Sep 17

Troubleshooting Multicast Routing

Posted by Petr Lapukhov, 4xCCIE/CCDE in IP Multicast,IP Multicast — 18 Comments

Tweet

Abstract

This publication illustrates some common techniques for troubleshooting multicast issues in IP networks. Common problems and their causes are discussed, troubleshooting techniques demonstrated. PIM Sparse mode is used for most of the examples, due to the fact that this is the most complicated mode of multicast signaling. The suggested troubleshooting approach separates control plane from data-plane troubleshooting and heavily relies on the **mroute** command for the control-plane verification. This publication requires solid understanding of intra-domain multicast routing technologies.

Common Reasons for Multicast Problems

In short, one common reason for all issues with multicast routing is the PIM and logical/physical topology incongruence. Ideally, multicast should be deployed in a single IGP domain with PIM enabled on all links running the IGP with all links preferably being point-to-point or broadcast multiple-access. If you have multicast running across the domain that has multiple IGPs, or you do not have PIM enabled on all links or finally you have NBMA links in the topology – you have open possibilities for a problem. Unfortunately, the "problem" conditions just described are very common in the CCIE lab exam environment.

The most common type of multicast issue is the **RPF Failure**. RPF checks are used both at the control and data plane of multicast routing. Control plane involves PIM signaling – some PIM messages are subject to RPF checks. For example, PIM (*,G) Joins are sent toward the shortest path to RP. Next, the BSR/RP address in the BSR messages is subject to RPF check as well. Notice that this logic does not apply to PIM Register messages – the unicast register packet may arrive on any interface. However, RPF check is performed on the encapsulated multicast source to construct the SPT toward the multicast source.

Data plane RPF checks are performed every time a multicast data packet is received for forwarding. The source IP address in the packet should be reachable via the receiving interface, or the packet is going to be dropped. Theoretically, with PIM Sparse-Mode RPF checks at the control plane level should preclude and eliminate the data-plane RPF failures, but data-plane RPF failures are common during the moments of IGP re-convergence and on multipoint non-broadcast interfaces.

PIM Dense Mode is different from SM in the sense that data-plane operations preclude control-plane signaling. One typical "irresolvable" RPF problem with PIM Dense mode is known as "split-horizon" forwarding, where packet received on one interface, should be forwarded back out of the same interface in the hub-and-spoke topology. The same problem may occur with PIM Sparse mode, but this type of signaling allows for treating the NBMA interface as a collection of point-to-point links by the virtue of PIM NBMA mode.

PIM SM Troubleshooting Routine

PIM SM Troubleshooting consists of checking the control plane first and validating the data plane after this. We outline the process step-by-step below and provide references for further breakdowns. Troubleshooting process is always centered on a sample multicast group "G" and a group of senders "S" and receivers "R".

