Copyright Information

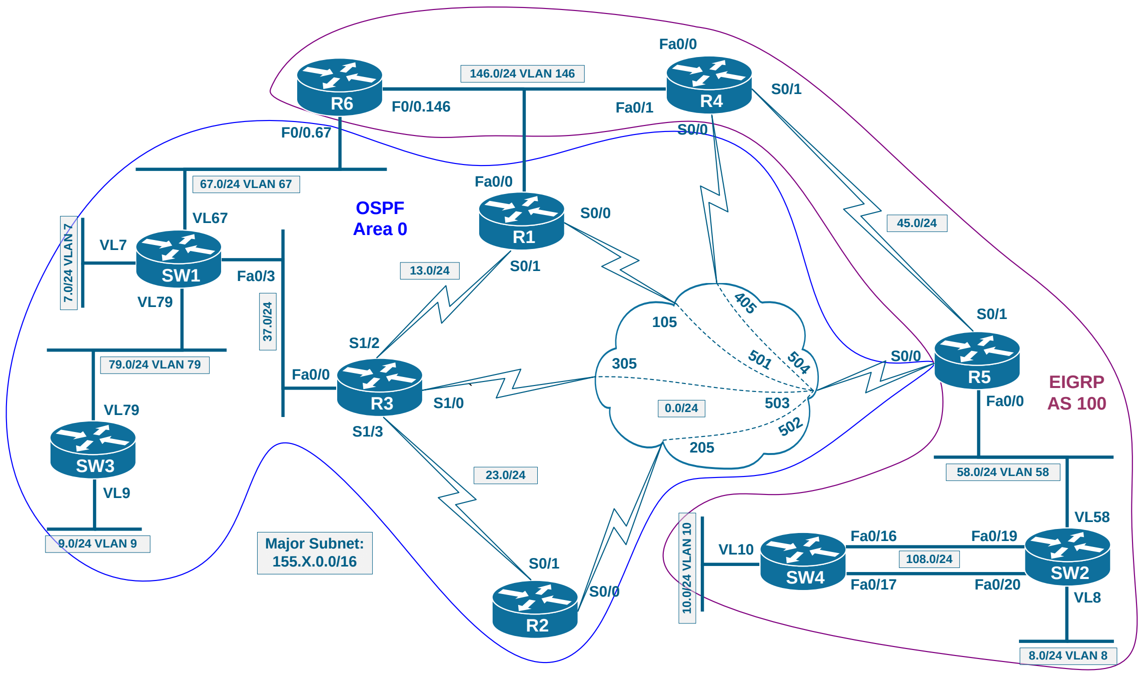
Disclaimer

Table of Contents

Multicast

Note

Load the *Multicast* initial configurations on all devices prior to starting. Refer to the diagram below when working with the following tasks.



8.1 PIM Dense Mode

- Enable dense-mode multicast delivery on the path between R6 and SW4.
- Do not enable PIM on the Frame-Relay cloud.
- To test this configuration, configure SW4's VLAN 10 interface to join the multicast group 224.10.10.10.
- Make sure R6 can ping this multicast group.

8.2 Multicast RPF Failure

- Disable PIM on the Serial link between R4 and R5 and activate it on Frame-Relay connection between R4 and R5.
- Ensure that R6 can still ping the group 224.10.10.10.
- Set R5's periodic RPF checks interval to 6 seconds.
- Configure R5 to perform triggered RPF check 10 ms after the topology change.
- Ensure the delay will never back off above 200 ms.

8.3 PIM Sparse Mode

- Remove the previous PIM Dense Mode configuration and replace it with Sparse Mode on all interfaces.
- Use R5's Loopback0 interface as the static Rendezvous Point address.
- Configure SW4's VLAN 10 interface to join the group 224.20.20.20, and ensure that SW3 can send multicast packets to this segment.
- Perform SPT switchover only when the traffic flow exceeds 128Kbps.

8.4 PIM Sparse-Dense Mode

- Remove the previous PIM Sparse Mode configuration and replace it with Sparse-Dense-Mode on all interfaces.
- Modify the static RP assignments so that R5 is only used as the RP for groups in the range 224.0.0.0/8.
- Configure SW4's VLAN 10 interface to join the groups 224.30.30.30 and 239.0.0.1, and ensure that SW3 can send multicast packets to this segment for both groups.

