are malfunctioning. To isolate the routing issue, let’s see if we can ping using the directly connected interfaces:

```
Rack1R5#ping 162.1.0.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 162.1.0.4, timeout is 2 seconds:
 ...
Success rate is 0 percent (0/5)

Rack1R5#ping 162.1.45.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 162.1.45.4, timeout is 2 seconds:
U.U.U
Success rate is 0 percent (0/5)
```

Both direct connections fail. The second connection shows us ICMP unreachables, which means the network is not even in the routing table and the interface is probably down. Let’s deal with each of the link failures separately, starting with the Frame-Relay link.

Using the bottom-up approach we test the physica link connectivity first:

```
Rack1R5#show interfaces serial 0/0/0
Serial0/0/0 is up, line protocol is up
  Hardware is GT96K Serial
  Internet address is 162.1.0.5/24
  MTU 1500 bytes, BW 1544 Kbit/sec, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation FRAME-RELAY, loopback not set
  Keepalive set (10 sec)
  CRC checking enabled
  LMI enq sent  8628, LMI stat recvd 8629, 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 7851/0, interface
broadcasts 5372
  Last input 00:00:00, output 00:00:01, output hang never
  Last clearing of "show interface" counters 1d01h
  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  0/2/256 (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
     22926 packets input, 1122227 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
     25561 packets output, 1266927 bytes, 0 underruns
     0 output errors, 0 collisions, 3 interface resets
     0 unknown protocol drops
     0 output buffer failures, 0 output buffers swapped out
     2 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

As usual, pay attention to LMI stats. LMI keepalives is what keeps the link up. So far everything looks good. Let’s see if we can ping another router across the Frame-Relay cloud:

```
Rack1R5#ping 162.1.0.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 162.1.0.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/59/68 ms
```

If the OSPF network-type used on the cloud would be point-to-multipoint, having connectivity to R3 would be enough for R4 to communicate with R5 across R3. However, it appears the OSPF network type is non-broadcast:

```
Rack1R5#show ip ospf interface serial 0/0/0
Serial0/0/0 is up, line protocol is up
  Internet Address 162.1.0.5/24, Area 0
  Process ID 1, Router ID 150.1.5.5, Network Type NON_BROADCAST, Cost:
64
  Transmit Delay is 1 sec, State DROTHER, Priority 0
<snip>
```

And thus R4 needs to have direct PVC to R5. From our initial tests, it looks like there is something in between R4 and R5. As we can see from the diagram, they use a separate PVC with DLCIs 405 and 504 for communicating. Let’s check the Frame-Relay mappings at both ends:

```
Rack1R5#show frame-relay map
Serial0/0/0 (up): ip 162.1.0.3 dlci 503(0x1F7,0x7C70), static,
              broadcast,
              CISCO, status defined, active
Serial0/0/0 (up): ip 162.1.0.4 dlci 504(0x1F8,0x7C80), static,
              broadcast,
              CISCO, status defined, active
Serial0/0/0 (up): ip 162.1.0.2 dlci 504(0x1F8,0x7C80), static,
              CISCO, status defined, active
Serial0/1/0 (up): ip 162.1.45.4 dlci 415(0x19F,0x64F0), static,
              CISCO, status deleted
```

There is something interesting about the DLCI 415, which shows as deleted but we’ll focus on the DLCI 504 for now. This shows up as static and active, which means the DLCI is being learned from the switch. The IP address mapped to this DLCI is 162.1.0.4 which is also correct. So let’s switch back to R4 and check the mappings there:

```
Rack1R4#show frame-relay map
Serial0/0/0 (up): ipv6 FEC0:234::3 dlci 403(0x193,0x6430), static,
              broadcast,
              CISCO, status defined, active
Serial0/0/0 (up): ipv6 FEC0:234::2 dlci 403(0x193,0x6430), static,
              CISCO, status defined, active
Serial0/0/0 (up): ip 162.1.0.3 dlci 403(0x193,0x6430), static,
              broadcast,
              CISCO, status defined, active
              RTP Header Compression (enabled), connections: 15
Serial0/0/0 (up): ip 162.1.0.2 dlci 403(0x193,0x6430), static,
              CISCO, status defined, active
Serial0/0/0 (up): ipv6 FE80::3 dlci 403(0x193,0x6430), static,
              CISCO, status defined, active
Serial0/0/0 (up): ipv6 FE80::2 dlci 403(0x193,0x6430), static,
              CISCO, status defined, active
Serial0/0/0 (up): ip 162.1.0.5 dlci 504(0x1F8,0x7C80), static,
              broadcast,
              CISCO, status deleted
Serial0/1/0 (up): ip 162.1.45.5 dlci 514(0x202,0x8020), static,
              CISCO, status deleted
```

We look for the IP address of R5 and find that it's mapped to DLCI 504 which appears to be deleted. As a matter of fact, the IP address of R5 should be mapped to DLCI 405 per the diagram. Let's fix that:

R4:
```
interface Serial 0/0/0
 no frame-relay map ip 162.1.0.5 504
 frame-relay map ip 162.1.0.5 405 broadcast
```

And see if we can ping between R4 and R5 now, using the Frame-Relay interface addresses:

```
Rack1R4#ping 162.1.0.5

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 162.1.0.5, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/56/60 ms
```

And the BGP session is up:

```
Rack1R5#show ip bgp summary
BGP router identifier 150.1.5.5, local AS number 500
BGP table version is 59, main routing table version 59
18 network entries using 2376 bytes of memory
18 path entries using 936 bytes of memory
6/5 BGP path/bestpath attribute entries using 1008 bytes of memory
4 BGP AS-PATH entries using 128 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
Bitfield cache entries: current 1 (at peak 2) using 32 bytes of memory
BGP using 4480 total bytes of memory
BGP activity 38/20 prefixes, 38/20 paths, scan interval 60 secs

Neighbor        V          AS MsgRcvd MsgSent   TblVer  InQ OutQ
Up/Down  State/PfxRcd
150.1.4.4       4        100    1403    1394       59    0    0
00:06:58       17
```

But we still have the problem with the Frame-Relay link between R4 and R5. Do we really need to fix it? Apparently yes, as the baseline says there is a need for BGP peering session redundancy. OK, so let’s check what might be keeping the Serial link between R4 and R5 down:

```
Rack1R5#show interfaces serial 0/1/0
Serial0/1/0 is up, line protocol is down
  Hardware is GT96K Serial
  Internet address is 162.1.45.5/24
  MTU 1500 bytes, BW 1544 Kbit/sec, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation FRAME-RELAY, loopback not set
  Keepalive set (10 sec)
  CRC checking enabled
  LMI enq sent  733, LMI stat recvd 0, LMI upd recvd 0, DTE LMI down
  LMI enq recvd 975, 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>
```

First of all, we see that this is a Frame-Relay link. Next, we see that LMI status is down. We see that LMI enquiries are being sent and received, but no LMI status reports are being sent or received. This is what actually might be keeping the connection down, as the physical layer seems to be working – interface shows as up/down. The fact that we receive LMI enquiries means that the other side is configured for Frame-Relay as well. We may check it just to make sure:

```
Rack1R4#show interfaces serial 0/1/0
Serial0/1/0 is up, line protocol is down (looped)
  Hardware is GT96K Serial
  Internet address is 162.1.45.4/24
  MTU 1500 bytes, BW 1544 Kbit/sec, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation FRAME-RELAY, loopback not set
  Keepalive set (10 sec)
  CRC checking enabled
  LMI enq sent  764, LMI stat recvd 0, LMI upd recvd 0, DTE LMI down
  LMI enq recvd 1016, 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 0/0, interface
broadcasts 0
  Last input 00:00:03, output 00:00:03, output hang never
  Last clearing of "show interface" counters 02:07:23
  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
     1018 packets input, 13383 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
     1018 packets output, 13361 bytes, 0 underruns
     0 output errors, 0 collisions, 255 interface resets
     0 unknown protocol drops
     0 output buffer failures, 0 output buffers swapped out
     512 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

The same picture on the other side. By the way, we notice that both interfaces are configured as Frame-Relay DTEs, which surely would not work. We need to either configure one side as Frame-Relay DCE or disable LMI messages. In addition to that we need to check the Frame-Relay mappings to make sure they match locally configured DLCI numbers. Let’s start by checking if the DLCI numbers are matching:

```
Rack1R4#show frame-relay map
Serial0/0/0 (up): ipv6 FEC0:234::3 dlci 403(0x193,0x6430), static,
              broadcast,
              CISCO, status defined, active
Serial0/0/0 (up): ipv6 FEC0:234::2 dlci 403(0x193,0x6430), static,
              CISCO, status defined, active
Serial0/0/0 (up): ip 162.1.0.3 dlci 403(0x193,0x6430), static,
              broadcast,
              CISCO, status defined, active
              RTP Header Compression (enabled), connections: 15
Serial0/0/0 (up): ip 162.1.0.2 dlci 403(0x193,0x6430), static,
              CISCO, status defined, active
Serial0/0/0 (up): ipv6 FE80::3 dlci 403(0x193,0x6430), static,
              CISCO, status defined, active
Serial0/0/0 (up): ipv6 FE80::2 dlci 403(0x193,0x6430), static,
              CISCO, status defined, active
Serial0/0/0 (up): ip 162.1.0.5 dlci 405(0x195,0x6450), static,
              broadcast,
              CISCO, status defined, active
Serial0/1/0 (up): ip 162.1.45.5 dlci 514(0x202,0x8020), static,
              CISCO, status deleted

Rack1R4#show frame-relay pvc | inc ST
DLCI = 401, DLCI USAGE = UNUSED, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0/0
DLCI = 402, DLCI USAGE = UNUSED, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0/0
DLCI = 403, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0/0
DLCI = 405, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0/0
DLCI = 413, DLCI USAGE = UNUSED, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0/0
DLCI = 415, DLCI USAGE = LOCAL, PVC STATUS = DELETED, INTERFACE =
Serial0/1/0
DLCI = 514, DLCI USAGE = LOCAL, PVC STATUS = DELETED, INTERFACE =
Serial0/1/0
```

OK, we can see that R4 maps the IP address of R5 to DLCI 514, and the locally configured DLCI is 415. Both DLCIs shows up as DELETED because they don't appear in LMI status reports. Let's see if R5's configuration matches this:

```
Rack1R5#show frame-relay map
Serial0/0/0 (up): ip 162.1.0.3 dlci 503(0x1F7,0x7C70), static,
              broadcast,
              CISCO, status defined, active
Serial0/0/0 (up): ip 162.1.0.4 dlci 504(0x1F8,0x7C80), static,
              broadcast,
              CISCO, status defined, active
Serial0/0/0 (up): ip 162.1.0.2 dlci 504(0x1F8,0x7C80), static,
              CISCO, status defined, active
Serial0/1/0 (up): ip 162.1.45.4 dlci 415(0x19F,0x64F0), static,
              CISCO, status deleted
```

```
Rack1R5#show frame-relay pvc | inc ST
DLCI = 501, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0/0
DLCI = 502, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0/0
DLCI = 503, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0/0
DLCI = 504, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0/0
DLCI = 513, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0/0
DLCI = 415, DLCI USAGE = LOCAL, PVC STATUS = DELETED, INTERFACE =
Serial0/1/0
DLCI = 514, DLCI USAGE = LOCAL, PVC STATUS = DELETED, INTERFACE =
Serial0/1/0
```

The mappings seem to be symmetric, mirroring each other's DLCIs. So the only thing we need to do now is to bring the protocol up. We'll do that by disabling LMI:

```
R4:
interface Serial 0/1/0
 no keepalive

R5:
interface Serial 0/1/0
 no keepalive
```

Check if the issue has been fixed now:

```
Rack1R4#ping 162.1.45.5

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 162.1.45.5, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/15/16 ms
```

Double check OSPF neighbors:

```
Rack1R4#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address
Interface
150.1.5.5         1   FULL/DR         00:01:40    162.1.45.5
Serial0/1/0
150.1.3.3         1   FULL/DR         00:01:37    162.1.0.3
Serial0/0/0
```

Fix the Issue

Fixing the issue consisted of two steps: correcting the Frame-Relay map and making the Back-to-Back Frame-Relay work correctly.

```
R4:
interface Serial 0/0/0
 no frame-relay map ip 162.1.0.5 504
 frame-relay map ip 162.1.0.5 405 broadcast
!
interface Serial 0/1/0
 no keepalive

R5:
interface Serial 0/1/0
 no keepalive
```

Verify

To verify, ensure you can ping the Loopback interfaces of both routers.

Ticket 3

Analyze the Symptoms

The first thing we may want to know is where everything brakes. As usual, our primary tool is the **`traceroute`** command.

```
Rack1SW1#traceroute 162.1.3.3

Type escape sequence to abort.
Tracing the route to 162.1.3.3

  1  *  *
```

Seems like we can’t even reach the first hop, so the problem might be local to SW1. Check SW1’s route for VLAN 3:

```
Rack1SW1#show ip route 162.1.3.0
% Subnet not in table
```

Check if the OSPF adjacency is healthy:

```
Rack1SW1#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address
Interface
150.1.2.2         0   2WAY/DROTHER    00:00:35    162.1.27.2
Vlan27
```

OK, so the problem seems to be related to the OSPF adjacency between SW1 and R2. Let’s keep working with that hypothesis at the isolation stage.

Isolate the Issue

First of all, notice that the adjacency seems to be stuck in the 2WAY state as SW1. This means that both routers see each other's OSPF HELLO packets. Thus, at least physical and link layers are working correctly. So the next thing to do is check the OSPF adjacency formation. Use the following commands on R2:

```
Rack1R2#debug ip ospf packet
OSPF packet debugging is on

Rack1R2#debug ip ospf adj
OSPF adjacency events debugging is on

Rack1R2#debug interface fastEthernet 0/0
Condition 1 set

Rack1R2#debug ip ospf events
OSPF events debugging is on

OSPF: rcv. v:2 t:1 l:48 rid:162.1.7.7
      aid:0.0.0.27 chk:AD74 aut:0 auk: from FastEthernet0/0

OSPF: rcv. v:2 t:1 l:48 rid:162.1.7.7
      aid:0.0.0.27 chk:AD74 aut:0 auk: from FastEthernet0/0

Rack1R2#
OSPF: rcv. v:2 t:1 l:48 rid:162.1.7.7
      aid:0.0.0.27 chk:AD74 aut:0 auk: from FastEthernet0/0

OSPF: rcv. v:2 t:1 l:48 rid:162.1.7.7
      aid:0.0.0.27 chk:AD74 aut:0 auk: from FastEthernet0/0
```

We see R2 receiving OSPF packets but the OSPF adjacency state is not changing. Let's see OSPF statistics for R2's Ethernet interface:

```
Rack1R2#show ip ospf interface fastEthernet 0/0
FastEthernet0/0 is up, line protocol is up
  Internet Address 162.1.27.2/24, Area 27
  Process ID 1, Router ID 150.1.2.2, Network Type BROADCAST, Cost: 1
  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 10, Dead 40, Wait 40, Retransmit 5
    oob-resync timeout 40
    Hello due in 00:00:09
  Supports Link-local Signaling (LLS)
  Index 1/3, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 1
  Last flood scan time is 0 msec, maximum is 4 msec
  Neighbor Count is 1, Adjacent neighbor count is 0
  Suppress hello for 0 neighbor(s)
```

OK what we see is that OSPF process sees no DR or BDR on the network and the network type is broadcast. Per the normal rules, if the a router does not detect a DR on the segment during the waiting period it declares itself the DR. R2 didn't do that, because it's Priority is zero. Let's compare the interface settings with SW1's:

```
Rack1SW1#show ip ospf interface vlan 27
Vlan27 is up, line protocol is up
  Internet Address 162.1.27.7/24, Area 27
  Process ID 1, Router ID 162.1.7.7, Network Type BROADCAST, Cost: 1
  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 10, Dead 40, Wait 40, Retransmit 5
    oob-resync timeout 40
    Hello due in 00:00:01
  Supports Link-local Signaling (LLS)
  Cisco NSF helper support enabled
  IETF NSF helper support enabled
  Index 2/2, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 0, maximum is 2
  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)
```

And we see the same picture here! None of the routers could be elected as DR/BDR and the routers don't synchronize with each other since they thing their partner is DROTHER. So what need to do is have at least one router with non zero priority to allow it becoming a DR. Let's do that with SW1 and see if adjacency comes up:

```
SW1:
interface Vlan 27
 ip ospf priority 1
```

This seems to fix the problem, as now our adjacency is healthy:

```
Rack1SW1#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address
Interface
150.1.2.2         0   FULL/DROTHER    00:00:38    162.1.27.2
Vlan27
```

Fix the Issue

The problem was that both routers had an OSPF priority of zero, which prevented DR election and synchronization on the broadcast segment. We fixed the problem by changing one router's priority to non-zero value. Can you think of another way except for changing the priority on any of the routers?

```
SW1:
interface Vlan 27
 ip ospf priority 1
```

Verify

Make sure SW1 can reach VLAN3 subnet now:

```
Rack1SW1#traceroute 162.1.3.3

Type escape sequence to abort.
Tracing the route to 162.1.3.3

  1 162.1.27.2 0 msec 8 msec 0 msec
  2 162.1.0.3 34 msec *  25 msec
```

Ticket 4

Analyze the Symptoms

One of the problems that we know about is that R3 does not see R1 as a BGP neighbor. We found that out when discovering IPv6 routing protocols during the initial analysis phase. We didn't track down the issue at that time, so this is our task now. Let's go ahead and isolate the issue that prevents IPv6 BGP peering session from working properly.

Isolate the Issue

Let's look at the summary of IPv6 BGP peering sessions at R3 and R1:

```
Rack1R1#show bgp ipv6 unicast summary
BGP router identifier 150.1.1.1, local AS number 200
BGP table version is 2, main routing table version 2
1 network entries using 149 bytes of memory
1 path entries using 76 bytes of memory
9/1 BGP path/bestpath attribute entries using 1116 bytes of memory
7 BGP AS-PATH entries using 168 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 1509 total bytes of memory
BGP activity 32/14 prefixes, 61/29 paths, scan interval 60 secs

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
2001:CC1E:1::3  4   300      39      38        0    0    0 00:15:23
(NoNeg)

Rack1R3#show bgp ipv6 unicast summary
BGP router identifier 150.1.3.3, local AS number 300
BGP table version is 6, main routing table version 6
5 network entries using 745 bytes of memory
5 path entries using 380 bytes of memory
12/3 BGP path/bestpath attribute entries using 1488 bytes of memory
9 BGP AS-PATH entries using 216 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 2829 total bytes of memory
BGP activity 22/0 prefixes, 34/3 paths, scan interval 60 secs

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
FEC0:234::2     4   300      24      28        6    0    0 00:15:52
1
FEC0:234::4     4   100      21      28        6    0    0 00:15:47
1
```

What this means is that there is a TCP session between R1 and R3, but for some reason IPv6 extension is not active in R3 for the session. You may check ALL address families enabled for this session in R3 using the following command:

```
Rack1R3#show bgp all neighbors 2001:CC1E:1::1
For address family: IPv4 Unicast
BGP neighbor is 2001:CC1E:1::1,  remote AS 200, external link
  BGP version 4, remote router ID 150.1.1.1
  BGP state = Established, up for 00:16:46
  Last read 00:00:46, last write 00:00:46, 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
    Address family IPv6 Unicast: received
  Message statistics:
    InQ depth is 0
    OutQ depth is 0
                         Sent       Rcvd
    Opens:                  1          1
    Notifications:          0          0
    Updates:                7          4
    Keepalives:            18         18
    Route Refresh:          0          0
    Total:                 26         23
  Default minimum time between advertisement runs is 30 seconds

<snip>

For address family: IPv6 Unicast

For address family: VPNv4 Unicast

For address family: IPv4 Multicast

For address family: IPv6 Multicast

For address family: NSAP Unicast
```

As you can see, the IPv6 address family is not active, and only the TCP transport settings are enabled for this session. Basically this means that the peer is not activated under IPv6 address family. Let’s go ahead and fix that:

R3:
```
router bgp 300
 address-family ipv6 unicast
  neighbor 2001:CC1E:1::1 activate
```

But then again, it’s still not working!

```
Rack1R2#traceroute 2001:CC1E:1:1::1

Type escape sequence to abort.
Tracing the route to 2001:CC1E:1:1::1

  1  *  *  *
  2  *  *
Rack1R2#
```

And it looks like R2 cannot see the prefix still. Let’s check R2’s BGP table first:

```
Rack1R2#show bgp ipv6 unicast
BGP table version is 21, 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
*>i2001:204:12:1::/64
                    FEC0:234::4              0    100      0 100 i
* i2001:CC1E:1:1::/64
                    2001:CC1E:1::1           0    100      0 200 i
*> 2001:CC1E:1:2::/64
                    ::                       0         32768 i
*>i2001:CC1E:1:3::/64
                    FEC0:234::3              0    100      0 i
*>iFEC0:234::/64    FEC0:234::3              0    100      0 i
```

We see that R1’s prefix is in the BGP table but is not marked as best, which prevents it from being installed into the routing table. Let’s check the detailed information for the prefix:

```
Rack1R2#show bgp ipv6 unicast 2001:CC1E:1:1::/64
BGP routing table entry for 2001:CC1E:1:1::/64, version 21
Paths: (1 available, no best path)
  Not advertised to any peer
  200
    2001:CC1E:1::1 (inaccessible) from FEC0:234::3 (150.1.3.3)
      Origin IGP, metric 0, localpref 100, valid, internal
```

We can see that the next-hop IPv6 address is not reachable! This is probably because R3 does not change the next-hop IPv6 address for eBGP routes when advertising them via iBGP. We’re going to go to R3 and fix that problem. There are two ways: either advertise the link subnet between R1 and R3 into BGP or change the next-hop to self on R3. We’ll use the second way:

R3:
```
router bgp 300
 address-family ipv6 unicast
  neighbor FEC0:234::2 next-hop-self
```

Fix the Issue

Fixing the issue consists of two steps: making sure IPv6 address family is activated for R3's BGP peering session with R1 and changing the BGP next-hop IPV6 address for routes learned by R3 from R1.

```
R3:
router bgp 300
address-family ipv6 unicast
  neighbor 2001:CC1E:1::1 activate
  neighbor FEC0:234::2 next-hop-self
```

Verify

Check the route trace from R2 to R1:

```
Rack1R2#traceroute 2001:CC1E:1:1::1

Type escape sequence to abort.
Tracing the route to 2001:CC1E:1:1::1

  1 2001:CC1E:1:3::3 44 msec 45 msec 44 msec
  2 2001:CC1E:1:1::1 88 msec 92 msec 88 msec
```

Additionally, make sure R1 can actually ping R2's Ethernet segment off its Ethernet:

```
Rack1R1#ping 2001:CC1E:1:2::2 source fastEthernet 0/0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:CC1E:1:2::2, timeout is 2
seconds:
Packet sent with a source address of 2001:CC1E:1:1::1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 116/116/117
ms
```

Ticket 5

Analyze the Symptoms

First, per the regular procedure, we need to find out where the connectivity breaks, using the traceroute command. We start with R5:

```
Rack1R5#traceroute 162.1.77.7

Type escape sequence to abort.
Tracing the route to 162.1.77.7

  1 162.1.0.3 28 msec 28 msec 28 msec
  2 162.1.0.3 !H  *  !H
```

OK, there is no need to check R3, we can see that it has no route for this host. Why does R5 route via R3?

```
Rack1R5#show ip route 162.1.77.0
% Subnet not in table

Rack1R5#show ip route
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 162.1.0.3 to network 0.0.0.0
<snip>
```

Oh, there is a default route pointing to R3. Now we go to R2 and see if this router can reach VLAN7:

```
Rack1R2#show ip route 162.1.77.0
% Subnet not in table
```

OK it doesn’t see the subnet either. But we know that the OSPF adjacency between R2 and SW1 is OK, so the problem might be at SW1, for example subnet not being advertised. Until we isolate the issue, we’ll just stick with our initial hypothesis: something misconfigured in SW1.

Isolate the Issue

We now go to SW1 and check the subnet there:

```
Rack1SW1#show ip route 162.1.77.0
Routing entry for 162.1.77.0/24
  Known via "connected", distance 0, metric 0 (connected, via
interface)
  Redistributing via ospf 1
  Advertised by ospf 1 subnets route-map CONNECTED_TO_OSPF
  Routing Descriptor Blocks:
  * directly connected, via Vlan7
      Route metric is 0, traffic share count is 1
```

Now there is some information. First we see the subnet as being connected, which means the respective interface is up. Secondly, we see the prefix being redistributed into OSPF using redistribution controlled by a route-map. Now let's see if an LSA being generated for this:

```
Rack1SW1#show ip ospf database external 162.1.77.0

            OSPF Router with ID (162.1.7.7) (Process ID 1)
```

Oops, there is no Type-5 LSA for an external prefix. Maybe the route-map is wrong?

```
Rack1SW1#show route-map CONNECTED_TO_OSPF
route-map CONNECTED_TO_OSPF, permit, sequence 10
  Match clauses:
    interface Vlan7
  Set clauses:
  Policy routing matches: 0 packets, 0 bytes

Rack1SW1#show interface Vlan7
Vlan7 is up, line protocol is up
  Hardware is EtherSVI, address is 0011.bb0a.c8c4 (bia 0011.bb0a.c8c4)
  Internet address is 162.1.77.7/24
```

Everything looks normal. Oh wait, there is another option – all areas could be stub, and thus no external LSAs will be generated. Let's check the OSPF areas on the router:

```
Rack1SW1#show ip ospf | beg Area
    Area 27
        Number of interfaces in this area is 2 (1 loopback)
        It is a NSSA area
        Area has no authentication
        SPF algorithm last executed 01:16:11.724 ago
        SPF algorithm executed 7 times
        Area ranges are
        Number of LSA 12. Checksum Sum 0x05F5E7
        Number of opaque link LSA 0. Checksum Sum 0x000000
        Number of DCbitless LSA 0
        Number of indication LSA 0
        Number of DoNotAge LSA 0
        Flood list length 0
```

So the only attached area is NSSA. And so we know what type of LSA to expect now:

```
Rack1SW1#show ip ospf database nssa-external 162.1.77.0

            OSPF Router with ID (162.1.7.7) (Process ID 1)

                Type-7 AS External Link States (Area 27)

  LS age: 391
  Options: (No TOS-capability, Type 7/5 translation, DC)
  LS Type: AS External Link
  Link State ID: 162.1.77.0 (External Network Number )
  Advertising Router: 162.1.7.7
  LS Seq Number: 80000003
  Checksum: 0xD576
  Length: 36
  Network Mask: /24
        Metric Type: 2 (Larger than any link state path)
        TOS: 0
        Metric: 20
        Forward Address: 150.1.7.7
        External Route Tag: 0
```

Let’s check if R2 has the same LSA in the database:

```
Rack1R2#show ip ospf database nssa-external 162.1.77.0

            OSPF Router with ID (150.1.2.2) (Process ID 1)

                Type-7 AS External Link States (Area 27)

  LS age: 682
  Options: (No TOS-capability, Type 7/5 translation, DC)
  LS Type: AS External Link
  Link State ID: 162.1.77.0 (External Network Number )
  Advertising Router: 162.1.7.7
  LS Seq Number: 80000003
  Checksum: 0xD576
  Length: 36
  Network Mask: /24
        Metric Type: 2 (Larger than any link state path)
        TOS: 0
        Metric: 20
        Forward Address: 150.1.7.7
        External Route Tag: 0
```

It does… but there is no corresponding route in the routing table. So what might be preventing it from being installed? The first thing you have to do with NSSA prefixes, or any external prefix, is check if the Forwarding Address (FA) is accessible. In our case this is 150.1.7.7:

```
Rack1R2#show ip route 150.1.7.7
% Subnet not in table
Rack1R2#
```

OK, so how is this address supposed to be in R2’s RIB? Most obviously OSPF process in SW1 should be advertising the prefix. Let’s go and check if this is true:

```
Rack1SW1#show ip ospf interface loopback 0
Loopback0 is up, line protocol is up
  Internet Address 150.1.7.7/24, Area 27
  Process ID 1, Router ID 162.1.7.7, Network Type LOOPBACK, Cost: 1
  Loopback interface is treated as a stub Host
```

Return to R2 and check if the corresponding LSA is in the LSDB. We need to know the neighbor’s router-ID, which could be found from inspecting the OSPF neighbors table:

```
Rack1R2#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address
Interface
150.1.3.3         1   FULL/DR         00:01:44    162.1.0.3
Serial0/0.1
162.1.7.7         1   FULL/DR         00:00:31    162.1.27.7
FastEthernet0/0
```

Now we check the links advertised by SW1:

```
Rack1R2#show ip ospf database router adv-router 162.1.7.7

            OSPF Router with ID (150.1.2.2) (Process ID 1)

                Router Link States (Area 27)

  Routing Bit Set on this LSA
  LS age: 1125
  Options: (No TOS-capability, DC)
  LS Type: Router Links
  Link State ID: 162.1.7.7
  Advertising Router: 162.1.7.7
  LS Seq Number: 80000008
  Checksum: 0xA74
  Length: 48
  AS Boundary Router
  Number of Links: 2

    Link connected to: a Transit Network
     (Link ID) Designated Router address: 162.1.27.7
     (Link Data) Router Interface address: 162.1.27.7
      Number of TOS metrics: 0
       TOS 0 Metrics: 1

    Link connected to: a Stub Network
     (Link ID) Network/subnet number: 150.1.7.7
     (Link Data) Network Mask: 255.255.255.255
      Number of TOS metrics: 0
       TOS 0 Metrics: 1
```

So we have the same situation here – the LSA is in the database, but the attached NLRI prefix is not. The only thing that may prevent information from LSA type-1 being used is RIB route filter. Let’s check if we have one:

```
Rack1R2#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 7
  Router ID 150.1.2.2
  It is an area border and autonomous system boundary router
  Redistributing External Routes from,
  Number of areas in this router is 2. 1 normal 0 stub 1 nssa
  Maximum path: 4
  Routing for Networks:
    150.1.2.2 0.0.0.0 area 0
    162.1.0.2 0.0.0.0 area 0
    162.1.27.2 0.0.0.0 area 27
 Reference bandwidth unit is 100 mbps
  Routing Information Sources:
    Gateway         Distance      Last Update
    150.1.5.5            110      00:50:05
    150.1.4.4            110      00:50:05
    150.1.3.3            110      00:50:05
    162.1.7.7            110      00:50:05
  Distance: (default is 110)
<snip>
```

Now we look at the access-list:

```
Rack1R2#show ip access-lists 7
Standard IP access list 7
    10 deny   150.1.7.7 (1 match)
    20 permit any (12 matches)
```

Which clearly shows SW1’s Loopback0 prefix being filtered out. This is causing the problem: with no FA in the routing table, the OSPF process will not install the route corresponding to type-7 LSA and will not translate it into type-5 LSA.

Fix the Issue

To fix the issue, stop R2 from filtering R7’s Loopback0 prefix:

```
R2:
ip access-list standard 7
 no 10
```

Verify

Check that R2 now has the route to VLAN7:

```
Rack1R2#show ip route 162.1.77.0
Routing entry for 162.1.77.0/24
  Known via "ospf 1", distance 110, metric 20, type NSSA extern 2,
forward metric 2
  Last update from 162.1.27.7 on FastEthernet0/0, 00:00:16 ago
  Routing Descriptor Blocks:
  * 162.1.27.7, from 162.1.7.7, 00:00:16 ago, via FastEthernet0/0
      Route metric is 20, traffic share count is 1
```

Check that the corresponding type-5 LSA is now generated:

```
Rack1R2#show ip ospf database external 162.1.77.0

            OSPF Router with ID (150.1.2.2) (Process ID 1)

                Type-5 AS External Link States

  LS age: 22
  Options: (No TOS-capability, DC)
  LS Type: AS External Link
  Link State ID: 162.1.77.0 (External Network Number )
  Advertising Router: 150.1.2.2
  LS Seq Number: 80000001
  Checksum: 0x1C52
  Length: 36
  Network Mask: /24
        Metric Type: 2 (Larger than any link state path)
        TOS: 0
        Metric: 20
        Forward Address: 150.1.7.7
        External Route Tag: 0
```

Check the route in R3's table:

```
Rack1R3#show ip route 162.1.77.0
Routing entry for 162.1.77.0/24
  Known via "ospf 1", distance 110, metric 20, type extern 2, forward
metric 783
  Redistributing via eigrp 200
  Advertised by eigrp 200 metric 10000 1000 100 1 1500
  Last update from 162.1.0.2 on Serial1/0, 00:02:45 ago
  Routing Descriptor Blocks:
  * 162.1.0.2, from 150.1.2.2, 00:02:45 ago, via Serial1/0
      Route metric is 20, traffic share count is 1
```

```
Rack1R3#traceroute 162.1.77.7

Type escape sequence to abort.
Tracing the route to 162.1.77.7

  1 162.1.0.2 28 msec 28 msec 28 msec
  2 162.1.27.7 32 msec *  28 msec
```

And finally trace the route from R5:

```
Rack1R5#traceroute 162.1.77.7

Type escape sequence to abort.
Tracing the route to 162.1.77.7

  1 162.1.0.4 28 msec 28 msec 28 msec
  2 162.1.0.3 48 msec 36 msec 40 msec
  3 162.1.0.2 76 msec 68 msec 64 msec
  4 162.1.27.7 68 msec *  72 msec
```

Ticket 6

Analyze the Symptoms

Start with looking at the routing table of R5, to see if this might be a routing issue.

```
Rack1R5#show ip route 112.0.0.0
% Network not in table
Rack1R5#
```

This makes the issue pretty clear, at least the initial guess – the routing is broken between BB3 and R5. We need to isolate the issue down to the router that actually breaks it.