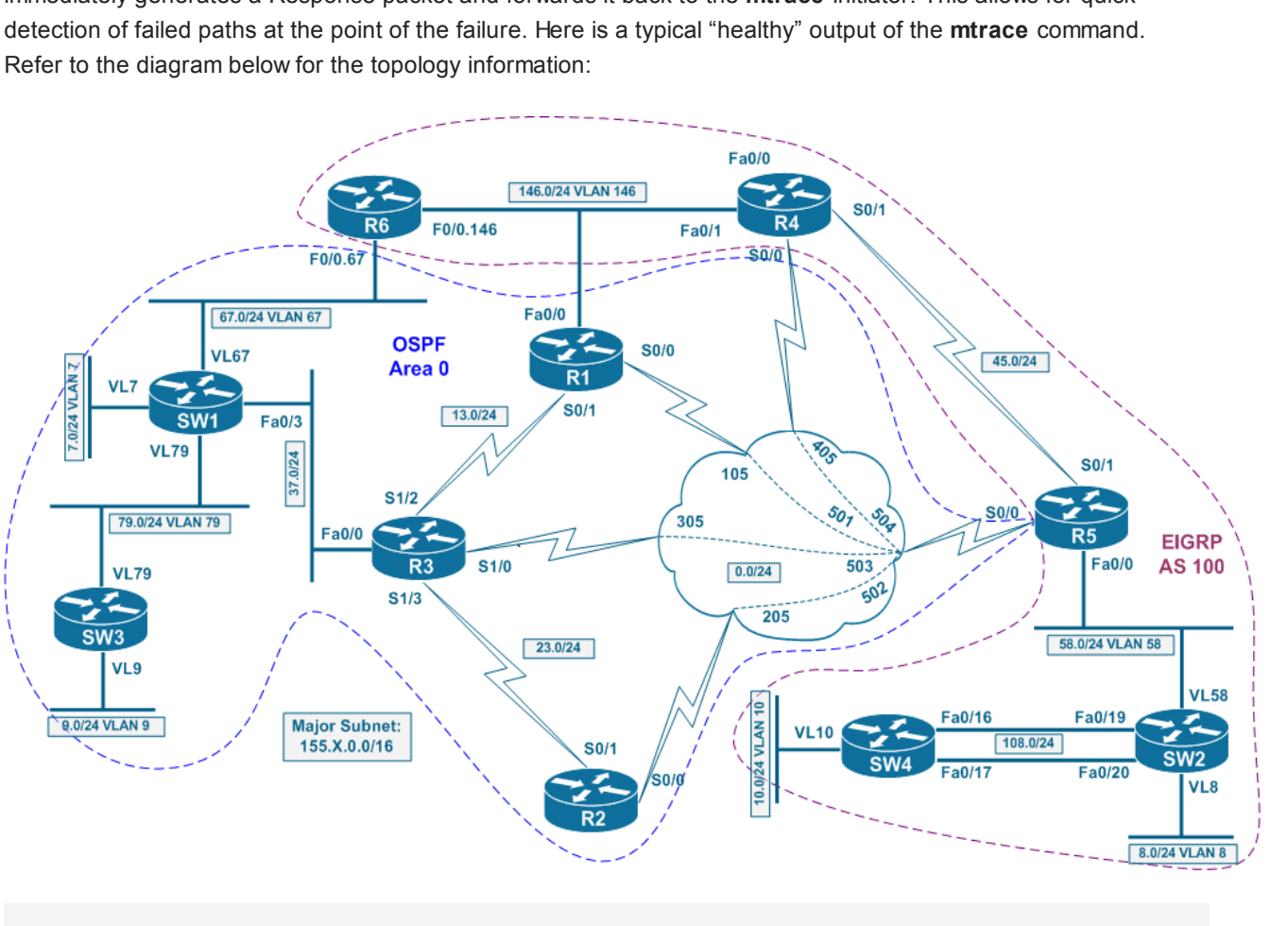
- **Step 1:** Ensure RP information propagation through the topology. First, confirm that the BSR/MA hears all the candidate RP announcements. If the MA/BSR collects all the information, make sure there is an RP for group "G" using the command **show ip pim rp mapping**. After this, proceed to every router in the domain, starting with the ones closest to BSR/MA and check that they have the RP mapping information. Refer to **Troubleshooting Auto-RP** and **Troubleshooting PIM BSR** for detailed techniques on fixing the Auto-RP/BSR problems.
- **Step 2:** Ensure all receivers "R" have joined the group "G". Use the command **show ip igmp groups** to validate this. If you don't have actual receivers, simulate them on the routers using the command **ip igmp join**. Next, from every leaf router that has a receiver attached, issue the **mtrace** command back to the RP address. Use the command **mtrace [RP-IP-Address]** to accomplish this. For more information about the **mtrace** command, refer to the section **Understanding the *mtrace* command**. If you can successfully **mtrace** back to the RP, this means the leaf router is able to join the (*,G) tree for this RP. If the **mtrace** breaks at some point, this most likely means an RPF failure occurs at this point. Read the **Resolving RPF failures in control plane** for information on fixing this problem. Fix all RPF problems so that receiver can join the RP-tree.
- **Step 3:** Ensure that DR can reach the RP using unicast packet exchange. Ping the RP address off the DR address for the source segment. Remember you can change the registration source address on the DR using the command **ip pim register-source [interface]**
- **Step 4:** This step should be performed on the leaf routers connecting the receivers "R" or emulating the receivers themselves. Use the **mtrace** command and trace the route back to the "S" addresses. This procedure ensures that every leaf node is able to switch to the shortest-tree upon receiving the first packet down the RP-tree. This operation is not required if you are not using the SPTs, i.e. if the command **ip pim spt-threshold infinity** is used at the respective leaf router.
- **Step 5:** Simulate ICMP echo traffic from the source to the multicast group "G" either from the actual sources "S" or from the routers attached directly to the sources and ensure you are receiving response from every receiver "R". In some cases, you are not required to receive the replies back, but only make traffic flow down to the receivers. It is more common, however, to ensure bi-directional traffic exchange between the senders and receivers. If this step fails, follow the procedures outlined in the **Troubleshooting multicast data-plane failures** section.

Understanding the *mtrace* command

The **mtrace** command is often overlooked, but very useful troubleshooting tool for multicast networks. Unlike the classic **traceroute** tool, this is mainly a control-plane verification tool. It does not generate real multicast packet flow but rather perform the "reverse path" query process. The process goes as following. Initiating router obtains the source (S), destination (D) and the group (G) multicast address to trace to from the operator. The Destination address could be omitted and in this case the route is traced down to the initiating router. For our purposes, we are going to supply just the source address, to query the RPF path back from the receiver to the source.

Here is a brief description of how the **mtrace** command works. The initiating router generates an internal Query for **mtrace** which is translated into a Request message with the (S,D,G) information encapsulated. This Request is forwarded to the next upstream router on the shortest path to S. Notice that this requires PIM to be enabled on this interface, and therefore **mtrace** is able to detect RPF failures due to PIM/IGP incongruence. Every next router in turn forwards the Request packet upstream to the first-hop router closest to the Source and adds local information, such as multicast routing protocol used, distance from the requesting router, incoming interface and so forth. The first-hop router that has the source directly attached generates a Response message, and passes its down to the Destination IP address. The return process does not add any information but simply delivers the response to the initiating router. It is important to understand that the whole query/response process is performed using unicast packets hop-by-hop, and the **mtrace** result is displayed upon the reception of the final Response packet.