8.5 PIM Assert

- Enable multicast routing in R1 and configure PIM sparse-dense mode on Ethernet interface of R1.
- Configure PIM sparse-dense mode on the connection between R1 and R5.
- Use the same RP assignment that was configured in other routers.
- Ensure R1 is always elected via PIM to flood the shared VLAN146 segment.

8.6 PIM Accept RP

- Ensure that SW2 and R5 will only accept (*,G) joins toward R5's Loopback0 for groups 224.10.10.10 and 224.110.110.110

8.7 PIM DR Election

- Ensure that R1 is responsible for multicast source registration with RP on VLAN146 segment.

8.8 PIM Accept Register

- Configure R5 so that the only source allowed on VLAN146 segment is R6.
- Your configuration should not affect sources in any other segments.

8.9 Multicast Tunneling

- Without configuring multicast routing on the transit nodes, make sure that receivers at SW3 may receive multicast feeds from R1.
- Do not configure any additional IGP adjacencies to accomplish this task.

8.10 PIM NBMA Mode

- Configure R3 to join the multicast group 224.110.110.110 and make sure R6 can ping it.

8.11 Auto-RP

- Remove all static RP settings and previous Accept-RP/Register configuration.
- Using Cisco proprietary technology, configure so that R5 advertises itself as the RP for all multicast groups.

8.12 Auto-RP - Multiple Candidate RPs

- Configure SW2 and SW4 as RPs for group ranges 224.0.0.0-231.255.255.255 and 232.0.0.0-239.255.255.255 respectively.
- In case of one RP failure, the other one should provide backup for its groups.
- The group 224.110.110.110 should be always switched in dense-mode.

8.13 Auto-RP - Filtering Candidate RPs

- Configure R5 to filter announces for the group 224.110.110.110 received from SW4.

8.14 Auto-RP Listener

- Switch from PIM SM/DM on all interfaces to PIM SM.
- Ensure all routers still hear Auto-RP announces.

8.15 Auto-RP and RP/MA Placement

- Configure R1 as the RP instead of R5.
- Ensure R3 can hear the RP discovery messages sent by R1.

8.16 Filtering Auto-RP Messages

- Configure R1 so that SW4 cannot hear the Auto-RP discovery messages.

8.17 Multicast Boundary

- Configure R5's connection to VLAN58 so that traffic to the group range 232.0.0.0/5 cannot reach SW2.
- Filter the Auto-RP messages to remove the information about this group range.

8.18 PIM Bootstrap Router

- Remove the Auto-RP configuration in SW2, SW4 and R1.
- Remove the multicast boundary in R5.
- Configure R5 to advertise itself as the RP for all multicast groups using the standards-based protocol.

8.19 BSR - Multiple RP Candidates

- Configure SW2 and SW4 to advertise themselves as RP candidates for all multicast groups.
- R5 should distribute this information and instruct all routers to load-balance multicast groups between the two RPs.
- Use the maximum possible hash mask length to evenly distribute the load across the RPs.

8.20 Filtering BSR Messages

- Configure your network so that R6 does not learn any RP information,

8.21 Stub Multicast Routing & IGMP Helper

- Remove the tunnel connecting R1 and R3. Configure R3 as the stub multicast router with SW1 emulating a host on the stub segment.

8.22 IGMP Filtering

- Only permit R3 to accept IGMP joins for groups in range 239.1.1.0/24 on its connection to SW1.
- Limit the number of concurrent IGMP states on the interface to 10.

8.23 IGMP Timers

- Configure the designated IGMP querier on VLAN146 segment so that failed multicast traffic receivers are detected and removed within 60 seconds.
- Every active receiver should respond to general IGMP queries within 4 seconds.
- Designated querier failures should be detected by the next candidate twice as fast than by default.
- Since there is just one receiver on R3's connection to SW1, configure the router to remove group states immediately - as soon as IGMP leave report is received.