Isolate the Issue

R5 is supposed to receive the prefixes from R4 via BGP. Quickly check if R4 has the prefix in question:

```
Rack1R4#show ip route 112.0.0.0
Routing entry for 112.0.0.0/8
  Known via "bgp 100", distance 20, metric 0
  Tag 54, type external
  Last update from 204.12.1.254 00:11:16 ago
  Routing Descriptor Blocks:
  * 204.12.1.254, from 204.12.1.254, 00:11:16 ago
      Route metric is 0, traffic share count is 1
      AS Hops 3
      Route tag 54
```

It does, now quickly check the connectivity:

```
Rack1R4#ping 112.0.0.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 112.0.0.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 84/86/88 ms
```

Looks like the problem lies between R4 and R5. Let's check the BGP peering session status:

```
Rack1R5#show ip bgp summary
BGP router identifier 150.1.5.5, local AS number 500
BGP table version is 2, main routing table version 2
1 network entries using 132 bytes of memory
1 path entries using 52 bytes of memory
2/1 BGP path/bestpath attribute entries using 336 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 520 total bytes of memory
BGP activity 1/0 prefixes, 1/0 paths, scan interval 60 secs

Neighbor        V          AS MsgRcvd MsgSent   TblVer  InQ OutQ
Up/Down  State/PfxRcd
150.1.4.4       4        100       0       0        0    0    0 never
Active
```

As we see, the BGP session never came up. Let's check the neighbor's detailed stats:

```
Rack1R5#show ip bgp neighbors 150.1.4.4
BGP neighbor is 150.1.4.4,  remote AS 100, external link
  BGP version 4, remote router ID 0.0.0.0
  BGP state = Active
  Last read 01:07:53, last write 01:07:53, hold time is 180, keepalive
interval is 60 seconds
  Message statistics:
    InQ depth is 0
    OutQ depth is 0

                         Sent       Rcvd
    Opens:                  0          0
    Notifications:          0          0
    Updates:                0          0
    Keepalives:             0          0
    Route Refresh:          0          0
    Total:                  0          0
  Default minimum time between advertisement runs is 30 seconds

 For address family: IPv4 Unicast
  BGP table version 2, neighbor version 0/0
  Output queue size : 0
  Index 1, Offset 0, Mask 0x2
  1 update-group member
  Community attribute sent to this neighbor
  Outbound path policy configured
  Route map for outgoing advertisements is NO_ADVERTISE
                                 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

  Address tracking is enabled, the RIB does have a route to 150.1.4.4
  Connections established 0; dropped 0
  Last reset never
  External BGP neighbor may be up to 255 hops away.
  Transport(tcp) path-mtu-discovery is enabled
  No active TCP connection
```

As we can see the TCP connection was never established. Additionally, we learn that the neighbor is configured for eBGP Multihop. Now let’s get the same output at R4 and compare that to the one we just inspected.

```
Rack1R4#show ip bgp neighbors 150.1.5.5
BGP neighbor is 150.1.5.5,  remote AS 500, external link
  BGP version 4, remote router ID 0.0.0.0
  BGP state = Idle
  Last read 00:23:45, last write 00:23:45, hold time is 180, keepalive
interval is 60 seconds
  Message statistics:
    InQ depth is 0
    OutQ depth is 0

                         Sent       Rcvd
    Opens:                  0          0
    Notifications:          0          0
    Updates:                0          0
    Keepalives:             0          0
    Route Refresh:          0          0
    Total:                  0          0
  Default minimum time between advertisement runs is 30 seconds

 For address family: IPv4 Unicast
  BGP table version 18, neighbor version 0/0
  Output queue size : 0
  Index 1, Offset 0, Mask 0x2
  1 update-group member
                                 Sent       Rcvd
  Prefix activity:               ----       ----
    Prefixes Current:              17          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

  Address tracking is enabled, the RIB does have a route to 150.1.5.5
  Connections established 0; dropped 0
  Last reset never
  External BGP neighbor not directly connected.
  Transport(tcp) path-mtu-discovery is enabled
  No active TCP connection
```

As we can see, R4 complains that R5 is not directly connected. Most likely, R4 is not configured for eBGP Multihop, and thus cannot establish a connection to R5. We fix that issue and get the BGP session up:

```
Rack1R5#show ip bgp summary
BGP router identifier 150.1.5.5, local AS number 500
BGP table version is 19, main routing table version 19
18 network entries using 2376 bytes of memory
18 path entries using 936 bytes of memory
6/5 BGP path/bestpath attribute entries using 1008 bytes of memory
4 BGP AS-PATH entries using 128 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
Bitfield cache entries: current 1 (at peak 1) using 32 bytes of memory
BGP using 4480 total bytes of memory
BGP activity 18/0 prefixes, 18/0 paths, scan interval 60 secs

Neighbor        V          AS MsgRcvd MsgSent   TblVer  InQ OutQ
Up/Down  State/PfxRcd
150.1.4.4       4        100      12       6       19    0    0
00:02:03       17
```

However we still cannot reach the prefix from R5:

```
Rack1R5#traceroute 112.0.0.1 source fastEthernet 0/0

Type escape sequence to abort.
Tracing the route to 112.0.0.1

  1 162.1.0.4 28 msec
    162.1.45.4 8 msec
    162.1.0.4 28 msec
  2  *  *
Rack1R5#
```

And we can quickly guess the reason – the subnet is not advertised to BB3.

```
Rack28R4#show ip bgp neighbors 204.12.28.254 advertised-routes
BGP table version is 21, local router ID is 150.28.4.4
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
<snip>
*> 150.28.10.0/24   162.28.0.2                             0 300 i
*> 162.28.7.0/24    162.28.0.2                             0 300 i

Total number of prefixes 12
```

If we check R5's BGP table, we wont find the prefix either:

```
Rack1R5#show ip bgp regexp ^$
BGP table version is 19, 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
*> 162.1.15.0/24    0.0.0.0                  0         32768 i
```

We can fix that by manually advertising the subnet into BGP on R4 – this is the simplest way if R4 has reachability to it, since there might be filtering configured between R4 and R5.

```
Rack1R4#show ip route 162.1.5.0
Routing entry for 162.1.5.0/24
  Known via "ospf 1", distance 110, metric 65, type intra area
  Redistributing via rip
  Advertised by rip metric 1
  Last update from 162.1.45.5 on Serial0/1/0, 00:38:30 ago
  Routing Descriptor Blocks:
    162.1.45.5, from 150.1.5.5, 00:38:30 ago, via Serial0/1/0
      Route metric is 65, traffic share count is 1
  * 162.1.0.5, from 150.1.5.5, 01:19:24 ago, via Serial0/0/0
      Route metric is 65, traffic share count is 1
```

```
R4:
router bgp 500
 network 162.1.5.0 mask 255.255.255.0
```

Fix the Issue

We fixed the issue by properly configuring the eBGP peering sessions and advertising R5's connected network into BGP on R4.

```
R4:
router bgp 100
 neighbor 150.1.5.5 ebgp-multihop
 network 162.1.5.0 mask 255.255.255.0
```

Verify

Test the end-to-end connectivity between the VLAN2005 interface and the remote network:

```
Rack1R5#ping 112.0.0.1 source fastEthernet 0/0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 112.0.0.1, timeout is 2 seconds:
Packet sent with a source address of 162.1.5.5
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 64/65/68 ms
```

Ticket 7

Analyze the Symptoms

Previously we have fixed the BGP peering issue between R4 and R5. Now, we have to deal with MPLS VPN deployed between the two directly connected routers. In order for MPLS VPN to function correctly, the following conditions should be satisfied:

1. There should be an end-to-end transport LSP path between the VPNv4 peering Loopback interfaces of PE routers.
2. The above mentioned LSP should function both ways (remember, MPLS LSPs are unidirectional).
3. The VPNv4 sessions should be active and the routes should be propagated into MP-BGP via redistribution.
4. MP-BGP routes should be tagged correctly with extended communities, so that they could be imported into respective VRFs.
5. The MP-BGP routes should have correct NLRI, meaning that correct labels should be generated for MP-BGP prefixes, so that a PE may process them once the transport LSP terminates.

Lets start using the above checklist to isolate the issue..

Isolate the Issue

R4 and R5 are directly connected. This means that the transport LSP is effectively using implicit-null labels generated for the respective Loopback0 interface – recall that R4 and R5 use their Loopback0's for eBGP peering. Let's see if there is LDP configured between R4 and R5.

```
Rack1R5#show mpls ldp neighbor
    Peer LDP Ident: 150.1.4.4:0; Local LDP Ident 162.1.15.5:0
        TCP connection: 150.1.4.4.646 - 162.1.15.5.13627
        State: Oper; Msgs sent/rcvd: 33/40; Downstream
        Up time: 00:12:01
        LDP discovery sources:
          Serial0/0/0, Src IP addr: 162.1.0.4
        Addresses bound to peer LDP Ident:
          204.12.1.4      162.1.0.4       162.1.45.4      150.1.4.4
```

```
Rack1R4#show mpls interfaces
Interface              IP            Tunnel   BGP Static Operational
Serial0/0/0            Yes (ldp)     No       No  No     Yes
Serial0/1/0            Yes (ldp)     No       No  No     Yes
Rack1R4#

Rack1R5#show mpls interfaces
Interface              IP            Tunnel   BGP Static Operational
Serial0/0/0            Yes (ldp)     No       No  No     Yes
Serial0/1/0            Yes (ldp)     No       No  No     Yes
```

There is an active LDP TCP connection between the Loopback0 interfaces of R4 and R5. Let's check the MPLS LFIB on both routers to check the labels generated for Loopback0 interfaces.

```
Rack1R5#show mpls forwarding-table 150.1.4.4
Local  Outgoing      Prefix            Bytes Label   Outgoing   Next
Hop
Label  Label or VC   or Tunnel Id      Switched      interface
20     No Label      150.1.4.4/32      0             Se0/0/0
162.1.0.4
       No Label      150.1.4.4/32      0             Se0/1/0
162.1.45.4
Rack1R5#

Rack1R4#show mpls forwarding-table 150.1.5.5
Local  Outgoing      Prefix            Bytes Label   Outgoing   Next
Hop
Label  Label or VC   or Tunnel Id      Switched      interface
26     No Label      150.1.5.5/32      0             Se0/0/0
162.1.0.5
       No Label      150.1.5.5/32      0             Se0/1/0
162.1.45.5
```

There are no outgoing labels for the remote prefixes at each end. What might be causing this? To start with, let's check the local labels that every router generates for Loopback0 interfaces. Start with R4:

```
Rack1R4#show mpls ldp bindings local 150.1.4.4 32

Rack1R4#
```

There is not label for this /32! Let's check the IP address on R4's Loopback0:

```
Rack1R4#show ip interface loopback 0 | inc address
  Internet address is 150.1.4.4/24
  Broadcast address is 255.255.255.255
  Helper address is not set
  Network address translation is disabled
```

It is actually /24 and there is a local label for this /24:

```
Rack1R4#show mpls ldp bindings local 150.1.4.0 24
  lib entry: 150.1.4.0/24, rev 28
        local binding:  label: imp-null
```

So why does R5 see this prefix as /32? The answer probably lies in the OSPF behavior. The Loopback0 is probably defaulted to OSPF stub network:

```
Rack1R4#sh ip ospf interface loopback 0
Loopback0 is up, line protocol is up
  Internet Address 150.1.4.4/24, Area 0
  Process ID 1, Router ID 150.1.4.4, Network Type LOOPBACK, Cost: 1
  Loopback interface is treated as a stub Host
```

OK, so what should we do? Of course, change the IP address to /32. The same issue probably pertains to R5.

```
Rack1R5#show ip ospf interface loopback 0
Loopback0 is up, line protocol is up
  Internet Address 150.1.5.5/24, Area 0
  Process ID 1, Router ID 150.1.5.5, Network Type LOOPBACK, Cost: 1
  Loopback interface is treated as a stub Host
```

Quickly fix both problems:

```
R4:
interface Loobpack0
 ip address 150.1.4.4 255.255.255.255

R5:
interface Loopback0
 ip address 150.1.5.5 255.255.255.255

Rack1R4#show mpls forwarding-table 150.1.5.5
Local  Outgoing      Prefix            Bytes Label   Outgoing   Next
Hop
Label  Label or VC   or Tunnel Id      Switched      interface
26     Pop Label     150.1.5.5/32      0             Se0/0/0
162.1.0.5
       No Label      150.1.5.5/32      0             Se0/1/0
162.1.45.5

Rack1R5#show mpls forwarding-table 150.1.4.4
Local  Outgoing      Prefix            Bytes Label   Outgoing   Next
Hop
Label  Label or VC   or Tunnel Id      Switched      interface
20     Pop Label     150.1.4.4/32      0             Se0/0/0
162.1.0.4
       No Label      150.1.4.4/32      0             Se0/1/0
162.1.45.4
```

Notice that the path across the Serial link may not be used for label switching even though MPLS is enabled there. The reason is that no IGP uses this link. However, maybe our actions have fixed the initial issue? Let's see if we can ping between the VPN interfaces:

```
Rack1R4#ping vrf VPN_A 172.16.5.5 source loopback 100

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.5.5, timeout is 2 seconds:
Packet sent with a source address of 172.16.4.4
 ...
Success rate is 0 percent (0/5)
```

No, we still need to troubleshoot more. We confirmed that both /32 LSP endpoints are now tagged using implicit null labels. This assures that the transport LSP is functional both ways. Now let's check VPNv4 settings and confirm that MP-BGP propagate our routes:

```
Rack1R5#show bgp vpnv4 unicast all summary
BGP router identifier 150.1.5.5, local AS number 500
BGP table version is 9, main routing table version 9
1 network entries using 156 bytes of memory
1 path entries using 68 bytes of memory
8/1 BGP path/bestpath attribute entries using 1344 bytes of memory
5 BGP AS-PATH entries using 152 bytes of memory
1 BGP extended 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
Bitfield cache entries: current 1 (at peak 2) using 32 bytes of memory
BGP using 1776 total bytes of memory
BGP activity 57/37 prefixes, 94/74 paths, scan interval 15 secs

Neighbor        V          AS MsgRcvd MsgSent   TblVer  InQ OutQ
Up/Down  State/PfxRcd
150.1.4.4       4        100    1632    1598        9    0    0
00:32:10        0
Rack1R5#

Rack1R4#show bgp vpnv4 unicast all summary
BGP router identifier 150.1.4.4, local AS number 100
<snip>

Neighbor        V          AS MsgRcvd MsgSent   TblVer  InQ OutQ
Up/Down  State/PfxRcd
150.1.5.5       4        500    1599    1632        5    0    0
00:32:27        1
```

We see two things here. First, the VPNv4 address family is activated between the two peers. Secondly, R5 does not receive a single prefix from R4, while R4 receives one prefix from R5. Let's see if R4 redistributes the VPN routes into MP-BGP:

```
Rack1R4#show bgp vpnv4 unicast all
BGP table version is 5, 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
Route Distinguisher: 100:45 (default for vrf VPN_A)
*> 172.16.4.0/24    0.0.0.0                  0         32768 ?
*> 172.16.5.0/24    150.1.5.5                0             0 500 ?
```

We can see that the Loopback100 prefix is being redistributed on R4. Is there a reason that R5 might reject it? The most usual case is that the local prefix is tagged with the Route-Target (RT) extended community not supported on the remote end. Let's see the RT associated with 172.16.4.0/24 on R4:

```
Rack1R4#show bgp vpnv4 unicast all 172.16.4.0
BGP routing table entry for 100:45:172.16.4.0/24, version 5
Paths: (1 available, best #1, table VPN_A)
  Advertised to update-groups:
        1
  Local
    0.0.0.0 from 0.0.0.0 (150.1.4.4)
      Origin incomplete, metric 0, localpref 100, weight 32768, valid,
sourced, best
      Extended Community: RT:100:45
      mpls labels in/out 35/nolabel(VPN_A)
Rack1R4#

Rack1R5#show ip vrf detail
VRF VPN_A; default RD 100:45; default VPNID <not set>
  Interfaces:
    Lo100
VRF Table ID = 1
  Export VPN route-target communities
    RT:100:45
  Import VPN route-target communities
    RT:100:45
  No import route-map
  No export route-map
  VRF label distribution protocol: not configured
  VRF label allocation mode: per-prefix
```

The prefix is tagged with RT 100:45, which is imported by VPN_A VRF in R5. So everything should be working fine, but it's not. Let's do BGP prefix import debugging (notice that you may need to log the debugging info into memory buffer).

```
Rack1R5#debug bgp vpnv4 unicast updates
BGP updates debugging is on for address family: VPNv4 Unicast

Rack1R4#clear bgp vpnv4 unicast * soft out
Rack1R5#
VPN(4): Scanning for import check is done.
Rack1R5#
BGP(4): 150.1.4.4 rcvd UPDATE w/ attr: nexthop 150.1.4.4, origin ?,
metric 0, merged path 100, AS_PATH
BGP(4): 150.1.4.4 rcvd 100:45:172.16.4.0/24, label 35 -- DENIED due to:
extended community not supported;
```

We can see that the remote peer sends R5 the update but the local router rejects it due to unsupported extended community. Usually this means that the prefix received does not have a locally configured RT. Let's see if R5 overrides RT values somehow:

```
Rack1R4#debug bgp vpnv4 unicast updates out
BGP updates debugging is on (outbound) for address family: VPNv4
Unicast

Rack1R4#clear bgp vpnv4 unicast * soft out
Rack1R4#
BGP(4): 150.1.5.5 send UPDATE (format) 100:45:172.16.4.0/24, next
150.1.4.4, label 35, metric 0, extended community RT:100:45
```

So we may assume one of the following: either there is a policy applied to R5's peer IP on R5 or the communities are not being sent for the peering session. To investigate this, we need to extract some parts of the running configuration. In this case, this step is unavoidable as it's hard to collect the outbound policy information using just the show commands.

```
Rack1R4#sh running-config | section include address-family vpnv4
 address-family vpnv4
  neighbor 150.1.5.5 activate
```

We use the "section" keyword to minimize the amount of output. We can see that there is no outbound policy, but at the same time, extended communities are not being sent! Allow the sending of the extended communities to R5:

```
R4:
router bgp 100
 address-family vpnv4 unicast
  neighbor 150.1.5.5 send-community both
!
Rack1R4#clear bgp vpnv4 unicast * soft out
```

Check R5's VPN_A routing table after this:

```
Rack1R5#show ip route vrf VPN_A

Routing Table: VPN_A
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

     172.16.0.0/24 is subnetted, 2 subnets
B       172.16.4.0 [20/0] via 150.1.4.4, 00:01:03
C       172.16.5.0 is directly connected, Loopback100

And we've got end-to-end connectivity now:

Rack1R5#ping vrf VPN_A 172.16.4.4 source loopback 100

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.4.4, timeout is 2 seconds:
Packet sent with a source address of 172.16.5.5
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/62/76 ms
```

Fix the Issue

We had to fix two problems: First, assign /32 IP addresses to the Loopback0 interfaces to ensure OSPF/LDP consistency, and next enable R4 to send extended communities to R5.

```
R4:
interface Loobpack0
 ip address 150.1.4.4 255.255.255.255

R5:
interface Loopback0
 ip address 150.1.5.5 255.255.255.255

R4:
router bgp 100
 address-family vpnv4 unicast
  neighbor 150.1.5.5 send-community both
```

Verify

Once again, double check the transport LSP, the MP-BGP prefixes, and the MPLS labels used to packet transportation:

```
Rack1R5#show ip cef vrf VPN_A 172.16.4.4
172.16.4.0/24
  nexthop 162.1.0.4 Serial0/0/0 label 35
  nexthop 162.1.45.4 Serial0/1/0 unusable: no label

Rack1R4#show ip cef vrf VPN_A 172.16.5.5
172.16.5.0/24
  nexthop 162.1.0.5 Serial0/0/0 label 27
  nexthop 162.1.45.5 Serial0/1/0 unusable: no label
```

The labels used by CEF are actually taken from MP-BGP updates:

```
Rack1R4#show bgp vpnv4 unicast all 172.16.5.0
BGP routing table entry for 100:45:172.16.5.0/24, version 8
Paths: (1 available, best #1, table VPN_A)
  Not advertised to any peer
  500
    150.1.5.5 (metric 65) from 150.1.5.5 (150.1.5.5)
      Origin incomplete, metric 0, localpref 100, valid, external, best
      Extended Community: RT:100:45
      mpls labels in/out nolabel/27

Rack1R5#show bgp vpnv4 unicast all 172.16.4.0
BGP routing table entry for 100:45:172.16.4.0/24, version 5
Paths: (1 available, best #1, table VPN_A)
  Not advertised to any peer
  100
    150.1.4.4 (metric 65) from 150.1.4.4 (150.1.4.4)
      Origin incomplete, metric 0, localpref 100, valid, external, best
      Extended Community: RT:100:45
      mpls labels in/out nolabel/35
```

Ticket 8

Analyze the Symptoms

Let’s check the BGP peering session status on SW4:

```
Rack1SW4#show ip bgp summary
BGP router identifier 150.1.10.10, local AS number 65034
BGP table version is 61, main routing table version 61
13 network entries using 1521 bytes of memory
13 path entries using 676 bytes of memory
12/5 BGP path/bestpath attribute entries using 1392 bytes of memory
9 BGP AS-PATH entries using 248 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 3837 total bytes of memory
BGP activity 31/18 prefixes, 52/39 paths, scan interval 60 secs

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
162.1.10.7      4 65001    2510    2507       61    0    0 1d17h
12
162.1.109.9     4 65034    2461    2463        0    0    0 00:42:47
Active
```

The session is active, let’s inspect it:

```
Rack1SW4#show ip bgp neighbors 162.1.109.9
BGP neighbor is 162.1.109.9,  remote AS 65034, internal link
  BGP version 4, remote router ID 0.0.0.0
  BGP state = Active
  Last read 00:42:56, last write 00:42:56, hold time is 180, keepalive
interval is 60 seconds
  Message statistics:
    InQ depth is 0
    OutQ depth is 0
                         Sent       Rcvd
    Opens:                  1          1
    Notifications:          1          0
    Updates:                8         11
    Keepalives:          2453       2449
    Route Refresh:          0          0
    Total:               2463       2461
  Default minimum time between advertisement runs is 0 seconds

 <snip>
```

```
  Connections established 1; dropped 1
  Last reset 00:42:58, due to BGP Notification sent, hold time expired
  No active TCP connection
```

As we can see, the local end cannot establish a TCP connection to the remote end. Let's see if we can ping the remote end:

```
Rack1SW4#ping 162.1.109.9

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 162.1.109.9, timeout is 2 seconds:
 ...
Success rate is 0 percent (0/5)
```

This means we have a problem on the link connecting the two switches, which should be a Layer 3 Etherchannel. We'll start troubleshooting this problem bottom-up.

Isolate the Issue

First, check the etherchannel status in SW4:

```
Rack1SW4#show etherchannel summary
Flags:  D - down        P - in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      f - failed to allocate aggregator
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port

Number of channel-groups in use: 3
Number of aggregators:           3

Group  Port-channel  Protocol    Ports ------+-------------
+-----------+-----------------------------------
14     Po14(SU)        -        Fa0/13(P)   Fa0/14(P)
41     Po41(RU)       LACP      Fa0/15(P)
43     Po43(RD)       LACP      Fa0/19(s)   Fa0/20(s)   Fa0/21(s)

Rack1SW4#show interfaces fastEthernet 0/19
FastEthernet0/19 is up, line protocol is down (suspended)
  Hardware is Fast Ethernet, address is 001a.a174.13c4 (bia
001a.a174.13c4)
  MTU 1504 bytes, BW 100000 Kbit, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/25
<snip>
```

As we can see, all three member links of the Etherchannel are suspended. Also, notice that LACP is used for Etherchannel negotiations. Let’s use some debugging and see what might be causing this behavior.

```
Rack1SW4#debug lacp event
Link Aggregation Control Protocol events debugging is on
Rack1SW4#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1SW4(config)#interface port-channel 43
Rack1SW4(config-if)#shutdown
Rack1SW4(config-if)#
LACP: Fa0/19 set to UNSELECTED
LACP: Fa0/20 set to UNSELECTED
LACP: Fa0/21 set to UNSELECTED

%LINK-5-CHANGED: Interface FastEthernet0/19, changed state to
administratively down
%LINK-5-CHANGED: Interface FastEthernet0/20, changed state to
administratively down
%LINK-5-CHANGED: Interface FastEthernet0/21, changed state to
administratively down
%LINK-5-CHANGED: Interface Port-channel43, changed state to
administratively down

Rack1SW4(config-if)#no shutdown
Rack1SW4(config-if)#
%LINK-3-UPDOWN: Interface Port-channel43, changed state to down
%LINK-3-UPDOWN: Interface FastEthernet0/19, changed state to up
%LINK-3-UPDOWN: Interface FastEthernet0/20, changed state to up
%LINK-3-UPDOWN: Interface FastEthernet0/21, changed state to up
Rack1SW4(config-if)#
LACP: Fa0/19 set to SELECTED
LACP: Fa0/20 set to SELECTED
LACP: Fa0/21 set to SELECTED
Rack1SW4(config-if)#
%EC-5-L3DONTBNDL2: Fa0/19 suspended: LACP currently not enabled on the
remote port.
%EC-5-L3DONTBNDL2: Fa0/21 suspended: LACP currently not enabled on the
remote port.
%EC-5-L3DONTBNDL2: Fa0/20 suspended: LACP currently not enabled on the
remote port.
```

Based on this output we may assume that the other side of the link (SW3, which we cannot access) is not configured for LACP. There are two options here – either the remote end is configured for PaGP or statically. Let’s configure the member links for PaGP first:

```
Rack1SW4(config)#interface range fastEthernet 0/19 - 21
Rack1SW4(config-if-range)#no channel-group 43 mode active
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/19,
changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/20,
changed state to up
Rack1SW4(config-if-range)#channel-group 43 mode desirable
Rack1SW4(config-if-range)#
%EC-5-L3DONTBNDL1: Fa0/19 suspended: PAgP not enabled on the remote
port.
%EC-5-L3DONTBNDL1: Fa0/21 suspended: PAgP not enabled on the remote
port.
%EC-5-L3DONTBNDL1: Fa0/20 suspended: PAgP not enabled on the remote
port.
```

The last option that we have is static Etherchannel configuration:

```
Rack1SW4(config)#interface range FastEthernet 0/19 - 21
Rack1SW4(config-if-range)#no channel-group 43 mode desirable
Rack1SW4(config-if-range)#channel-group 43 mode on
Rack1SW4(config-if-range)#
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/19,
changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/20,
changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/21,
changed state to up
Rack1SW4(config-if-range)#

%PIM-5-NBRCHG: neighbor 162.1.109.9 UP on interface Port-channel43
%LINEPROTO-5-UPDOWN: Line protocol on Interface Port-channel43, changed
state to up
Rack1SW4(config-if-range)#^Z
Rack1SW4#
%BGP-5-ADJCHANGE: neighbor 162.1.109.9 Up
```

As we can see, this fixes the problem and the BGP peering session is now up.

Fix the Issue

The issue was the misconfigured Ether-Channel between SW3 and SW4. We fixed it by configuring the channel in static mode:

```
SW4:
interface range FastEthernet 0/19 - 21
 channel-group 43 mode on
```

Verify

Check the EtherChannel status and the BGP peering session

```
Rack1SW4#show etherchannel summary
Flags:  D - down        P - in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      f - failed to allocate aggregator
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port


Number of channel-groups in use: 3
Number of aggregators:           3

Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------
14     Po14(SU)         -        Fa0/13(P)   Fa0/14(P)
41     Po41(RU)        LACP      Fa0/15(P)
43     Po43(RU)         -        Fa0/19(P)   Fa0/20(P)   Fa0/21(P)

Rack1SW4#show ip bgp summary
BGP router identifier 150.1.10.10, local AS number 65034
BGP table version is 66, main routing table version 66
18 network entries using 2106 bytes of memory
29 path entries using 1508 bytes of memory
12/8 BGP path/bestpath attribute entries using 1392 bytes of memory
9 BGP AS-PATH entries using 248 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 5254 total bytes of memory
BGP activity 36/18 prefixes, 69/40 paths, scan interval 60 secs

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
162.1.10.7      4 65001    2773    2770       66    0    0 1d22h
12
162.1.109.9     4 65034    2485    2484       66    0    0 00:13:40
16
```

Ticket 9

Analyze the Symptoms

Let’s check what may be causing EIGRP route loss on SW2. Start with checking the adjacency health:

```
Rack1SW2#show ip eigrp neighbors
EIGRP-IPv4:(200) neighbors for process 200
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q
Seq
                                            (sec)         (ms)
Cnt Num
0   162.1.38.3              Fa0/15            14 00:00:05    1  4500  2
0
```

As we can see the adjacency is up, but the Q (queue depth) is non-zero, which means the local end has some unsent/unconfirmed packets. The fact that the adjacency is up means that the local router heard an EIGRP HELLO from the remote end. You may also notice the following syslog messages on the console:

```
%DUAL-5-NBRCHANGE: EIGRP-IPv4:(200) 200: Neighbor 162.1.38.3
(FastEthernet0/15) is down: Interface PEER-TERMINATION received
%DUAL-5-NBRCHANGE: EIGRP-IPv4:(200) 200: Neighbor 162.1.38.3
(FastEthernet0/15) is up: new adjacency
```

Which clearly means some problems with the adjacency. This is our initial hypothesis, and will continue researching it.

Isolate the Issue

Let’s start with checking the link between R3 and SW2. Since we receive the EIGRP HELLO packets, it’s working at least in one direction. Let’s check if R3 sees SW2:

```
Rack1R3#show ip eigrp neighbors
IP-EIGRP neighbors for process 200
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q
Seq
                                            (sec)         (ms)
Cnt Num
1   162.1.38.8              Fa0/0             14 00:00:35    1  5000  2
81
0   162.1.13.1              Se1/1            176 1d23h      58  1140  0
26
```

We see the same issue at R3. So the root cause might be that multicast packets make it through, while unicast don’t. Let’s check if we can ping across the link:

```
Rack1R3#ping 162.1.38.8

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 162.1.38.8, timeout is 2 seconds:
 ...
Success rate is 0 percent (0/5)
```

There is something blocking the packet exchange between R3 and SW2. This might be a packet filter; to further isolate the issue, let's see if we have ARP entries for both endpoints:

```
Rack1SW2#show ip arp 162.1.38.3
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  162.1.38.3              0   0011.2093.e560  ARPA
FastEthernet0/15
Rack1SW2#

Rack1R3#show ip arp 162.1.38.8

Rack1R3#
```

Notice the Age for the ARP entry in SW2 – it means the entry has been just requested. We see two strange things here:

1. SW2 sees an ARP entry for R3 while R3 does not see an ARP entry for SW2.
2. The entry for SW2's IP in R3 is not shown as "Incomplete" like it should be in case if ARP times out. It's simply empty, which looks like ARP is not working.

Let's see ARP settings for R3's VLAN 38 interface:

```
Rack1R3#show ip interface fastEthernet 0/0 | inc ARP
  Proxy ARP is enabled
  Local Proxy ARP is disabled
  Authorized ARP is enabled
```

An interesting detail – Authorized ARP is enabled on this interface! Which means only DHCP generated entries are allowed to install ARP entries for the interface that matches the DHCP pool. The easiest way to resolve this problem is by disabling Authorized ARP in R3. However, we are explicitly prohibited from doing this.

Thus, another acceptable solution is configuring SW2 to obtain an IP address on VLAN38 via DHCP.

SW2:
```
interface FastEthernet 0/15
 ip address dhcp
```

Fix the Issue

The issue was that Authorized DHCP has been configured on R3. However, since we cannot disable this, we had to enable DHCP autoconfiguration for SW2.

```
SW2:
interface FastEthernet 0/15
 ip address dhcp
```

Verify

Check EIGRP neighbors, ping between SW2 and R3 and check EIGRP routes in SW2:

```
Rack1SW2#show ip eigrp neighbors
EIGRP-IPv4:(200) neighbors for process 200
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q
Seq
                                            (sec)         (ms)
Cnt Num
0   162.1.38.3              Fa0/15            14 00:00:18   13   200  0
151

Rack1SW2#ping 162.1.38.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 162.1.38.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/4/9 ms

Rack1SW2#show ip route eigrp
D EX 204.12.1.0/24 [170/1027563] via 162.1.38.3, 00:01:00,
FastEthernet0/15
     54.0.0.0/24 is subnetted, 1 subnets
D EX    54.1.1.0 [170/60848991] via 162.1.38.3, 00:01:00,
FastEthernet0/15
     162.1.0.0/24 is subnetted, 14 subnets
D EX    162.1.45.0 [170/1027563] via 162.1.38.3, 00:01:00,
FastEthernet0/15
D EX    162.1.55.0 [170/1027563] via 162.1.38.3, 00:01:00,
FastEthernet0/15
D       162.1.3.0 [90/82020] via 162.1.38.3, 00:01:00, FastEthernet0/15
D EX    162.1.0.0 [170/1027563] via 162.1.38.3, 00:01:00,
FastEthernet0/15
D       162.1.13.0 [90/60592991] via 162.1.38.3, 00:01:00,
FastEthernet0/15
D EX    162.1.27.0 [170/1027563] via 162.1.38.3, 00:01:00,
<snip>
```

Ticket 10

Analyze the Symptoms

This ticket is special in one sense. We cannot truly verify the configuration “in action” as there is no real server capturing netflow statistics that we can access. However, what we could do is use some show commands and try to match the information against the baseline, making the necessary changes.

Isolate the Issue

The first thing we should check is the export configuration. This includes the export destination, the destination port and the Netflow version used.

```
Rack1R6#show ip flow export
Flow export v1 is enabled for main cache
  Export source and destination details :
  VRF ID : Default
    Destination(1)  192.10.1.100 (9999)
  Version 1 flow records
  0 flows exported in 0 udp datagrams
  0 flows failed due to lack of export packet
  0 export packets were sent up to process level
  0 export packets were dropped due to no fib
  0 export packets were dropped due to adjacency issues
  0 export packets were dropped due to fragmentation failures
  0 export packets were dropped due to encapsulation fixup failures
```

We see two things that need to be changed immediately - the remote destination port and the export version.

```
R6:
no ip flow-export destination 192.10.1.100 9999
ip flow-export destination 192.10.1.100 9996
ip flow-export version 5
```

OK, so this takes care of the export settings, which now match the baseline specification. Next we need to ensure that HTTP flows for VLAN6 subnets are captured and reported to the NMS. The ticket explicitly mentions that the amount of netflow information should be kept to the minimum required. This implies the use of Netflow filters, as by default all flows ingress or egress on the particular interface are reported. Use the following command to discover the interfaces configured for Netflow:

```
Rack1R6#show ip flow interface
Serial0/0/0.1
  egress MQC netflow-sampler NETFLOW
```

This output shows that Netflow sampler is applied to a particular MQC class by means of egress service policy. To see the detailed information use the regular MQC commands:

```
Rack1R6#show policy-map interface serial 0/0/0.1

 Serial0/0/0.1

  Service-policy output: WAN_OUT

    Class-map: NETFLOW_OUT (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: access-group name NETFLOW_SUBNETS
      netflow-sampler: NETFLOW

    Class-map: class-default (match-any)
      758 packets, 127887 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any

Rack1R6#show ip access-lists NETFLOW_SUBNETS
Extended IP access list NETFLOW_SUBNETS
    10 permit ip 162.1.6.0 0.0.0.255 any

Rack1R6#show flow-sampler

 Sampler : NETFLOW, id : 1, packets matched : 0, mode : random sampling
mode
  sampling interval is : 1
```

We can see that the flow sampler is set correctly, sampling every packet in the flow. This prevents any information loss. However, the MQC class is defined incorrectly. The policy is “egress” and the access-list matches traffic “from any to VLAN6” while it should be matching in the opposite direction. In addition to this, only HTTP flows should be matched. Thus, we can edit the access-list as following:

```
R6:
ip access-list extended NETFLOW_SUBNETS
 no permit ip 162.1.6.0 0.0.0.255 any
 permit tcp 162.1.6.0 0.0.0.255 any eq 80
```

Are we done here? Not yet. If you look at the current configuration once again, you will see that it only captures egress HTTP flows, but not the returning traffic, as required for bi-directional capturing. To accomplish this goal, we need an additional MQC traffic class and an ingress service-policy on the Frame-Relay link interface.

```
R6:
ip access-list extended NETFLOW_SUBNETS_IN
 permit tcp any eq 80 162.1.6.0 0.0.0.255
!
class-map NETFLOW_IN
 match access-group name NETFLOW_SUBNETS_IN
!
policy-map WAN_IN
 class NETFLOW_IN
  flow-sampler NETFLOW
!
interface Serial 0/0/0.1
 service-policy input WAN_IN
```

This accomplishes all the ticket requirements. Notice that this particular scenario was more “analysis” oriented rather then search and find the issue.

Fix the Issue

The complete fixup includes changing the export settings, modifying the Netflow filter and applying the netflow collection inbound on R6’s Frame-Relay link.

Verify

There are no specific verification commands with except to the ones we used during the analysis and isolation steps.

Lab 5 Solutions

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.