If any router on the path toward the source is not able to forward the Request upstream due to an RPF failure, it immediately generates a Response packet and forwards it back to the **mtrace** initiator. This allows for quick detection of failed paths at the point of the failure. Here is a typical "healthy" output of the **mtrace** command. Refer to the diagram below for the topology information:




```
Rack29R5#mtrace 150.29.10.10
Type escape sequence to abort.
mtrace from 150.29.10.10 to 155.29.58.5 via RPF
From source (?) to destination (?)
Querying full reverse path...
 0  155.29.58.5
-1  155.29.58.5 PIM  [150.29.10.0/24]
-2  155.29.58.8 PIM  [150.29.10.0/24]
-3  155.29.108.10 PIM  [150.29.10.0/24]
-4  150.29.10.10
```

The first column displays the amount of the hops that the responding router is away from the receiver. The first entry has the hop count of zero – this entry is inserted by the local router in response to the initial Query message. It has neither multicast routing protocol information nor RPF validation entry. The second line with the hop-count "-1" corresponds to the RPF entry on the initiating router – it displays the IP address of the RPF interface (155.29.58.5) the protocol used for multicast routing (PIM) and the RPF information – prefix 150.29.10.0/24 that is used for RPF check at this hop. The third line with the hop count of "-2" is the next upstream router toward the source we are tracing to. It displays the IP address of the interface that *received* the query (155.29.58.8), the protocol used for multicast forwarding and the RPF route – 150.29.10.0/24. The process continues till the first-hop router directly attached to the source, which is displayed without any RPF information at the line having the hop count of "-4".

Next, look at the sample of "broken" **mtrace**, which demonstrates lack of PIM enabled on a transit link with healthy IGP. PIM is disabled on SW4's interface connecting to SW2:

```
Rack29R5#mtrace 150.29.10.1
Type escape sequence to abort.
mtrace from 150.29.10.1 to 155.29.58.5 via RPF
From source (?) to destination (?)
Querying full reverse path...
 0  155.29.58.5
-1  155.29.58.5 PIM  [150.29.10.0/24]
-2  155.29.58.8 PIM  [150.29.10.0/24]
-3  155.29.108.10 PIM Multicast disabled [150.29.10.0/24]
```

Notice that hop "-3" (SW4) reports the exact failure – PIM multicast disabled but provides the RPF information. Essentially, SW2 (155.29.58.8) still forwarded the Request to SW4, but the latter correctly detected lack of multicast adjacency.

Here is a more interesting example. We program SW2 with a static **mroute** that sets the RPF information for 150.29.10.0/24 to point toward R5. The route, in turn, belongs to SW4, which creates IGP/Static RPF discrepancy. The **mtrace** output will look like this:

```
Rack29R5#mtrace 150.29.10.10
Type escape sequence to abort.
mtrace from 150.29.10.10 to 155.29.58.5 via RPF
From source (?) to destination (?)
Querying full reverse path...
 0  155.29.58.5
-1  155.29.58.5 PIM  [150.29.10.0/24]
-2  155.29.58.8 PIM/Static  [150.29.10.0/24]
-3  155.29.58.5 PIM  [150.29.10.0/24]
-4  155.29.58.8 PIM/Static  [150.29.10.0/24]
-5  155.29.58.5 PIM  [150.29.10.0/24]
-6  155.29.58.8 PIM/Static  [150.29.10.0/24]
...
```

As you can see, the Request packet cycles between R5 and R8 following the "false" static **mroute** information. Once again, the **mtrace** points at the static routing information which is the source of the RPF check loops.

Being a powerful tool, **mtrace** remains a control-plane only utility. That is, it may not detect data-plane RPF failures, such as split-horizon forwarding on NBMA interfaces. From the **mtrace** perspective, there is no problem following the path back the same interface the Request was received onto. Another good example is multicast rate-limit or multicast boundary that drops all multicast packets – such configuration will not be detected by **mtrace**. Therefore, simply validating the control-plane functionality may not be enough, and you may need to apply procedures described in **Troubleshooting multicast data plane failures** paragraph.

Resolving RPF failures in control plane

As soon as you have identified problems using the **mtrace** tool, you need to fix them using either of the following methods:

- Enable PIM on the RPF interface if it has not been enabled. Often, the problem could be that PIM is not enabled on some links connecting receiver to the source, or it is enabled on the links that do not constitute shortest path.
- Install a static **mroute** on the router that experiences RPF failure. This is probably the easiest way, if the use of static **mroute**s is allowed by the scenario. Static **mroute**s do not affect unicast forwarding, and therefore are least intrusive in the terms of configuration modification.
- Tune IGP metrics so that the shortest path changes to the interface running PIM. This may require changing IGP protocol administrative distance or protocol's native metrics. The main drawback is that unicast forwarding is affected in result, which may have undesired side-effects, such as routing loops or suboptimal traffic flow. If you do change the IGP preferences, try to be as selective as possible and apply changes only to the prefixes for the source subnets.
- Create M-BGP session and advertise the prefixes for the source subnets via M-BGP to modify RPF information. This is very similar to creating the static **mroute**s, but does not involve any static configuration. M-BGP learned prefixes are preferred for RPF checks over unicast routing information, plus you may flexibly change the next-hop value for BGP updates. However, similar to static **mroute**s, you may not be allowed to create new BGP peering sessions or processes.