8.24 Multicast Helper Map

- When R6 sends broadcast UDP packets on port 5000, those packets should be transported across the network and broadcasted on VLAN37 segment.
- Use the group 239.1.1.100 to accomplish this task and use the DNS broadcasts for testing.
- You may use static mroutes if needed to accomplish the task.

8.25 Multicast Rate Limiting

- Configure R3 to limit the aggregate rate of the multicast traffic leaving VLAN37 interface to 1Mbps.
- Any source sending to the group IP 239.1.1.7 should be limited to 128Kbps.
- Multicast feed from R6 to the IP address 239.1.1.100 should be limited to 256Kbps.

8.26 Bidirectional PIM

- The group range 238.0.0.0/8 is used for network video-conferencing with many participants.
- Configure the network so that this group uses single shared tree rooted in R5 for multicast traffic delivery.

8.27 Source Specific Multicast

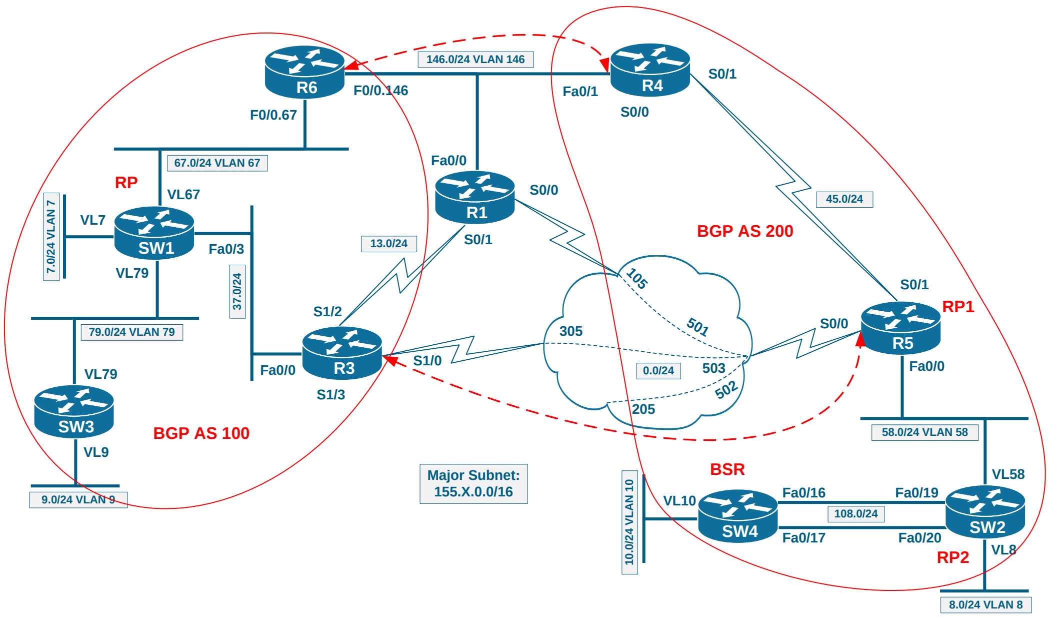
- Using the default multicast group range enable PIM SSM functionality in your network.
- R6 should join the multicast feed (150.1.10.10,232.6.6.6).

8.28 DVMRP Interoperability

- There is a UNIX server on the VLAN146 segment running mrouted daemon. This server connects your network to the MBONE.
- Allow the server to learn about multicast sources known to R4 via EIGRP.
- R4 should be able to use received DVMRP updates for RPF checks.
- Disable DVMRP auto-summarization and only send sources in the ranges 155.X.0.0/16 to the server.

Note

At this point, erase all devices running configs and load the *Inter Domain Multicast* initial configuration files. Refer to the diagram below, when working with the tasks that follow.



8.29 Multicast BGP Extension

- Enable multicast exchange between AS 100 and AS200 on both peering links.
- Multicast traffic should preferably be routed across the Frame-Relay link, using the Ethernet link as backup.

8.30 MSDP

- Configure your network to enable exchange of multicast traffic between sources located in different ASes.

8.31 Anycast RP

- Instead of relying on BSR protocol to distribute load between the RPs in AS 200, implement a solution that hides both RPs behind the same RP IP address 150.1.100.100.
- Every RP should inform the other one of the active sources registered with them.