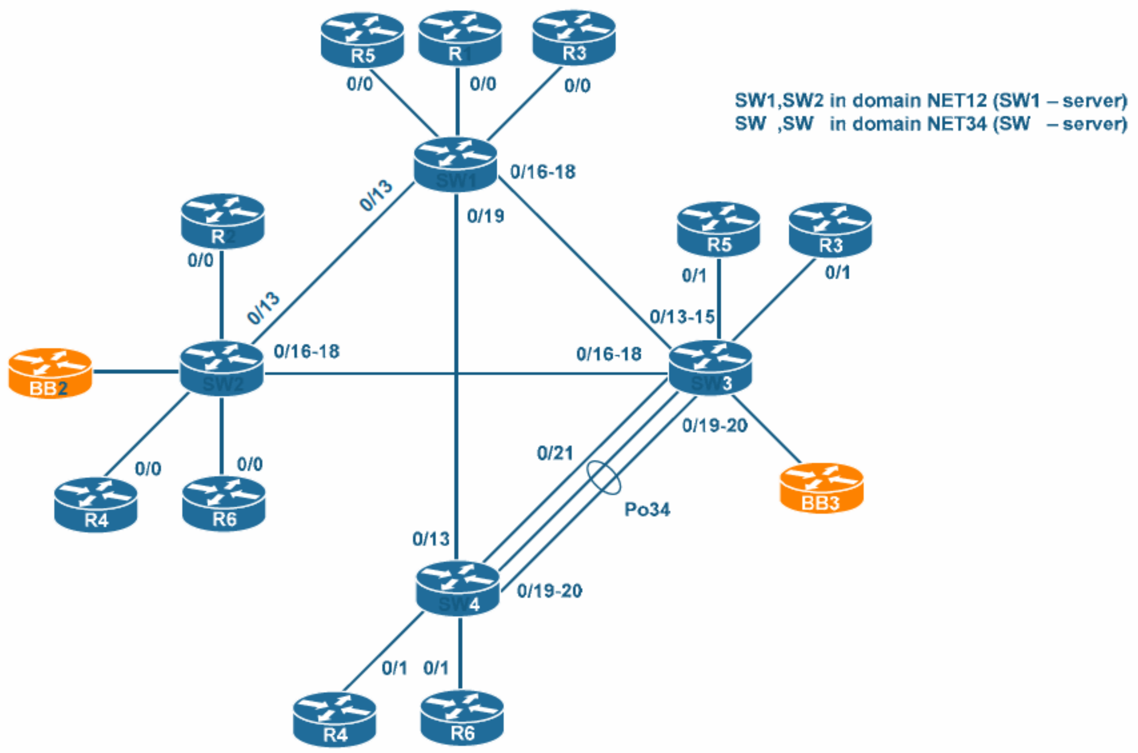


Fig 1

The baseline does not provide any detailed information on L2 topology, so it's up to you to use the commands **`show cdp neighbor`**, **`show interfaces trunk`** and **`show etherchannel summary`** to discover the active layer 2 connections. When looking at the diagram you may notice that layer 2 topology has some loops, and therefore some of the links will be blocking.

BGP Diagram

Based on the information supplied with the scenario we can build the BGP peering diagram. Notice that you will need to use the commands **`show ip bgp summary`** in order to discover the BGP peering sessions.

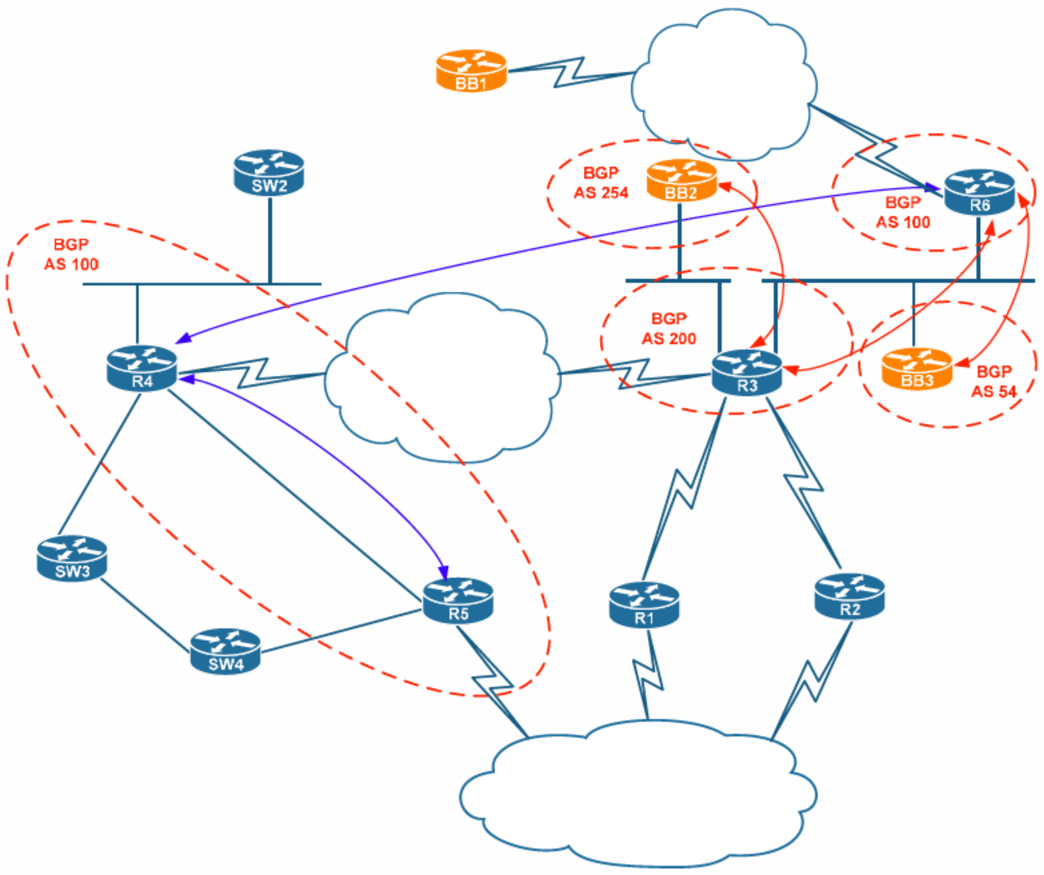
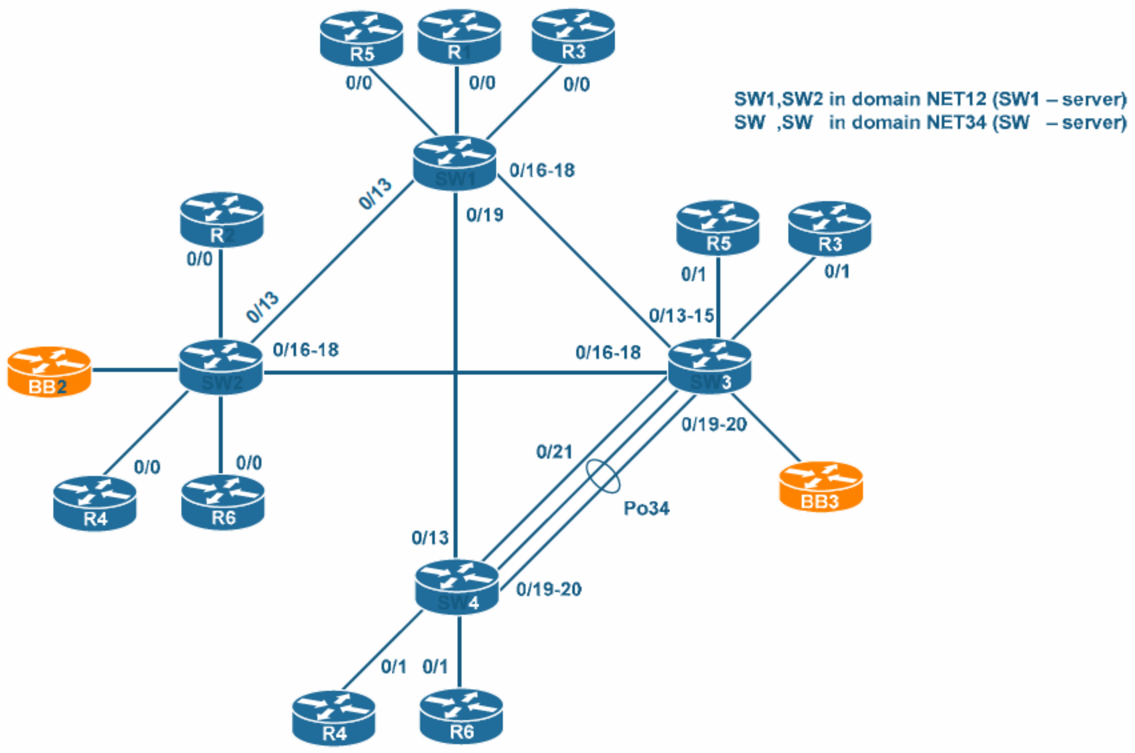


Fig 2

MPLS VPN Diagram

Based on the baseline information and using the commands **`show ip vrf, show bgp vpnv4 unicast all summary`** you may discover the VPN interfaces and BGP peerings used for MP-BGP. Use the main diagram to create a special drawing for MPLS VPNs.



Having a separate MPLS VPN diagram is important in situations where you have more than one VPN provisioned, as marking VPN relationships might become cumbersome when using just a single diagram.

Ticket 1

Analyze the Symptoms

From the baseline information we can find that R4 connects to R6 using QinQ configuration. Based on our L2 diagram, we can see that SW4, SW3 and SW2 are involved in the connection. Based on the ticket statement, we may start assuming that there is a connectivity problem along the path. We can quickly verify this:

```
Rack1R4#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address
Interface
150.1.5.5         0   FULL/  -           -        191.1.45.5
OSPF_VL1
150.1.3.3         0   FULL/  -           -        191.1.34.3
OSPF_VL0
150.1.3.3         0   FULL/  -        00:00:32    191.1.34.3
Serial0/0/0
150.1.5.5         1   FULL/DR         00:00:33    191.1.45.5
FastEthernet0/0.45
```

```
Rack1R4#ping 191.1.48.8

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 191.1.48.8, timeout is 2 seconds:
 ...
Success rate is 0 percent (0/5)
```

Initial hypothesis: Problem in the path between R4 and SW2
Problem Scope: R4, SW4, SW3 and SW2

Isolate the Issue

Let's verify the QinQ configuration first. Double tagging involves using a transport VLAN, which should be created in every transit switch. Let's check the configur-ation applied to the port connecting to R4:

```
Rack1SW4#show interfaces fastEthernet 0/4 switchport
Name: Fa0/4
Switchport: Enabled
Administrative Mode: tunnel
Operational Mode: tunnel
Administrative Trunking Encapsulation: negotiate
Operational Trunking Encapsulation: native
Negotiation of Trunking: Off
Access Mode VLAN: 100 (VLAN0100)
Trunking Native Mode VLAN: 1 (default)
Administrative Native VLAN tagging: enabled
Voice VLAN: none
Administrative private-vlan host-association: none
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
```

So the transport VLAN number is 100. Let’s check that this VLAN is active in every transit switch:

```
Rack1SW4#show vlan id 100

VLAN Name                             Status    Ports
---- -------------------------------- --------- -----------------------
--------
100  VLAN0100                         active    Fa0/4, Fa0/13, Fa0/21,
Po34

VLAN Type  SAID       MTU   Parent RingNo BridgeNo Stp  BrdgMode Trans1
Trans2
---- ----- ---------- ----- ------ ------ -------- ---- -------- ------
------
100  enet  100100     1500  -      -      -        -    -        0
0
```

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

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

Rack1SW3#show vlan id 100

```
VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
--------
100  VLAN0100                         active    Fa0/5, Fa0/13, Fa0/14,
Fa0/15
                                                Fa0/16, Fa0/17, Fa0/21,
Po34

VLAN Type  SAID       MTU   Parent RingNo BridgeNo Stp  BrdgMode Trans1
Trans2
---- ----- ---------- ----- ------ ------ -------- ---- -------- ------
------
100  enet  100100     1500  -      -      -        -    -        0
0

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

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

The VLAN should appear on every trunk link and also on every “tunnel” port (ports 0/4 and 0/18 on SW4 and SW3 in our case). If you study the output carefully, you will notice that port 0/18 in SW1 is not in VLAN 100. Let’s check the respective port status in details:

Rack1SW3#show interfaces fa0/18 switchport

```
Name: Fa0/18
Switchport: Enabled
Administrative Mode: tunnel
Operational Mode: tunnel
Administrative Trunking Encapsulation: dot1q
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: 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
```

Looks like this port does not have the correct transport VLAN applied and belongs to the default VLAN 1. We need to go ahead and set the access VLAN to 100:

```
SW3:
interface fastEthernet 0/18
 switchport acces vlan 100
```

Has that fixed our issue? Let’s see if we can ping across now and check the OSPF adjacency:

```
Rack1R4#ping 191.1.48.8

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 191.1.48.8, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms

Rack1R4#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address
Interface
150.1.5.5         0   FULL/  -           -        191.1.45.5
OSPF_VL1
150.1.3.3         0   FULL/  -           -        191.1.34.3
OSPF_VL0
150.1.3.3         0   FULL/  -        00:00:32    191.1.34.3
Serial0/0/0
150.1.5.5         1   FULL/DR         00:00:38    191.1.45.5
FastEthernet0/0.45
150.1.8.8         1   FULL/DR         00:00:38    191.1.48.8
FastEthernet0/1
```

Confirm this by checking if SW2's prefix is in R4's routing table:

```
Rack1R4#show ip route 150.1.8.8
Routing entry for 150.1.8.8/32
  Known via "ospf 1", distance 110, metric 2, type intra area
  Last update from 191.1.48.8 on FastEthernet0/1, 00:00:28 ago
  Routing Descriptor Blocks:
  * 191.1.48.8, from 150.1.8.8, 00:00:28 ago, via FastEthernet0/1
      Route metric is 2, traffic share count is 1
```

Fix the issue

The solution was to assign the correct access VLAN to the tunnel port:

```
SW3:
interface fastEthernet 0/18
 switchport acces vlan 100
```

Verify

Verification consists of validating existence of SW2's OSPF-advertised prefixes in R4's routing table:

```
Rack1R4#show ip route 150.1.8.8
Routing entry for 150.1.8.8/32
  Known via "ospf 1", distance 110, metric 2, type intra area
  Last update from 191.1.48.8 on FastEthernet0/1, 00:00:28 ago
  Routing Descriptor Blocks:
  * 191.1.48.8, from 150.1.8.8, 00:00:28 ago, via FastEthernet0/1
      Route metric is 2, traffic share count is 1
```

Ticket 2

Analyze the Symptoms

Just as usual, start by checking the OSPF adjacency health. Per the PVC outline, R5 should have an adjacency with R1 and R1 should hear R2:

```
Rack1R5#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address
Interface
150.1.1.1         0   FULL/DROTHER    00:00:15    191.1.125.1
Serial0/0/0
150.1.4.4         0   FULL/  -           -        191.1.45.4
OSPF_VL0
150.1.4.4         1   FULL/BDR        00:00:31    191.1.45.4
FastEthernet0/1.45
```

```
Rack1R1#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address
Interface
N/A               0   ATTEMPT/DROTHER    -        191.1.125.2
Serial0/0
150.1.5.5         1   FULL/DR         00:00:19    191.1.125.5
Serial0/0
```

There is definitely a problem in OSPF adjacency establishment between R1 and R2. We need to find out whether the problem relates to L2/L3 or OSPF misconfiguratations.

Initial hypothesis: Either a Frame-Relay communication issue or OSPF misconfiguration.
Problem Scope: R1 and R2 as well as the Frame-Relay cloud.

Isolate the Issue

Start by pinging across the Frame-Relay links to rule out L2/L3 issues:

```
Rack1R1#ping 191.1.125.5

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 191.1.125.5, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/57/60 ms
```

```
Rack1R1#ping 191.1.125.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 191.1.125.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/59/60 ms
```

Alright. Now, based on the OSPF show outputs from the above, we may want to check OSPF network types and neighbor settings on R1 and R2.

```
Rack1R1#show ip ospf interface serial 0/0
Serial0/0 is up, line protocol is up
  Internet Address 191.1.125.1/24, 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
  Designated Router (ID) 150.1.5.5, Interface address 191.1.125.5
  No backup designated router on this network
  Timer intervals configured, Hello 5, Dead 20, Wait 20, Retransmit 5
    oob-resync timeout 40
    Hello due in 00:00:01
  Supports Link-local Signaling (LLS)
  Index 1/2, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 1
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 150.1.5.5  (Designated Router)
  Suppress hello for 0 neighbor(s)
  Message digest authentication enabled
    Youngest key id is 1
```

```
Rack1R2#show ip ospf interface serial 0/0
Serial0/0 is up, line protocol is up
  Internet Address 191.1.125.2/24, Area 0
  Process ID 1, Router ID 150.1.2.2, 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 5, Dead 20, Wait 20, Retransmit 5
    oob-resync timeout 40
    Hello due in 00:00:00
  Supports Link-local Signaling (LLS)
  Index 2/4, 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)
  Message digest authentication enabled
    Youngest key id is 1
```

Pause for a moment and carefully examine the information provided in the output. We see the following:

1. NON-BROADCAST network type is in use. This means DR must be elected for information synchronization.
2. All interface states are shown as DROTHER. At the same time, there is a discrepancy. R1 shows R5 as the DR, and R2 shows no DR/BDR elected.
3. Notice that Hello/Dead timers have been tuned from their regular values; however, they match among all three routers so it’s not a problem.

It appears to be that R2 has no contact with DR or BDR. However, why can’t R1 and R2 establish an adjacency? If you look at the show command output, you will notice that both routers have their OSPF priorities set to zero. This eliminates the possibility of bringing an adjacency up on a NON-BROADCAST network.

In order to correct the problem, we need to ensure that R1, which is the hub of the Frame-Relay cloud, is configured as the DR on the shared segment, not R5.

```
R1:
interface serial 0/0
 ip ospf priority 1
!
router ospf 1
 neighbor 191.1.125.2
 neighbor 191.1.125.5

R5:
interface serial 0/0/0
 ip ospf priority 0
```

```
Rack1R1#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address
Interface
150.1.2.2         0   FULL/DROTHER    00:00:15    191.1.125.2
Serial0/0
150.1.5.5         0   FULL/DROTHER    00:00:18    191.1.125.5
Serial0/0
```

Now let's see if R5 can reach SW1's prefixes:

```
Rack1R5#show ip route ospf | inc 77
O IA    191.1.177.0/24 [110/66] via 191.1.125.2, 00:00:35, Serial0/0/0
O IA    191.1.77.0/24 [110/66] via 191.1.125.2, 00:00:35, Serial0/0/0
```

```
Rack1R5#traceroute 191.1.177.7

Type escape sequence to abort.
Tracing the route to 191.1.177.7

  1 191.1.125.1 28 msec 28 msec 28 msec
  2 191.1.125.2 64 msec 56 msec 56 msec
  3 191.1.27.7 56 msec *  56 msec
```

Conclusion: NBMA and DR placement might be complicated in partial-mesh topologies. If you are using Hub-and-Spoke, make sure DR is always the hub, not any of the spokes, which might be a result of adjusting priorities. Another thing to look for is the neighbor statements.

Fix the issue

The solution was fixing OSPF priorities and setting correct OSPF neighbors in the hub router:

```
R1:
interface serial 0/0
 ip ospf priority 1
!
router ospf 1
 neighbor 191.1.125.2
 neighbor 191.1.125.5

R5:
interface serial 0/0/0
 ip ospf priority 0
```

Verify

Make sure R5 can reach SW1's prefixes across the Frame-Relay cloud.

`Rack1R5#traceroute 191.1.177.7`

```
Type escape sequence to abort.
Tracing the route to 191.1.177.7

  1 191.1.125.1 28 msec 28 msec 28 msec
  2 191.1.125.2 64 msec 56 msec 56 msec
  3 191.1.27.7 56 msec *  56 msec
```

Ticket 3

Analyze the Symptoms

Most likely the issue is related to BGP misconfiguration. Let’s check the link status real quick. If you look at the BGP diagram, you will see that AS254 has to go across AS 200 and AS 100 to reach AS 54. Let’s start by quickly checking the peering links health:

```
Rack1R3#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 = 1/3/4 ms
```

Let’s see if there is a BGP peering session established:

```
Rack1R3#show ip bgp summary
BGP router identifier 150.1.3.3, local AS number 200
BGP table version is 4, main routing table version 4
3 network entries using 351 bytes of memory
3 path entries using 156 bytes of memory
5/1 BGP path/bestpath attribute entries using 620 bytes of memory
3 BGP AS-PATH entries using 88 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 1215 total bytes of memory
BGP activity 69/66 prefixes, 77/74 paths, scan interval 60 secs

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
192.10.1.254    4   254    1090    1107        4    0    0 00:00:28        3
204.12.1.6      4   100      49      55        4    0    0 00:00:29        0
```

```
Rack1R6#show ip bgp neighbors 204.12.1.254 | inc Estab
  BGP state = Established, up for 00:17:22
```

This rules out any Layer 2 and Layer 3 issues. We may suggest just two potential issues: traffic filtering or BGP misconfiguration.

Initial hypothesis: Traffic filtering and/or BGP misconfiguration.
Problem Scope: R3, R6, BB3 and transit Layer 2 devices (filtering may be applied).

Isolate the Issue

Since we cannot do a traceroute or BGP table lookup in BB2, let's try to see what routes R3 advertises to BB2:

```
Rack1R3#sh ip bgp neighbors 192.10.1.254 advertised-routes
BGP table version is 13, 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
*> 150.1.3.0/24     0.0.0.0                  0         32768 ?
*> 191.1.23.0/24    0.0.0.0                  0         32768 ?
*> 191.1.34.0/24    0.0.0.0                  0         32768 ?
*> 192.10.1.0       0.0.0.0                  0         32768 ?
*> 204.12.1.0       0.0.0.0                  0         32768 ?
*> 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 ?

Total number of prefixes 8
```

There are a number of /24 prefixes advertised locally (Next Hop 0.0.0.0). However, we don't see any prefixes originated in AS54. Let's see if R6 learns them:

```
Rack1R6#show ip bgp regexp _54_
BGP table version is 86, 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
*> 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
```

And if they are advertised to R3:

```
Rack1R6#show ip bgp neighbors 204.12.1.3 advertised-routes
BGP table version is 86, 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
*> 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

Total number of prefixes 8
```

R3 seems to be filtering incoming AS54 updates. Let’s see if there are any inbound filters configured for R6 in R3:

```
Rack1R3#show ip bgp neighbors 204.12.1.6 | beg Inbound
                                 Outbound    Inbound
  Local Policy Denied Prefixes:    --------    -------
    AS_PATH too long:                   n/a          8
    Total:                                0          8
  Number of NLRIs in the update sent: max 6, min 2

  Connections established 6; dropped 5
  Last reset 00:12:11, due to User reset
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: 204.12.1.3, Local port: 46827
Foreign host: 204.12.1.6, Foreign port: 179
```

Notice how we filtered the Inbound policy denials. As you can see, 8 prefixes are filtered due to long AS_PATH. This particular notification is related to just one single command configurable under BGP process – **bgp maxas-limit**. We may disable this limit and see what happens:

```
R3:
router bgp 200
 no bgp maxas-limit
```

Reset BGP peering sessions and see the results:

```
Rack1R3#show ip bgp regexp _54_
BGP table version is 12, 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
*> 112.0.0.0        204.12.1.254                           0 100 300
100 300 54 50 60 i
*> 113.0.0.0        204.12.1.254                           0 100 300
100 300 54 50 60 i
*> 114.0.0.0        204.12.1.254                           0 100 300
100 300 54 i
*> 115.0.0.0        204.12.1.254                           0 100 300
100 300 54 i
*> 116.0.0.0        204.12.1.254                           0 100 300
100 300 54 i
*> 117.0.0.0        204.12.1.254                           0 100 300
100 300 54 i
*> 118.0.0.0        204.12.1.254                           0 100 300
100 300 54 i
*> 119.0.0.0        204.12.1.254                           0 100 300
100 300 54 i
```

This seems to have fixed the issue – R3 now advertises AS54 prefixes to AS254.

```
Rack1R3#show ip bgp neighbors 192.10.1.254 advertised-routes
BGP table version is 17, 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
*> 112.0.0.0        204.12.1.254                           0 100 300 100 300 54
50 60 i
*> 113.0.0.0        204.12.1.254                           0 100 300 100 300 54
50 60 i
*> 114.0.0.0        204.12.1.254                           0 100 300 100 300 54
i
*> 115.0.0.0        204.12.1.254                           0 100 300 100 300 54
i
*> 116.0.0.0        204.12.1.254                           0 100 300 100 300 54
i
*> 117.0.0.0        204.12.1.254                           0 100 300 100 300 54
i
*> 118.0.0.0        204.12.1.254                           0 100 300 100 300 54
i
*> 119.0.0.0        204.12.1.254                           0 100 300 100 300 54
i
*> 150.1.3.0/24     0.0.0.0                  0         32768 ?
   Network          Next Hop            Metric LocPrf Weight Path
*> 191.1.23.0/24    0.0.0.0                  0         32768 ?
*> 191.1.34.0/24    0.0.0.0                  0         32768 ?
*> 192.10.1.0       0.0.0.0                  0         32768 ?
*> 204.12.1.0       0.0.0.0                  0         32768 ?
*> 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 ?
```

Just to make sure everything works, we need to make sure AS100 advertises the prefixes to AS54.

```
Rack1R6#show ip bgp neighbors 204.12.1.254 advertised-
routes
BGP table version is 103, 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
*> 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
*> 150.1.3.0/24     204.12.1.3               0             0 200 ?
*> 191.1.23.0/24    204.12.1.3               0             0 200 ?
*> 191.1.34.0/24    204.12.1.3               0             0 200 ?
*> 192.10.1.0       204.12.1.3               0             0 200 ?
r> 204.12.1.0       204.12.1.3               0             0 200 ?
*> 205.90.31.0      204.12.1.3                             0 200 254 ?
*> 220.20.3.0       204.12.1.3                             0 200 254 ?
*> 222.22.2.0       204.12.1.3                             0 200 254 ?

   Network          Next Hop            Metric LocPrf Weight Path
Total number of prefixes 16
```

Thus both AS254 and 54 can see each other's routes.

Conclusion: Some rare BGP commands may have undesired effects on BGP prefix propagation.

Fix the issue

Since we could not modify R6's configuration, we have to remove AS-path limit set up in R3:

```
R3:
router bgp
 no bgp maxas-limit
```

Verify

We already verified our configuration by making sure AS54 and AS254 prefixes make it through.

Ticket 4

Analyze the Symptoms

IPv6 autoconfiguration uses ICMPv6 messages sent to special link-local multicast addresses on the segment. Aside from misconfiguration, there could be some sort of filtering preventing the multicast messages from reaching their ultimate destination. However, since the ticket does not mention any switches we assume it's some sort of IPv6 misconfiguration. We'll proceed to the issue isolation phase by configuring SW1 as IPv6 host set up for autoconfiguration.

Initial hypothesis: IPv6 misconfiguration
Problem Scope: R5

Isolate the Issue

Set up SW1 as an IPv6 host. You may need to change the SDM profile to accomplish this:

```
SW1:
sdm prefer dual-ipv4-and-ipv6 routing
!
interface Vlan5
 ipv6 address autoconfig
```

Right after this, check the IPv6 address for this interface:

```
Rack1SW1#show ipv6 interface vlan 5
Vlan5 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::211:BBFF:FE0A:C8C2
  No global unicast address is configured
  Joined group address(es):
    FF02::1
    FF02::1:FF0A:C8C2
  MTU is 1500 bytes
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
  Output features: Check hwidb
  ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 30000 milliseconds
```

Now we need to read the IPv6 settings for R5's VLAN5 interface:

```
Rack1R5#show ipv6 interface fastEthernet 0/0
  IPv6 is enabled, link-local address is FE80::213:1AFF:FE68:B04C
  No Virtual link-local address(es):
  Global unicast address(es):
    2001:CC1E:1:5::5, subnet is 2001:CC1E:1:5::/64
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::1:FF00:5
    FF02::1:FF68:B04C
  MTU is 1500 bytes
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
  ICMP unreachables are sent
  ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 30000 milliseconds (using 22917)
  Default router is FE80::215:62FF:FE41:FD9C on FastEthernet0/1.45
```

As we can see the interface is up and running and has a global IPv6 address. Let’s see if we can ping SW1’s link-local address –this will rule out any link problems.

```
Rack1R5#ping FE80::211:BBFF:FE0A:C8C2
Output Interface: FastEthernet0/0
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to FE80::211:BBFF:FE0A:C8C2, timeout is
2 seconds:
Packet sent with a source address of FE80::213:1AFF:FE68:B04C
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 0/4/8 ms
```

Next, let’s see what IPv6 prefixes are being advertised in RAs:

```
Rack1R5#show ipv6 interface fastEthernet 0/0 prefix
IPv6 Prefix Advertisements FastEthernet0/0
Codes: A - Address, P - Prefix-Advertisement, O - Pool
       U - Per-user prefix, D - Default
       N - Not advertised, C - Calendar

PD default [LA] Valid lifetime 2592000, preferred lifetime 604800
AD 2001:CC1E:1:5::/64 [LA] Valid lifetime 2592000, preferred lifetime
604800
```

Which appears to be valid. Now let’s make sure IPv6 RA is not suppressed on the interface:

```
R5:
interface FastEthernet 0/0
 no ipv6 nd ra suppress
```

And shutdown/no-shutdown SW1's VLAN5 interface:

```
Rack1SW1(config)#interface vlan 5
Rack1SW1(config-if)#shutdown
Rack1SW1(config-if)#no shutdown
```

Let's see whether the RA messages are being sent at all: We may want to tune up the RA sending interval to the minimum:

```
Rack1R5#debug ipv6 nd
  ICMP Neighbor Discovery events debugging is on
Rack1R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1R5(config)#interface fastEthernet 0/0
Rack1R5(config-if)#ipv6 nd ra interval 4
```

There is not debugging output for R5's VLAN5 interface. This means R5 is not sending any RAs on this interface. One good reason for this might be disabled IPv6 routing in R5. Keeping debugs enabled, type the following command

```
Rack1R5(config)#ipv6 unicast-routing
Rack1R5(config)#
ICMPv6-ND: Created RA context for FE80::213:1AFF:FE68:B04C
ICMPv6-ND: Created RA context for FE80::213:1AFF:FE68:B04D
ICMPv6-ND: Removed default to FE80::215:62FF:FE41:FD9C on
FastEthernet0/1.45
ICMPv6-ND: Request to send RA for FE80::213:1AFF:FE68:B04C
ICMPv6-ND: Sending RA from FE80::213:1AFF:FE68:B04C to FF02::1 on
FastEthernet0/0
ICMPv6-ND:     MTU = 1500
ICMPv6-ND:     prefix = 2001:CC1E:1:5::/64 onlink autoconfig
ICMPv6-ND:             2592000/604800 (valid/preferred)
```

Enabling IPv6 routing made R5 start sending RAs out of all eligible interfaces. Now check if SW1 has autoconfigured its IPv6 interface:

```
Rack1SW1#show ipv6 interface vlan 5
Vlan5 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::211:BBFF:FE0A:C8C2
  Global unicast address(es):
    2001:CC1E:1:5:211:BBFF:FE0A:C8C2, subnet is 2001:CC1E:1:5::/64
[PRE]
<snip>
```

Conclusion: IPv6 show command output is not always informative. Sometimes it may require you to run through a simple checklist for IPv6 features. Number one thing to do – make sure IPv6 unicast routing is enabled!

Fix the issue

We combine all commands we used during the isolation process into one “solution”:

```
R5:
ipv6 unicast-routing
!
interface FastEthernet 0/0
 no ipv6 nd ra suppress
```

Verify

We already verified our solution during the isolating phase. The key to the verification here is configuring SW1 as an IPv6 host.

Ticket 5

Analyze the Symptoms

Referring to MPLS VPN diagram you may notice that VPN_A spans two routers – R4 and R6, and uses EIGRP for PE-CE routing. Notice that there is a tunnel interface connecting R4 to R6, using VRF-lite style configuration. The ticket states that R4 cannot reach the prefixes behind BB1. There could be a number of issues involved here, including tunnel link failure, EIGRP misconfiguration, prefix filtering, traffic-filtering and so on. We need to do some basic issue isolation prior to forming a hypothesis.

Initial hypothesis: Not defined
Problem Scope: R4, R6; tunnel interface between R4 and R6; Frame-Relay link to BB1.

Isolate the Issue

Let's start by verifying link integrity. Ping BB1 from R6 and ping across the tunnel link between R4 and R6:

`Rack1R6#ping vrf VPN_A 54.1.3.254`

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 54.1.3.254, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/29/32 ms
```

`Rack1R6#ping vrf VPN_A 191.1.46.4`

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 191.1.46.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 68/71/72 ms
```

This rules out all L2 and L3 issues. Now for the control plane – routing protocols.

`Rack1R6#show ip route vrf VPN_A`

```
Routing Table: VPN_A
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

     191.1.0.0/24 is subnetted, 1 subnets
C       191.1.46.0 is directly connected, Tunnel46
     54.0.0.0/24 is subnetted, 1 subnets
C       54.1.3.0 is directly connected, Serial0/0/0
```

```
Rack1R6#show ip eigrp neighbors
IP-EIGRP neighbors for process 10
```

There are no EIGRP adjacencies between the three routers involved in the process. Let’s check if EIGRP has been configured properly for the VPN_A interfaces:

```
Rack1R6#show ip protocols vrf VPN_A
Routing Protocol is "bgp 100"
  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
  Maximum path: 1
  Routing Information Sources:
    Gateway         Distance      Last Update
  Distance: external 20 internal 200 local 200

Routing Protocol is "eigrp"
  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
  EIGRP NSF-aware route hold timer is 240s
  Automatic network summarization is in effect
  Maximum path: 4
  Routing for Networks:
    54.1.3.6/32
    191.1.46.6/32
  Routing Information Sources:
    Gateway         Distance      Last Update
  Distance: internal 90 external 170
```

The networks configured seem to match the interfaces IP addressing. Let’s see EIGRP interfaces:

```
Rack1R6#show ip eigrp vrf VPN_A interfaces tunnel 46
Rack1R6#show ip eigrp vrf VPN_A interfaces serial 0/0/0
```

For some reason EIGRP is not active on the VRF interfaces.

Let’s more to R4 for a while and see the situation there:

```
Rack1R4#show ip protocols vrf VPN_A
Routing Protocol is "bgp 100"
  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
  Maximum path: 1
  Routing Information Sources:
    Gateway         Distance      Last Update
  Distance: external 20 internal 200 local 200

Routing Protocol is "eigrp 10"
  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 10
  EIGRP NSF-aware route hold timer is 240s
  Automatic network summarization is in effect
  Maximum path: 4
  Routing for Networks:
    0.0.0.0
  Routing Information Sources:
    Gateway         Distance      Last Update
  Distance: internal 90 external 170
```

```
Rack1R4#show ip eigrp vrf VPN_A interfaces
IP-EIGRP interfaces for process 10

                        Xmit Queue   Mean   Pacing Time   Multicast
Pending
Interface        Peers  Un/Reliable  SRTT   Un/Reliable   Flow Timer
Routes
Fa0/0.4            0        0/0         0       0/1            0
0
Tu46               0        0/0         0       6/6            0
0
```

As opposed to R6, EIGRP is active on the interfaces now. So what is the difference between the two routers? It is often helpful comparing the two outputs and noticing any mismatches. If you match the output line by line, you will notice that the routing process in R4 has EIGRP AS# configured, while R6 has not.