Troubleshooting Auto-RP

Auto-RP uses two dense-mode groups to flood RP and RP-to-Group mapping information: 224.0.1.39 for RP announcements and 224.0.1.40 for RP discovery. Every router joins the 224.0.1.40 group for listening to the Auto-RP discovery messages and the RP candidates flood their presence via 224.0.1.39. Only the mapping agents listen to the 224.0.1.39 group so they can create RP to group mappings. Troubleshooting Auto-RP consists of the following steps:

- **Step 1:** Ensure the Auto-RP groups are allowed to flood. You have three different options: using PIM sparse-dense mode, using PIM SM and PIM Auto-RP listener and lastly statically configuring an RP for the Auto-RP groups. If you think at least one of these conditions has been met, you may move to the next step.
- **Step 2:** Ensure the MA is able to hear all RP announces listening to the group 224.0.1.39. Use the command **show ip pim rp mapping** on the MA to validate this. You may also use the command **show ip pim autorp** to learn about Auto-RP statistics information.

  ```
  Rack29R5#show ip pim autorp
  AutoRP Information:
    AutoRP is enabled.

  PIM AutoRP Statistics: Sent/Received
    RP Announce: 0/0, RP Discovery: 0/0
  ```

  If the MA does not learn all or some of the RP announcements, perform **mtrace** from the MA to the failing RP's IP address and see if there are any problems on the path. Resolve the problems, if any, using the methods for fixing RPF failures. However, if the **mtrace** does not detect any issue, and the information is still not being learned, proceed to data-plane debugging as follows.

  Start at the MA, and issue the command **debug ip mpacket**. You may use an access-list along with this command to permit just the packets going from the RP address to the group 224.0.1.39 – the RP announcements. This is especially useful in production where you need to be careful about debugging output load on the router's CPU. If the MA is having any RPF issues receiving the RP announces, you will see the output similar to the following:

  ```
  IP(0): s=150.29.6.6 (Serial0/1/0) d=224.0.1.39 id=9123, ttl=8, prot=17, len=52(48), not RPF interface
  IP(0): s=150.29.6.6 (Serial0/1/0) d=224.0.1.39 id=9129, ttl=8, prot=17, len=52(48), not RPF interface
  ```

  If the MA does not show any problems, proceed to the next one, but use the command **debug ip mpacket fastswitch 224.0.1.39** on the transit routers. This command will enable tracing the fast-switched multicast packets – fast-switching the default behavior for transit nodes. Keep in mind that RP announcements are not sent very often, and you may want to change the sending interval to as shorter value using the command similar to the following one on the RP: **ip pim send-rp-announce loopback 0 scope 10 interval 3**. Following this process you will be able to locate the point where RPF failure happens. After this, you may use any of the above described methods to resolve the RPF failure.

- **Step 3:** After the MA has learned all RP information, you need to make sure all routers are able to receive the MA RP discovery messages via the group 224.0.1.40. This procedure is very similar to verifying the MA: use the command **show ip pim rp mapping** to check the routers, followed by **mtrace** to the MA address and finally data-plane troubleshooting. However, this time you should be looking to trace the group 224.0.1.40 along with the source IP address of the MA. You may want to tune the MA discovery advertisement interval on the MA using the command **ip pim send-rp-discovery … interval XX**

As a heads-up, since Auto-RP uses PIM DM (typically) for multicast flooding it often has issues when running on NBMA segments. Common problems include inability of PIM DM to pass traffic across the hub in the hub-and-spoke topology due to the split-horizon forwarding problem and improper PIM Assert winner selection on the NBMA segment. If the MA is located at any of the spokes, the Auto-RP discovery messages will not make it through to the other spokes – the solution is either moving the MA to the hub, or using tunnel interfaces from hub to "other" spokes. As an alternative to using the tunnels, in production network you may want to split the multipoint interface to the collection of logical point-to-point subinterfaces. Frame-Relay implementation in Cisco IOS even supports the use of the same address on different subinterfaces to support such "splitting" procedure. As for PIM Assert winning, make sure the hub router, or the router having direct layer 2 connections with every other node on the NBMA segment always has the best PIM priority on the segment. This will prevent a non-fully connected node from being elected as Assert winner by mistake and improper segment flooding.

What makes Auto-RP troubleshooting slightly more complicated is that RP information is distributed using data-plane multicast messages, not special control-plane signaling. This often results in the need to troubleshoot the data-plane behavior, which is more time-consuming compared to pure control-plane troubleshooting.

Troubleshooting PIM BSR

PIM BSR uses PIM protocol messages for conveying the RP announcements and bootstrap information. The BSR operates in three major steps. First, the BSR routers broadcast their presence and IP address to every router in the network. This procedure uses hop-by-hop flooding and RPF checks on the BSR addresses. Eventually, only one BSR remains active, while others give up their roles. After discovering the BSR, candidate RPs start periodically *unicasting* their presence and group mapping to the BSR. Lastly, the BSR continues flooding BSR announcements with the RPs and their corresponding multicast groups plus some other parameters. Troubleshooting RP information dissemination for PIM BSR is similar, yet simpler than Auto-RP:

- **Step 1:** Ensure that every RP in the domain learns of the BSR. Use the command **show ip pim bsr-router** to validate this. If some RPs appear to be missing this information, perform **mroute** off the candidate RP back to the BSR IP address and see if there are any problems on the path. Fix any problems detected by **mtrace** using the RPF fixup tools described above. If the problem persists, use the command **debug ip pim bsr** on all routers on the reverse path to the BSR and see if there are any RPF failure messages similar to the one below:

  ```
  PIM-BSR(0): bootstrap (150.29.3.3) on non-RPF path Serial0/0/0 or from non-RPF neighbor 155.29.45.4 discarded
  ```

  This message displays the interface that received the BSR message and the expected RPF next-hop. Using this information you may fix the problem per the RPF problem resolution procedures. Move along the path toward the BSR using the same debugging command for troubleshooting of BSR-related data-plane problems. Typical issue could be NBMA split-horizon forwarding problem, which could be solved using PIM NBMA mode.