8.32 Catalyst IGMP Snooping

- For hosts on VLAN146, enable the IGMPv2 processing that immediately removes the port from the flood list once a Leave message is detected.
- Configure SW1 for this task.

8.33 Catalyst Multicast VLAN Registration

- Configure SW1 so that multicast traffic sent by R1 on VLAN146 is received by R5 on VLAN58.
- Use the configuration that allows R1 dynamically tracking sources in other VLANs.

8.34 Catalyst IGMP Profiles

- Configure SW4 to permit only the IGMP reports for groups in range 232.0.0.0/8 from R4.

Multicast Solutions

Note

Load the *Multicast* initial configurations prior to starting.

8.1 PIM Dense Mode

- Enable dense-mode multicast delivery on the path between R6 and SW4.
- Do not enable PIM on the Frame-Relay cloud.
- To test this configuration, configure SW4's VLAN 10 interface to join the multicast group 224.10.10.10.
- Make sure R6 can ping this multicast group.

Configuration

Note

Multicast traffic distribution uses the concept of "constrained" flooding. This means that packets destined to a particular multicast group are flooded only on the links that lead to the actual group subscribers. Constrained flooding procedure consists of two parts

1) Building a multicast distribution tree for a group – aka Shortest Path Tree or SPT.
2) Flooding the actual multicast packets down the SPT, using Reverse Path Forwarding (RPF).

We are going to work mostly with Protocol Independent Multicast (PIM) Dense Mode and Sparse Mode. PIM is a special signaling protocol that does not distribute any routing information on its own. Rather, PIM uses the unicast routing table to perform RPF checks. This procedure is the core of the multicast routing. When the router receives a multicast packet, it looks at the source IP address for this packet. The router looks up the source IP address in the unicast routing table, and determines the outgoing interface for this packet. If this outgoing interface does not match the interface where the packet was received, the router drops the packet. This behavior intends to eliminate potential packet loops, which may occur when routers flood multicast packets. Only the immediate upstream router (with respect to the source of the multicast stream) is allowed to send us multicast packets.

If the packet passes the RPF check, the router will determine the outgoing interfaces list (OIL) for this packet, i.e. the branches of the multicast distribution tree. The OIL never includes the input interface – this is the well-known split horizon rule. The SPT should be signaled by PIM based on the active subscribers across the network. However, PIM Dense Mode (PIM DM) does not use any explicit signaling to join a multicast distribution tree. Instead of that, it uses inverse logic approach. All routers initially flood packets out of all their multicast-enabled interfaces. If the downstream router determines that it does not have any directly connected subscribers or other routers willing to receive multicast traffic, it sends a special PIM Prune message to the upstream router. The upstream router then excludes the particular interface from the OIL for the pruned group.

PIM DM works fine in small networks with a lot of subscribers. However, it has one inherent scalability limitation – excessive flooding. Every Pruned interface state expires in 3 minutes by default, and the flooding out of this interface resumes again, until the upstream does not receive another PIM Prune message on the interface. This is needed to ensure that no node in the network potentially misses multicast traffic. This periodic flood and prune behavior is what makes PIM Dense mode non-scalable.

The obvious benefit of PIM DM is its “plug-and-play” behavior. As soon as you configure PIM DM in your network, you can start sending multicast traffic and the traffic is flooded to all interested subscribers.

R4:
```
ip multicast-routing
!
interface FastEthernet 0/1
 ip pim dense-mode
!
interface Serial 0/1
 ip pim dense-mode
```

R6:
```
ip multicast-routing
!
interface FastEthernet 0/0.146
 ip pim dense-mode
```

R5:
```
ip multicast-routing
!
interface Serial 0/1
 ip pim dense-mode
!
interface FastEthernet 0/0
 ip pim dense-mode
```

SW2:
```
ip multicast-routing distributed
!
interface Vlan 58
 ip pim dense-mode
!
interface Port-Channel1
 ip pim dense-mode
```

SW4:
```
ip multicast-routing
!
interface Port-Channel1
 ip pim dense-mode
!
interface Vlan 10
 ip igmp join-group 224.10.10.10
 ip pim dense-mode
```