```
Routing Protocol is "eigrp 10"
Vs
Routing Protocol is "eigrp"
```

As you remember, for multi-VRF EIGRP configuration you need to specify an AS# for every address-family. It seems like this hasn't been done in R6. To find out the global process number use the following command:

```
Rack1R6#show ip protocols | beg eigrp
Routing Protocol is "eigrp 10"
  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 10
  EIGRP NSF-aware route hold timer is 240s
  Automatic network summarization is not in effect
  Maximum path: 4
  Routing for Networks:
  Routing Information Sources:
    Gateway         Distance      Last Update
  Distance: internal 90 external 170
```

Now set up the per-VRF AS#:

```
R6:
router eigrp 10
 address-family ipv4 vrf VPN_A
  autonomous-system 10
```

And see if this has fixed the issue:

```
Rack1R6#show ip eigrp vrf VPN_A neighbors
IP-EIGRP neighbors for process 10
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q
Seq
                                            (sec)         (ms)
Cnt Num
1   54.1.3.254              Se0/0/0           13 00:00:08 1247  5000  0
3
0   191.1.46.4              Tu46              13 00:00:08 1042  5000  0
3
```

Looks like it has helped. Let's see if R4 learns the BB1 EIGRP prefixes now:

Rack1R4#show ip route vrf VPN_A

```
Routing Table: VPN_A
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

D    200.0.0.0/24 [90/27520000] via 191.1.46.6, 00:02:49, Tunnel46
     191.1.0.0/24 is subnetted, 2 subnets
C       191.1.46.0 is directly connected, Tunnel46
C       191.1.4.0 is directly connected, FastEthernet0/0.4
D    54.0.0.0/8 [90/27392000] via 191.1.46.6, 00:02:49, Tunnel46
D    200.0.1.0/24 [90/27520000] via 191.1.46.6, 00:02:49, Tunnel46
D    200.0.2.0/24 [90/27520000] via 191.1.46.6, 00:02:49, Tunnel46
D    200.0.3.0/24 [90/27520000] via 191.1.46.6, 00:02:49, Tunnel46
```

Conclusion: The issue in this scenario was clearly a result of not following the configuration checklist properly. However, sometimes, instead of remembering the checklist it might be helpful to match the show commands from "non-working" and "hope to be working" routers to see the difference and find the clue.

Fix the issue

In order to fix the issue we had to use just a single command:

R6:
```
router eigrp 10
 address-family ipv4 vrf VPN_A
  autonomous-system 10
```

Verify

Just an added verification step – make sure you can ping the EIGRP prefixes behind BB1.

Rack1R4#ping vrf VPN_A 200.0.1.1 source fastEthernet 0/0.4

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 200.0.1.1, timeout is 2 seconds:
Packet sent with a source address of 191.1.4.4
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 100/100/100
ms
```

```
Rack1R4#traceroute vrf VPN_A 200.0.1.1 source fastEthernet
0/0.4

Type escape sequence to abort.
Tracing the route to 200.0.1.1

  1 191.1.46.6 44 msec 40 msec 44 msec
  2 54.1.3.254 56 msec *  64 msec
```

Ticket 6

Analyze the Symptoms

If you look at the MPLS VPN diagram you will see that SW3 and SW4 form a dual-home site and use OSPF as the routing protocol. Notice that there are two separate OSPF processes – the MPLS VPN core one, used by SP (OSPF Area 45) and the CE routing protocol (OSPF area 90). There may be quite a few issues preventing the MPLS core from being used as a primary path:

1. Core MPLS LSP breakdown (prevents data-plane connectivity)
2. Improper route propagation through MP-BGP (wrong import statement, no redistribution)
3. No sham-link configured or the sham-link is not working. A sham-link is required is a single area is deployed between the CE sites
4. Sham link is configured but OSPF costs are not set properly

Initial hypothesis: Try all four hypotheses in sequence, which loosely reflects the “bottom-up” approach.
Problem Scope: R4, R5, SW3 and SW4.

Isolate the Issue

Let’s see if MPLS forwarding in the core is working properly. First, check if LDP is used for label exchange:

```
Rack1R4#show mpls interfaces
Interface              IP            Tunnel   BGP Static Operational
FastEthernet0/0.45     Yes (tdp)     No       No  No     Yes
```

```
Rack1R4#show mpls ldp neighbor

Rack1R4#
```

```
Rack1R5#show mpls interfaces
Interface              IP            Tunnel   BGP Static Operational
FastEthernet0/1.45     Yes (ldp)     No       No  No     Yes
```

```
Rack1R5#show mpls ldp neighbor

Rack1R5#
```

At least we can quickly spot a problem here R4 uses TDP while R5 uses LDP for label exchange. Since LDP is not the default protocol, we may want to configure R4 to use the same protocol as R5:

```
R4:
mpls label protocol ldp
```

```
Rack1R4#show mpls ldp neighbor
    Peer LDP Ident: 150.1.5.5:0; Local LDP Ident 150.1.4.4:0
        TCP connection: 150.1.5.5.18035 - 150.1.4.4.646
        State: Oper; Msgs sent/rcvd: 21/21; Downstream
        Up time: 00:00:02
        LDP discovery sources:
          FastEthernet0/0.45, Src IP addr: 191.1.45.5
        Addresses bound to peer LDP Ident:
          191.1.45.5      150.1.5.5       191.1.125.5     191.1.5.5
```

Now check the MPLS forwarding tables at both routers:

```
Rack1R4#sh mpls forwarding-table
Local  Outgoing      Prefix            Bytes Label   Outgoing   Next Hop
Label  Label or VC   or Tunnel Id      Switched      interface
16     Pop Label     191.1.125.0/24    0             Fa0/0.45   191.1.45.5
17     18            191.1.27.0/24     0             Fa0/0.45   191.1.45.5
18     19            191.1.23.0/24     0             Fa0/0.45   191.1.45.5
19     No Label      150.1.3.0/24      0             Se0/0/0    191.1.34.3
20     23            150.1.2.0/24      0             Fa0/0.45   191.1.45.5
21     No Label      150.1.8.8/32      0             Fa0/1      191.1.48.8
22     Pop Label     150.1.5.5/32      0             Fa0/0.45   191.1.45.5
23     Pop Label     191.1.5.0/24      0             Fa0/0.45   191.1.45.5
24     No Label      191.1.49.0/24[V]  0             aggregate/VPN_B
25     26            150.1.1.0/24      0             Fa0/0.45   191.1.45.5
26     27            191.1.177.0/24    0             Fa0/0.45   191.1.45.5
27     28            191.1.77.0/24     0             Fa0/0.45   191.1.45.5
28     29            191.1.7.0/24      0             Fa0/0.45   191.1.45.5
29     No Label      150.1.6.0/24      0             Se0/0/0    191.1.34.3
```

```
Rack1R5#show mpls forwarding-table
Local  Outgoing      Prefix            Bytes Label   Outgoing   Next Hop
Label  Label or VC   or Tunnel Id      Switched      interface
16     Pop Label     191.1.48.0/24     0             Fa0/1.45   191.1.45.4
17     Pop Label     191.1.34.0/24     0             Fa0/1.45   191.1.45.4
18     No Label      191.1.27.0/24     0             Se0/0/0    191.1.125.2
19     No Label      191.1.23.0/24     0             Se0/0/0    191.1.125.2
20     Pop Label     150.1.4.4/32      0             Fa0/1.45   191.1.45.4
21     21            150.1.8.8/32      0             Fa0/1.45   191.1.45.4
22     19            150.1.3.0/24      0             Fa0/1.45   191.1.45.4
<snip>
```

We need to make sure that the Loopback prefixes used for MP-BGP peering are tagged (in our case, using implicit null labeling).

```
Rack1R4#show bgp vpnv4 unicast all summary
BGP router identifier 150.1.4.4, local AS number 100
<snip>

Neighbor        V          AS MsgRcvd MsgSent   TblVer  InQ OutQ
Up/Down  State/PfxRcd
150.1.5.5       4        100    1353    1346       11    0    0
22:32:58        4
```

```
Rack1R5#show bgp vpnv4 unicast all summary
BGP router identifier 150.1.5.5, local AS number 100
<snip>

Neighbor        V          AS MsgRcvd MsgSent   TblVer  InQ OutQ
Up/Down  State/PfxRcd
150.1.4.4       4        100    1345    1352       11    0    0
22:32:17        1
```

Look back into the MPLS forwarding tables and notice that both peering IP addresses are MPLS labeled. This ensures the data-plane connectivity. Now, proceeding to the next section of our checklist – ensuring the CE routes are propagated via MP-BGP.

```
Rack1R4#show bgp vpnv4 unicast vrf VPN_B
BGP table version is 19, 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
Route Distinguisher: 100:2 (default for vrf VPN_B)
*> 150.1.9.9/32     191.1.49.9               2         32768 ?
*>i150.1.10.10/32   150.1.5.5                2    100      0 ?
*>i150.1.55.55/32   150.1.5.5                0    100      0 i
*> 191.1.49.0/24    0.0.0.0                  0         32768 ?
*>i191.1.50.0/24    150.1.5.5                0    100      0 ?
* i191.1.109.0/24   150.1.5.5            10000    100      0 ?
*>                  191.1.49.9           10000         32768 ?
```

```
Rack1R5#show bgp vpnv4 unicast vrf VPN_B
BGP table version is 13, 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
Route Distinguisher: 100:2 (default for vrf VPN_B)
*>i150.1.9.9/32     150.1.4.4                2    100      0 ?
*> 150.1.10.10/32   191.1.50.10              2         32768 ?
*> 150.1.55.55/32   0.0.0.0                  0         32768 i
*>i191.1.49.0/24    150.1.4.4                0    100      0 ?
*> 191.1.50.0/24    0.0.0.0                  0         32768 ?
* i191.1.109.0/24   150.1.4.4            10000    100      0 ?
*>                  191.1.50.10          10000         32768 ?
```

You can see each CE router's Loopback being propagated across MP-BGP here. Let's see if they reach the PE-routers VRF tables:

```
Rack1R4#show ip route vrf VPN_B

Routing Table: VPN_B
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

     191.1.0.0/24 is subnetted, 3 subnets
O       191.1.50.0 [110/10001] via 191.1.49.9, 00:03:05,
FastEthernet0/0.49
C       191.1.49.0 is directly connected, FastEthernet0/0.49
O       191.1.109.0 [110/10000] via 191.1.49.9, 00:03:15,
FastEthernet0/0.49
     150.1.0.0/32 is subnetted, 4 subnets
B       150.1.55.55 [200/0] via 150.1.5.5, 22:40:03
C       150.1.44.44 is directly connected, Loopback1
O       150.1.10.10 [110/10001] via 191.1.49.9, 00:03:05,
FastEthernet0/0.49
O       150.1.9.9 [110/2] via 191.1.49.9, 00:03:58, FastEthernet0/0.49
```

Rack1R5#show ip route vrf VPN_B

```
Routing Table: VPN_B
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

     191.1.0.0/24 is subnetted, 3 subnets
C       191.1.50.0 is directly connected, FastEthernet0/1.50
O       191.1.49.0 [110/10001] via 191.1.50.10, 00:03:43,
FastEthernet0/1.50
O       191.1.109.0 [110/10000] via 191.1.50.10, 00:03:53,
FastEthernet0/1.50
     150.1.0.0/32 is subnetted, 3 subnets
C       150.1.55.55 is directly connected, Loopback1
O       150.1.10.10 [110/2] via 191.1.50.10, 00:03:53,
FastEthernet0/1.50
O       150.1.9.9 [110/10001] via 191.1.50.10, 00:03:43,
FastEthernet0/1.50
```

Notice that both PE routers prefer reaching the CE prefixes across the PE-CE links, not the MPLS core. Since OSPF routes are being propagated into MPBGP, the only problem that may still remain is non-configured or broken sham-link. Check if we have any configured:

Rack1R5#show ip ospf sham-links
```
Sham Link OSPF_SL0 to address 150.1.44.44 is down
Area 90 source address 150.1.55.55
  Run as demand circuit
  DoNotAge LSA allowed. Cost of using 1 State DOWN,
  Timer intervals configured, Hello 10, Dead 40, Wait 40,
```

OK, let’s check if the sham-link endpoints are reachable:

Rack1R5#ping vrf VPN_B 150.1.44.44

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.44.44, timeout is 2 seconds:
 ...
Success rate is 0 percent (0/5)
```

This gives us a clue that the other endpoint is not being advertised into BGP:

```
Rack1R5#show ip bgp vpnv4 vrf VPN_B 150.1.44.44
% Network not in table
```

Let’s get back to R4 and advertise this subnet:

```
Rack1R4#sh ip route vrf VPN_B 150.1.44.44
Routing entry for 150.1.44.44/32
  Known via "connected", distance 0, metric 0 (connected, via
interface)
  Routing Descriptor Blocks:
  * directly connected, via Loopback1
      Route metric is 0, traffic share count is 1
```

```
R4:
router bgp 100
 address-family ipv4 vrf VPN_B
   network 150.1.44.44 mask 255.255.255.255
```

At this point, we may see that the sham-link is up and running:

```
Rack1R4#show ip ospf sham-links
Sham Link OSPF_SL0 to address 150.1.55.55 is up
Area 90 source address 150.1.44.44
  Run as demand circuit
  DoNotAge LSA allowed. Cost of using 1 State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40,
    Hello due in 00:00:05
    Adjacency State FULL (Hello suppressed)
    Index 2/2, retransmission queue length 1, number of retransmission
0
    First 0x674FFC84(20)/0x0(0) Next 0x674FFC84(20)/0x0(0)
    Last retransmission scan length is 0, maximum is 0
    Last retransmission scan time is 0 msec, maximum is 0 msec
    Link State retransmission due in 3636 msec
```

This exhausts our checklist and we may go to the CE routers to see their routing tables:

```
Rack1SW3#show ip route vrf VPN_B

Routing Table: VPN_B
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

     191.1.0.0/24 is subnetted, 3 subnets
O       191.1.50.0 [110/3] via 191.1.49.4, 00:01:36, Vlan49
C       191.1.49.0 is directly connected, Vlan49
C       191.1.109.0 is directly connected, Vlan109
     150.1.0.0/16 is variably subnetted, 4 subnets, 2 masks
O E2    150.1.55.55/32 [110/1] via 191.1.49.4, 00:01:36, Vlan49
O E2    150.1.44.44/32 [110/1] via 191.1.49.4, 00:01:36, Vlan49
O       150.1.10.10/32 [110/4] via 191.1.49.4, 00:01:36, Vlan49
C       150.1.9.0/24 is directly connected, Loopback0
Rack1SW3#
```

Rack1SW4#show ip route vrf VPN_B

```
Routing Table: VPN_B
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

     191.1.0.0/24 is subnetted, 4 subnets
C       191.1.50.0 is directly connected, Vlan50
O       191.1.49.0 [110/3] via 191.1.50.5, 00:01:49, Vlan50
C       191.1.10.0 is directly connected, Vlan10
C       191.1.109.0 is directly connected, Vlan109
     150.1.0.0/16 is variably subnetted, 4 subnets, 2 masks
O E2    150.1.55.55/32 [110/1] via 191.1.50.5, 00:01:49, Vlan50
O E2    150.1.44.44/32 [110/1] via 191.1.50.5, 00:01:49, Vlan50
O       150.1.9.9/32 [110/4] via 191.1.50.5, 00:01:49, Vlan50
C       150.1.10.0/24 is directly connected, Loopback0
```

Now both CE routes prefer the paths across the PEs.

Conclusion: MPLS VPNs are built around hierarchical concepts: LSPs, PE-CE routing, MP-BGP etc. It most cases you may quickly write out a checklist and follow it in bottom-up manner, which ensures that both control and data planes are fully functional. With MPLS VPNs, data and control plane may work somewhat independently – a working control plane does not guarantee working data plane.

Fix the issue

In order to fix the issue we used two commands: the first one to configure uniform LDP between the two routers and the second one to fix the sham-link:

```
R4:
mpls lable protocol ldp
!
router bgp 100
 address-family ipv4 vrf VPN_B
  network 150.1.44.44 mask 255.255.255.255
```

Verify

We already verified the control plane in the CE routers. Now check the data plane by pinging and tracing the routes between the CE routers' Loopback interfaces:

```
Rack1SW4#traceroute vrf VPN_B 150.1.9.9

Type escape sequence to abort.
Tracing the route to 150.1.9.9

  1 191.1.50.5 0 msec 0 msec 0 msec
  2 191.1.49.4 0 msec 0 msec 9 msec
  3 191.1.49.9 0 msec *  0 msec
Rack1SW4#
```

```
Rack1SW3#traceroute vrf VPN_B 150.1.10.10

Type escape sequence to abort.
Tracing the route to 150.1.10.10

  1 191.1.49.4 0 msec 0 msec 9 msec
  2 191.1.50.5 0 msec 0 msec 0 msec
  3 191.1.50.10 8 msec *  0 msec
```

```
Rack1SW3#ping vrf VPN_B 150.1.10.10 source loopback 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.10.10, timeout is 2 seconds:
Packet sent with a source address of 150.1.9.9
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/9 ms
```

Ticket 7

Analyze the Symptoms

We know that R5 and R1 are configured for stub multicast routing. What this means is that even though PIM should be enabled between the two routers, R1 should ignore any PIM messages coming from R5, and R5 should not participate in building any multicast SPTs. In fact, R5 acts as a dumb multicast relay, which forwards IGMP messages from the directly attached clients to the upstream multicast router. We may suspect some components for stub multicasting to be improperly configured.

The scenario mentions R3, which is the source of multicast feeds. PIM SM is configured between R1 and R3, with R3 being the RP (statically configured). Per stub multicast routing design, R5 should be configured using PIM dense mode to flood all multicast flows coming through.

Initial hypothesis: PIM/IGMP misconfiguration or lower-level (L2, L3) failure.
Problem Scope: R1, R3, R5 the Frame-Relay link between them. R5's VLAN5 interface.

Isolate the Issue

We know the link between R1 and R5 should be healthy, as we've been troubleshooting an OSPF issue across it before. Therefore, we may start with checking PIM settings:

```
Rack1R5#show ip pim interface

Address          Interface                Ver/   Nbr    Query  DR
DR
                                          Mode   Count  Intvl  Prior
191.1.125.5      Serial0/0/0              v2/D   1      30     1
191.1.125.5
191.1.5.5        FastEthernet0/0          v2/D   0      30     1
191.1.5.5
```

```
Rack1R5#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
191.1.125.1       Serial0/0/0              00:19:05/00:01:25 v2    1 /
S
```

```
Rack1R1#show ip pim interface

Address          Interface                Ver/   Nbr    Query  DR
DR
                                          Mode   Count  Intvl  Prior
191.1.125.1      Serial0/0                v2/S   0      30     100
191.1.125.1
191.1.13.1       Serial0/1                v2/S   1      30     1
0.0.0.0
```

```
Rack1R1#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
191.1.13.3        Serial0/1                03:39:31/00:01:30 v2    1 /
S
```

Everything looks good at R5, but R1 does not see R5 as a PIM neighbor. However, this is expected behavior, due to the PIM neighbor-filter configured – R1 should ignore PIM messages from R5, and enabling PIM on the downstream interface is needed for multicast forwarding only.

```
Rack1R1#show ip pim interface serial 0/0 detail
Serial0/0 is up, line protocol is up
  Internet address is 191.1.125.1/24
  Multicast switching: fast
  Multicast packets in/out: 0/0
  Multicast TTL threshold: 0
  PIM: enabled
    PIM version: 2, mode: dense
    PIM DR: 191.1.125.1 (this system)
    PIM neighbor count: 0
    PIM Hello/Query interval: 30 seconds
    PIM Hello packets in/out: 0/49
    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
```

```
Rack1R1#show ip access-lists 1
Standard IP access list 1
    10 deny   191.1.125.5 (50 matches)
    20 permit any
```

Now, let’s emulate a multicast receiver with SW1 and try sending a multicast feed from R3:

```
SW1:
interface Vlan5
 ip address 191.1.5.7 255.255.255.0
 ip igmp join-group 239.1.1.1
 ip pim dense-mode
!
ip mroute 0.0.0.0 0.0.0.0 191.1.5.5
```

Rack1R3#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:
...
```

Let’s trace the multicast distribution tree back from the receiver. Start with R5 – we should see the client joining the multicast group:

Rack1R5#show ip igmp groups

```
IGMP Connected Group Membership
Group Address    Interface                Uptime    Expires   Last
Reporter   Group Accounted
239.1.1.1        FastEthernet0/0          00:01:31  00:01:28  191.1.5.7
224.0.1.40       Serial0/0/0              04:30:11  00:02:36
191.1.125.5
```

Check the multicast routing states:

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,
       V - RD & Vector, v - Vector
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), 00:32:08/00:02:52, RP 0.0.0.0, flags: DC
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Dense, 00:32:08/00:00:00
    Serial0/0/0, Forward/Dense, 00:32:08/00:00:00
```

It appears no packets are coming to R5, as the incoming interface is set to Null. Let’s check the multicast states in R1:

```
Rack1R1#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), 00:01:54/00:02:51, RP 150.1.3.3, flags: SP
  Incoming interface: Serial0/1, RPF nbr 191.1.13.3
  Outgoing interface list: Null
```

There is only (*,G) state in this router, which means R1 does not see the tree being joined from downstream. With stub multicast routing this means IGMP messages have not been forwarded to R1. Check active IGMP groups to make this clear:

```
Rack1R1#show ip igmp groups
IGMP Connected Group Membership
Group Address    Interface                Uptime    Expires   Last
Reporter   Group Accounted
224.0.1.40       Serial0/0                21:02:58  00:01:59
191.1.125.1
```

Most likely there is a problem with IGMP helper in R5. The helper address should be set on the VLAN5 interface. Verify this:

```
Rack1R5#show ip igmp interface fastEthernet 0/0
FastEthernet0/0 is up, line protocol is up
  Internet address is 191.1.5.5/24
  IGMP is enabled on interface
  Current IGMP host version is 2
  Current IGMP router version is 2
  IGMP query interval is 60 seconds
  IGMP querier timeout is 120 seconds
  IGMP max query response time is 10 seconds
  Last member query count is 2
  Last member query response interval is 1000 ms
  Inbound IGMP access group is not set
  IGMP activity: 8 joins, 7 leaves
  Multicast routing is enabled on interface
  Multicast TTL threshold is 0
  Multicast designated router (DR) is 191.1.5.7
  IGMP querying router is 191.1.5.5 (this system)
  IGMP helper address is 191.1.152.1
  No multicast groups joined by this system
```

As we can see, the IGMP helper point to a wrong system. Fix this:

```
R5:
interface FastEthernet 0/0
 ip igmp helper-address 191.1.125.1
```

Now get back to R3 to find out the pings are still not working. Check the multicast routing states in R5 once again:

```
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,
       V - RD & Vector, v - Vector
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), 00:38:20/stopped, RP 0.0.0.0, flags: DC
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Dense, 00:38:20/00:00:00
    Serial0/0/0, Forward/Dense, 00:38:20/00:00:00

(150.1.3.3, 239.1.1.1), 00:00:18/00:02:41, flags:
  Incoming interface: Null, RPF nbr 191.1.45.4
  Outgoing interface list:
    FastEthernet0/0, Forward/Dense, 00:00:18/00:00:00
    Serial0/0/0, Forward/Dense, 00:00:18/00:00:00
```

As you can see, now there is an (S,G) state for our feed. However, the RPF interface points to R4! This means R3's Loopback0 is not reachable via the multicast uplink to R1. The scenario allows the use of a single mroute, which we use to fix the problem:

```
R5:
ip mroute 150.1.3.3 255.255.255.255 191.1.125.1
```

Now this last step fixes the problem – R3 can now ping the group:

```
Reply to request 268 from 191.1.5.7, 68 ms
Reply to request 269 from 191.1.5.7, 68 ms
Reply to request 270 from 191.1.5.7, 72 ms
Reply to request 271 from 191.1.5.7, 69 ms
Reply to request 272 from 191.1.5.7, 68 ms
```

Conclusion: Troubleshooting multicast issues includes tracing the multicast distribution tree from the receiver to the source. Check for RPF failures and multicast routing states. Almost all the time, troubleshooting the multicast problem includes simulating "live" multicast feeds.

Fix the issue

There were two issues: one that prevented IGMP messages from being forwarded to R1 and another causing multicast RPF failure for the packets sent off R3's Loopback0.

Verify

We already verified the configuration by sourcing packet stream off R3 and receiving it at SW1 emulating a host on VLAN5. The key to multicast troubleshooting is live simulation.

Ticket 8

Analyze the Symptoms

VoIP QoS problems are usually caused by any of the following:

1. Interface oversubscription, shaping not properly configured
2. LLQ not configured for VoIP traffic
3. VoIP traffic not matching LLQ
4. Fragmentation not configured on slow WAN link

We are going to check all these possibilities in sequence. Notice that a preferred method in real life would be using IP SLA and traffic simulation, but due to the rack hardware restrictions this might not be an optimal solution.

Initial hypothesis: QoS misconfiguration in R3/R4.
Problem Scope: R3, R4 and the Frame-Relay link between them.

Isolate the Issue

Let's start by verifying QoS policies applied to the Frame-Relay interfaces in R3 and R4. In our case, shaping is not necessary, as CIR equals the physical interface rates.

Rack1R3#show policy-map interface serial 1/0

```
 Serial1/0

  Service-policy output: QOS

    Class-map: RTP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: protocol rtp
      Queueing
        Output Queue: Conversation 41
        Bandwidth 25 (%)
        Bandwidth 192 (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)
      83828 packets, 5574880 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
Rack1R3#
```

Rack1R4#show policy-map interface serial 0/0/0

```
 Serial0/0/0

  Service-policy output: QOS

    queue stats for all priority classes:

      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 0/0

    Class-map: RTP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: protocol rtp
      Priority: 25% (192 kbps), burst bytes 4800, b/w exceed drops: 0


    Class-map: class-default (match-any)
      29 packets, 10251 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any

      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 29/2171
```

As we can see, both policies have classes that match RTP protocol. However, we can see a mismatch here: R4's class is configured for "bandwidth" not "priority". This effectively disables LLQ behavior for the mentioned class. Obviously, this has to be fixed:

R3:
```
policy-map QOS
 class RTP
  no bandwidth
  priority percent 25
```

If you recall properly, we've seen similar problem back in Lab 1 of the workbook. Never hurts to redo the similar scenario, as this verifies how good you learned the previous lesson. OK, so far we've went through sections 1-3 of our checklist. The last thing to see is whether fragmentation has been enabled. Since the interface rates are set to 768K, fragmentation is mandatory. The recommended fragment size is:

```
FRAG_SIZE = 768000/8*10ms=960 bytes
```

Check fragmentation settings:

```
Rack1R3#show frame-relay fragment
interface                dlci frag-type  size in-frag    out-frag
dropped-frag
```

```
Rack1R4#show frame-relay fragment
interface                dlci frag-type  size in-frag    out-frag
dropped-frag
```

This means we need to enable it. There are two ways to enable FRF – using the MQC-compatible interface-level command **`frame-relay fragment end-to-end`** or legacy **`map-class`** command. In our case, MQC configuration is in use, and thus we apply the following commands:

```
R3:
interface Serial 1/0
 frame-relay fragment 960 end-to-end
```

```
R4:
interface Serial 0/0/0
 frame-relay fragment 960 end-to-end
```

Now one thing to remember here – when you enable interface-level fragmentation, it applies to all PVCs. Therefore, you need to make sure all routers attached to the same cloud also have frame-relay fragmentation configured. In our scenario, there are only two routers attached to the Frame-Relay link, so our new configuration will not cause any problems.

Conclusion: Creating checklists for every typical scenario (e.g. OSPF, QoS, BGP peering) is one of the core components to proper troubleshooting.

Fix the issue

It took two commands to fix the QoS problem (per out checklist):

```
R3:
policy-map QOS
 class RTP
  no bandwidth
  priority percent 25
```

```
R3:
interface Serial 1/0
 frame-relay fragment 960 end-to-end
```

```
R4:
interface Serial 0/0/0
 frame-relay fragment 960 end-to-end
```

Verify

Check the policy-maps output and frame-relay fragmentation status:

```
Rack1R3#show frame-relay fragment
interface                 dlci frag-type  size in-frag    out-frag
dropped-frag
Se1/0                     301  end-to-end 960  0          0          0
Se1/0                     302  end-to-end 960  0          0          0
Se1/0                     304  end-to-end 960  0          0          0
Se1/0                     305  end-to-end 960  0          0          0
Rack1R3#

Rack1R4#show frame-relay fragment
interface                 dlci frag-type  size in-frag    out-frag
dropped-frag
Se0/0/0                   401  end-to-end 960  0          0          0
Se0/0/0                   402  end-to-end 960  0          0          0
Se0/0/0                   403  end-to-end 960  0          0          0
Se0/0/0                   405  end-to-end 960  0          0          0
Se0/0/0                   413  end-to-end 960  0          0          0

Rack1R4#show policy-map interface serial 0/0/0

 Serial0/0/0

  Service-policy output: QOS

    queue stats for all priority classes:

      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 0/0

    Class-map: RTP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: protocol rtp
      Priority: 25% (192 kbps), burst bytes 4800, b/w exceed drops: 0


    Class-map: class-default (match-any)
      566 packets, 254386 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any

      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 566/29634
Rack1R4#
```

Rack1R3#show policy-map interface serial 1/0

```
 Serial1/0

  Service-policy output: QOS

    Class-map: RTP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: protocol rtp
      Queueing
        Strict Priority
        Output Queue: Conversation 264
        Bandwidth 25 (%)
        Bandwidth 192 (kbps) Burst 4800 (Bytes)
        (pkts matched/bytes matched) 0/0
        (total drops/bytes drops) 0/0

    Class-map: class-default (match-any)
      84917 packets, 5650718 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
```

Ticket 9

Analyze the Symptoms

There could be two general problems causing this situation: SSH protocol misconfiguration or access-control restrictions. In the first case, the problem is limited to R6. In the second case, any router on the path to R6 could filter traffic. However, the task says to use R5 as a verification agent, so we're limited only to the paths between R5 and R6.

Initial hypothesis: SSH misconfiguration or access-control rules.
Problem Scope: R6, paths between R5 and R6.

Isolate the Issue

Let's see if SSH2 has been configured in R6:

```
Rack1R6#show ip ssh
SSH Enabled - version 1.5
Authentication timeout: 120 secs; Authentication retries: 3
Minimum expected Diffie Hellman key size : 1024 bits
```

As you can see, only version 1.5 has been enabled. In order to enable SSH2, a key length of 768 bits is required. Let's re-regenerate the key and try testing connectivity:

```
R6:
crypto key generate rsa modulus 768
```

We can run a test off R5 quickly:

```
Rack1R5#ssh -l cisco -v 2 150.1.6.6

[Connection to 150.1.6.6 closed by foreign host]
```

What now? Let's see if we can get some debugging output in R5:

```
Rack1R5#debug ip ssh
Incoming SSH debugging is on
```

And repeat our connection attempt from R5:

```
Rack1R5#
%SSH: Could not get a vty line for incoming session
```

Well this isn't too helpful, but at least this points us towards checking our VTY lines settings.

```
Rack1R6#show line vty 0
   Tty Line Typ     Tx/Rx    A Modem  Roty AccO AccI  Uses  Noise
Overruns  Int
    514  514 VTY              -    -      -    -    -     0      0
0/0      -

Line 514, Location: "", Type: ""
Length: 24 lines, Width: 80 columns
Baud rate (TX/RX) is 9600/9600
Status: No Exit Banner
Capabilities: none
Modem state: Idle
Group codes:    0
Special Chars: Escape  Hold  Stop  Start  Disconnect  Activation
                ^^x    none   -     -       none
Timeouts:      Idle EXEC    Idle Session   Modem Answer  Session
Dispatch
               00:10:00        never                        none
not set
                            Idle Session Disconnect Warning
                              never
                            Login-sequence User Response
                             00:00:30
                            Autoselect Initial Wait
                              not set
Modem type is unknown.
Session limit is not set.
Time since activation: never
Editing is enabled.
History is enabled, history size is 20.
DNS resolution in show commands is enabled
Full user help is disabled
Allowed input transports are telnet.
Allowed output transports are lat pad telnet rlogin lapb-ta mop v120
ssh.
Preferred transport is lat.
No output characters are padded
No special data dispatching characters
```

If you look at the output carefully, you may notice that the only input transport allowed is telnet. This prevents SSH from establishing any incoming connection with this router. Fix this issue:

```
R6:
line vty 0 4
 transport input ssh telnet

And try connecting once again:

Rack1R5#ssh -l cisco -v 2 150.1.6.6

Password:

[Connection to 150.1.6.6 closed by foreign host]
```

The same debugging output reveals:

```
Rack1R6#
SSH2 0: MAC compared for #9 :ok
SSH2 0: input: padlength 9 bytes
SSH2 0: authentication failed for userid (code=1)
```

It seems to be an authentication issue. Let's see if we can login using the same credentials via telnet:

```
Rack1R5#telnet 150.1.6.6
Trying 150.1.6.6 ... Open


User Access Verification

Password: cisco
Rack1R6>
```

This gives us a hint: password authentication is configured for the line, not local, which is required for SSH. Fix this:

```
R6:
line vty 0 4
 login local
```

And now it seems to be working:

```
Rack1R5#ssh -l cisco -v 2 150.1.6.6

Password:

Rack1R6>
```

Conclusion: In this scenario, it would be very helpful going through the "SSH verification" checklist; i.e. generating an RSA key, configuring local authentication and enabling SSH as transport. Remember, keeping in mind configuration steps for a given feature greatly assists in troubleshooting.

Fix the issue

We had to re-generate the RSA key, enable SSH as the input transport and enable local authentication for VTY lines.

R6:
```
crypto key generate rsa modulus 768
!
line vty 0 4
 login local
 transport input ssh telnet
```

Verify

We already verified the solution by connecting to R6 from R5 using SSHv2.

Ticket 10

Analyze the Symptoms

The most probable issue is some misconfiguration in R2. The best way to troubleshoot the scenario is by emulating the dynamic acl behavior and seeing what goes wrong. Prior to this we need to gather some more information on the dynamic access-list configuration.

Initial hypothesis: dynamic access-list misconfiguration in R2
Problem Scope: R2

Isolate the Issue

Let's check the access-list applied to R2's VLAN27 interface:

```
Rack1R2#show ip interface fastEthernet 0/0 | inc access
Rack1R2#show ip interface fastEthernet 0/0 | inc access
  Outgoing access list is not set
  Inbound  access list is VLAN27_IN
  IP access violation accounting is disabled
```

```
Rack1R2#show ip access-list VLAN27_IN
Extended IP access list VLAN27_IN
    10 Dynamic dynacl permit tcp any host 191.1.32.100 eq www
    20 deny ip any host 191.1.32.100
    30 permit ip any any (1 match)
```

The access-list seems to be in order. Now let's connect to R2 from SW1 and try triggering the dynamic access-list.

```
Rack1SW1#telnet 150.1.2.2
Trying 150.1.2.2 ... Open


User Access Verification

Username: dynacl
Password: cisco

Rack1R2>
```

Instead of triggering the dynamic access entry the router logs the user in. Just for the sake of clarity, check the access-list again and notice there is no dynamic entry created.

```
Rack1R2#show ip access-list VLAN27_IN
Extended IP access list VLAN27_IN
    10 Dynamic dynacl permit tcp any host 191.1.32.100 eq www
    20 deny ip any host 191.1.32.100
    30 permit ip any any (117 matches)
```

The first thing that comes to mind – there is not autocommand associated with the username "dynacl":

```
Rack1R2#sh running-config | inc dynacl
username dynacl password 0 cisco
 dynamic dynacl permit tcp any host 191.1.32.100 eq ww
```

Indeed, there is no auto-command associated with this user. Fix this issue, remember to use the "host" keyword to create individual entries (the ticket mentions multiple users using the same credentials).

```
R2:
username dynacl autocommand access-enable host
```

However, logging to R2 once again yields no result – we are still allowed to login, and the command is not triggered.