- **Step 2:** Ensure that BSR receives all RP announcements. Use the command **show ip pim rp mapping** and **show ip pim bsr-router** on the BSR to verify this. The RPs unicast their information to the BSR, so it is enough using the **ping** command from the BSR to every RP's address sourced off the BSR's IP address to test the connectivity.
- **Step 3:** Verify that BSR announcements reach to every router in the topology, not just the candidate RPs. Troubleshooting this process follows the same guidance as used in **Step 1**. There is a caveat with BSR troubleshooting – you cannot change BSR advertisement interval in Cisco IOS, it is fixed to 60 seconds. Therefore, when using the command **debug ip pim bsr** you may waste valuable time if you follow router to router along the **mtrace** path. You may want to adjust your strategy and enable the debugging command on every router along the path at the same time and save debugging information in the syslog buffer.

In general, deploying PIM BSR creates fewer issues compared to Auto-RP. BSR supports operations over NBMA interfaces by the virtue of PIM NBMA mode and it does not rely on dense-mode flooding. Therefore, BSR is significantly easier to troubleshoot in many cases.

Troubleshooting Multicast Data Plane

mroute is an ideal tool for validating almost every aspect of multicast control plane. However, like we mentioned previously, it does not detect all faults, e.g. it does not reveal split-horizon forwarding issues or data-plane filters. Sometimes, it may be necessary to start the actual data plane traffic flows and validate that there are no problems forwarding the multicast packets. Make sure you completed control-plane validation before digging into the data-plane troubleshooting.

Data-plane troubleshooting process starts by joining every router attached to the receiver to the multicast group you are troubleshooting – the group "G". If you have the actual receivers in your network, configure them to join the group. The next step is making the source(s) generate constant flow of ICMP Echo messages. For the sake of simplicity, we assume there is just one source and there are no data-plane filters blocking ICMP traffic in the network. Essentially, we are only concerned with the issues related to multicast routing as opposed to the filtering configuration and such.

Note: If you are using routers to generate multicast traffic flows, there is a caveat. When you issue a **ping** command for a multicast destination address, the router generates multicast packets out of EVERY multicast-enabled interface. On every non-Loopback interface this will result in the respective DR attempting multicast source registration with the RP. If the router sending the multicast flows is DR itself, it will originate PIM Register messages on its own. On contrary, Loopback interfaces enabled for multicast forwarding will simply cycle the multicast packet back to the router and the router will switch it further down the multicast forwarding path. This behavior may result in the extraneous amount of multicast traffic generated off the router. As opposed to using a router with multiple PIM-enabled interfaces, you may want to utilize a stub host device, e.g. a router with a single multicast-enabled interface to produce multicast flows.

Depending on your situation, after you started pinging, you may not receive any responses at all, or you may receive only a few initial responses, followed by abrupt cut in the flow of response. The first condition typically signalizes that the multicast traffic flow does not make it down the shared (S,G) tree. The second condition typically means that the shared tree works, but the shortest-path tree from the receiver to the source fails.

Case 1: Shared Tree Failure. If you cannot get even the first few responses, follow the "divide and conquer" approach and verify whether the sources can register with the RP. Use the command **debug ip pim** on the RP to ensure the Register messages are received and check that the (*,G) state exists in the RP. If the state does not exist in the RP, this means the receivers cannot join it. It is possible to have the (*,G) state if at least one of the receivers have joined but not the others.

Step 1: Run the **mtrace** command from the leaf router toward the RP to identify the nodes on the shortest path. These are the routers you will have to test, starting with the leaf.

Step 2: Use the command **show ip mroute G** on the router you are testing, and ensure there is an (*,G) entry in the multicast routing table. Most likely the entry should be there, or otherwise the **mtrace** command would have failed.

If you previously configured the leaf router to join the multicast group using the command **ip igmp join-group** then you should execute the command **debug ip mpacket**. If the router is not the leaf, or the leaf router did not join the group, use the command **debug ip mpacket fastswitch G** to display the transit multicast packets.

Step 3: Ensure the multicast packets flowing from your source appear in the debugging output, and there are no RPF failures observed. If there are no matches, move one node upstream and repeat the procedure, until you find the point where packets fail the RPF check.

Case 2: Shortest Tree Failure

As mentioned previously, shortest path tree failure usually manifests itself by successful initial pings, followed by failing pings after switching to the SPT. Troubleshooting process follows *the same routine* used to troubleshoot the shared tree, but applies to the tree rooted at the source of multicast traffic. Instead of identifying the reverse path to the RP, trace the path to the source, and use the same **show ip mroute**, **debug ip mpacket** and **debug ip mpacket fastswitch** commands upstream the path to the source.