Verification

Note

Start your basic verification looking for PIM neighbors and PIM interfaces. Notice the PIM version and mode on every interface and make sure they match the requirements.

```
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
155.1.45.4        Serial0/1                00:09:43/00:01:22 v2    1 /
S
155.1.58.8        FastEthernet0/0          00:09:28/00:01:35 v2    1 /
DR S

Rack1R5#show ip pim interface

Address          Interface                Ver/   Nbr    Query  DR
DR
                                          Mode   Count  Intvl  Prior
155.1.45.5       Serial0/1                v2/D   1      30     1
0.0.0.0
155.1.58.5       FastEthernet0/0          v2/D   1      30     1
155.1.58.8
```

Note

Ping the group and confirm that the packets reach the destination.

```
Rack1R6#ping 224.10.10.10 repeat 100

Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 224.10.10.10, timeout is 2 seconds:

Reply to request 0 from 155.1.108.10, 32 ms
Reply to request 1 from 155.1.108.10, 32 ms
Reply to request 2 from 155.1.108.10, 28 ms
```

Note

Now verify multicast route states created by the traffic flow. You should see (S,G) entries corresponding to the dense-mode SPT. Notice that (*,G) states have all potential interfaces within their OILs. Look for RPF neighbor for each entry, and make sure they match the traffic flow. The RPF neighbor of 0.0.0.0 means this router is connected to the source.

```
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

(*, 224.10.10.10), 00:00:24/stopped, RP 0.0.0.0, flags: D
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/1, Forward/Dense, 00:00:24/00:00:00
    FastEthernet0/1, Forward/Dense, 00:00:24/00:00:00

(155.1.146.6, 224.10.10.10), 00:00:24/00:02:57, flags: T
  Incoming interface: FastEthernet0/1, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/1, Forward/Dense, 00:00:25/00:00:00

(*, 224.0.1.40), 00:02:26/00:02:05, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/1, Forward/Dense, 00:01:49/00:00:00
    FastEthernet0/1, Forward/Dense, 00:02:26/00:00:00

Rack1R4#
```

```
Rack1R5#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

(*, 224.10.10.10), 00:00:42/stopped, RP 0.0.0.0, flags: D
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Dense, 00:00:42/00:00:00
    Serial0/1, Forward/Dense, 00:00:42/00:00:00

(155.1.146.6, 224.10.10.10), 00:00:42/00:02:57, flags: T
  Incoming interface: Serial0/1, RPF nbr 155.1.45.4
  Outgoing interface list:
    FastEthernet0/0, Forward/Dense, 00:00:43/00:00:00

(*, 224.0.1.40), 00:02:07/00:02:59, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Dense, 00:01:52/00:00:00
    Serial0/1, Forward/Dense, 00:02:07/00:00:00

Rack1R5#
Rack11AS>8
[Resuming connection 8 to sw2 ... ]

Rack1SW2#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
       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

(*, 224.10.10.10), 00:00:53/stopped, RP 0.0.0.0, flags: D
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Port-channel1, Forward/Dense, 00:00:53/00:00:00
    Vlan58, Forward/Dense, 00:00:53/00:00:00
```

```
(155.1.146.6, 224.10.10.10), 00:00:53/00:02:52, flags: T
  Incoming interface: Vlan58, RPF nbr 155.1.58.5
  Outgoing interface list:
    Port-channel1, Forward/Dense, 00:00:53/00:00:00

(*, 224.0.1.40), 00:01:48/00:02:49, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Port-channel1, Forward/Dense, 00:01:34/00:00:00
    Vlan58, Forward/Dense, 00:01:48/00:00:00

Rack1SW2#
```

```
Rack1SW4#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
       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

(*, 224.10.10.10), 00:01:32/stopped, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Port-channel1, Forward/Dense, 00:01:32/00:00:00
    Vlan10, Forward/Dense, 00:01:32/00:00:00

(155.1.146.6, 224.10.10.10), 00:01:07/00:02:59, flags: LT
  Incoming interface: Port-channel1, RPF nbr 155.1.108.8
  Outgoing interface list:
    Vlan10, Forward/Dense, 00:01:07/00:00:00, H

(*, 224.0.1.40), 00:01:33/00:02:16, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Port-channel1, Forward/Dense, 00:01:33/00:00:00
```

8.2 Multicast RPF Failure

- Disable PIM on the Serial link between R4 and R5 and activate it on Frame-Relay connection between R4 and R5.
- Ensure that R6 can still ping the group 224.10.10.10.
- Set R5's periodic RPF checks interval to 6 seconds.
- To reduce the load on router's CPU configure R5 to perform triggered RPF check 10 ms after the topology change but don't make the delay bigger than 200ms.