```
Rack1SW1#telnet 150.1.2.2
Trying 150.1.2.2 ... Open


User Access Verification

Username: dynacl
Password: cisco

Rack1R2>
```

Enable AAA subsystem debugging in R2 and try logging in once again. The reason for AAA debugging is that all shell commands are authorized prior to being allowed or executed. This holds true for all commands, be they entered via CLI or triggered via auto-commands.

```
Rack1R2#debug aaa authentication
AAA Authentication debugging is on

Rack1R2#debug aaa authorization
AAA Authorization debugging is on

AAA/BIND(00000007): Bind i/f
AAA/AUTHEN/LOGIN (00000007): Pick method list 'default'
AAA/AUTHOR (00000007): Method list id=0 not configured. Skip author
```

What this output says is that default authorization list is not configured. Hence, the auto-command is not authorized or allowed. This is the direct consequence of the default AAA behavior, which disallows auto-commands authorization. Enable local exec authorization to fix this issue:

```
R2:
aaa authorization exec default local
```

```
Rack1SW1#telnet 150.1.2.2
Trying 150.1.2.2 ... Open


User Access Verification

Username: dynacl
Password: cisco

[Connection to 150.1.2.2 closed by foreign host]
Rack1SW1#
```

And verify everything once again:

```
Rack1R2#show ip access-lists VLAN27_IN
Extended IP access list VLAN27_IN
    10 Dynamic dynacl permit tcp any host 191.1.32.100 eq www
       permit tcp host 191.1.27.7 host 191.1.32.100 eq www
    20 deny ip any host 191.1.32.100
    30 permit ip any any (1159 matches)
```

Verify

We already verified the solution by emulating dynamic access-list functionality.

Lab 6 Solutions

Ticket 1

The key to resolving this ticket is finding that SW3 and SW4 are connected using SW1 for VLAN tunneling. Both SW3 and SW4 are set to use LACP for channel negotiation. Both use two links for channeling. However, the problem is that in SW1 three different ports are assigned in the same VLAN 100.

Per Etherchannel tunneling rules, you need to have a single separate VLAN for every pair or “opposing” channel links. Essentially, every VLAN is used to emulate a point-to-point link. In this scenario, two VLANs are used: VLAN 100 and VLAN101. To fix the problem, configure the ports 17 and 20 in VLAN 100 and ports 18 and 21 in VLAN 101. This will ensure no channel member receives superfluous LACP messages, which prevents proper channel negotiation.

```
SW1:
interface range FastEthernet0/17, Fa0/20
 switchport access vlan 100
!
interface range FastEthernet0/18, Fa0/21
 switchport access vlan 101
```

There is another problem, still. Even though you recover EIGRP peering between SW3 and SW4, R3 will prefer using the path to VLAN 73 via RIP, since EIGRP external routes have the admin distance of 170. To fix that, change RIP’s distance for VLAN73 prefix to be higher than EIGRP’s:

```
R3:
router rip
 distance 171 0.0.0.0 255.255.255.255 VLAN73
!
ip access-list standard VLAN73
 permit 204.12.1.0
```

Ticket 2

The key is starting bottom-up troubleshooting and finding that R4 cannot ping R1. The problem is incorrect frame-relay mapping in R1:

R1:
```
interface Serial0/0
 no frame-relay map ip 148.1.0.5 104 broadcast
 frame-relay map ip 148.1.0.4 104 broadcast
```

What interesting is the fact that broadcasts are still mapped to DLCI 104, which allows R1 seeing RIP routes. However, as you can quickly find, only redistributed OSPF routes are learned by R4. Even though R1 learns routes from R2 it does not propagate them to R4. The issue is split-horizon enabled on R1's Frame-Relay interface. By default it's disabled on physical Frame-Relay interface, but this time is has been enabled on purpose.

R1:
```
interface Serial 0/0
 no ip split-horizon
```

Ticket 3

Since BGP is a high-level protocol, issues at any layer of the OSI model might cause loss of BGP connectivity. It is a good to trace the routes from the source (AS 54) checking BGP peering session health and digging down into any lower-level issues if needed. While progressing toward the BGP route receiver, trace any issues that prevent routes from properly propagating through BGP tables. In this trouble-ticket scenario, R6 has Frame-Relay LMI disabled on its connecting to the Frame-Relay cloud. Additionally, SW2's peering sessions are not configured for BGP route reflection. The whole ticket is relatively easy, based on the fact that we know the scope of the problem is AS 100.

```
R6:
interface Serial 0/0/0
 keepalive

SW2:
router bgp 100
 neighbor 150.1.1.1 route-reflector-client
 neighbor 150.1.6.6 route-reflector-client
```

Ticket 4

All hosts are configured with IPv6 addressing matching the IPv4 source networks 150.1.Y.Y for rack1, not your rack necessarily. So if you happen to get this sceanario on rack other then 1, you need to find the proper embedded IPv4 address.

For example, if you are rack 28, the prefix would be 961C:Y0Y where Y is your router number and 0x1C = 28. While fixing IPv6 addressing on R6, you will notice that R6 is using incorrect prefix: 6901 as opposed to 9601. Renumbering will fix this discrepancy, but non of the hosts will still be able to ping R6, even though they would be able to ping each other. The second problem is IPv4 filtering configured on R1 – it does not permit IPv6 in IP tunnel traffic, and you need to allow this manually, i.e. permit protocol 41.

```
!
! You only need to “fix” R3, R4 and R5 if your rack is not rack 1.
! The solution below assumes rack 28.
!
R3:
interface Tunnel3456
 ipv6 address 2002:961C:303:3456::3/64
 tunnel source Loopback0

R4:
interface Tunnel3456
 ipv6 address 2002:961C:404:3456::4/64
 tunnel source Loopback0

R5:
interface Tunnel3456
 ipv6 address 2002:961C:505:3456::5/64
 tunnel source Loopback0

R6:
!
! R6 has incorrect IPv6 prefix.
!
interface Tunnel3456
 ipv6 address 2002:9601:606:3456::6/64
 tunnel source Loopback0

R1:
!
! R1 should permit protocol 41
!
ip access-list extended FR_INBOUND
 5 permit 41 any any

ip access-list extended FR_OUTBOUND
 permit 41 any any
```

Ticket 5

This ticket requires you to perform basic multicast topology analysis. PIM should be enabled on path from R3 down to R6. You will find that a tunnel is provisioned between R1 and R6 across the non-multicast capable Frame-Relay cloud. After this, you may isolate the problem to the following issues:

PIM not enabled between R6 and SW2:

SW2:
```
interface Vlan 68
 ip pim dense-mode
```

RPF failure at R1, fixable by applying an mroute:

R1:
```
ip mroute 0.0.0.0 0.0.0.0 Tunnel0
```

R1 should permit the GRE tunnel traffic coming from R3.

Ticket 6

It is very common to have application-level protocols broken down due to firewalls. Another common problem is simple protocol misconfiguration. In our case, we know that FTP is to be used for file transfer. At the same time, we have been told the FTP server is configured correctly to use the credentials specified. First of all, we may want to check the credentials used by the FTP client, which appears to be OK.

Next, we may want to check if there is packet filtering configured along the path to the file-server. You may quickly discover that R1 applies reflexive access-list filtering on its Frame-Relay interface. This filter permits outgoing TCP sessions, which should permit passive FTP. However, if you check R6's FTP client configuration you will find that passive-FTP has been disabled. This prevents DATA session establishment, as active FTP will not bypass the packet filter. Therefore, to fix the issue, use one single command:

R6:
```
ip ftp passive
```

Ticket 7

After a quick investigation you may find that R5 synchronizes time with BB3. It does additionally peer for NTP with SW1. This ticket requires thorough understanding of NTP security model. Specifically, R5 has authentication key configured for BB3 which is not configured as trusted. Additionally, NTP access-list in R5 does not permit synchronization with the local master clock, denoted using the IP address 127.127.1.1. This is mandatory to permit when the local router is configured as NTP master.

```
R5:
access-list 5 permit 127.127.1.1
!
ntp trusted-key 1
```

In the end you should have NTP synchronized on R5:

```
Rack1R5#show ntp status
Clock is synchronized, stratum 6, reference is 204.12.1.254
nominal freq is 250.0000 Hz, actual freq is 250.0142 Hz, precision is
2**24
reference time is C032D4F9.EED53377 (06:13:13.932 UTC Fri Mar 8 2002)
clock offset is -0.0238 msec, root delay is 0.03 msec
root dispersion is 0.02 msec, peer dispersion is 0.00 msec
loopfilter state is 'CTRL' (Normal Controlled Loop), drift is -
0.000056827 s/s
system poll interval is 64, last update was 93 sec ago.
```

Ticket 8

This troubleshooting scenario requires you to have good understanding of load-balancing NAT configuration. First of all, the “real” servers address pool should be configured as “type rotary”. Secondly, the access-list specifying the traffic to the virtual server should be “mirrored” – it should match traffic from sources to the virtual server’s IP address.

R3:
```
no ip nat pool REAL_SERVERS 148.1.3.110 148.1.3.112 prefix-length 24
!
ip nat pool REAL_SERVERS 148.1.3.110 148.1.3.112 prefix-length 24 type
rotary
!
no ip access-list extended OLD_WEB_SERVER
ip access-list extended OLD_WEB_SERVER
 permit tcp any host 148.1.3.100 eq www
 permit tcp any host 148.1.3.100 eq 8080
 permit tcp any host 148.1.3.100 eq 443
```

Ticket 9

There are two issues hidden in this ticket. The first thing, which is easy to spot, is that RIP distance in SW1 has been set to 255, which prevents any RIP routes from BB3 being used by SW1. However, if you check RIP’s database (it’s separate from the routing table) you will notice that it’s empty nonetheless.

SW1:
```
router rip
 no distance 255
```

Checking the link integrity you will notice that you can still ping BB3. Therefore, there is something filtering RIP updates specifically. If you track through L2 devices connecting SW1 to BB3 and check for port/VLAN filters you will notice that there is a VLAN filter in SW3 dropping all UDP packets. Remove this filter

SW3:
```
no vlan filter VLAN73 vlan-list 73
```

Ticket 10

The key to this ticket is noticing that the issue only occurs after a router reload. There are just a few IOS features that affect convergence in this situation – e.g. stub LSA advertisement or interface dampening. Another hint could be the fact that the different behavior depends on R5’s interface: the servers connect to VLAN5 interface and the Internet is accessible via the other Ethernet interface. Comparing the two interface configuration you may quickly spot the mismatches.

R5:
```
interface FastEthernet 0/1
 no dampening 30 1000 17956 125 restart 17956
```

Lab 7 Solutions

Ticket 1

There are two issues in the initial configuration that affect EIGRP adjacency formation. First, you may quickly spot that R4 cannot reach R5 over the FR cloud. After subsequent check you will notice that R4 does not learn any PVCs via LMI due to disabled keeplalives. Enabling LMI will not help though, and R4 won't be able to get to R5 over the FR link still. If you revert to the other end of the link and inspect R5's FR interface status you will quickly spot that it is in the "standby" state, which clearly indicates there is a backup interface configuration. Using the "show backup" command you may discover that the Serial interface is the primary and the FR is the backup. Per the task's logic, the relation should be reversed. Change the backup configuration and achieve the desired behavior.

```
R5:
interface Serial0/1/0
  no backup interface Serial0/0/0
!
interface Serial0/0/0.54 point-to-point
  backup interface Serial0/1/0

R4:
interface Serial 0/0/0
 keepalive
```

Ticket 2

This ticket requires Ticket 1 to allow for the Frame-Relay cloud to be used as a backup path. The real issue is flapping OSPF adjacency on R3. Comparing the OSPF network types on all three participating interfaces you will notice that R3 uses point-to-point network type, which does not match R1/SW1 settings. Additional setting is tuning R3's hello interval from the default 30 to 10 seconds, per SW1 and R1 settings.

```
R3:
interface FastEthernet0/0
 ip ospf network point-to-multipoint
 ip ospf hello-interval 10
```

Ticket 3

Quickly performing multicast discovery you will notice that there is a tunnel between R4 and SW1 used to traverse the non-multicast capable cloud. You will quickly notice the tunnel is up but the PIM adjacency is not forming. If you try to ping across the tunnel you will notice that R4's end is not configured for any IP address, which disables PIM processing.

```
R4:
interface Tunnel47
 ip unnumbered FastEthernet 0/0
```

However, when you try to ping across the tunnel to the mentioned group, you will notice there is no response. Doing multicast route listing you will notice there are no packets getting to R4 at all, as the SPT entries never appear. This means packets are being filtered somehow. Inspecting the interface configurations you will find that R4's Tunnel end has an access-list that blocks all multicast packet. Based on the scenario requirement to keep the mutlcast traffic to the minimum, you need to add a single entry in the list permitting the group mentioned in the scenario.

```
R4:
no ip access-list extended MULTICAST
ip access-list extended MULTICAST
 permit ip any host 227.69.53.7
 deny ip any 224.0.0.0 15.255.255.255
 permit ip any any
```

Ticket 4

The scenario wording might make you think of TCP intercept feature being used for SYN flood prevention. However, you will quickly discover that neither TCP intercept nor CBAC are configured. The only other way to limit SYN flooding is by applying a rate-limiting configuration, using either legacy or MQC syntax. If you inspect the rate limits applied to R5's interface using the command show interface X/Y rate-limit you will notice there is a rule configured. However, both actions are set to drop which block all traffic to the server.

```
R5:
interface FastEthernet0/1
 no rate-limit output access-group 192 496000 4000 4000 conform-action
drop exceed-action drop
 rate-limit output access-group 192 496000 4000 4000 conform-action
transmit exceed-action drop
```

Ticket 5

NAC is an obscure topic, which does not really belong to R&S lab blueprint. However, you should be ready for the “wildcard” situations similar to this task. If you are not familiar with something, refer to the documentation for more information. In our case, locate the NAC feature implementation in IOS routers. There is a sample configuration in this page, which you may want to compare to your router’s configuration. Specifically, you will find that the access-list applied to the customer-facing interface misses a line for EAPoUDP packets. This prevents Cisco Trust agents from communicating with the authentication server.

R4:
```
no access-list 102
access-list 102 permit udp any any eq 21862
access-list 102 deny ip any any
```

In the real exam, if you encounter a wildcard scenario, it is more reasonable to skip it and return back after you have finished all other tickets.

Ticket 6

Start with bottom-up troubleshooting for this ticket and you will quickly find that both sides of the connection have the protocol down state. As usual with serial lines, check for clock settings to find that R3 has no clock command applied.

R3:
```
interface Serial 1/2
 clock rate 64000
```

However, the protocol remains down after this. Noticing that the protocol used for line encapsulation is PPP, enable PPP negotiation debugging. You will notice failed authentication, initiated by R1. Looking into the configurations you will notice that R3 authenticates R1 using CHAP and R1 is configured for remote party authentication using PAP. At the same time, R3 has no PAP credentials configured. There are two options here – configure PAP credentials on R3 or remove PAP authentication on R1. We choose configuring R3, as removing authentication might be a bad idea.

R3:
```
interface Serial1/2
 ppp pap sent-username Rack1R3 password CISCO
```

Ticket 7

This ticket requires familiarity with zone-based firewall's configuration concepts. Use the show commands to discover the security zone and zone to interface mappings. After that, discover the zone-pairs and the policies applied. You will need to use the following commands:

```
show zone security
show zone-pair security
show policy-map type inspect zone-pair
```

To correctly resolve the misconfiguration, you need to map the configuration to task requirement. First of all, the R5's Loopback belongs to the "Serial" zone. However, the zone pair OutSerial does not have a class for FTP protocol – this class is mistakenly configured for the OutEthernet zone-pair.

The second problem relates to the OutEthernet zone-pair policy. HTTP traffic is being "passed" and not inspected. There is no policy applied in the opposite direction, and this prevents HTTP sessions traffic from returning.

R4:
```
policy-map type inspect Serial
 class VLAN4Web
  inspect
 class VLAN4FTP
  inspect
!
policy-map type inspect Ethernet
 class VLAN4Web
  inspect
 no class VLAN4FTP
```

Ticket 8

One might think that monitor logging has been disabled using the command no logging monitor. However, the ticket states that some messages occasionally appear on the screen, just their amount is too low for normal debugging. In order to properly resolve this ticket, you may need to be familiar with the logging rate limit feature or peek through the running-configuration looking for any configuration related to "logging". You will quickly reveal the logging messages rate-limit setting of one, which is obviously incorrect.

R4:
```
no logging rate-limit
```

Ticket 9

All non-HTTP and non-VoIP traffic maps to the class-default of the CBWFQ policy. This ticket requires good understanding of CBWFQ behavior in pre-HQF images. Contrary to many beliefs, class-default does not get the remaining bandwidth guaranteed in case of interface congestion. In our case, class-default is configured for fair queueing, which has similar effect. All user-configured classes have better weights than class-default, and thus in case of congestion class-default will starve. In order to resolve this syndrome, explicitly allocate bandwidth to class-default, making its weight competitive with other user-defined classes. Since there are no other classes in the configuration, allocate the 60 remaining percents to class-default.

```
R1:
policy-map CBWFQ_DLCI_105
class class-default
  no fair-queue
  bandwidth remaining percent 60
```

Notice that recent IOS versions, which implement HQF, do not suffer from this behavior and properly allocate the remaining bandwidth to class-default.

Ticket 10

This ticket has two issues underneath the problem. The first one is the fact that R4 is configured as a ZFW, and there is no security policy in place between Serial/Ethernet interfaces. Since there are no special requirements, you may just configured a pass-all policy in both directions.

```
R4:
access-list 179 permit ip any any
!
class-map type inspect All
 match access-group 179
!
policy-map type inspect PermitAll
  class All
    pass
!
zone-pair security WAN-WAN1 source WAN-Serial destination WAN-Ethernet
  service-policy type inspect PermitAll
!
zone-pair security WAN-WAN2 source WAN-Ethernet destination WAN-Serial
  service-policy type inspect PermitAll
```

However, there is yet one problem remaining. The next-hops for the prefixes received from BB2 is not accessible from R4 or R5. There is a number of ways to make it reachable – we simply instruct R5 to change the next hop in updates.

R5:
```
router bgp 200
 neighbor 150.1.4.4 next-hop-self
 neighbor 173.1.46.6 next-hop-self
```

Lab 8 Solution

Ticket 1

This ticket demonstrates an issue that could be really hard to spot. First of all, it's hard to trace connectivity issues at layer 2 due to the "spanning" nature of layer 2 domains. You should start with drawing a physical connectivity diagram, which lists all switches and the links interconnecting them. Use the **`show interfaces status`** command to discover these ports. After that, create a VLAN7 SVI in SW4 and assign it an IP address. Ping SW1's VLAN7 IP address to confirm the failure. Create similar SVIs in all switches on the path to SW1 and keep pinging. This would allow you to isolate the fault domain to the links connecting SW3 and SW4.

The fact that SW3 and SW4 see each other via CDP hints that there is no physical link issue – they are connected using two links, ports 19 and 20. These links are even trunking, and the output signifies the trunking encapsulation was negotiated.

```
RSRack1SW3#show interfaces trunk

Port        Mode             Encapsulation  Status        Native vlan
Fa0/19      auto             n-isl          trunking      1
Fa0/20      auto             n-isl          trunking      1

Port        Vlans allowed on trunk)
```

At the same time spanning-tree is not enabled on the interfaces, even though STP for VLAN1 is surely on:

```
RSRack1SW3#show spanning-tree interface fastEthernet 0/19
no spanning tree info available for FastEthernet0/19

RSRack1SW3#show spanning-tree interface fastEthernet 0/20
no spanning tree info available for FastEthernet0/20
```

RSRack1SW3#show spanning-tree vlan 1

```
VLAN0001
  Spanning tree enabled protocol ieee
  Root ID    Priority    32769
             Address     0011.bb0a.c880
             Cost        19
             Port        15 (FastEthernet0/13)
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    32769  (priority 32768 sys-id-ext 1)
             Address     0018.7392.0980
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time 300

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- ----------------------
Fa0/13              Root FWD 19        128.15   P2p
Fa0/14              Altn BLK 19        128.16   P2p
Fa0/15              Altn BLK 19        128.17   P2p
```

The fact that STP is disabled for the ports even though ports are active and STP instance is not off globally points to the FlexLink feature. When you enable Flex-Link on a pair or ports, STP is automatically disabled on these same ports. Since SW4 cannot ping SW1, there is some misconfiguration here. Use the command **`show interfaces swtichport backup`** to discover the FlexLinks configuration. You will quickly notice the configuration inconsistency – SW3 uses port 20 to backup port 19 and SW4 uses port 19 to backup port 20. Configure either switch to match the configuration of the other switch.

SW3:

```
interface FastEthernet0/19
 switchport mode dynamic auto
 no switchport backup interface FastEthernet 0/20
!
interface FastEthernet0/20
 switchport mode dynamic auto
 switchport backup interface FastEthernet 0/19
```

Ticket 2

Frame-Relay switching is significantly easier to debug compared to Ethernet due to point-to-point nature of Frame-Relay. Start by making sure that both physical and protocol states are up for all involved interfaces. This means that clocking and framing is correct and LMI keepalives are being answered. Using the commands **`show interface`** and **`show frame-relay pvc interface`** You will quickly find that R2 lists the local DLCI 231 as DELETED in the switch:

```
RSRack1R2#show frame-relay pvc interface serial 0/1

PVC Statistics for interface Serial0/1 (Frame Relay DTE)

              Active     Inactive      Deleted       Static
  Local          0            0            1            0
  Switched       0            0            0            0
  Unused         0            0            0            0

DLCI = 231, DLCI USAGE = LOCAL, PVC STATUS = DELETED, INTERFACE =
Serial0/1

  input pkts 0             output pkts 0            in bytes 0
  out bytes 0              dropped pkts 0           in pkts dropped 0
```

Since the protocol is down, there are some problems with getting LMI responses to the keepalive message. Use the following commands to observe the LMI message exchange:

```
RSRack1R2#debug interface serial 0/1
RSRack1R2#debug frame-relay lmi
Frame Relay LMI debugging is on
Displaying all Frame Relay LMI data
```

Soon after you entered these commands you will find that R3 is not responding to any LMI queries from R2. If you use the command **`show interface`** on R3 you will find that LMI has been explicitly disabled on R3's connection to R2:

```
RSRack1R3#show interfaces serial
Serial1/3 is up, line protocol is up
  Hardware is CD2430 in sync mode
  MTU 1500 bytes, BW 128 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation FRAME-RELAY, loopback not set
  Keepalive not set
```

```
R3:
interface Serial 1/3
 keepalive
```

Now R2's Frame-Relay connection to R3 is up/up but still the PVC configured locally in R2 shows up as DELETED:

```
RSRack1R2#show frame-relay pvc interface serial 0/1 | inc
ST
DLCI = 231, DLCI USAGE = LOCAL, PVC STATUS = DELETED, INTERFACE =
Serial0/1
```

This means the Frame-Relay switch is not configured to terminate or switch this particular DLCI, and does not list it in LMI response. Get back to R3 and inspect the Frame-Relay switching configuration. There are two ways to set up Frame-Relay switching: using the legacy **`frame-relay route`** command or the modern **`connect`** syntax:

```
RSRack1R3#show frame-relay route
Input Intf      Input Dlci      Output Intf     Output Dlci     Status
```

```
RSRack1R3#show connection all

ID   Name               Segment 1            Segment 2           State
========================================================================
1    R1_R2             Se1/2 132            Se1/1 132            OPER
DOWN
```

Carefully inspect the output of these two commands. The connections list shows that the connection is down. As you can see both segment endpoints use the DLCI 132, while R2 is configured for 231. At the same time, the second segment terminates at the interface Serial 1/1 while it should be using interface Serial 1/3. This is the reason for the connection being down, by the way – interface Serial 1/1 is not configured for Frame-Relay encapsulation:

```
R3:
no connect R1_R2 Serial 1/2 132 Serial 1/1 132
connect R1_R2 Serial 1/2 132 Serial 1/3 231
```

After this, you should be able to ping across the Frame-Relay connection.

Ticket 3

Check the Serial link status using the **`show interface`** command. You will notice the link uses PPP, the protocol is down and PPP is stuck in LCP negotiations:

```
RSRack1R4#show interfaces serial 0/1/0
Serial0/1/0 is up, line protocol is down
  Hardware is GT96K Serial
  Internet address is 149.1.45.4/24
  MTU 1500 bytes, BW 1544 Kbit/sec, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation PPP, LCP REQsent, loopback not set
```

LCP normally negotiates link parameters such as compression, authentication and other options independent of any network protocol, such as IP. To figure out the problem, use the command **`debug ppp negotiation`** at both endpoints. Be careful, as the link is dedicated and using this debug command may produce a lot of output. If you find your router unresponsive due to high amount of debug output either shut down the other end of the link or access the router via telnet and disable debugging. In the debugging output you will notice the following repeating lines:

```
RSRack1R4#
Se0/1/0 LCP: O CONFREQ [REQsent] id 1 len 14
Se0/1/0 LCP:    AuthProto PAP (0x0304C023)
Se0/1/0 LCP:    MagicNumber 0x1A38E4D1 (0x05061A38E4D1)
Se0/1/0 LCP: I CONFREJ [REQsent] id 1 len 8
Se0/1/0 LCP:    AuthProto PAP (0x0304C023)
```

Which means the local endpoint offers PAP as the authentication protocol and R5 rejects it. This means R4 is the authenticator and R5 is the authentication client which refuses PAP. Look at R5's interface configuration:

```
RSRack1R5#sh running-config interface serial 0/1/0
Building configuration...

Current configuration : 166 bytes
!
interface Serial0/1/0
 ip address 149.1.45.5 255.255.255.0
 encapsulation ppp
 ip ospf cost 9999
 ppp pap sent-username R5PPP password 0 CISCO
 ppp pap refuse
```

The router is configured with PAP credentials but refuses PAP. Remove the PAP refuse command:

```
R5:
interface Serial 0/1/0
 no ppp pap refuse
```

However the link protocol remains down, and the debugging output on R4 shows the following:

```
Se0/1/0 PAP: I AUTH-REQ id 2 len 16 from "R5PPP"
Se0/1/0 PAP: Authenticating peer R5PPP
Se0/1/0 PPP: Phase is FORWARDING, Attempting Forward
Se0/1/0 PPP: Phase is AUTHENTICATING, Unauthenticated User
Se0/1/0 PAP: O AUTH-NAK id 2 len 26 msg is "Authentication failed"
Se0/1/0 PPP: Sending Acct Event[Down] id[7E86]
```

This means R5 is sending PAP credentials but R4 cannot validate them. From the show command output above we know that R5 sends the username/password pair R5PPP/CISCO. See how R4 validates them:

```
RSRack1R4#debug ppp authentication
PPP authentication debugging is on

RSRack1R4#debug aaa authentication
AAA Authentication debugging is on

AAA/BIND(00007E8C): Bind i/f Serial0/1/0
Se0/1/0 PPP: Authorization NOT required
Se0/1/0 PAP: I AUTH-REQ id 8 len 16 from "R5PPP"
Se0/1/0 PAP: Authenticating peer R5PPP
AAA/AUTHEN/PPP (00007E8C): Pick method list 'default'
Se0/1/0 PPP: Sent PAP LOGIN Request
RSRack1R4#
Se0/1/0 PPP: Received LOGIN Response FAIL
Se0/1/0 PAP: O AUTH-NAK id 8 len 26 msg is "Authentication failed"
```

Use the following command to extract AAA information and the locally configured usernames:

```
RSRack1R4#sh running-config | section include aaa|username
aaa new-model
aaa group server tacacs+ MYTACACS
 server-private 149.1.4.100 key CISCO
 ip tacacs source-interface Loopback0
aaa authentication login CONSOLE none
aaa authentication ppp default group MYTACACS
aaa session-id common
username R5PPP password 0 CISCO
```

Looking at the AAA configuration you may see that PPP is configured to authenticate use TACACS+ only. There is also a local username matching R5's configuration. Since AAA is configured for TACACS+, check the TACACS+ private group status:

```
RSRack1R4#show aaa servers private

TACACS+: id 0, priority 0, host 149.1.4.100, auth-port 0, acct-port 0
     State: current DEAD, duration 23s, previous duration 66s
     Dead: total time 233s, count 8
```

The server is dead and local fallback is not configured in the AAA list. This is why authentication fails. Fixing this problem will resolve the ticket:

```
R4:
aaa authentication ppp default group MYTACACS local
```

Ticket 4

The ticket specifies rather wide scope for the problem. The first thing you may want to do is discover the BGP topology in AS 100. Using the respective show commands you will find that R5 is the router-reflector peering with all other BGP routers. AS 54 may access AS 100 via two different ways: either coming over the Frame-Relay link connected to R6 or coming through AS 200. The first path should be more preferred due to shorter AS path. Now, see if the target prefix for VLAN88 originates into BGP at R6:

```
RSRack1SW2#show ip bgp 149.1.88.0
BGP routing table entry for 149.1.88.0/24, version 14
Paths: (1 available, best #1, table Default-IP-Routing-Table)
Flag: 0x820
  Advertised to update-groups:
     1
  Local
    0.0.0.0 from 0.0.0.0 (150.1.8.8)
      Origin incomplete, metric 0, localpref 100, weight 32768, valid,
sourced, best
```

And check that R6 advertises it to AS 54:

```
RSRack1R6#show ip bgp neighbors 54.1.2.254 advertised-
routes
BGP table version is 23, 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
*>i149.1.4.0/24     150.1.4.4                0    100      0 ?
*>i149.1.5.0/24     149.1.0.5                0    100      0 ?
*>i149.1.8.0/24     149.1.0.8                0    100      0 ?
*>i149.1.88.0/24    149.1.0.8                0    100      0 ?
<snip>

Total number of prefixes 10
```

Which is true. Next, we need to make sure SW2 sees the prefixes from AS 54:

```
RSRack1SW2#show ip bgp regexp _54_
BGP table version is 24, local router ID is 150.1.8.8
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   54.1.2.254               0    100      0 54 i
* i28.119.17.0/24   54.1.2.254               0    100      0 54 i
* i112.0.0.0        54.1.2.254               0    100      0 54 50 60 i
* i113.0.0.0        54.1.2.254               0    100      0 54 50 60 i
* i114.0.0.0        54.1.2.254               0    100      0 54 i
* i115.0.0.0        54.1.2.254               0    100      0 54 i
* i116.0.0.0        54.1.2.254               0    100      0 54 i
* i117.0.0.0        54.1.2.254               0    100      0 54 i
* i118.0.0.0        54.1.2.254               0    100      0 54 i
* i119.0.0.0        54.1.2.254               0    100      0 54 i
```

None of the paths appear to be best. Let’s see what the reason might be:

```
RSRack1SW2#show ip bgp 112.0.0.0
BGP routing table entry for 112.0.0.0/8, version 23
Paths: (1 available, no best path)
  Not advertised to any peer
  54 50 60
    54.1.2.254 (inaccessible) from 149.1.0.5 (150.1.5.5)
      Origin IGP, metric 0, localpref 100, valid, internal
      Originator: 150.1.6.6, Cluster list: 150.1.5.5
```

The next-hop IP address is inaccessible. There are two options here – either R6 does not change the next-hop to itself or SW2 does not receive the link subnet via IGP. The second option appears to be valid, as we receive the prefixes from the router-reflector, which means the RR has the next-hop IP in its RIB. So the problem is connected to SW2’s IGP routing. SW2 is supposed to use OSPF on VLAN568 segment:

```
RSRack1SW2#show ip ospf neighbor

RSRack1SW2#show ip ospf interface vlan 568
Vlan568 is up, line protocol is up
  Internet Address 149.1.0.8/22, Area 568
  Process ID 1, Router ID 150.1.8.8, Network Type NON_BROADCAST, Cost:
1
  Transmit Delay is 1 sec, State DROTHER, Priority 0
```

There are no OSPF neighbors, and the network type for this segment is set to Non-broadcast. Even though we know BGP works over this segment, quickly ping the OSPF neighbors to validate connectivity:

RSRack1SW2#ping 149.1.0.5

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 149.1.0.5, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/8 ms
```

RSRack1SW2#ping 149.1.0.6

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 149.1.0.6, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/9 ms
```

This drives us to a conclusion there is some problem with OSPF configuration on the segment. Looking around VLAN568 OSPF routers you will find that R6 is configured as the DR:

RSRack1R6#show ip ospf neighbor

```
Neighbor ID     Pri   State           Dead Time   Address
Interface
N/A               0   ATTEMPT/DROTHER    -        149.1.0.8
FastEthernet0/0
150.1.5.5         2   FULL/DR         00:01:47    149.1.0.5
FastEthernet0/0
```

And that it cannot establish the adjacency with SW2. Use the following debugging commands on SW2 and R6 to find out the root cause:

RSRack1R6#debug ip ospf adj
```
OSPF adjacency events debugging is on
```

RSRack1R6#debug ip ospf events
```
OSPF events debugging is on
```

Notice that it may take some time to receive the packets as R6 is slowly polling SW2. You may want to shutdown R6’s connection to VLAN568 to force it restarting the adjacency:

```
RSRack1SW2#
OSPF: Rcv hello from 150.1.6.6 area 568 from Vlan568 149.1.0.6
OSPF: Mismatched hello parameters from 149.1.0.6
OSPF: Dead R 120 C 120, Hello R 30 C 30  Mask R 255.255.255.0 C
255.255.252.0
```

As you can see, there is parameter mismatch – R6 announces netmask /24 and SW2 is configured with the netmask /22. Fix the netmask configuration:

```
SW2:
interface Vlan 568
 ip address 149.1.0.8 255.255.255.0
```

This will allow SW2 learning the next-hop IP address and correctly installing the BGP prefixes.

Ticket 5

Start by looking for the aggregated prefix:

```
RSRack1R3#sh running-config | include aggregate
 aggregate-address 149.1.0.0 255.255.128.0 as-set summary-only
```

Check the aggregated prefix attributes:

```
RSRack1R3#show ip bgp 149.1.0.0 255.255.128.0
BGP routing table entry for 149.1.0.0/17, version 31
Paths: (1 available, best #1, table Default-IP-Routing-Table)
Multipath: eBGP
  Advertised to update-groups:
     2          3
  {100,300}, (aggregated by 200 150.1.3.3)
    0.0.0.0 from 0.0.0.0 (150.1.3.3)
      Origin incomplete, localpref 100, weight 32768, valid,
aggregated, local, best
```

What we can see is that there is AS_SET attribute associated with this prefix. If you check R3's BGP table you will notice that both AS 100 and AS 300 prefixes are aggregated, seeing them having the "suppressed" flag set. Aggregating prefixes from different autonomous systems is not the best idea, as now AS 100 and AS 300 would not accept this prefix thus losing routing information from another AS.

We are not allowed to change the summarized prefix, but we may attempt changing its attributes. First of all, we notice that AS 100 is dual-connected to AS 54. Thus, the aggregate prefix from R2 may re-enter AS 100 by travelling through AS 54. Since the aggregate encompasses AS100's own prefixes, we need to make sure the prefix does not enter AS 100 back, as this may lead to routing loops. This means we should retain AS 100 in the AS_SET attribute. However, we may drop AS 300, as this AS has only connection to AS 200 and no other AS, and thus no routing loops may occur. We accomplish this selective construction of AS_SET attribute using BGP advertise map which does not include information from AS 300 prefixes into the summary prefix:

```
R3:
router bgp 200
 aggregate-address 149.1.0.0 255.255.128.0 as-set summary-only
advertise-map ONLY_AS_100
!
ip as-path access-list 2 permit ^100$
```

This solves the problem of AS 300 receiving AS100's routing information. However, AS 100 doest not receive AS 300's individual prefixes since those are suppressed:

```
RSRack1R3#show ip bgp regexp 300$
BGP table version is 82, 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
s  149.1.7.0/24     149.1.123.2              2             0 300 ?
s>                  149.1.123.1              1             0 300 ?
s> 149.1.77.0/24    149.1.123.2              1             0 300 ?
s                   149.1.123.1              2             0 300 ?
s  149.1.127.0/24   149.1.123.2                            0 300 ?
s>                  149.1.123.1                            0 300 ?
*  150.1.1.0/24     149.1.123.2                            0 300 ?
*>                  149.1.123.1              0             0 300 ?
*  150.1.2.0/24     149.1.123.2              0             0 300 ?
*>                  149.1.123.1                            0 300 ?
*  150.1.7.0/24     149.1.123.2                            0 300 ?
*>                  149.1.123.1                            0 300 ?
```

To resolve this problem, we may configure R3 to un-suppress the specific prefixes on the BGP peering session between R3 and R5 using an unsuppress map:

```
R3:
ip as-path access-list 3 permit ^300$
!
route-map UNSUPPRESS permit 10
 match as-path 3
!
router bgp 200
 neighbor 149.1.254.5 unsuppress-map UNSUPPRESS
```

And now the situation looks correct:

```
RSRack1SW1#show ip bgp 149.1.0.0
BGP routing table entry for 149.1.0.0/17, version 85
Paths: (2 available, best #2, table Default-IP-Routing-Table)
Flag: 0x820
  Not advertised to any peer
  200 100, (aggregated by 200 150.1.3.3)
    149.1.127.2 from 149.1.127.2 (150.1.2.2)
      Origin incomplete, metric 0, localpref 100, valid, internal,
atomic-aggregate
  200 100, (aggregated by 200 150.1.3.3)
    149.1.127.1 from 149.1.127.1 (150.1.1.1)
      Origin incomplete, metric 0, localpref 100, valid, internal,
atomic-aggregate, best
```

```
RSRack1R5#show ip bgp regexp 300$
BGP table version is 98, 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
*> 149.1.7.0/24     149.1.254.3                            0 200 300 ?
*> 149.1.77.0/24    149.1.254.3                            0 200 300 ?
*> 149.1.127.0/24   149.1.254.3                            0 200 300 ?
*> 150.1.1.0/24     149.1.254.3                            0 200 300 ?
*> 150.1.2.0/24     149.1.254.3                            0 200 300 ?
*> 150.1.7.0/24     149.1.254.3                            0 200 300 ?
```

You may validate end-to-end connectivity using **traceroute**:

```
RSRack1SW2#traceroute
Protocol [ip]:
Target IP address: 149.1.7.7
Source address: 149.1.88.8
Numeric display [n]:
Timeout in seconds [3]:
Probe count [3]:
Minimum Time to Live [1]:
Maximum Time to Live [30]:
Port Number [33434]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Type escape sequence to abort.
Tracing the route to 149.1.7.7

  1 149.1.0.5 8 msec 0 msec 0 msec
  2 149.1.254.3 34 msec 33 msec 34 msec
  3 149.1.123.1 50 msec 59 msec 58 msec
  4 149.1.127.7 [AS 300] 59 msec *  51 msec
```

Ticket 6

Use the **`show ipv6 protocols`** command to discover the routing protocols used

```
RSRack1R3#show ipv6 protocols
IPv6 Routing Protocol is "connected"
IPv6 Routing Protocol is "static"
IPv6 Routing Protocol is "bgp 200"
  IGP synchronization is disabled
  Redistribution:
    None
IPv6 Routing Protocol is "bgp multicast"
  IGP synchronization is disabled
  Redistribution:
    None
IPv6 Routing Protocol is "rip RIPng"
  Interfaces:
    Serial1/0
    Loopback100
  Redistribution:
    None
```

Check RIPng routes on R1 and R2 – those two should be learning Loopback100's IPv6 prefix from R3. However, their tables are empty. This signifies some problem with IPv6 routing. Start isolating the issue by checking the IPv6 links connectivity. Use the command show frame-relay map to make sure IPv6 addresses are mapped. Pay special attention to link-local addresses and broadcast mappings:

```
RSRack1R3#show frame-relay map
Serial1/0 (up): ipv6 FE80::2 dlci 302(0x12E,0x48E0), static,
              CISCO, status defined, active
Serial1/0 (up): ipv6 2001:149:1:123::1 dlci 301(0x12D,0x48D0), static,
              broadcast,
              CISCO, status defined, active
Serial1/0 (up): ipv6 2001:149:1:123::2 dlci 302(0x12E,0x48E0), static,
              broadcast,
              CISCO, status defined, active
Serial1/0 (up): ip 149.1.123.1 dlci 301(0x12D,0x48D0), static,
              broadcast,
              CISCO, status defined, active
Serial1/0 (up): ip 149.1.123.2 dlci 302(0x12E,0x48E0), static,
              broadcast,
              CISCO, status defined, active
Serial1/0 (up): ipv6 FE80::1 dlci 301(0x12D,0x48D0), static,
              CISCO, status defined, active
Serial1/1 (up): ip 149.1.254.5 dlci 315(0x13B,0x4CB0), static,
              broadcast,
              CISCO, status defined, active
```

Make sure you can ping across the links:

```
RSRack1R3#ping ipv6 2001:149:1:123::1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:149:1:123::1, timeout is 2
seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/57/60 ms
```

```
RSRack1R3#ping ipv6 2001:149:1:123::2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:149:1:123::2, timeout is 2
seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/58/60 ms
```

This signifies problem with RIPng settings. Debug RIPng packets on R3. You will notice that R3 only displays its own packets being sent.