There is one common data-plane problem often found in the IOS routers, pertaining to the multicast *fast-switching*. It manifests itself with a clean, working control plane and multicast packets being silently discarded by a forwarding hop, typically the one having NBMA connection. Typically you see upstream node connected to an NBMA cloud forwarding packets downstream using fast-switching but the downstream never receiving the packets. The working resolution is disabling multicast fast-switching using the command **no ip mroute-cache** on the upstream router's interface that receives the multicast packets. The problem seems to be fixed in the recent IOS versions that utilize MFIB switching, but it was quite common in the older IOS releases.

Decoding the *show ip mroute* command output

The multicast routing state display could be very helpful, provided that you can read it properly. Take a look at the sample output below, produced on a router that has a single interface enabled for PIM and no actual multicast traffic flows. Notice that the router joins the group 224.0.1.40 to listen to the Auto-RP discovery messages – this is the default behavior for Cisco routers and you cannot change it.

```
Rack29SW3#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
       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

(*, 224.0.1.40), 00:03:53/00:02:11, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Vlan79, Forward/Sparse, 00:03:52/00:02:11
```

There is a single (*,G) entry for the group 224.0.1.40 which is Auto-RP Discovery group address. Look at the times next to the group address –these are Uptime/Expire times, the first one shows how long the group state has been created and the second one showing how soon the group state will expire if not refreshed. Next field is the "RP" which is the RP address for the (*,G) entry. The value of "0.0.0.0" means self, and it appears in the output if the router is the RP itself, or the (*,G) state has been created for a dense group. Keep in mind that IOS always creates an (*,G) state for dense traffic flows, even though it is not used for actual traffic forwarding. The flags "DCL" mean that the multicast state is forwarded in dense mode, "C" means there is a group-member directly connected and "L" means the router itself joined the group. The "Incoming interface" is set to Null, which means there is no incoming traffic for this group. Also, the RPF neighbor is set to 0.0.0.0 which means "self". Finally, there is an outgoing interface list (OIL) listing the outgoing interface, possibly the next-hop router (in PIM NBMA mode) and Uptime/Expire timers for this forwarding adjacency.

Let's have a look at a more informative example:

```
Rack29R6#show ip mroute 224.0.1.39
IP Multicast Routing Table
...

(*, 224.0.1.39), 00:11:44/stopped, RP 0.0.0.0, flags: D
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0.146, Forward/Sparse-Dense, 00:11:44/00:00:00

(150.29.6.6, 224.0.1.39), 00:11:44/00:02:56, flags: T
  Incoming interface: Loopback0, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0.146, Forward/Sparse-Dense, 00:11:44/00:00:00
```

There is an (S,G) entry in this table, which has the flag "T" meaning it's a shortest-path and not a shared tree construct. The incoming interface is set to Loopback0 and RPF neighbor to "0.0.0.0" which means the local router is the traffic source. Have a look at the output for the same traffic flow on transit router:

```
Rack29R4#show ip mroute 224.0.1.39
IP Multicast Routing Table
…

(*, 224.0.1.39), 21:24:46/stopped, RP 0.0.0.0, flags: D
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/1/0, Forward/Sparse, 21:24:46/00:00:00
    FastEthernet0/1, Forward/Sparse-Dense, 21:24:46/00:00:00

(150.29.6.6, 224.0.1.39), 21:21:42/00:02:55, flags: T
  Incoming interface: FastEthernet0/1, RPF nbr 155.29.146.6
  Outgoing interface list:
    Serial0/1/0, Forward/Sparse, 21:21:42/00:00:00
```

The (S,G) entry now has an incoming interface and RPF neighbor IP address in the output. Here is another interesting entry:

```
Rack29R4#show ip mroute 224.0.1.40
IP Multicast Routing Table
...

(*, 224.0.1.40), 21:35:41/stopped, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/1/0, Forward/Sparse, 21:35:41/00:00:00
    FastEthernet0/1, Forward/Sparse, 21:35:41/00:00:00

(150.29.5.5, 224.0.1.40), 00:02:26/00:00:33, flags: L
  Incoming interface: Null, RPF nbr 155.29.0.5
  Outgoing interface list:
    FastEthernet0/1, Forward/Sparse, 00:02:26/00:00:00
    Serial0/1/0, Forward/Sparse, 00:02:26/00:00:00
```

Look at the second entry that has an incoming interface value of "Null" and RPF neighbor 155.29.0.5. This typically means there is an RPF failure for this particular source. Indeed if you an **mtrace** it will confirm our guess:

```
Rack29R4#mtrace 150.29.5.5
Type escape sequence to abort.
mtrace from 150.29.5.5 to 155.29.0.4 via RPF
From source (?) to destination (?)
Querying full reverse path...
 0  155.29.0.4
-1  155.29.0.4 None No route
```

Therefore, simply reading the **mroute** state table may point to an RPF failure. Using a static **mroute** will provide a fix, as shown in the output below:

```
Rack29R4#show ip mroute 224.0.1.40
IP Multicast Routing Table
...

(150.29.5.5, 224.0.1.40), 00:01:44/00:01:15, flags: L
  Incoming interface: Serial0/1/0, RPF nbr 155.29.45.5, mroute
  Outgoing interface list:
    FastEthernet0/1, Forward/Sparse, 00:01:44/00:00:00
```

Two other interesting flags are "J" and "F". The J flag means the respective (*,G) state is to be switched the SPT by the leaf router. The "F" flag is typically found for the states created at the PIM DR router – it signalizes the forwarding states that correspond to the flows being registered with the RP. If the "F" flag persists, then your router is most likely not receiving the PIM Register-Stop messages back from the RP, and thus there are sources that has not switched to the SPT tree.