Configuration

Note

The RPF check procedure has been discussed in the previous task. RPF failures occur in two general situations: packet flooded out of wrong interface (good, prevents looping) or unicast shortest paths not matching multicast distribution trees (bad, incorrect multicast routing configuration). The situations of the second type generally occur in the following cases:

1) Duplicate paths to the multicast source exist and PIM is not enabled on all links. In this situation, the router may receive multicast packets on the interface that is not on the shortest path to the source. As an extreme case of this situation, PIM might be enabled on the links where IGP is not running.

2) There are multiple routing protocols in the network, and the router might have paths to the source learned via different protocols. In this situation, the router may receive multicast packets on the interface running RIP, while thinking the shortest path to the source is via OSPF.

3) Use of static routes may change local router's perception of the shortest paths and force it to reject otherwise valid multicast packets.

Additionally, temporary RPF failures may happen in situations when the network is unstable and the routers keep changing their shortest path tables. The best way to find and isolate the RPF issue in your lab is probably to run the **`debug ip mpacket`** command on all routers, starting from the one closest to the destination and moving upstream to the source. This command will show you process-switches multicast packets and signal any RPF issues. Note that you will need to enter the command **`no ip mroute-cache`** on all interfaces where multicast packets are received, to disable fast-switching of the multicast packets.

The easiest way to fix RPF failure is to manually provide RPF information using the **ip mroute** command. This command is local to the router and specifies the RPF next-hop or interface for the given subnet. The syntax for the command is **ip mroute <IP> <Mask> {RPF-IP-Address|<Interface-Name>} [distance]**. It is important to understand that this command does not specify any forwarding rules. Instead, it creates the ordered table of entries to look up for RPF information. When a multicast packet comes in, the router will first try finding a matching entry in the mroute table, to determine the RPF interface. Note that the mroute table is ordered in the same way you enter the ip mroute commands, so you should enter the most specific mroutes first. When the router finds RPF information in both mroute table and the unicast routing table, it prefers mroute information, since it has better distance (zero). However, connected routes still have preference over any other RPF source. You may change the mroute administrative distance to make it less preferred than the unicast routing table information.

When a network topology change occurs, the routers will delay their RPF interface re-computation. You may change this delay using the commands **ip multicast rpf backoff <min-delay ms> <max-delay ms>.** The **min-delay** value specifies the amount of milliseconds to wait after the topology change to re-calculate RPF interfaces. This delay is doubled after every consequent topology change that happens within the **max-delay** window. The delay never becomes larger that the **max-delay** field. Additionally, you tune periodic RPF interface computations using the command **ip multicast rpf interval <interval in seconds>**.

```
R4:
interface Serial 0/1
 no ip pim dense-mode
!
interface Serial 0/0.1
 ip pim dense-mode

R5:
interface Serial 0/1
 no ip pim dense-mode
!
interface Serial 0/0
 ip pim dense-mode
!
ip multicast rpf interval 6
ip multicast rpf backoff 10 200
```

Verification

Note

Try pinging the group before you applied the mroute fixup.

```
Rack1R6#ping 224.10.10.10 repeat 100

Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 224.10.10.10, timeout is 2 seconds:
..
```

Note

Check the multicast routing table in R5. Notice that the (S,G) state for our flow has no Incoming Interface field (it's Null) means the traffic is arriving not on the RPF interface.

```
Rack1R5#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

(*, 224.10.10.10), 00:05:57/stopped, RP 0.0.0.0, flags: D
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/0, Forward/Dense, 00:05:57/00:00:00
    FastEthernet0/0, Forward/Dense, 00:05:57/00:00:00

(155.1.146.6, 224.10.10.10), 00:05:57/00:00:25, flags:
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Dense, 00:05:58/00:00:00

(*, 224.0.1.40), 00:34:33/00:02:34, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/0, Forward/Dense, 00:06:16/00:00:00
    FastEthernet0/0, Forward/Dense, 00:34:18/00:00:00
```