```
RIPng: Sending multicast update on Loopback100 for RIPng
       src=FE80::211:20FF:FE93:E560
       dst=FF02::1 (Loopback100)
       sport=520, dport=520, length=52
       command=2, version=1, mbz=0, #rte=2
       tag=0, metric=1, prefix=2001:220:20:3::/64
       tag=0, metric=1, prefix=2001:149:1:123::/64
RIPng: Sending multicast update on Serial1/0 for RIPng
       src=FE80::3
       dst=FF02::1 (Serial1/0)
       sport=520, dport=520, length=52
       command=2, version=1, mbz=0, #rte=2
       tag=0, metric=1, prefix=2001:220:20:3::/64
       tag=0, metric=1, prefix=2001:149:1:123::/64
```

Make sure RIPng is enabled on R1’s and R2’s Frame-Relay interfaces. E.g. the output below shows that it is enabled on R1. So at least R3 should be hearing RIPng packets from R1.

```
RSRack1R1#show ipv6 rip
RIP process "RIPng", port 521, multicast-group FF02::9, pid 212
     Administrative distance is 120. Maximum paths is 16
     Updates every 30 seconds, expire after 180
     Holddown lasts 0 seconds, garbage collect after 120
     Split horizon is on; poison reverse is off
     Default routes are not generated
     Periodic updates 2759, trigger updates 1
  Interfaces:
    Serial0/0
    FastEthernet0/0
  Redistribution:
    None
```

This seems to be a dead lock. In situations like this, resort to verbatim comparing protocol settings on both endpoints:

```
RSRack1R3#show ipv6 rip
RIP process "RIPng", port 520, multicast-group FF02::1, pid 213
     Administrative distance is 120. Maximum paths is 16
     Updates every 30 seconds, expire after 180
     Holddown lasts 0 seconds, garbage collect after 120
     Split horizon is off; poison reverse is off
     Default routes are not generated
     Periodic updates 2719, trigger updates 2
  Interfaces:
    Loopback100
    Serial1/0
```

You will notice that R3 uses the UDP port 520 and multicast group of FF02::1 while R1 uses the default port 521 and multicast group FF02::9. Reset R3 to the default settings:

```
R3:
ipv6 router rip RIPng
 no  port 520 multicast-group ff02::1
```

And this would fix the issue:

```
RSRack1R1#show ipv6 route rip
IPv6 Routing Table - 7 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   2001:220:20:3::/64 [120/2]
     via FE80::3, Serial0/0
```

Ticket 7

Sending configuration periodically could be achieved in two different ways – either by using kron or by configuring EEM. Check kron occurrences in R5 to validate your hypothesis:

```
RSRack1R5#show kron schedule
RSRack1R5#
```

Now check for registered EEM policies:

```
RSRack1R5#show event manager policy registered
No.  Class     Type    Event Type          Trap  Time Registered
Name
1    applet    user    timer watchdog      Off   Mon Jan 4 23:56:33
2010   SHOW_RUN_EVERY_5MIN
 1.0: timer watchdog: time 300.000
 maxrun 20.000
 action 1.0 cli command "show run"
 action 2.0 syslog msg "Configuration Saved"
 action 3.0 mail server "149.1.0.100" to "noc@INE.com" from
"r5@INE.com" subject "Configuration" body "$cli_result"
```

And this looks like the script used to report the running configuration. It is registered with the watchdog event to be run every 300 seconds. Reconfigure it to run every minute for the ease of debugging:

```
R5:
event manager applet SHOW_RUN_EVERY_5MIN
 no event tag 1.0
 event timer watchdog time 60
```

Now run some debugging in R5:

```
RSRack1R5#debug event manager action cli
Debug EEM action cli debugging is on

RSRack1R5#debug event manager action mail
Debug EEM action mail debugging is on

RSRack1R5#debug event manager detector timer
Debug EEM Timer Event Detector debugging is on
```

We picked up those command to track the CLI commands being run, as those are being used in the script.

```
%HA_EM-6-LOG: SHOW_RUN_EVERY_5MIN : DEBUG(cli_lib) : : CTL : cli_open
called.
%HA_EM-6-LOG: SHOW_RUN_EVERY_5MIN : DEBUG(cli_lib) : : OUT : RSRack1R5>
%HA_EM-6-LOG: SHOW_RUN_EVERY_5MIN : DEBUG(cli_lib) : : IN  :
RSRack1R5>show run
%HA_EM-6-LOG: SHOW_RUN_EVERY_5MIN : DEBUG(cli_lib) : : OUT :
^
```

```
%HA_EM-6-LOG: SHOW_RUN_EVERY_5MIN : DEBUG(cli_lib) : : OUT : % Invalid
input detected at '^' marker.
%HA_EM-6-LOG: SHOW_RUN_EVERY_5MIN : DEBUG(cli_lib) : : OUT :
%HA_EM-6-LOG: SHOW_RUN_EVERY_5MIN : DEBUG(cli_lib) : : OUT : RSRack1R5>
%HA_EM-6-LOG: SHOW_RUN_EVERY_5MIN: Configuration Saved
%HA_EM-3-FMPD_UNKNOWN_ENV: fh_parse_var: could not find environment
variable: cli_result
%HA_EM-3-FMPD_ERROR: Error executing applet SHOW_RUN_EVERY_5MIN
statement 3.0
RSRack1R5#
%HA_EM-6-LOG: SHOW_RUN_EVERY_5MIN : DEBUG(cli_lib) : : CTL : cli_close
called.
```

There are two main issues that could be observed in the debugging output: first, the **`show run`** command didn't execute properly, as the router was in non-privileged mode, and secondly there was an unknown variable **`$cli_result`**. The last error actually stopped script execution. To fix the first issue we need to put the "**`enable`**" command before "**`show run`**" has been executed. The second problem is actually due to the wrong variable name. The built-in well-know system variable is named "$_cli_result" has been used with the wrong name – this prevents the corresponding command from running properly. Fix both issues as follows, and don't forget to restore the original event timer:

```
R5:
no event manager applet SHOW_RUN_EVERY_5MIN
event manager applet SHOW_RUN_EVERY_5MIN
 event tag 1.0 timer watchdog time 300
 action 1.0 cli command "enable"
 action 2.0 cli command "show run"
 action 3.0 syslog msg "Configuration Saved"
 action 4.0 mail server "149.1.0.100" to "noc@INE.com" from
"r5@INE.com" subject "Configuration" body " $_cli_result"
```

This would allow the script to collect the information and run properly. The script will be unable to contact the mail server, but this is in accordance with the scenarios requirements.

Ticket 8

This is a kind of the ticket that requires you to have thorough understanding of the technology and all associated configuration steps. When you use RSVP for the purpose of admission control and QoS policy installation, you need to remember the following things:

1) RSVP should have reservable bandwidth configured on the interface
2) Fair queue should be configured on WAN interface or on PVC (in case of Frame-Relay)
3) RSVP should be configured to use resource provider as either PVC WFQ or interface WFQ

In this scenario, we can see that R5 has a multipoint interface terminating PVCs from R3 and R4. At the same time, Frame-Relay traffic shaping is enabled on both R4 and R5:

RSRack1R5#show traffic-shape

```
Interface   Se0/0/0
       Access Target    Byte   Sustain   Excess    Interval  Increment
Adapt
VC     List   Rate      Limit  bits/int  bits/int  (ms)      (bytes)
Active
503           56000     875    7000      0         125       875
-
502           56000     875    7000      0         125       875
-
501           56000     875    7000      0         125       875
-
513           56000     875    7000      0         125       875
-
504           512000    640    5120      0         10        640
-
```

What this means is that R4's Frame-Relay interface queue is set to FIFO, and thus cannot support WFQ. However, per-VC queues could be tuned to WFQ and the RSVP resource provider could be set to PVC WFQ. If you look at the existing configuration for the Frame-Relay PVCs you will notice that PVC's queueing is set to FIFO:

RSRack1R5#show frame-relay pvc 504

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

DLCI = 504, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0/0

  input pkts 453           output pkts 348          in bytes 38828
  out bytes 28504          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 183       out bcast bytes 13700
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
  pvc create time 00:43:46, last time pvc status changed 00:43:27
  cir 512000    bc 5120      be 0         byte limit 640    interval 10
  mincir 256000    byte increment 640   Adaptive Shaping none
  pkts 348       bytes 28504     pkts delayed 0         bytes delayed 0
  shaping inactive
  traffic shaping drops 0
  Queueing strategy: fifo
  Output queue 0/40, 0 drop, 0 dequeued
```

This means you need to modify queue settings for PVCs to turn them to WFQ and properly set the resource provider:

R4:
```
map-class frame-relay SHAPE_512
 frame-relay fair-queue
!
interface Serial 0/0/0
 ip rsvp resource-provider wfq pvc
```

R5:
```
map-class frame-relay SHAPE_512
 frame-relay fair-queue
!
interface Serial 0/0/0
 ip rsvp resource-provider wfq pvc
```

This ticket is a good example of the issue that could be resolved by following a technology configuration checklist.

Ticket 9

Check HSRP status in the both routers:

```
RSRack1R1#show standby
FastEthernet0/0 - Group 1
  State is Standby
    4 state changes, last state change 1d01h
  Virtual IP address is 149.1.127.254
  Active virtual MAC address is 0000.0c07.ac01
    Local virtual MAC address is 0000.0c07.ac01 (v1 default)
  Hello time 3 sec, hold time 10 sec
    Next hello sent in 1.486 secs
  Preemption enabled
  Active router is 149.1.127.2, priority 100 (expires in 8.422 sec)
  Standby router is local
  Priority 10 (configured 110)
    Track object 1 state Down decrement 100
  IP redundancy name is "HSRP" (cfgd)

RSRack1R2#show standby
FastEthernet0/0 - Group 1
  State is Active
    5 state changes, last state change 1d01h
  Virtual IP address is 149.1.127.254 (learnt)
  Active virtual MAC address is 0000.0c07.ac01
    Local virtual MAC address is 0000.0c07.ac01 (v1 default)
  Hello time 3 sec, hold time 10 sec
    Next hello sent in 0.015 secs
  Preemption enabled
  Active router is local
  Standby router is 149.1.127.1, priority 10 (expires in 7.075 sec)
  Priority 100 (default 100)
  IP redundancy name is "HSRP" (cfgd)
```

From the output we can see that R1 is configured with higher priority than R2, but the actual priority is being 10, because of decrement due to failed tracked object. So the problem is somehow connected to the object tracking:

```
RSRack1R1#show track 1
Track 1
  Response Time Reporter 1 state
  State is Down
    3 changes, last change 1d01h
  Latest operation return code: Unknown
  Tracked by:
    HSRP FastEthernet0/0 1
```

The object tracks an IP SLA (RTR) operation number one, which appears to be down. Following further to the IP SLA object we find that it is ICMP echo monitor, but there is no statistics history.

```
RSRack1R1#show ip sla monitor configuration 1
IP SLA Monitor, Infrastructure Engine-II.
Entry number: 1
Owner:
Tag:
Type of operation to perform: echo
Target address: 149.1.132.3
Source address: 149.1.123.1
Request size (ARR data portion): 28
Operation timeout (milliseconds): 1000
Type Of Service parameters: 0x0
Verify data: No
Operation frequency (seconds): 1
Next Scheduled Start Time: Pending trigger
Group Scheduled : FALSE
Life (seconds): 3600
Entry Ageout (seconds): never
Recurring (Starting Everyday): FALSE
Status of entry (SNMP RowStatus): notInService
Threshold (milliseconds): 5000
Number of statistic hours kept: 2
Number of statistic distribution buckets kept: 1
Statistic distribution interval (milliseconds): 20
Number of history Lives kept: 0
Number of history Buckets kept: 15
History Filter Type: None
Enhanced History:
```

```
RSRack1R1#show ip sla monitor statistics 1
Round trip time (RTT)   Index 1
Number of successes: Unknown
Number of failures: Unknown
Operation time to live: 0
```

This means the operation has either not been scheduler or has expired. The latter is not the case, as there is not statistics collected. So apparently the operation was never scheduled. Fix this issue:

R1:
```
ip sla monitor schedule 1 start-time now life forever
```

However, the tracked object is still down:

```
RSRack1R1#show track 1
Track 1
  Response Time Reporter 1 state
  State is Down
    3 changes, last change 1d01h
  Latest operation return code: Timeout
  Tracked by:
    HSRP FastEthernet0/0 1
```

If you check the object statistics you will notice constant timeouts. This could be a result of mis-configuration or ICMP filtering in the target router.

```
RSRack1R1#show ip sla monitor statistics 1
Round trip time (RTT)   Index 1
        Latest RTT: NoConnection/Busy/Timeout
Latest operation start time: *06:48:53.599 UTC Sat Mar 2 2002
Latest operation return code: Timeout
Number of successes: 0
Number of failures: 8
Operation time to live: Forever
```

Check the SLA operation configuration once again:

```
RSRack1R1#show ip sla monitor configuration 1
IP SLA Monitor, Infrastructure Engine-II.
Entry number: 1
Owner:
Tag:
Type of operation to perform: echo
Target address: 149.1.132.3
Source address: 149.1.123.1
```

From the output you can see the target address is incorrectly set to 149.X.132.3 as opposed to 149.X.123.3. This is causing the problems for IPS SLA object. Reconfigure the operation, based on the existing configuration:

R1:
```
no ip sla monitor 1
ip sla monitor 1
 type echo protocol ipIcmpEcho 149.1.123.3 source-ipaddr 149.1.123.1
 timeout 1000
 frequency 1
!
ip sla monitor schedule 1 life forever start-time now
```

This brings R1 back to the normal HSRP active state:

```
RSRack1R1#show track 1
Track 1
  Response Time Reporter 1 state
  State is Up
    4 changes, last change 00:00:05
  Latest operation return code: OK
  Latest RTT (millisecs) 40
  Tracked by:
    HSRP FastEthernet0/0 1

RSRack1R1#show standby
FastEthernet0/0 - Group 1
  State is Active
    5 state changes, last state change 00:00:08
  Virtual IP address is 149.1.127.254
  Active virtual MAC address is 0000.0c07.ac01
    Local virtual MAC address is 0000.0c07.ac01 (v1 default)
  Hello time 3 sec, hold time 10 sec
    Next hello sent in 0.648 secs
  Preemption enabled
  Active router is local
  Standby router is unknown
  Priority 110 (configured 110)
    Track object 1 state Up decrement 100
  IP redundancy name is "HSRP" (cfgd)
```

Ticket 10

This situation clearly points to an application level error. However, it might be a result of many different issues, e.g. a packet filter. The first thing you may want to do is connect to R6 on HTTP port 80. Enable TCP transaction debugging to observe TCP packet exchange:

```
RSRack1R4#debug ip tcp transactions
TCP special event debugging is on

RSRack1R4#telnet 150.1.6.6 80
Trying 150.1.6.6, 80 ...
% Connection refused by remote host

RSRack1R4#
TCB676D6810 created
TCB676D6810 setting property TCP_VRFTABLEID (20) 6772102C
TCB676D6810 setting property TCP_TOS (11) 67720FC8
TCB676D6810 setting property TCP_RTRANSTMO (31) 67720EF8
TCB676D6810 setting property TCP_GIVEUP (34) 67720EFC
TCP: Random local port generated 29131, network 1
TCB676D6810 bound to UNKNOWN.29131
TCB676D6810 setting property unknown (24) 67720F28
Reserved port 29131 in Transport Port Agent for TCP IP type 1
TCP: sending SYN, seq 1575808476, ack 0
TCP0: Connection to 150.1.6.6:80, advertising MSS 536
TCP0: state was CLOSED -> SYNSENT [29131 -> 150.1.6.6(80)]
Released port 29131 in Transport Port Agent for TCP IP type 1 delay
240000
TCP0: state was SYNSENT -> CLOSED [29131 -> 150.1.6.6(80)]
TCP0: bad seg from 150.1.6.6 -- closing connection: port 29131 seq 0
ack 1575808477 rcvnxt 0 rcvwnd 0 len 0
TCP0: connection closed - remote sent RST
TCB 0x676D6810 destroyed
```

Notice that the remote endpoint sent us an RST packet. This means that the service mostly is not configured on this port. A packet filter would drop the session packets silently or notice us via an ICMP message. In either case, that would not result in RST packets send back. Check the listening TCP ports in R6:

```
RSRack1R6#show tcp brief all
TCB       Local Address               Foreign Address
(state)
49FCDA3C  149.1.0.6.179               149.1.0.5.11294             ESTAB
49FBC4B4  54.1.2.6.37862              54.1.2.254.179              ESTAB
485628D0  *.443                       *.*
LISTEN
48561EC0  *.443                       *.*
LISTEN
4855F8F8  *.179                       149.1.0.5.*
LISTEN
48546844  *.179                       54.1.2.254.*
LISTEN
```

There is no HTTP port listed here. Now it’s time to validate HTTP server configuration:

```
RSRack1R6#show ip http server all
HTTP server status: Disabled
HTTP server port: 80
HTTP server active supplementary listener ports:
HTTP server authentication method: enable
HTTP server digest algorithm: md5
HTTP server access class: 80
HTTP server base path:
HTTP server help root:
Maximum number of concurrent server connections allowed: 2
Server idle time-out: 180 seconds
Server life time-out: 180 seconds
Maximum number of requests allowed on a connection: 1
HTTP server active session modules: ALL
HTTP secure server capability: Present
HTTP secure server status: Enabled
HTTP secure server port: 443
HTTP secure server ciphersuite: 3des-ede-cbc-sha des-cbc-sha rc4-128-
md5 rc4-128-sha
HTTP secure server client authentication: Disabled
HTTP secure server trustpoint:
HTTP secure server active session modules: ALL
```

Apparently, the HTTP server is disabled and secure HTTP is enabled. Fix this by enabling the server:

```
R6:
ip http server
```

However, repeating the previous connection test you will notice that R6 refuses the connection again. But this time we know the service is on, and still it refuses to accept the incoming section. This might be a result of service access-control. From the **`show ip http server all`** output above you may figure out the HTTP server access-class list of 80. If you check the access-list you will notice the statement ordering error there:

```
RSRack1R6#show ip access-lists 80
Standard IP access list 80
    10 deny   any (2 matches)
    20 permit 149.1.0.0, wildcard bits 0.0.255.255
    30 permit 150.1.0.0, wildcard bits 0.0.255.255
```

Fix it by rebuilding the access-list:

```
R6:
no access-list 80
access-list 80 permit 149.1.0.0 0.0.255.255
access-list 80 permit 150.1.0.0 0.0.255.255
```

After this, you should be able connect to the HTTP server. Try transferring the file from R6's flash memory. You need to authenticate to the server - from the **`show ip http server all`** command output you will see that the server uses "enable" password authentication. Use this for authentication with the HTTP service:

```
RSRack1SW2#copy http://cisco:cisco@150.1.6.6/c2800nm-
adventerprisek9-mz.124-24.T.bin null:
%Error opening http://*****:*****@150.1.6.6/c2800nm-adventerprisek9-
mz.124-24.T.bin (No such file or directory)
```

Provided that the filename was correct, this means some misconfiguration in the HTTP server. The fact that the server responds "no such file" means that the server's document root is not configured or configured incorrectly. Reset the HTTP server root directory using the following command:

```
R6:
ip http path flash:
```

And after this, the transfer finally succeeds.

Lab 9 Solutions

Ticket 1

The ticket prompts towards the issue with Ethernet switching. This might be a false path though, and thus we have to perform a thorough isolation process for every issue mentioned. Starting with R2 and SW1 connection, ensure you cannot ping SW1. The next thing you may want to do is trace the layer 2 path from R2 to SW1. If you perform a logical layer 2 topology discovery you will notice that all switches connect to SW4 in a start topology. Thus, in order to get to SW1, packets from R2 need to traverse across SW2 and SW4. Check the connectivity across the path:

```
Rack1SW2#show int fa 0/2
FastEthernet0/2 is up, line protocol is up (connected)
  Hardware is Fast Ethernet, address is 0018.738d.9e04
<snip>
```

```
Rack1SW2#show interfaces trunk

Port        Mode             Encapsulation  Status        Native vlan
Fa0/19      on               802.1q         trunking      1
Fa0/20      on               802.1q         trunking      1
Fa0/21      on               802.1q         trunking      1
```

```
Rack1SW2#show spanning-tree vlan 27

MST2
  Spanning tree enabled protocol mstp
  Root ID    Priority    32770
             Address     0018.738d.9e00
             This bridge is the root
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    32770  (priority 32768 sys-id-ext 2)
             Address     0018.738d.9e00
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- -----------------------
Fa0/2               Desg FWD 200000    128.4    P2p
Fa0/19              Desg FWD 200000    128.21   P2p
Fa0/20              Desg FWD 200000    128.22   P2p
Fa0/21              Desg FWD 200000    128.23   P2p
```

The above output illustrates that there are no obvious level 2 problems on SW2 for the VLAN connecting R2 and SW1. There could be some sort of filtering though, but we might want to check it later, after we fully verified the end-to-end connectivity between R2 and SW1. Next layer 2 hop in turn is SW4:

```
Rack1SW4#show interfaces trunk

Port        Mode             Encapsulation  Status        Native vlan
Fa0/14      on               802.1q         trunking      1
Fa0/16      on               802.1q         trunking      1
Fa0/17      on               802.1q         trunking      1
Fa0/18      on               802.1q         trunking      1
Fa0/20      on               802.1q         trunking      1
```

```
Rack1SW4#show spanning-tree vlan 27

MST2
  Spanning tree enabled protocol mstp
  Root ID    Priority    32770
             Address     0018.738d.9e00
             Cost        200000
             Port        18 (FastEthernet0/16)
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    32770  (priority 32768 sys-id-ext 2)
             Address     001a.a174.1380
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

Interface        Role Sts Cost      Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------
Fa0/14           Altn BKN*200000    128.16   P2p Bound(PVST) *PVST_Inc
Fa0/16           Root FWD 200000    128.18   P2p
Fa0/17           Altn BLK 200000    128.19   P2p
Fa0/18           Altn BLK 200000    128.20   P2p
Fa0/20           Desg FWD 200000    128.22   P2p
```

Now there is a problem here. The port connected to SW1 is up and functional, but the VLAN connecting R2 and SW2 is blocking on this port. The reason says something about PVST inconsistency. Another important observation is the ports cost which is 200000. This is not normal with PVST or R-PSVT, and we quickly figure out that the local switch runs MSTP. Therefore, we may conclude that SW1 seems to run PVST, as the port is marked as PVST boundary. Confirm that:

```
Rack1SW1#show spanning-tree vlan 27

VLAN0027
  Spanning tree enabled protocol ieee
  Root ID    Priority    24603
             Address     0011.bb0a.c880
             This bridge is the root
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    24603  (priority 24576 sys-id-ext 27)
             Address     0011.bb0a.c880
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time 15

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- -----------------------
Fa0/19              Desg FWD 19        128.21   P2p
```

```
Fa0/20              Desg FWD 19         128.22   P2p
Fa0/21              Desg FWD 19         128.23   P2p
```

The other side does not report any issues. Indeed, the protocol use is IEEE (which maps to Cisco's PVST+) and the port connected to SW4's Fa0/14 is up and runs as designated. Another important piece of information is the fact that SW1 declares itself as root for VLAN 27.

Per the PVST+ and MSTP interaction rules, the PVST+ domain must either host root switches for ALL VLANs or none. At the same time, Cisco's implementation of MSTP does not support the PVST+ hosting any of the root bridges. In fact, when you "network" PVST+ and MSTP domains, all root bridges for all VLANs should belong to the MSTP domain. The MSTP border ports will emulate PVST+ packets based on the instance configuration, and the priority values for all instances should override any of the STP priorities configured for PVST+ STP instances.

In order to fix this problem, we may either convert SW1 to MSTP or change the priority settings in SW1 to the highest (numerically) values. The second option involves less configuration steps and thus is preferred:

```
SW1:
spanning-tree vlan 1-4094 priority 61440
```

Soon after this the connectivity will be restored. And you should notice the message "%SPANTREE-2-PVSTSIM_OK: PVST Simulation inconsistency cleared on port FastEthernet0/14." in SW4.

```
Rack1R2#ping 142.1.27.7

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 142.1.27.7, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 1/3/4 ms
```

Now for the second issue, concerning R3 and R4. Since R3's port connect to SW1 you may hope that the above fix should have helped here as well. And indeed it looks alright now.

```
Rack1R3#ping 142.1.34.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 142.1.34.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

```
Rack1R3#show ip bgp summary | inc 34.4
142.1.34.4      4   300      61     151      531    0    0 00:02:52
10
```

Ticket 2

Start by figuring out the R2/R3 connection's properties. It seems that both R2 and R3 connect to R1, yet the link appears to be point-to-point connecting R2 and R3 directly. This could be a result of Frame-Relay switching, bridging over Frame-Relay or tunneling. List the interfaces and their IP addresses on R2 and R3 for a quick discovery:

```
Rack1R2#show ip interface brief
Interface                  IP-Address      OK? Method Status
Protocol
FastEthernet0/0            142.1.27.2      YES manual up
up
Serial0/0                  unassigned      YES manual up
up
Serial0/1                  unassigned      YES TFTP   up
down
BVI1                       142.1.23.2      YES manual up
up
Loopback0                  150.1.2.2       YES manual up
up
Tunnel0                    unassigned      YES unset  up
up
```

So it appears to be bridging over Frame-Relay. Both R2 and R3 use IRB mode and BVI interfaces. Clear the counter on both BVI interfaces and ping from R3 to R2. Use flood ping just to generate traffic and see if R2 receives it or not:

```
Rack1R3#clear counters bvI 1
Clear "show interface" counters on this interface [confirm]

Rack1R3#ping 142.1.23.2 repeat 1000 timeout 0

Type escape sequence to abort.
Sending 1000, 100-byte ICMP Echos to 142.1.23.2, timeout is 0 seconds:
 ....................................................................
 ....................................................................

Rack1R2#sho int bvi 1
BVI1 is up, line protocol is up
  Hardware is BVI, address is 0000.0c36.9290 (bia 0011.20fd.6e00)
  Internet address is 142.1.23.2/31
<snip>
     0 packets input, 0 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
     0 packets output, 0 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
```

```
Rack1R3#sh ip arp 142.1.23.2

Rack1R3#
```

What this means is that both unicast and broadcast packets don't make it to R2. Now it's time to check the intermediate node. Check for the bridge group configured and the spanning tree status:

```
Rack1R1#show bridge

Total of 300 station blocks, 299 free
Codes: P - permanent, S - self

Bridge Group 1:

Bridge Group 2:

    Address       Action   Interface       Age   RX count   TX count
0000.0c0e.2777   forward   Serial0/0.103     0        137          0
```

```
Rack1R1#show spanning-tree

 Bridge group 1 is executing the ieee compatible Spanning Tree protocol
  Bridge Identifier has priority 32768, address 0000.0cd5.3da4
  Configured hello time 2, max age 20, forward delay 15
  Current root has priority 32768, address 0000.0cbc.932c
  Root port is 10 (Serial0/0.102), cost of root path is 647
  Topology change flag not set, detected flag not set
  Number of topology changes 0 last change occurred 20:02:00 ago
  Times:  hold 1, topology change 35, notification 2
          hello 2, max age 20, forward delay 15
  Timers: hello 0, topology change 0, notification 0, aging 300

 Port 10 (Serial0/0.102) of Bridge group 1 is forwarding
   Port path cost 647, Port priority 128, Port Identifier 128.10.
   Designated root has priority 32768, address 0000.0cbc.932c
   Designated bridge has priority 32768, address 0000.0cbc.932c
   Designated port id is 128.5, designated path cost 0
   Timers: message age 1, forward delay 0, hold 0
   Number of transitions to forwarding state: 1
   BPDU: sent 2, received 35999


 Bridge group 2 is executing the ieee compatible Spanning Tree protocol
  Bridge Identifier has priority 32768, address 0000.0cd5.3db5
  Configured hello time 2, max age 20, forward delay 15
  Current root has priority 32768, address 0000.0c4f.284b
  Root port is 11 (Serial0/0.103), cost of root path is 647
  Topology change flag not set, detected flag not set
  Number of topology changes 2 last change occurred 13:51:10 ago
  Times:  hold 1, topology change 35, notification 2
          hello 2, max age 20, forward delay 15
  Timers: hello 0, topology change 0, notification 0, aging 300
 Port 11 (Serial0/0.103) of Bridge group 2 is forwarding
   Port path cost 647, Port priority 128, Port Identifier 128.11.
   Designated root has priority 32768, address 0000.0c4f.284b
   Designated bridge has priority 32768, address 0000.0c4f.284b
   Designated port id is 128.6, designated path cost 0
   Timers: message age 1, forward delay 0, hold 0
   Number of transitions to forwarding state: 2
   BPDU: sent 3, received 11400
```

First of all, we notice that there are two bridge groups, applied to the subinterfaces of the Frame-Relay interface. Clearly this does not allow for bridging between the PVCs.

```
R1:
interface serial 0/0.103
 no bridge-group 2
 bridge-group 1
```

Did that help? Firs check the spanning tree status in R1:

```
Rack1R1#show spanning-tree 1

 Bridge group 1 is executing the ieee compatible Spanning Tree protocol
  Bridge Identifier has priority 32768, address 0000.0cd5.3da4
  Configured hello time 2, max age 20, forward delay 15
  Current root has priority 32768, address 0000.0c4f.284b
  Root port is 11 (Serial0/0.103), cost of root path is 647
  Topology change flag set, detected flag not set
  Number of topology changes 1 last change occurred 00:00:30 ago
          from Serial0/0.103
  Times:  hold 1, topology change 35, notification 2
          hello 2, max age 20, forward delay 15
  Timers: hello 0, topology change 0, notification 0, aging 15

 Port 10 (Serial0/0.102) of Bridge group 1 is forwarding
   Port path cost 647, Port priority 128, Port Identifier 128.10.
   Designated root has priority 32768, address 0000.0c4f.284b
   Designated bridge has priority 32768, address 0000.0cd5.3da4
   Designated port id is 128.10, designated path cost 647
   Timers: message age 0, forward delay 0, hold 0
   Number of transitions to forwarding state: 1
   BPDU: sent 33, received 36092

 Port 11 (Serial0/0.103) of Bridge group 1 is forwarding
   Port path cost 647, Port priority 128, Port Identifier 128.11.
   Designated root has priority 32768, address 0000.0c4f.284b
   Designated bridge has priority 32768, address 0000.0c4f.284b
   Designated port id is 128.6, designated path cost 0
   Timers: message age 1, forward delay 0, hold 0
   Number of transitions to forwarding state: 1
   BPDU: sent 1, received 32
```

This appears to be in order; however, you will notice only one MAC address learned on the “ports” – the one from R3’s BVI interface.

Rack1R1#show bridge 1

```
Total of 300 station blocks, 299 free
Codes: P - permanent, S - self

Bridge Group 1:

    Address       Action   Interface       Age   RX count   TX count
0000.0c0e.2777   forward   Serial0/0.103     0         14          0
```

```
Now repeat pinging from R3 again:
```

Rack1R3#ping 142.1.23.2

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 142.1.23.2, timeout is 2 seconds:
 ...
Success rate is 0 percent (0/5)
```

Rack1R3#sh ip arp 142.1.23.2

```
Rack1R3#
```

At the same time, the RX bridged packet counter is incrementing on R2 – which means the packets are making it to R2 but not back.