Summary

This paper presents a systematic approach to troubleshooting single-AS PIM-SM failures. The method features separate troubleshooting of control and data plane, with control-plane troubleshooting precluding the data plane. The central troubleshooting tool for control-plane validation is the **mroute** command that is commonly overlooked in many troubleshooting guides. The same approach could be used for more complex scenarios, such as multicast VPNs or Inter-AS multicast – just some additional actions might be needed to validate MSDP or per-VRF specific features.

Tags: auto-rp, ccie, multicast troubleshooting, pim bsr, pim dm, pim sm

Download this page as a PDF

About Petr Lapukhov, 4xCCIE/CCDE:
Petr Lapukhov's career in IT begain in 1988 with a focus on computer programming, and progressed into networking with his first exposure to Novell NetWare in 1991. Initially involved with Kazan State University's campus network support and UNIX system administration, he went through the path of becoming a networking consultant, taking part in many network deployment projects. Petr currently has over 12 years of experience working in the Cisco networking field, and is the only person in the world to have obtained four CCIEs in under two years, passing each on his first attempt. Petr is an exceptional case in that he has been working with all of the technologies covered in his four CCIE tracks (R&S, Security, SP, and Voice) on a daily basis for many years. When not actively teaching classes, developing self-paced products, studying for the CCDE Practical & the CCIE Storage Lab Exam, and completing his PhD in Applied Mathematics.
Find all posts by Petr Lapukhov, 4xCCIE/CCDE | Visit Website

You can leave a response, or trackback from your own site.

18 Responses to "Troubleshooting Multicast Routing"

Vik — September 17, 2010 at 9:05 am

Hi Petr,

Thank you for posting this article. Would it be possible for you to confirm a few queries I had on this article.

1. Are you referring to the mtrace or "sh ip mroute" command in the sections listed below

"
Step 2: ………. Next, from every leaf router that has a receiver attached, issue the **mroute** command back to the RP address. Use the command mtrace [RP-IP-Address] to accomplish this. For more information about the **mroute** command, refer to the section **Understanding the mtrace command**.

Step 3: …. From the RP corresponding to group "G", use the **mroute** command"

Step 4: ….. Use the **mroute** command and trace the route back to the "S" addresses.
"

Another thing I wanted to confirm was with regards to the "F" flag. Isn't this flag indicative that you are the first hop router and hence you need to do the register. My understanding was that it never gets cleared even if you receive the register stop. Could you please confirm this.

Finally all the diagrams on the blog always come out really sharp and clear. Can you please post which program do you use to create diagrams. I have not been able to recreate the same polished look on my diagrams using visio and there is always some discoloration when converting to jpg etc. Thx for your help.

Reply

Petr Lapukhov, CCIE #16379 — September 17, 2010 at 9:26 am

@Vik

My apologies, I was doing search/replace and messed up some keywords :) I fixed the issues by now, so please take another look.

Reply

Ashish — September 17, 2010 at 9:41 am

Halfway through the article I couldnt stop myself checking out who is the author…Great article..Thanks Peter

Reply

Vik — September 17, 2010 at 10:58 am

Hi Petr,

Thanks for responding so quickly. Could you possibly answer the question about the F flag when you get a chance. Thx

Reply

ovidiu — September 17, 2010 at 3:13 pm

Great article Petr !!!!! it will help us a lot in achieving 100% in multicast.
Would be great if you can write a similar one for MPLS TE :)
br
Ovidiu

Reply

Arnaud — September 17, 2010 at 5:30 pm

This is a best article I read on multicast routing troubleshooting.
I am now more organized on my troubleshooting approach after reading your great work

Thanks Petr.

Reply

Nadeem Rafi — September 20, 2010 at 1:40 am

One of the bes articles on MC…….

Reply

Blog Post Catalogue | CCIE Blog — September 21, 2010 at 1:21 am

[…] Troubleshooting Multicast Routing […]

Reply

ssa — October 3, 2010 at 4:59 am

Hi Petr,
I was trying to lab what you mention about rpf checks that is applied to register messages
'The same applies to PIM Register message, i.e. the encapsulated multicast frame's source IP address should be reachable over the interface where packet was received'
However i dont see any rpf failures please check the below.

R3 is the RP, another router will be sending register message to R3
the source address in the encapculated multicast packet is 6.6.6.6 In the R3 the rpf interface is s 1/0 as shown below.
R3#sh ip rpf 6.6.6.6
RPF information for ? (6.6.6.6)
RPF interface: Serial1/0
RPF neighbor: ? (10.1.34.4)
RPF route/mask: 6.6.6.6/32
RPF type: unicast (ospf 100)
RPF recursion count: 0
Doing distance-preferred lookups across tables
R3#

since I did route manipulation R3 will recieve the register message from e0/0. I dont see any rpf failure here and pings are success full.
Please check the details below.
What I expect to see here is RPF failure for register message since rpf interface is s1/0 while packet is coming from e 0/0.