Note

You can easily see RPF failures when you enable multicast packet debugging.

```
Rack1R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1R5(config)#interface Serial 0/0
Rack1R5(config-if)#no ip mroute-cache

Rack1R5#debug ip mpacket
IP multicast packets debugging is on
Rack1R5#
IP(0): s=155.1.146.6 (Serial0/0) d=224.10.10.10 id=300, ttl=253,
prot=1, len=104(100), not RPF interface
Rack1R5#
IP(0): s=155.1.146.6 (Serial0/0) d=224.10.10.10 id=301, ttl=253,
prot=1, len=104(100), not RPF interface
Rack1R5#
```

Note

Now apply the fixup using static multicast route in R5. Static default mroute would be enough in this situation.

```
Rack1R5(config)#ip mroute 0.0.0.0 0.0.0.0 155.1.0.4
Rack1R5(config)#
Rack11AS>6
[Resuming connection 6 to r6 ... ]
...
Reply to request 98 from 155.1.108.10, 48 ms
Reply to request 99 from 155.1.108.10, 48 ms
Rack1R6#
```

Note

Now the corresponding mroute state has valid input interface and OIL is not empty. Notice that RPF neighbor information is taken from the mroute cache, per the output of the show command.

```
Rack1R5#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

(*, 224.10.10.10), 00:09:16/stopped, RP 0.0.0.0, flags: D
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Dense, 00:09:16/00:00:00
    Serial0/0, Forward/Dense, 00:09:16/00:00:00

(155.1.146.6, 224.10.10.10), 00:02:52/00:02:56, flags: T
  Incoming interface: Serial0/0, RPF nbr 155.1.0.4, Mroute
  Outgoing interface list:
    FastEthernet0/0, Forward/Dense, 00:02:53/00:00:00

(*, 224.0.1.40), 00:37:52/00:02:13, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Dense, 00:37:37/00:00:00
    Serial0/0, Forward/Dense, 00:09:35/00:00:00
```

8.3 PIM Sparse Mode

- Remove PIM Dense Mode configuration and replace it with Sparse Mode on all interfaces.
- Use R5's Loopback0 interface as the static Rendezvous Point address.
- Configure SW4's VLAN 10 interface to join the group 224.20.20.20, and ensure that SW3 can send multicast packets to this segment.
- Perform SPT switchover only when the traffic flow exceeds 128Kbps.

Configuration

Note

PIM Sparse Mode is the scalable version of the multicast signaling protocol. Instead of flooding multicast traffic to all potential receivers, PIM SM builds explicit multicast distribution trees from the receivers to the sources. This model does not consume large amounts of network resources like PIM DM's flood-and-prune behavior does.

In order to build an explicit tree to the source, receivers and sources should have a way to dynamically discover each other. This is facilitated by the use of Rendezvous Points or RPs. RP is a special router known to both sources and receivers. If a receiver is interested in traffic for multicast group G, it first builds a multicast distribution tree towards the RP as the fictive "source". This is facilitated by PIM Join messages, and the resulting tree is called shared tree and designated as (*,G). When a source appears in the network the closest multicast router will contact the RP using PIM Register messages. The PIM Register messages are sent toward the RP across the unicast shortest path. PIM should be enabled along the shortest path to the RP, or RPF check for the RP would fail and the registration process cannot be completed.

When registration completes, the RP builds a new SPT towards the source using PIM Join messages and start forwarding received multicast traffic down the (*,G) tree. When the receivers see traffic going down the (*,G) tree from the actual source they initiate PIM Join toward the source, building another SPT designated as (S,G) tree. This new tree follows the optimal path to the source and allows avoiding the RP as the "bottleneck" of all multicast traffic flows. This process of switching from (*,G) to (S,G) tree is called multicast SPT switchover. Finally, the receiving router will send a special Prune message towards the RP designated as (S, G, RPbit). This will prune the router from receiving duplicate traffic down the (*,G) tree from the RP.