Rack1R2#show bridge 1 verbose

```
Total of 300 station blocks, 299 free
Codes: P - permanent, S - self

BG Hash      Address      Action  Interface      VC    Age   RX count
TX count
 1 50/0   0000.0c0e.2777 forward  Serial0/0       201    0        822
                                  0

Flood ports (BG 1)           RX count    TX count
Serial0/0                           0           0
```

Let's see if there is an access-group on the BVI interface:

Rack1R2#show ip interface bvI 1 | in acce
```
  Outgoing access list is BVI_OUT
  Inbound  access list is BVI_IN
  IP access violation accounting is disabled
```

There is a couple indeed. However, if you remove the lists and try pinging again it will not work still. Besides, if you look into the access-lists they permit EIGRP packets, yet the adjacency does not form. Even if you keep pinging R2 from R3 and enable ICMP packet debugging using **`debug ip icmp`** – you wont see anything. In the context of bridging this might mean that the IP pack-ets don't
make it to the IP level. The only reason for that, provided that we removed

```
R2:
bridge 1 route ip
```

Immediately after this that IP connectivity is restored. Don't forget to apply back the access-lists and check for the EIGRP adjacency:

```
Rack1R2#show ip eigrp neighbors
IP-EIGRP neighbors for process 100
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q
Seq
                                            (sec)         (ms)
Cnt Num
0   142.1.23.3              BV1               10 00:00:41  841  5000  0
55
```

Ticket 3

Start with R5 and check the routing table for the prefixes injected from BGP. Since R5 does not run BGP, the routing information for AS 54 is delivered as either via redistributed BGP routes or default routing information.

```
Rack1R5#sh ip route ospf
O E2 119.0.0.0/8 [110/1] via 142.1.0.3, 00:02:54, Serial0/0/0
O E2 118.0.0.0/8 [110/1] via 142.1.0.3, 00:02:54, Serial0/0/0
O    204.12.1.0/24 [110/1001] via 142.1.0.3, 00:02:54, Serial0/0/0
O E2 117.0.0.0/8 [110/1] via 142.1.0.3, 00:02:54, Serial0/0/0
O E2 116.0.0.0/8 [110/1] via 142.1.0.3, 00:02:54, Serial0/0/0
O E2 115.0.0.0/8 [110/1] via 142.1.0.3, 00:02:54, Serial0/0/0
O E2 114.0.0.0/8 [110/1] via 142.1.0.3, 00:02:54, Serial0/0/0
     142.1.0.0/16 is variably subnetted, 8 subnets, 3 masks
O       142.1.0.4/32 [110/64] via 142.1.0.4, 00:03:25, Serial0/0/0
O       142.1.0.3/32 [110/1000] via 142.1.0.3, 00:02:54, Serial0/0/0
O E2    142.1.27.0/24 [110/20] via 142.1.0.3, 00:02:54, Serial0/0/0
O E2    142.1.23.2/31 [110/20] via 142.1.0.3, 00:02:54, Serial0/0/0
O       142.1.46.0/24 [110/1002] via 142.1.0.3, 00:02:54, Serial0/0/0
     28.0.0.0/24 is subnetted, 2 subnets
O E2    28.119.17.0 [110/1] via 142.1.0.3, 00:02:54, Serial0/0/0
O E2    28.119.16.0 [110/1] via 142.1.0.3, 00:02:54, Serial0/0/0
     150.1.0.0/16 is variably subnetted, 3 subnets, 2 masks
O E2    150.1.2.0/24 [110/20] via 142.1.0.3, 00:02:54, Serial0/0/0
O       150.1.6.6/32 [110/1002] via 142.1.0.3, 00:02:54, Serial0/0/0
```

Looking at the routes, you may identify AS 54 prefixes. If you look at R3 and R4 configuration, you will notice that R4 redistributes BGP into OSPF. The external metric of 1 also signifies this feature is configured. Now the question is: why routes redistributed by R4 are routed via R4?

If you check R4's configuration you will see that it indeed performs redistribution. We check R5's physical interface OSPF network type and find it to be point-to-multipoint:

```
Rack1R5#sh ip ospf interface serial 0/0/0
Serial0/0/0 is up, line protocol is up
  Internet Address 142.1.0.5/24, Area 345
  Process ID 1, Router ID 150.1.5.5, Network Type POINT_TO_MULTIPOINT,
Cost: 64
  Transmit Delay is 1 sec, State POINT_TO_MULTIPOINT
  Timer intervals configured, Hello 30, Dead 120, Wait 120, Retransmit
5
    oob-resync timeout 120
    Hello due in 00:00:01
  Supports Link-local Signaling (LLS)
  Cisco NSF helper support enabled
  IETF NSF helper support enabled
  Index 2/2, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 11
  Last flood scan time is 4 msec, maximum is 4 msec
  Neighbor Count is 2, Adjacent neighbor count is 2
    Adjacent with neighbor 150.1.4.4
    Adjacent with neighbor 150.1.3.3, cost is 1000
  Suppress hello for 0 neighbor(s)
  Simple password authentication enabled
```

Per this output, the cost to reach R3 is 1000, which is higher that interface cost of 64. However, the path via R3 is still preferred! Let's check the OSPF database in R5 for the external prefixes redistributed by R3 and R4.

```
Rack1R5#sh ip ospf database external 118.0.0.0

            OSPF Router with ID (150.1.5.5) (Process ID 1)

                Type-5 AS External Link States

  Routing Bit Set on this LSA
  LS age: 1435
  Options: (No TOS-capability, DC)
  LS Type: AS External Link
  Link State ID: 118.0.0.0 (External Network Number )
  Advertising Router: 150.1.4.4
  LS Seq Number: 80000003
  Checksum: 0x81F1
  Length: 36
  Network Mask: /8
        Metric Type: 2 (Larger than any link state path)
        TOS: 0
        Metric: 1
        Forward Address: 142.1.46.6
        External Route Tag: 100
```

```
Rack1R5#show ip route 142.1.46.6
Routing entry for 142.1.46.0/24
  Known via "ospf 1", distance 110, metric 1002, type intra area
  Last update from 142.1.0.3 on Serial0/0/0, 00:19:18 ago
  Routing Descriptor Blocks:
  * 142.1.0.3, from 150.1.6.6, 00:19:18 ago, via Serial0/0/0
      Route metric is 1002, traffic share count is 1
```

The external prefix has a FA set, and the FA is reachable via R3. That's the reason the path via R3 is being preferred. So why not prefer going to VLAN46 over R4, provided that is has better cost to reach R4? The reason might be linked to the cost that R4 advertises for VLAN46. Check R5's OSPF database for VLAN46.

```
Rack1R5#sh ip ospf database router adv-router 150.1.4.4

            OSPF Router with ID (150.1.5.5) (Process ID 1)

                Router Link States (Area 345)

  Routing Bit Set on this LSA
  LS age: 1357
  Options: (No TOS-capability, DC)
  LS Type: Router Links
  Link State ID: 150.1.4.4
  Advertising Router: 150.1.4.4
  LS Seq Number: 80000046
  Checksum: 0x99ED
  Length: 84
  AS Boundary Router
  Number of Links: 5

    Link connected to: another Router (point-to-point)
     (Link ID) Neighboring Router ID: 150.1.3.3
     (Link Data) Router Interface address: 142.1.0.4
      Number of TOS metrics: 0
       TOS 0 Metrics: 65535

    Link connected to: another Router (point-to-point)
     (Link ID) Neighboring Router ID: 150.1.5.5
     (Link Data) Router Interface address: 142.1.0.4
      Number of TOS metrics: 0
       TOS 0 Metrics: 65535

    Link connected to: a Stub Network
     (Link ID) Network/subnet number: 142.1.0.4
     (Link Data) Network Mask: 255.255.255.255
      Number of TOS metrics: 0
       TOS 0 Metrics: 0

    Link connected to: a Transit Network
     (Link ID) Designated Router address: 142.1.46.6
     (Link Data) Router Interface address: 142.1.46.4
      Number of TOS metrics: 0
       TOS 0 Metrics: 65535
```

We see that R4 advertises connection to R6 (the DR) with the maximum possible OSPF cost! This, naturally, excludes R4 from path computations to reach R4. Now get to R6 and figure out what could be causing this.

```
Rack1R4#show ip ospf interface fastEthernet 0/0
FastEthernet0/0 is up, line protocol is up
  Internet Address 142.1.46.4/24, Area 345
  Process ID 1, Router ID 150.1.4.4, Network Type BROADCAST, Cost: 1
  Transmit Delay is 1 sec, State BDR, Priority 1
  Designated Router (ID) 150.1.6.6, Interface address 142.1.46.6
  Backup Designated router (ID) 150.1.4.4, Interface address 142.1.46.4
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    oob-resync timeout 40
    Hello due in 00:00:05
  Supports Link-local Signaling (LLS)
  Cisco NSF helper support enabled
  IETF NSF helper support enabled
  Index 2/2, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 6, 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 150.1.6.6  (Designated Router)
  Suppress hello for 0 neighbor(s)
```

The interface cost equal 1, so there should be another feature causing the maximum cost for the router links. This feature is known as Max Cost LSA advertising, which is normally used for graceful router shutdown:

```
Rack1R4#show ip ospf | inc [Mm]ax
 Originating router-LSAs with maximum metric
 Maximum wait time between two consecutive SPFs 10000 msecs
```

This prevents R4 from being used for any traffic transit. Disable this feature in R4 to fix the problem:

```
R4:
router ospf 1
 no max-metric router-lsa
```

Immediately after this R5 falls back to the primary path via R4:

```
Rack1R5#sh ip route ospf
O E2 119.0.0.0/8 [110/1] via 142.1.0.4, 00:00:30, Serial0/0/0
O E2 118.0.0.0/8 [110/1] via 142.1.0.4, 00:00:30, Serial0/0/0
O    204.12.1.0/24 [110/66] via 142.1.0.4, 00:00:30, Serial0/0/0
O E2 117.0.0.0/8 [110/1] via 142.1.0.4, 00:00:30, Serial0/0/0
O E2 116.0.0.0/8 [110/1] via 142.1.0.4, 00:00:30, Serial0/0/0
O E2 115.0.0.0/8 [110/1] via 142.1.0.4, 00:00:30, Serial0/0/0
O E2 114.0.0.0/8 [110/1] via 142.1.0.4, 00:00:30, Serial0/0/0
     142.1.0.0/16 is variably subnetted, 9 subnets, 3 masks
O       142.1.0.4/32 [110/64] via 142.1.0.4, 00:34:18, Serial0/0/0
O       142.1.0.3/32 [110/65] via 142.1.0.4, 00:00:30, Serial0/0/0
O E2    142.1.27.0/24 [110/20] via 142.1.0.4, 00:00:30, Serial0/0/0
O E2    142.1.23.2/31 [110/20] via 142.1.0.4, 00:00:30, Serial0/0/0
O       142.1.46.0/24 [110/65] via 142.1.0.4, 00:00:30, Serial0/0/0
O       142.1.34.0/24 [110/65] via 142.1.0.4, 00:00:30, Serial0/0/0
     28.0.0.0/24 is subnetted, 2 subnets
O E2    28.119.17.0 [110/1] via 142.1.0.4, 00:00:30, Serial0/0/0
O E2    28.119.16.0 [110/1] via 142.1.0.4, 00:00:30, Serial0/0/0
     150.1.0.0/16 is variably subnetted, 3 subnets, 2 masks
O E2    150.1.2.0/24 [110/20] via 142.1.0.4, 00:00:30, Serial0/0/0
O       150.1.6.6/32 [110/66] via 142.1.0.4, 00:00:30, Serial0/0/0
```

Ticket 4

The first thing to do is figure out IPv6 topology. Using the respective show commands you will find out that both R2 and R5 have IPv6 enabled on Ethernet interfaces, and there is an IPv6 tunnel connecting them. You may figure out the tunnel type, which is 6to4. After this, you need to check if you can ping across the tunnel.

```
Rack1R5#ping 2002:9601:202::2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2002:9601:202::2, timeout is 2
seconds:
 ...
Success rate is 0 percent (0/5)
```

There is a problem here. The first thing you need to do is trace IPv4 path between R2 and R5:

```
Rack1R5#traceroute 150.1.2.2

Type escape sequence to abort.
Tracing the route to 150.1.2.2

  1 142.1.0.4 28 msec 28 msec 28 msec
  2 142.1.34.3 28 msec 28 msec 32 msec
  3  *
```

The **`traceroute`** stops after R3. As we remember from Ticket 2, there is an access-list applied to R2's BVI interface. This list could be blocking the traceroute and the IPv6 tunnel! However, as you check the access list, you will see protocol 41 permitted there. This protocol type is used for IPv6 over IPv6 tunnels, so the traffic should make it. If you are not sure about the access-list, removed it and ping across the IPv6 tunnel again to find out it's not the root cause.

Now check the tunnel configuration. Since those are 6to4 tunnels, it is important to have the IPv6 addresses set accordingly to the source IPv4 interface.

```
Rack1R5#sh running-config interface tunnel 0
Building configuration...

Current configuration : 141 bytes
!
interface Tunnel0
 no ip address
 no ip redirects
 ipv6 address 2002:6901:505::5/64
 tunnel source Loopback0
 tunnel mode ipv6ip 6to4
```

The embedded IPv4 prefix 6901:505 corresponds to the IPv4 address 105.1.5.5, while it should be 150.1.5.5 to match Loopback0's IP address. Fix the IPv6 address on the tunnel:

```
R5:
interface Tunnel0
 noipv6 address 2002:6901:505::5/64
 ipv6 address 2002:9601:505::5/64
```

Immediately after this you can ping across the tunnel.

```
Rack1R5#ping 2002:9601:202::2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2002:9601:202::2, timeout is 2
seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 216/216/216
ms
```

However, our ultimate goal is providing reachability between the Ethernet segments of R2 and R5. This does not work still, if you take R2's IPv6 address and ping it from R5:

```
Rack1R5#ping ipv6 2002:8e01:3502:27::2 source fastEthernet
0/0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2002:8E01:3502:27::2, timeout is 2
seconds:
Packet sent with a source address of 2002:8E01:505:5::5
 ...
Success rate is 0 percent (0/5)
```

In order to route across the tunnel, static routes are needed at both R2 and R5. Let’s look at the endpoints configuration:

```
Rack1R5#sh ipv6 route static
IPv6 Routing Table - Default - 6 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, M - MIPv6, R - RIP, I1 - ISIS L1
       I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary, D - EIGRP
       EX - EIGRP external
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF
ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
S   2002::/16 [1/0]
     via Tunnel0, directly connected
```

```
Rack1R2#show ipv6 route static
IPv6 Routing Table - 7 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
S   2002::/16 [1/0]
     via ::, Tunnel0
```

There are two /16 prefix routed to the 6to4 tunnel at every router. That seems to be sufficient, but apparently is non-functional. What could be the problem? To understand it, you need to recall how 6to4 encapsulates IPv6 packets. When you send a packet to **`2002:8e01:3502:27::2`** over the IPv6 tunnel it will be encapsulated using an IPv4 address of 142.1.53.2 based on the embedded IPv4 address. The packet will be further dropped as no such prefix exists in the network. The same story would happen when sending packets to R5’s Ethernet interface IPv6 address.

To resolve this problem, you need to recall a special recursive encapsulation feature supported by IOS. If you configure and IPv6 route via an 6to4 tunnel with an explicit IPv6 next-hop address, the IPv4 address embedded in the next-hop will be used for encapsulation. In our case, you need to apply the following static IPv6 routes:

```
R2:
ipv6 route 2002:8E01:505:5::/64 2002:9601:505::5
```

```
R5:
ipv6 route 2002:8E01:3502:27::/64 2002:9601:202::2
```

Immediately after this the connectivity between the two segments is restored:

```
Rack1R5#ping ipv6 2002:8e01:3502:27::2 source fastEthernet
0/0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2002:8E01:3502:27::2, timeout is 2
seconds:
Packet sent with a source address of 2002:8E01:505:5::5
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 216/216/216
ms
```

Ticket 5

Start troubleshooting by checking for BGP sessions health, AS 254 routes are learned via BGP.

```
Rack1R1#show ip bgp summary
BGP router identifier 150.1.1.1, local AS number 200
BGP table version is 266, main routing table version 266
3 network entries using 351 bytes of memory
3 path entries using 156 bytes of memory
2/1 BGP path/bestpath attribute entries using 248 bytes of memory
1 BGP AS-PATH 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 779 total bytes of memory
BGP activity 19/16 prefixes, 19/16 paths, scan interval 60 secs

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
150.1.3.3       4   300    1393    1318        0    0    0 06:56:42
Active
192.10.1.254    4   254     418     432      266    0    0 03:56:27
3
```

The problem manifests itself between R1 and R3. Check the Serial link status:

```
Rack1R1#show interfaces serial 0/1
Serial0/1 is up, line protocol is down
  Hardware is PowerQUICC Serial
  Internet address is 142.1.13.1/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 246/255, txload 1/255, rxload 1/255
  Encapsulation PPP, LCP TERMsent, loopback not set
  Keepalive set (10 sec)
```

You will notice that the link protocol cycles between “up” and “down” constantly. With PPP, which is the link’s encapsulation, this normally means that LCP cannot finish negotiations and establish link parameters. Look at PPP negotiation debugging, but watch out for excessive debugging output (you may want to rate-limit the console messages or simply shut down link at R3 when you’re over).

```
Rack1R1#debug ppp negotiation
PPP protocol negotiation debugging is on
Se0/1 LCP: Timeout: State TERMsent
Se0/1 LCP: State is Closed
Se0/1 PPP: Phase is DOWN
Se0/1 PPP: Phase is ESTABLISHING, Passive Open
Se0/1 LCP: State is Listen
Se0/1 LCP: Timeout: State Listen
Se0/1 LCP: O CONFREQ [Listen] id 10 len 10
Se0/1 LCP:    MagicNumber 0x1ACC4FE5 (0x05061ACC4FE5)
Se0/1 LCP: I CONFREQ [REQsent] id 104 len 14
Se0/1 LCP:    AuthProto EAP (0x0304C227)
Se0/1 LCP:    MagicNumber 0x186C2F2B (0x0506186C2F2B)
Se0/1 LCP: O CONFACK [REQsent] id 104 len 14
Se0/1 LCP:    AuthProto EAP (0x0304C227)
Se0/1 LCP:    MagicNumber 0x186C2F2B (0x0506186C2F2B)
Se0/1 LCP: I CONFACK [ACKsent] id 10 len 10
Se0/1 LCP:    MagicNumber 0x1ACC4FE5 (0x05061ACC4FE5)
Se0/1 LCP: State is Open
Se0/1 PPP: Phase is AUTHENTICATING, by the peer
Se0/1 EAP: I REQUEST  IDENTITY id 6 len 5
Se0/1 EAP: O RESPONSE IDENTITY id 6 len 12 from "ROUTER1"
Se0/1 EAP: I FAILURE id 6 len 4
Se0/1 LCP: I TERMREQ [Open] id 105 len 4
Se0/1 LCP: O TERMACK [Open] id 105 len 4
Se0/1 PPP: Sending Acct Event[Down] id[1E08]
Se0/1 PPP: Phase is TERMINATING
```

Per this output, we can see that R3 uses EAP to authenticate R1. EAP is a type of authentication protocol that generally is supposed to be used with an AAA server. The idea is to tunnel the authentication information to the AAA server and free the router from the burden of implementing an authentication protocol. Thus, with EAP, a router acts as a simple proxy, relaying EAP packets between the authentication client and the AAA server. Thus, our next goal is to check R3's configuration for EAP. If you run the following debugging in R3 you will notice that the router attempts to authenticate the remote endpoint locally, but fails.

```
Rack1R3#debug aaa authentication
AAA Authentication debugging is on

Rack1R3#debug ppp negotiation
PPP protocol negotiation debugging is on

Se1/2 LCP: State is Open
Se1/2 PPP: Phase is AUTHENTICATING, by this end
Se1/2 EAP: O REQUEST  IDENTITY id 108 len 5
Se1/2 EAP: I RESPONSE IDENTITY id 108 len 12 from "ROUTER1"
Se1/2 PPP: Phase is FORWARDING, Attempting Forward
Se1/2 PPP: Phase is AUTHENTICATING, Unauthenticated User
AAA/AUTHEN/PPP (00001E76): Pick method list 'Permanent Local'
Se1/2 EAP: O FAILURE id 108 len 4
```

First, check for AAA servers in R3, then look for local usernames:

```
Rack1R3#show aaa servers
Rack1R3#
```

```
Rack1R3#show running-config | section username
username ROUTER1 password 0 CISCO
```

This means that no AAA settings have been configured, and thus R3's only option is local authentication. If you check R1's Serial interface configuration, you will notice that it is configured to send the EAP identity and password matching the name configured in R3.
At this point, even if your have little understanding of EAP, you may want to go to R3's Serial interface and dump it's current configuration.

```
Rack1R3#sh running-config interface serial 1/2
Building configuration...

Current configuration : 197 bytes
!
interface Serial1/2
 ip address 142.1.13.3 255.255.255.0
 ip pim sparse-mode
 encapsulation ppp
 clock rate 64000
 ppp authentication eap
 ppp eap identity ROUTER3
```

After this, enter the Serial interface configuration mode and use the "?" prompt to list the available "ppp eap" sub-commands. This is a "last resort" move you may want to use if you have no clue how to do something. Luckily enough, the command "ppp eap local" seems to be what we need on R3 in order to force EAP using local database for authentication.

```
R3:
interface Serial 1/2
 ppp eap local
```

That seems to bring the link up. However, AS254's routes are still not in R3's BGP table!

```
Rack1R3#show ip bgp regexp _254_

Rack1R3#
```

This is obviously a BGP problem as you can see the prefixes in R1's BGP table:

```
Rack1R1#show ip bgp regexp _254_
BGP table version is 316, 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
*> 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 ?
```

In order to figure out the problem, enable BGP updates debugging in R3:

```
Rack1R3#debug ip bgp updates
BGP updates debugging is on for address family: IPv4 Unicast

Rack1R3#clear ip bgp 150.1.1.1

%BGP-5-ADJCHANGE: neighbor 150.1.1.1 Up
BGP(0): 150.1.1.1 send UPDATE (format) 119.0.0.0/8, next 150.1.3.3,
metric 0, path 100 54
BGP(0): 150.1.1.1 send UPDATE (prepend, chgflags: 0x0) 118.0.0.0/8,
next 150.1.3.3, metric 0, path 100 54
BGP(0): 150.1.1.1 send UPDATE (prepend, chgflags: 0x0) 117.0.0.0/8,
next 150.1.3.3, metric 0, path 100 54
BGP(0): 150.1.1.1 send UPDATE (prepend, chgflags: 0x0) 116.0.0.0/8,
next 150.1.3.3, metric 0, path 100 54
BGP(0): 150.1.1.1 send UPDATE (prepend, chgflags: 0x0) 115.0.0.0/8,
next 150.1.3.3, metric 0, path 100 54
BGP(0): 150.1.1.1 send UPDATE (prepend, chgflags: 0x0) 114.0.0.0/8,
next 150.1.3.3, metric 0, path 100 54
BGP(0): 150.1.1.1 send UPDATE (prepend, chgflags: 0x0) 28.119.17.0/24,
next 150.1.3.3, metric 0, path 100 54
BGP(0): 150.1.1.1 send UPDATE (prepend, chgflags: 0x0) 28.119.16.0/24,
next 150.1.3.3, metric 0, path 100 54
BGP(0): 150.1.1.1 send UPDATE (format) 113.0.0.0/8, next 150.1.3.3,
metric 0, path 100 54 50 60
BGP(0): 150.1.1.1 send UPDATE (prepend, chgflags: 0x0) 112.0.0.0/8,
next 150.1.3.3, metric 0, path 100 54 50 60
BGP(0): 150.1.1.1 rcv UPDATE w/ attr: nexthop 150.1.1.1, origin ?,
originator 0.0.0.0, path 200 300 254, community , extended community
BGP(0): 150.1.1.1 rcv UPDATE about 205.90.31.0/24 -- DENIED due to: AS-
PATH contains our own AS;
BGP(0): 150.1.1.1 rcv UPDATE about 220.20.3.0/24 -- DENIED due to: AS-
PATH contains our own AS;
BGP(0): 150.1.1.1 rcv UPDATE about 222.22.2.0/24 -- DENIED due to: AS-
PATH contains our own AS;
```

Per this information it appears that AS 200 injects AS 300 in the AS_PATH attribute for prefixes sent to AS 300. This effectively prevents AS 300 from accepting this prefixes. There are two solutions to this problem – either fix the policy in AS 200 (R1) or configure R3 to accept its own AS. Both options are equally “incorrect”, but the second one seems to be simpler to apply. Thus, we configure R3 to accept prefixes with its own AS# from R1:

```
R3:
router bgp 300
neighbor 150.1.1.1 allowas-in
```

This will allow AS254 prefix propagation end-to-end. Even though this “solution” does not look correct, it demonstrates that a ticket could be resolved in an “indirect” way. Additionally, the point of this ticket is to show you that sometimes its worth trying the “last-resort” methods before you give up on a problem!

Ticket 6

This is an “IP Service” type ticket, which clearly specifies the problem scope. Unlike connectivity problems, you don’t have to isolate the problem area, as it is already clearly stated in the ticket. As a matter of fact, tickets like that require more of configuration skills than actual troubleshooting process. First, list the issues stated in the tickets:

1) NMS cannot receive traps
2) NMS cannot reload the router
3) Interface Index information is not consistent

This leads us to the following configuration steps:

1) Ensure sending SNMP traps is enabled
2) Allow the NMS to reload the router. The NMS should have RW access to the SNMP MIB and router shutdown should be permitted.
3) Enable SNMP IfIndex persistence, as this is most likely the cause.

Start by inspecting the existing SNMP configuration, and see what’s missing:

```
Rack1R5#show running-config | include snmp
snmp-server community CISCORW RO 11
snmp-server location San Jose, CA US
snmp-server contact CCIE Lab R5
snmp-server host 142.5.5.100 CISCOTRAP
snmp-server host 142.5.58.100 CISCOTRAP
```

Inspecting the configuration, you will see that both hosts are configured as trap destinations using the correct community value. However, sending traps is not enabled globally. Furthermore, the access-list 11 correctly lists the NMS IP addresses, but is associated with "RO" mode. Next, for the router reboot, you need to configure the "shutdown" option for the SNMP server. And last, the Interface Index persistence is not enabled in the running configuration.

R5:
```
snmp-server ifindex persist
no snmp-server community CISCORO RO 11
snmp-server community CISCORW RW 11
snmp-server shutdown
```

The above configuration should fix all the listed issue. The only difficulty is that you cannot really verify that unless you have an SNMP agent handy, which is not available in the lab.

Ticket 7

This is another “IP Service” troubleshooting ticket. You may not be familiar with the feature and if that’s the case you may want to skip this ticket and push it back to the end of the troubleshooting section. The “service” oriented tickets are normally “leaves” in the dependency tree, so you will not affect any other ticket in the lab. If you pushed the ticket back and then found you have enough time for it, you may resort to looking through the documentation CD for this particular feature and find the configuration examples.
If you inspect the running configuration related to MAC notifications, you will find that three main things are missed:

1) No notifications configured for removed MAC addresses (expired addresses)
2) MAC address table changes are not enabled globally
3) SNMP traps are not configured along with the management stations IP address.

The third item poses some additional problems. We need to know the SNMP community names used in the trap messages. To get this information, you need to go back to the previous ticket, which specifies the SNMP trap community value used for this particular NMS.

SW2:
```
interface FastEthernet0/2
 snmp trap mac-notification removed
!
interface FastEthernet0/6
 snmp trap mac-notification removed
!
mac-address-table notification
!
snmp-server enable traps mac-notification
snmp-server host 142.1.5.100 CISCOTRAP
```

The thing to learn from this ticket is that you should always build configuration checklist for issues like that one. If you don’t remember all configuration steps by heart, you can resort to DocCD, but only do so without sacrificing too much time.

Ticket 8

This ticket involves a special technology used to transport local broadcasts across a multicast-enabled network in a dedicated multicast group. The first thing you need to do is discover the multicast topology and the multicast domain characteristics, such as PIM mode used, RP(s), BSR/MA(s) and so on. Use the commands **`show ip pim interface`** and **`show ip pim neighbor`** on every router to discover the topology, and the commands show ip pim rp mapping to make sure the RP information has propagated to all nodes.

The discovery process would demonstrate healthy multicast network state, provided that you resolved the pre-requisite ticket. The next step is finding out the group used for broadcast packet transportation and seeing if it works.

```
Rack1R4#sh running-config interface fastEthernet 0/0
Building configuration...

Current configuration : 170 bytes
!
interface FastEthernet0/0
 ip address 142.1.46.4 255.255.255.0
 ip pim sparse-mode
 ip multicast helper-map broadcast 239.1.1.100 TRAFFIC
 duplex auto
 speed auto
```

The group used is 239.1.1.100. Join R1’s Ethernet interface to this group and see if you can ping it from R4’s VLAN 46 interface.

```
R1:
interface FastEthernet 0/0
 ip igmp join-group 239.1.1.100
```

```
Rack1R4#ping 239.1.1.100 source fastEthernet 0/0 repeat 100

Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 239.1.1.100, timeout is 2 seconds:
Packet sent with a source address of 142.1.46.4

Reply to request 0 from 142.1.13.1, 32 ms
Reply to request 0 from 142.1.13.1, 52 ms
Reply to request 1 from 142.1.13.1, 28 ms
Reply to request 2 from 142.1.13.1, 28 msõ
```

```
Rack1R3#show ip mroute 239.1.1.100
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

(*, 239.1.1.100), 01:43:00/stopped, RP 150.1.4.4, flags: S
  Incoming interface: FastEthernet0/0, RPF nbr 142.1.34.4
  Outgoing interface list:
    Serial1/2, Forward/Sparse, 01:43:00/00:03:07

(142.1.46.4, 239.1.1.100), 00:00:13/00:03:27, flags: T
  Incoming interface: FastEthernet0/0, RPF nbr 142.1.34.4
  Outgoing interface list:
    Serial1/2, Forward/Sparse, 00:00:13/00:03:17
```

The output above signifies that indeed multicast packets make it through. Don't forget to remove the IGMP join command when you're done. Next, it's time to feed some broadcast packets onto VLAN 46 and see if they get converted to multicast and then transported end-to-end.

Configure R6 to send broadcast UDP packets on VLAN46. Notice that this operation didn't work properly in many versions of IOS 12.3T and 12.4 but functions normally in IOS 12.4T.

```
R6:
ip sla 1
 udp-echo 142.1.46.255 5000 control disable
 frequency 5
ip sla schedule 1 life forever start-time now
!
interface FastEthernet0/1
 ip broadcast-address 142.1.46.255
 ip directed-broadcast
```

Now, enable the following debugging in R4 to observe the broadcast packets received:

```
R4:
access-list 100 permit udp any any 5000
```

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

Rack1R4#debug ip mpacket
IP multicast packets debugging is on
Rack1R4#

IP: s=142.1.46.6 (FastEthernet0/0), d=142.1.46.255, len 44, input
feature
    UDP src=50810, dst=5000, MCI Check(63), rtype 0, forus FALSE,
sendself FALSE, mtu 0
FIBipv4-packet-proc: route packet from FastEthernet0/0 src 142.1.46.6
dst 142.1.46.255
FIBfwd-proc: Default:142.1.46.255/32 receive entry
FIBipv4-packet-proc: packet routing failed
IP: tableid=0, s=142.1.46.6 (FastEthernet0/0), d=142.1.46.255
(FastEthernet0/0), routed via RIB
IP: s=142.1.46.6 (FastEthernet0/0), d=142.1.46.255 (FastEthernet0/0),
len 44, rcvd 3
    UDP src=50810, dst=5000
IP: s=142.1.46.6 (FastEthernet0/0), d=142.1.46.255, len 44, stop
process pak for forus packet
    UDP src=50810, dst=5000
```

From this output, you can see that the broadcast packet is received, but no multicast packet is generated. To see what the problem might be, check the following:

1) Whether UDP port 5000 traffic is configured for forwarding. To validate this, issue the following command:

```
Rack1R4#sh running-config | inc ip forward
ip forward-protocol nd
ip forward-protocol udp 5000
```

2) If the access-list associated with the multicast helper permits the traffic. Check the access-list you saw before in the interface configuration:

```
Rack1R4#show ip access-lists TRAFFIC
Extended IP access list TRAFFIC
    10 permit udp any any eq 9000
```

Now this is where the problem is. The port specified is 9000, not 5000. Fix that:

```
R4:
ip access-list extended TRAFFIC
 no permit udp any any eq 9000
 permit udp any any eq 5000
```

And now you can see the multicast packets being forwarded to R3.