*Mar 1 00:22:31.583: PIM(0): Received v2 Register on Ethernet0/0 from 10.1.56.6
*Mar 1 00:22:31.587: for 6.6.6.6, group 239.3.3.3
*Mar 1 00:22:31.595: PIM(0): Insert (6.6.6.6,239.3.3.3) join in nbr 10.1.34.4's queue
*Mar 1 00:22:31.599: PIM(0): Forward decapsulated data packet for 239.3.3.3 on Ethernet0/0
*Mar 1 00:22:31.603: PIM(0): Building Join/Prune packet for nbr 10.1.34.4
*Mar 1 00:22:31.603: PIM(0): Adding v2 (6.6.6.6/32, 239.3.3.3), S-bit Join
*Mar 1 00:22:31.607: PIM(0): Send v2 join/prune to 10.1.34.4 (Serial1/0)

ping
Protocol [ip]:
Target IP address: 239.3.3.3
Repeat count [1]: 10
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Interface [All]: loopback0
Time to live [255]:
Source address: 6.6.6.6
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]: r
Number of hops [9]:
Loose, Strict, Record, Timestamp, Verbose[RV]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 10, 100-byte ICMP Echos to 239.3.3.3, timeout is 2 seconds:
Packet sent with a source address of 6.6.6.6
Packet has IP options: Total option bytes= 39, padded length=40
Record route:
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)

Request 0 timed out
Reply to request 1 from 10.1.34.3, 272 ms
Reply to request 2 from 10.1.34.3, 332 ms
Reply to request 3 from 10.1.34.3, 288 ms
Reply to request 4 from 10.1.34.3, 592 ms
Reply to request 5 from 10.1.34.3, 396 ms
Reply to request 6 from 10.1.34.3, 600 ms
Reply to request 7 from 10.1.34.3, 632 ms
Reply to request 8 from 10.1.34.3, 708 ms
Reply to request 9 from 10.1.34.3, 668 ms

Reply

Petr Lapukhov, 4xCCIE/CCDE — October 3, 2010 at 7:43 am

@ssa

I think your point is correct: the RP actually treats the packet as arriving on a "tunnel" interface sourced at the DR. However, on second look, the RPF check does apply to the encapsulated source address in oerder to build the SPT tree toward the multicast source. You can see it in your debugging output , which says that PIM Join is being sent out of Serial 0/1 (RPF) interface.

Thanks for your help here, I will edit the text!

Reply

ssa — October 5, 2010 at 10:18 am

Hi Petr,

i dont understant clearly.
In the example the source adress in the encapsulated packet is 6.6.6.6 for 6.6.6.6 the rpf interface is Serial1/0 however unicast packet register packet (which includes encapsulated multicast packet) arrives at e 0/0. What sort of rpf check applied to this source (6.6.6.6) ?

Thanks
Sinan

Reply

Petr Lapukhov, 4xCCIE/CCDE — October 5, 2010 at 4:09 pm

@ssa

Well if I remember it right, you were making the register message arrive on the interface that was NOT RPF for encapsulated packet. That encapsulation packet did not have any RPF checks applied, indeed. However, when the RP starts building the SPT toward the source it still looks up the RPF interface to send a PIM Join message out based on the encapsulated packet's source IP address.

Reply

ssa — October 7, 2010 at 1:13 am

Then we can say we shouldnt worry about RPF interface for register message. Either for encapsulated packet or register packet itself.

Reply

lee_maynard2003 — February 14, 2011 at 3:33 am

Hi I have an example of the Shortest Tree Failure old IOS bug fixed with no ip mroute-cache if anyone is interested.

http://ieoc.com/forums/p/14694/128402.aspx#128402

THANKS

Reply

lee_maynard2003 — February 22, 2011 at 7:03 am

Hi Petr,

First off I love this doc but I bet you cant explain this …

http://ieoc.com/forums/t/13471.aspx

Thanks,

Lee

Reply

Troubleshooting IP Multicast Routing (Reference blog.ine.com) | dtechquest — March 16, 2013 at 6:03 am

[…] with the ones closest to BSR/MA and check that they have the RP mapping information. Refer to Troubleshooting Auto-RP and Troubleshooting PIM BSR for detailed techniques on fixing the Auto-RP/BSR […]

Reply

Lamontra — September 4, 2013 at 6:09 pm

Why before the creation of the SPT-Tree theres is no RPF failure in Sparse-Mode or even when using ip pim spt-threshold infinity?

Reply

ccie: Troubleshooting Multicast Routing | ipv46.com — February 7, 2014 at 8:48 am

[…] http://blog.ine.com/2010/09/17/troubleshooting-multicast-routing/ […]

Reply

Leave a Reply

Name (required)

Mail (will not be published) (required)

Submit Comment

Search

Search
Submit

Categories

Select Category

CCIE Bloggers

Brian Dennis, CCIEx5 #2210
- Routing & Switching
- Voice
- Security
- Service Provider
- ISP Dial

Brian McGahan, CCIEx4 #8593, CCDE #2013::13
- Design
- Data Center
- Routing & Switching
- Security
- Service Provider

Mark Snow , CCIEx3 #14073
- Data Center
- Voice
- Security

Petr Lapukhov, CCIEx4 #16379, CCDE #2010::7
- Design
- Routing & Switching
- Security
- Service Provider
- Voice

Popular Posts

- CCIE RSv5 ATC & Baby 3.0 Status Update
- MPLS 101 for CCIE Candidates - Free to Attend
- #CCIE RSv5 ATC Continues Today - Week of 2014-07-07