```
Rack1R4#debug ip mpacket
IP multicast packets debugging is on

IP(0): s=142.1.46.6 (FastEthernet0/0) d=239.1.1.100 (FastEthernet0/1)
id=0, ttl=254, prot=17, len=44(44), mforward
Rack1R4#
IP(0): s=142.1.46.6 (FastEthernet0/0) d=239.1.1.100 (FastEthernet0/1)
id=0, ttl=254, prot=17, len=44(44), mforward
Rack1R4#
IP(0): s=142.1.46.6 (FastEthernet0/0) d=239.1.1.100 (FastEthernet0/1)
id=0, ttl=254, prot=17, len=44(44), mforward
Rack1R4#
IP(0): s=142.1.46.6 (FastEthernet0/0) d=239.1.1.100 (FastEthernet0/1)
id=0, ttl=254, prot=17, len=44(44), mforward
```

Now how would you check that the packets are received on VLAN 12. Since BB2 is out of our reach, configure SW1 with an SVI in VLAN12 and enable IP packet debugging for the packets in question:

SW2:
```
interface Vlan 12
 ip address 192.10.1.8 255.255.255.0
!
access-list 101 permit udp any any eq 5000
!
```
Rack1SW2#debug ip packet 101 detail
```
IP packet debugging is on (detailed) for access list 101
```

You will notice no debugging output for the packets matching the list. There is a problem obviously, and most likely related to R1. There could be filtering configured too – this is why we picked up SW2, which connects directly to BB2. Implement the following “packet detection” technique in R1 now:

R1:
```
access-list 101 permit udp any any eq 5000 log
access-list 101 permit ip any any
!
interface FastEthernet 0/0
 ip access-group 101 out
```

There are no packets leaving R1. If you implement multicast packets debugging here (debug ip mpacket) you will notice the multicast packets being received regularly. So the problem may relate to incomplete multicast helper configuration. On the egress endpoint, you need the following:

1) Directed Broadcasts Enabled on the egress interface
2) Multicast helper enabled on the ingress interface with proper access-list.

Check for (1) using the following command:

```
Rack1R1#show ip interface fastEthernet 0/0 | inc
[Bb]roadcast
  Broadcast address is 192.10.1.255
  Directed broadcast forwarding is disabled
```

That's an issue. Fix it by enabling directed broadcasts on the interface:

```
R1:
interface FastEthernet 0/0
 ip directed-broadcast
```

Immediately after this you will start seeing packets being logged:

```
%SEC-6-IPACCESSLOGP: list 101 permitted udp 142.1.46.6(64659) ->
192.10.1.255(5000), 1 packet
Rack1R1#
%SEC-6-IPACCESSLOGP: list 101 permitted udp 142.1.46.6(53586) ->
192.10.1.255(5000), 1 packet
Rack1R1#
%SEC-6-IPACCESSLOGP: list 101 permitted udp 142.1.46.6(61629) ->
192.10.1.255(5000), 1 packet
Rack1R1#
%SEC-6-IPACCESSLOGP: list 101 permitted udp 142.1.46.6(63446) ->
192.10.1.255(5000), 1 packet
Rack1R1#
```

Double check the debugging output at SW2 to make sure you fixed everything:

```
FIBipv4-packet-proc: route packet from Vlan12 src 142.1.46.6 dst
192.10.1.255
FIBfwd-proc: Default:192.10.1.255/32 receive entry
FIBipv4-packet-proc: packet routing failed
IP: tableid=0, s=142.1.46.6 (Vlan12), d=192.10.1.255 (Vlan12), routed
via RIB
IP: s=142.1.46.6 (Vlan12), d=192.10.1.255 (Vlan12), len 44, rcvd 3
    UDP src=57394, dst=5000
IP: s=142.1.46.6 (Vlan12), d=192.10.1.255, len 44, stop process pak for
forus packet
    UDP src=57394, dst=5000
```

And this assures us that end-to-end broadcast forwarding now works.

Ticket 9

Start by checking the EIGRP adjacency between SW1 and R2 and inspecting SW1's routing table:

```
Rack1SW1#show ip eigrp neighbors
EIGRP-IPv4:(100) neighbors for process 100
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q
Seq
                                            (sec)         (ms)
Cnt Num
0   142.1.27.2              Vl27              11 02:49:55    1   200  0
86
```

```
Rack1SW1#show ip route
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

     142.1.0.0/16 is variably subnetted, 3 subnets, 2 masks
C       142.1.27.0/24 is directly connected, Vlan27
D       142.1.23.2/31 [90/1786112] via 142.1.27.2, 02:49:32, Vlan27
C       142.1.107.0/24 is directly connected, Vlan107
C    192.10.1.0/24 is directly connected, Vlan12
     150.1.0.0/24 is subnetted, 2 subnets
C       150.1.7.0 is directly connected, Loopback0
D       150.1.2.0 [90/130816] via 142.1.27.2, 02:49:32, Vlan27
```

Compare this to R2's routing table:

```
Rack1R2#show ip route eigrp
D EX 119.0.0.0/8 [170/1788416] via 142.1.23.3, 03:03:59, BVI1
D EX 118.0.0.0/8 [170/1788416] via 142.1.23.3, 03:03:59, BVI1
D EX 222.22.2.0/24 [170/1788416] via 142.1.23.3, 02:50:30, BVI1
D EX 204.12.1.0/24 [170/2560128256] via 142.1.23.3, 03:03:59, BVI1
D EX 117.0.0.0/8 [170/1788416] via 142.1.23.3, 03:03:59, BVI1
D EX 220.20.3.0/24 [170/1788416] via 142.1.23.3, 02:50:30, BVI1
D EX 116.0.0.0/8 [170/1788416] via 142.1.23.3, 03:03:59, BVI1
D EX 115.0.0.0/8 [170/1788416] via 142.1.23.3, 03:03:59, BVI1
D EX 114.0.0.0/8 [170/1788416] via 142.1.23.3, 03:03:59, BVI1
     142.1.0.0/16 is variably subnetted, 11 subnets, 3 masks
D       142.1.13.0/24 [90/20640000] via 142.1.23.3, 03:03:59, BVI1
D       142.1.13.1/32 [90/20640000] via 142.1.23.3, 03:03:59, BVI1
D EX    142.1.5.0/24 [170/2560128256] via 142.1.23.3, 02:50:57, BVI1
D EX    142.1.0.5/32 [170/2560128256] via 142.1.23.3, 02:52:43, BVI1
D EX    142.1.0.4/32 [170/2560128256] via 142.1.23.3, 02:52:43, BVI1
D EX    142.1.0.0/24 [170/2560128256] via 142.1.23.3, 03:04:00, BVI1
D EX    142.1.46.0/24 [170/2560128256] via 142.1.23.3, 03:04:00, BVI1
D EX    142.1.34.0/24 [170/2560128256] via 142.1.23.3, 02:51:04, BVI1
D EX    142.1.58.0/24 [170/2560128256] via 142.1.23.3, 02:52:45, BVI1
     28.0.0.0/24 is subnetted, 2 subnets
D EX    28.119.17.0 [170/1788416] via 142.1.23.3, 03:04:00, BVI1
D EX    28.119.16.0 [170/1788416] via 142.1.23.3, 03:04:00, BVI1
     150.1.0.0/16 is variably subnetted, 7 subnets, 2 masks
D       150.1.7.0/24 [90/156160] via 142.1.27.7, 02:50:31,
FastEthernet0/0
D       150.1.3.0/24 [90/1913856] via 142.1.23.3, 03:04:01, BVI1
D       150.1.1.0/24 [90/20768000] via 142.1.23.3, 03:04:01, BVI1
D EX    150.1.6.6/32 [170/2560128256] via 142.1.23.3, 03:04:01, BVI1
D EX    150.1.5.5/32 [170/2560128256] via 142.1.23.3, 02:52:46, BVI1
D EX    150.1.4.4/32 [170/2560128256] via 142.1.23.3, 02:52:46, BVI1
D EX 205.90.31.0/24 [170/1788416] via 142.1.23.3, 02:50:32, BVI1
```

What SW1 could see are the networks directly connected to R2. Next, check for route filtering in R2 and SW1:

```
Rack1SW1#show ip protocols
*** IP Routing is NSF aware ***

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
  Redistributing: eigrp 100

Address Family Protocol EIGRP-IPv4:(100)
  EIGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
  EIGRP maximum hopcount 100
  EIGRP maximum metric variance 1
  EIGRP NSF-aware route hold timer is 240
  Topologies : 0(base)

  Automatic network summarization is not in effect
  Maximum path: 4
  Routing for Networks:
    142.1.27.7/32
    150.1.7.7/32
  Routing Information Sources:
    Gateway         Distance      Last Update
    Gateway         Distance      Last Update
    142.1.27.2            90      03:03:12
  Distance: internal 90 external 170
```

```
Rack1R2#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
  EIGRP stub, connected, summary
  Redistributing: eigrp 100
  EIGRP NSF-aware route hold timer is 240s
  Automatic network summarization is not in effect
  Maximum path: 4
  Routing for Networks:
    142.1.23.2/32
    142.1.27.2/32
    150.1.2.2/32
  Routing Information Sources:
    Gateway         Distance      Last Update
    142.1.27.7            90      03:05:07
    142.1.23.3            90      03:05:08
  Distance: internal 90 external 170
```

There don't seem to be any filters configured. You may also want to check for offset lists and distance manipulation, or even dump the routing process configuration using the command:

```
Rack1SW1#sh running-config | begin router eigrp
router eigrp 100
 no auto-summary
 network 142.1.27.7 0.0.0.0
 network 150.1.7.7 0.0.0.0

or on the router:

Rack1R2#show running-config | section router eigrp
router eigrp 100
 network 142.1.23.2 0.0.0.0
 network 142.1.27.2 0.0.0.0
 network 150.1.2.2 0.0.0.0
 no auto-summary
 eigrp stub connected summary
```

Now the second output may prompt a question – why is R2 configured as a stub router, even though it's transit? This would prevent SW1 from querying it for any non-local prefix and effectively filter all prefixes behind R2 from getting to SW1. Obviously, R2 should be un-configured from being an EIGRP stub.

Is there another way to detect if R2 is a stub router, without resorting to the running configuration inspection? You may see it from the detailed neighbor report:

```
Rack1SW1#show ip eigrp neighbors detail
EIGRP-IPv4:(100) neighbors for process 100
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q
Seq
                                            (sec)         (ms)
Cnt Num
0   142.1.27.2              Vl27              12 03:17:14    1   200  0
86
   Version 12.4/1.2, Retrans: 3, Retries: 0, Prefixes: 2
   Topology-ids from peer - 0
   Stub Peer Advertising ( CONNECTED SUMMARY ) Routes
   Suppressing queries
```

The solution is to disable EIGRP stub feature in R2:

```
R2:
router eigrp 100
 no eigrp stub
```

This will instantly make SW1 see all EIGRP prefixes:

```
Rack1SW1#show ip route eigrp
D EX 119.0.0.0/8 [170/1788672] via 142.1.27.2, 00:00:00, Vlan27
D EX 118.0.0.0/8 [170/1788672] via 142.1.27.2, 00:00:00, Vlan27
D EX 222.22.2.0/24 [170/1788672] via 142.1.27.2, 00:00:00, Vlan27
D EX 204.12.1.0/24 [170/2560128512] via 142.1.27.2, 00:00:00, Vlan27
D EX 117.0.0.0/8 [170/1788672] via 142.1.27.2, 00:00:00, Vlan27
D EX 220.20.3.0/24 [170/1788672] via 142.1.27.2, 00:00:00, Vlan27
D EX 116.0.0.0/8 [170/1788672] via 142.1.27.2, 00:00:00, Vlan27
D EX 115.0.0.0/8 [170/1788672] via 142.1.27.2, 00:00:00, Vlan27
D EX 114.0.0.0/8 [170/1788672] via 142.1.27.2, 00:00:01, Vlan27
     142.1.0.0/16 is variably subnetted, 12 subnets, 3 masks
D       142.1.13.0/24 [90/20640256] via 142.1.27.2, 00:00:01, Vlan27
D       142.1.13.1/32 [90/20640256] via 142.1.27.2, 00:00:01, Vlan27
D EX    142.1.5.0/24 [170/2560128512] via 142.1.27.2, 00:00:01, Vlan27
D EX    142.1.0.5/32 [170/2560128512] via 142.1.27.2, 00:00:01, Vlan27
D EX    142.1.0.4/32 [170/2560128512] via 142.1.27.2, 00:00:01, Vlan27
D EX    142.1.0.0/24 [170/2560128512] via 142.1.27.2, 00:00:01, Vlan27
D       142.1.23.2/31 [90/1786112] via 142.1.27.2, 00:00:05, Vlan27
D EX    142.1.46.0/24 [170/2560128512] via 142.1.27.2, 00:00:01, Vlan27
D EX    142.1.34.0/24 [170/2560128512] via 142.1.27.2, 00:00:01, Vlan27
D EX    142.1.58.0/24 [170/2560128512] via 142.1.27.2, 00:00:01, Vlan27
     28.0.0.0/24 is subnetted, 2 subnets
D EX    28.119.17.0 [170/1788672] via 142.1.27.2, 00:00:01, Vlan27
D EX    28.119.16.0 [170/1788672] via 142.1.27.2, 00:00:01, Vlan27
     150.1.0.0/16 is variably subnetted, 7 subnets, 2 masks
D       150.1.3.0/24 [90/1914112] via 142.1.27.2, 00:00:02, Vlan27
D       150.1.2.0/24 [90/130816] via 142.1.27.2, 00:00:06, Vlan27
D       150.1.1.0/24 [90/20768256] via 142.1.27.2, 00:00:02, Vlan27
D EX    150.1.6.6/32 [170/2560128512] via 142.1.27.2, 00:00:02, Vlan27
D EX    150.1.5.5/32 [170/2560128512] via 142.1.27.2, 00:00:02, Vlan27
D EX    150.1.4.4/32 [170/2560128512] via 142.1.27.2, 00:00:02, Vlan27
D EX 205.90.31.0/24 [170/1788672] via 142.1.27.2, 00:00:02, Vlan27
```

Ticket 10

This is yet another ticket on IP Service configuration. This time you are required to understand Netflow filters and basic Netflow configuration. You may dump the relevant global Netflow settings using the filtered running-config contents, e.g. using the command **`show run | inc flow`**. This will allow you to find out that the Netflow export destination is not configured in R6. Additionally, the parameter responsible for min/max packet size capture is not set in the configuration.

```
R6:
ip flow-export destination 142.1.5.100 9996
!
ip flow-capture packet-length
```

The last problem is a bit less obvious. Since the router is configured for Netflow filters, the flow collection feature is actually enabled using MQC classes/Netflow samplers and MQC policy. Normally, you would use the commands **`ip flow ingress|egress`** to collect information on all flows. In case of MQC-specific configuration, you need to be careful and enable flow collection for all classes that you want to monitor. In this particular scenario, flow collection is enabled for the class matching all TCP traffic. At the same time, the egress policy also has a separate class matching HTTP packets, which has CBWFQ weight set. The configuration misses Netflow sampler applied in this class, which results in egress HTTP flows not being monitored by Netflow. By applying the Netflow sampler to this class we effectively fix the problem.

R6:
```
policy-map SERIAL_OUT
 class HTTP_TRAFFIC
  netflow-sampler NORMAL
```

Tickets similar to this one require careful inspection of the feature configuration and thorough understanding of all configuration options. Thus, even though the problem might be relatively simple one, tickets of this type may take a lot of time to complete. The only sure way to be prepared for scenarios like this one is understand all relevant technologies and configuration commands in-depth.

Lab 10 Solutions

Ticket 1

You would notice that RIP is used for routing between R4, R5 and SW2. Check R4's routing table and make sure routes are learned from SW2. You will notice that the only prefixes learned from SW2 are either 172.X.0.0/24 subnets or /16 prefixes for 150.X.0.0/16 and 139.X.0.0/16. This most obviously points to summarization of some kind on SW2. When you would inspect SW2's RIP configuration you will notice auto-summarization enabled. This makes more specific prefixes being learned from R5 and slow link being preferred instead of the primary. When you fix that issue, the path across SW2 would be preferred due to better metrics as R4 applies offset-list to the updates learned on the slow link.

```
SW2:
router rip
 no auto-summary
```

Ticket 2

Even though this ticket states a problem it actually calls for a configuration task, not troubleshooting. This is probably the only configuration-style ticket in the whole workbook and the purpose is to illustrate you some complicated issues that may arise with improper OSPF design. In this case we need to make R1 prefer the path to Area 1 link across Area 0. This is against the natural OSPF path selection which prefers intra-areas over inter-area routes. In order to defeat that, you need a tunnel going across area 0 and belonging to area 1. Then, you tune the metric on this link and on the interfaces connecting R2 and R5 so that their summary cost is lower than cost of the link between R1 and R5. This will steer traffic going to VLAN5 across the tunnel link. Notice that we source the tunnel off the physical interfaces as the loopback interfaces belong to area 1.

```
R1:
interface Tunnel 12
 tunnel source Serial 0/1
 tunnel destination 139.1.23.2
 ip unnumbered Loopback 0
 ip ospf cost 1

R2:
interface Tunnel 12
 tunnel source Serial 0/1
 tunnel destination 139.1.13.1
 ip unnumbered Loopback 0
 ip ospf cost 1
!
interface Serial 0/1
 ip ospf cost 1
```

R5:

```
interface Serial 0/0/0.502
 ip ospf cost 1
```

Ticket 3

The fact that the target prefixes are BGP-learned leads us towards exploring BGP tables. The first thing we want to do is explore the BGP tables or R2 and R5 as those are the two nodes forming a loop in the traceroute output.

```
Rack1R5#show ip bgp
BGP table version is 22, 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
*>i28.119.16.0/24   150.1.6.6                0    100      0 54 i
*>i28.119.17.0/24   150.1.6.6                0    100      0 54 i
*>i112.0.0.0        150.1.6.6                0    100      0 54 50 60 i
*>i113.0.0.0        150.1.6.6                0    100      0 54 50 60 i
*>i114.0.0.0        150.1.6.6                0    100      0 54 i
*>i115.0.0.0        150.1.6.6                0    100      0 54 i
*>i116.0.0.0        150.1.6.6                0    100      0 54 i
*>i117.0.0.0        150.1.6.6                0    100      0 54 i
*>i118.0.0.0        150.1.6.6                0    100      0 54 i
*>i119.0.0.0        150.1.6.6                0    100      0 54 i
*>i139.1.0.0        150.1.6.6                0    100      0 i
```

```
Rack1R2#show ip bgp
BGP table version is 22, 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   150.1.4.4                0    100      0 54 i
*>i28.119.17.0/24   150.1.4.4                0    100      0 54 i
*>i112.0.0.0        150.1.4.4                0    100      0 54 50 60 i
*>i113.0.0.0        150.1.4.4                0    100      0 54 50 60 i
*>i114.0.0.0        150.1.4.4                0    100      0 54 i
*>i115.0.0.0        150.1.4.4                0    100      0 54 i
*>i116.0.0.0        150.1.4.4                0    100      0 54 i
*>i117.0.0.0        150.1.4.4                0    100      0 54 i
*>i118.0.0.0        150.1.4.4                0    100      0 54 i
*>i119.0.0.0        150.1.4.4                0    100      0 54 i
```

The above output shows that R2 resolves all prefixes via the next-hop of R4 and R5 resolves the next-hop of R6. This is actually creating a loop, since R5 will reach R6 via R2 and R2 will reach R4 via R5. To investigate the root cause of this problem we should discover the full BGP topology in the AS. Using normal BGP discovery commands you will find that R4 and R6 are ASBRs and Route Reflectors. R1 and R2 are the RR clients to R4 and R3 and R5 are the RR clients to R6.

Based on this BGP topology, let's trace out an AS54 update as it goes to R5 and R2. Start with R2 – it is supposed to learn the paths from R4 and R4 learns AS54 prefixes from BB3 and R6. R4 will select the path via BB3 since it's an eBGP path and advertise it to R2. Notice that R2 has no idea about the path via R6 as R4 advertises only the best routes. R5 would learn AS54 paths from R6, which in turn learns the paths from BB1 and R4 and then selects the path via BB1. Again, R5 only knows about a single exit point and installs a best path with the next hop of R6. It would make sense to peer R5 with R4 and peer R1, R2 with R6, based on their "IGP proximity". That would resolve the loop issue illustrated above, as every RR client would resolve the prefixes towards the closes RR's IP next-hop.

The problem illustrated in this ticket outlines one important BGP RR design rule: you should always peer clients with their "closest" RRs in terms of IGP metrics. Normally this translates in peering with geographically closest RR, as IGP metrics are normally tuned to select the shortest geographic part, which is normally the fastest way. In order to fix the issue in our case, reconfigure the route reflectors (R4, R6) and RR clients (R1, R2 and R5)

R4:
```
router bgp 100
 no neighbor 150.1.1.1
 no neighbor 150.1.2.2
 neighbor 150.1.5.5 peer-group IBGP
```

R5:
```
router bgp 100
 no neighbor 150.1.6.6
 neighbor 150.1.4.4 remote-as 100
 neighbor 150.1.4.4 update-source Loopback0
```

R6:
```
router bgp 100
 no neighbor 150.1.5.5
 neighbor 150.1.1.1 remote-as 100
 neighbor 150.1.1.1 update-source Loopback0
 neighbor 150.1.1.1 next-hop-self
 neighbor 150.1.1.1 route-reflector-client
 neighbor 150.1.2.2 remote-as 100
 neighbor 150.1.2.2 update-source Loopback0
 neighbor 150.1.2.2 next-hop-self
 neighbor 150.1.2.2 route-reflector-client
```

R1, R2:
```
router bgp 100
 no neighbor 150.1.4.4
 neighbor 150.1.6.6 remote-as 100
 neighbor 150.1.6.6 update-source Loopback0
```

Ticket 4

There are a number of IPv6 mis-configurations that you need to resolve. The best approach is to discover the IPv6 topology to find out that R4 and R5 connect to R3 using ISATAP tunnels. Next, proceed to checking that the tunnels provide end-to-end connectivity just to find out that R4 cannot ping R3. The issue is a simple tunnel misconfiguration.

```
R4:
interface Tunnel345
 no tunnel mode ipv6ip auto-tunnel
 tunnel mode ipv6ip isatap
```

The remaining issues relate to the fact that no router behind R3 can ping to R4 and R5's loopback. Since there is no dynamic routing of any kind, check for static routing issues on R3, R4 and R5 and you'll find static routing on R3 to be misconfigured, so that R4'Loopback is routed to R5 and vice versa.

```
R3:
no ipv6 route 2001:CC1E:1:4::/64 2001:CC1E:1:345:0:5EFE:9601:505
no ipv6 route 2001:CC1E:1:5::/64 2001:CC1E:1:345:0:5EFE:9601:404
!
ipv6 route 2001:CC1E:1:4::/64 2001:CC1E:1:345:0:5EFE:9601:404
ipv6 route 2001:CC1E:1:5::/64 2001:CC1E:1:345:0:5EFE:9601:505
```

Ticket 5

The first thing you need to find out is where the DHCP server is. By looking for DHCP pools on every router connected to VLAN 367 you find none of them is configured as a DHCP server. This implies that at least one of the routers is configured to forward DHCP messages. Use the command show ip interface to find about the helper-address configuration. This will point towards R1 as the DHCP server. Inspecting the DHCP pools in there you will find a subnet for VLAN367 segment. However, if you simulate a host in VLAN367 using any switch with an SVI set for DHCP autoconfiguration, it will not receive an IP address. Check all DHCP settings in R1 to see what could be wrong. You will find that all addresses on VLAN367 are excluded from allocation using DHCP exclusion command. You can fix it as follows:

```
R1:
no ip dhcp excluded-address 139.1.0.1 139.1.0.254
```

However, a question remains if the command was entered on purpose. Normally, it is a good practice to exclude all routers' IP addresses from dynamic IP address allocation. In addition to this, If you inspect R3, SW1 and R6's interface IP configuration in depth you will find HSRP configured on R1 and R6 using a virtual IP address 136.X.0.1. Thus, the original configuration may have had intention to exclude four IP addresses from allocation. However, this way of thinking could be treated as "making assumptions". We are explicitly told to resolve the issue that prevents hosts on VLAN367 from autoconfiguration via DHCP, and by default the DHCP server would ping the address before allocation. This type of ambiguity could be resolved by asking the proctor, but if you're running out of time you may want just to keep the simplest solution in place.

Ticket 6

This ticket is particularly interesting in the way that it presents the QoS problem. There are no QoS settings of any kind explicitly stated in the ticket, e.g. no CIRs, port speeds and so on. Based on the principle of minimal assumptions this should mean that all QoS settings are to be configured correctly and there could be just some small issue that prevents them from working correctly. Since the ticket explicitly states that the problem affects calls from VLAN5 to VLAN2 your subject of inspection would be R5's QoS configuration. First of all, you should check Frame-Relay traffic-shaping configuration, which could be done in four ways: legacy GTS, FRTS, MQC FRTS or MQC CBTS. In our case, FRTS is in use but you will see all PVCs having the CIR of 56Kbps. This means the following main interface command is missing:

```
R5:
interface Serial 0/0/0
 frame-relay traffic-shaping
```

However, after this you will notice that the PVC to R2 is still shaped at 56Kbps. This means that either map-class is not applied or a wrong class is applied to the PVC. If you inspect the interface configuration and the map-classes available in the system you will notice that the map-class name is wrong.

```
R5:
interface Serial 0/0/0.502
 frame-relay class DLCI_502
```

And this completes the troubleshooting. There is no priority queue set up for the PVC, but the Bc value conforms to best-practice of 10ms delay. You don't have to configure a voice LLQ as this is not required in the task in any way. Doing so would violate the minimum needed configuration principle.

Ticket 7

When you start looking for the problem, you will notice that OSPF adjacency between SW4 and R2 is stuck in the “EXSTART” state. This means both routers have seen each other in OSPF HELLO messages but cannot exchange the database. At the very least this means the path between R2 and SW4 is functional and there are no physical or Layer 2 misconfigurations. By inspecting the CDP neighbors table you will confirm this fact.

However, if you try to ping R2 from SW2 or vice versa the ping will fail. This suggests an idea that there could be filtering configured in SW4, R2 or SW2 – the transit switch between them. We know of the following general filtering techniques:

- Acess-lists
- VLAN Filters
- MAC address drops
- QoS rate-limiting

You will not find any access-lists that might block unicast traffic between R2 and SW4 nor there are any VLAN filters configured. There are some multicast drop entries configured in SW2 though, but as we have observed they do not affect multicast HELLO exchange between the two endpoints. And finally there are no QoS settings that might affect unicast packet exchange. This seems to be a dead end. In order to gather more information, try to enable ICMP packet debugging then ping from R2 to SW2 and vice versa. We want to see if any of the routers still receive the packets, i.e. determine if problem is polarized. You will notice that SW4 receives the packets from R2 but R2 does not receive the ICMP packets from SW2.

This means we need to trace the path that the packets from SW4 take to see how they reach R2. To accomplish this, find out the ARP entry for R2 in SW4 and trace it through SW2.

```
Rack1SW4#sh ip arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  139.1.2.2               -   0012.43a6.0c60  ARPA
Internet  139.1.2.22              -   000f.f769.3a80  ARPA   Vlan2
Rack1SW4#
```

```
Rack1SW2#show mac address-table static add 0012.43a6.0c60
          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
```

And apparently this MAC address is not learned on any port. Check the MAC address assigned to R2’s interface:

```
Rack1R2#show interfaces fastEthernet 0/0
FastEthernet0/0 is up, line protocol is up
  Hardware is AmdFE, address is 0012.43a6.06c0 (bia 0012.43a6.06c0)
  Internet address is 139.1.2.2/24
```

This means SW4 has the wrong ARP entry for R2, which is most likely result of incorrect static configuration. Remove the static ARP entry from SW4, and this will fix the problem:

```
SW4:
no arp 139.1.2.2 0012.43a6.0c60 ARPA
```

Ticket 8

Initially it may appear that the problem is simply due to the fact that R3 is not learning any routes from BB2. There are access-lists applied on R3's connection to BB2 but they do not block RIP updates. After you inspect RIPv2 settings you will find that the only problem could be incorrect authentication. To validate this hypothesis use the following command:

```
Rack1R3#debug ip rip
RIP protocol debugging is on
Rack1R3#
RIP: sending v2 update to 224.0.0.9 via FastEthernet0/1 (192.10.1.3)
RIP: build update entries
        139.1.0.0/16 via 0.0.0.0, metric 1, tag 0
        150.1.0.0/16 via 0.0.0.0, metric 1, tag 0
        172.1.0.0/16 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 192.10.1.254 (sourced from one of our
addresses)
```

The output does not signify any authentication errors but rather states that we have the BB2's IP address! Check the interfaces configuration:

```
Rack1R3#show ip interface brief
Interface                  IP-Address      OK? Method Status
Protocol
FastEthernet0/0            139.1.0.3       YES manual up
up
FastEthernet0/1            192.10.1.3      YES manual up
up
Serial1/0                  unassigned      YES manual administratively
down down
Serial1/1                  unassigned      YES manual administratively
down down
Serial1/2                  139.1.13.3      YES manual up
up
Serial1/3                  139.1.23.3      YES manual up
up
Loopback0                  150.1.3.3       YES manual up
up
Loopback100                unassigned      YES unset  up
up
Tunnel35                   139.1.0.3       YES TFTP   up
up
Tunnel345                  unassigned      YES unset  up
up
```

You wont see BB2's IP address in this list. Check the MAC address associated with the IP 192.10.X.254:

```
Rack1R3#sh ip arp 192.10.1.254
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  192.10.1.254            -   0011.920e.5621  ARPA
FastEthernet0/1
```

And if you look on SW3 you will notice that this MAC address is associated with R3, and it is dynamic.

```
Rack1SW3#show mac address-table address 0011.920e.5621
          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
  32    0011.920e.5621    DYNAMIC     Fa0/3
Total Mac Addresses for this criterion: 1
```

So there is some other way the BB2's IP address has gotten associated with R3. Inspect the NAT table:

```
Rack1R3#show ip nat translations
```

Nothing in there. There is one last command that shows all IP addresses associated with the router. Look at the following output:

```
Rack1R3#sh ip aliases
Address Type             IP Address      Port
Alias                    192.10.1.254   23
Interface                139.1.13.3
Interface                139.1.0.3
Interface                150.1.3.3
Interface                139.1.23.3
Interface                192.10.1.3
```

In fact you may prefer this command as alternative to show ip interface brief to quickly find out about all IP addresses associated with the router. As you can see, there is an alias (secondary IP address) configured to map port 23 for IP address 192.10.1.254 to the local router. You may find the actual configuration command:

```
Rack1R3#sh running-config | inc ip alias
ip alias 192.10.1.254 23
```

Removing this command will fix the problem:

R3:

```
no ip alias 192.10.1.254 23
```

Notice that the “**`ip alias`**” command cannot be entered if a router has ARP mapping for the aliased IP address. Normally, to implement this trick you may need to shut the interface down, then enter the command and finally unshut the interface.

Ticket 9

When you look at the console output of R3 you may notice the following message:

```
Rack1R3#
%OSPF-5-ADJCHG: Process 1, Nbr 150.1.7.7 on FastEthernet0/0 from
EXSTART to DOWN, Neighbor Down: Too many retransmissions
Rack1R3#
%OSPF-5-ADJCHG: Process 1, Nbr 150.1.7.7 on FastEthernet0/0 from DOWN
to DOWN, Neighbor Down: Ignore timer expired
```

Most likely this signifies some problems between R3 and SW1, because at the same time R3 and R6 establish OSPF adjacency successfully. This could be well-known MTU issues or something else. Use the OSPF debugging command on R3 and SW1 to find out more information:

```
Rack1R3#
OSPF: Send DBD to 150.1.7.7 on FastEthernet0/0 seq 0x23EF opt 0x52 flag
0x7 len 32
OSPF: Retransmitting DBD to 150.1.7.7 on FastEthernet0/0 [10]
OSPF: rcv. v:2 t:1 l:52 rid:150.1.7.7
      aid:0.0.0.0 chk:8F7E aut:0 auk: from FastEthernet0/0
OSPF: Rcv hello from 150.1.7.7 area 0 from FastEthernet0/0 139.1.0.7
OSPF: End of hello processing

OSPF: Send DBD to 150.1.7.7 on FastEthernet0/0 seq 0x23EF opt 0x52 flag
0x7 len 32
OSPF: Retransmitting DBD to 150.1.7.7 on FastEthernet0/0 [11]
OSPF: rcv. v:2 t:1 l:52 rid:150.1.6.6
      aid:0.0.0.0 chk:8F7E aut:0 auk: from FastEthernet0/0

OSPF: Rcv DBD from 150.1.3.3 on Vlan367 seq 0x23EF opt 0x52 flag 0x7
len 32  mtu 1500 state EXSTART
OSPF: First DBD and we are not SLAVE

OSPF: Send DBD to 150.1.3.3 on Vlan367 seq 0x900 opt 0x52 flag 0x7 len
32
OSPF: Retransmitting DBD to 150.1.3.3 on Vlan367 [19]
```

`Rack1SW1#`

```
OSPF: rcv. v:2 t:1 l:52 rid:150.1.3.3
      aid:0.0.0.0 chk:8F7D aut:0 auk: from Vlan367
OSPF: Rcv hello from 150.1.3.3 area 0 from Vlan367 139.1.0.3
OSPF: End of hello processing

OSPF: rcv. v:2 t:2 l:32 rid:150.1.3.3
      aid:0.0.0.0 chk:E906 aut:0 auk: from Vlan367
OSPF: Rcv DBD from 150.1.3.3 on Vlan367 seq 0x23EF opt 0x52 flag 0x7
len 32  mtu 1500 state EXSTART
```

```
OSPF: First DBD and we are not SLAVE
OSPF: Send hello to 224.0.0.5 area 0 on Vlan367 from 139.1.0.7
OSPF: Send DBD to 150.1.3.3 on Vlan367 seq 0x900 opt 0x52 flag 0x7 len
32
OSPF: Retransmitting DBD to 150.1.3.3 on Vlan367 [20]
```

As we can see, both routers hear each others hello messages but cannot seem to exchange the DBD packets. This sounds to be similar to the issue we had before – the link seems to be fine but only multicast packets make it through. Once again, enable ICMP packets debugging and ping SW1 from R3 and vice versa. You will notice that none of the routers receive the packets from the other side. If you look for standard filtering techniques on R3 and SW1 (R3 physically connects to SW1) you will notice no access-lists or VLAN filters of any sort. If you validate ARP entries like in the ticket we did before, they both look legit. What would you do now?

It this situation, we still have no idea where the packets disappear, we only know those are unicast packets. It would be nice to use packet sniffer or tracer to further isolate it, but all we have are IOS commands. So we could try doing low-level packet debugging and see how they get processed on routers and the transit switch.

R3 & SW1:
```
access-list 100 permit ospf any any
```

Rack1SW1#debug ip packet detail 100
```
IP packet debugging is on (detailed) for access list 100
Rack1SW1#

IP: s=139.1.0.3 (Vlan367), d=139.1.0.7, len 64, rcvd 0, proto=89
IP: s=139.1.0.6 (Vlan367), d=224.0.0.5, len 84, rcvd 0, proto=89
```

This means that SW1 does receive the unicast OSPF packets from R3. However, the same debugging performed on R3 will reveal the following output among the other:

```
IP: s=139.1.0.7 (FastEthernet0/0), d=139.1.0.3, len 64, unicast rpf
failed, proto=89
IP: s=139.1.0.7 (FastEthernet0/0), d=139.1.0.3, len 84, unicast rpf
failed, proto=89
```

And now we received some hint! uRPF seems to be enabled on the R3's interfaces and it denies packets from SW1. uRPF is yet another filtering method that you may not expect sometimes. Check the uRPF configuration on VLAN367's interface and the next-hop for SW1's IP address on R3:

Rack1R3#show ip interface fastEthernet 0/0 | inc verify
```
  IP verify source reachable-via RX, allow default
```

```
Rack1R3#show ip route 139.1.0.7
Routing entry for 139.1.0.7/32
  Known via "static", distance 1, metric 0 (connected)
  Routing Descriptor Blocks:
  * directly connected, via Null0
      Route metric is 0, traffic share count is 1
```

And we see the problem – there is a static route for 139.X.0.7/32 to null. Removing it will resolve the problem:

```
R3:
no ip route 139.1.0.7 255.255.255.255
```

Ticket 10

When you discover the multicast topology, you will notice a tunnel connecting R3 and R5 and configured for multicasting. There is no other multicast path. If you inspect QoS policies on the path that the tunnel takes (R3, R2 and R5) you will notice that the PVC between R2 and R5 has provisioned CIR of 512Kbps while the Serial link between R2 and R3 has link bandwidth of 384Kbps. There is a QoS policy applied egress on R3's connection to R2 that allocates 256Kbps to video traffic classified base on RTSP protocol.

There is no QoS policy on the Frame-Relay connection between R2 and R5, but this is in accordance with the ticket's statement that only the Serial link between R2 and R3 is congested. So there should be something in the QoS policy configured that applies to the Serial link. If you look it at carefully though, It makes sense as the requested 256Kbps are allocated to the RTSP traffic. There is no need to guarantee LLQ scheduling to video traffic as video streams are more tolerable to delays and losses than audio and because video traffic is not a good candidate for LLQ.

However, we know that the video multicast traffic is routed inside the GRE tunnel connecting R3 and R5 and thus the policy should be applied inside the tunnel. This, however, will void the policy's effect on other traffic going across the Serial link. In order to optimally resolve this problem, QoS pre-classification should be enabled on the tunnel interface to properly apply physical-interface level QoS policy to the traffic sent across the tunnel.

```
R3:
interface Tunnel 0
 qos pre-classify
```