Configuring RP for multicast groups is a crucial part of PIM SM setup. In this scenario, we use static RP configuration on every router using the command **ip pim rp-address <IP> [<ACL>] [override]**. The ACL parameter lists the groups that are mapped to this particular RP. You could have multiple RP using different group lists on every router, for the purpose of load-balancing. Later we will see ways of automatically disseminating RP information using Auto-RP and BSR protocols. For now, keep in mind that **override** parameter will force the router to retain static information even if different RP for the group is learned via Auto-RP.

Additionally, the task requires that the switchover STP should be built only when the traffic down the (*,G) tree exceeds 128Kbps. By default, the "leaf" router – the one closest to the receiver – will switch over as soon as it receives any multicast traffic down the (*,G) tree. This threshold could be changed using the command **ip pim spt-threshold {<Rate in Kbps>|infinity}.** By setting the rate to infinity, you effectively disable the use of shortest-path tree, and all information is received down the shared tree.

In this scenario, there is a hidden issue. When you start pinging from R6, and run debug ip mpacket in R4 (the router that registers the source) you might see something like the following:

```
IP(0): MAC sa=0011.bbbd.64a0 (FastEthernet0/1), IP last-hop=155.1.146.6
IP(0): IP tos=0x0, len=100, id=201, ttl=254, prot=1
IP(0): s=155.1.146.6 (FastEthernet0/1) d=224.10.10.10 id=201, ttl=254,
prot=1, len=114(100), RPF lookup failed
```

The problem is that R5's Loopback0 is known via OSPF, as a /32 route and know as /24 subnet via EIGRP. PIM will attempt to register across the OSPF path, which is not enabled for PIM. Thus, RPF check toward the RP will fail, and R4 will never register the source. To fix up this problem, we add **the ip ospf network point-to-point** statement in R5's Loopback0 configuration, to make OSPF advertise /24 route as well.

R4:

```
ip multicast-routing
ip pim rp-address 150.1.5.5
!
interface FastEthernet 0/1
 ip pim sparse-mode
!
interface Serial 0/1
 ip pim sparse-mode
!
interface Serial 0/0.1
 no ip pim dense-mode
```

R5:
```
ip multicast-routing
ip pim rp-address 150.1.5.5
!
interface Serial 0/1
 ip pim sparse-mode
!
interface FastEthernet 0/0
 ip pim sparse-mode

!
! Needed to fix RPF check for R5’s Loopback0
!
interface Loopback0
 ip ospf network point-to-point
```

R6:
```
ip multicast-routing
ip pim rp-address 150.1.5.5
!
interface FastEthernet 0/0.146
 ip pim sparse-mode
```

SW2:
```
ip multicast-routing distributed
ip pim rp-address 150.1.5.5
!
interface Vlan 58
 ip pim sparse-mode
!
interface Port-Channel1
 ip pim sparse-mode
```

SW4:
```
ip multicast-routing
ip pim rp-address 150.1.5.5
ip pim spt-threshold 128
!
interface Port-Channel1
 ip pim sparse-mode
!
interface Vlan 10
 ip igmp join-group 224.10.10.10
 ip pim sparse-mode
```

Verification

Note

Before you source any traffic, check that SW4 has joined the shared tree toward the RP. Notice the (*,G) states and the RP address for the group.

```
Rack1SW4#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
       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

(*, 224.10.10.10), 00:00:58/00:02:06, RP 150.1.5.5, flags: SJCL
  Incoming interface: Port-channel1, RPF nbr 155.1.108.8
  Outgoing interface list:
    Vlan10, Forward/Sparse, 00:00:56/00:02:06, H)

Rack1SW2#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
       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

(*, 224.10.10.10), 00:02:41/00:02:47, RP 150.1.5.5, flags: S
  Incoming interface: Vlan58, RPF nbr 155.1.58.5
  Outgoing interface list:
    Port-channel1, Forward/Sparse, 00:02:41/00:02:47
```

Note

The RP itself shows the RPF neighbor of 0.0.0.0 for the (*,G) state, meaning it is the RPF neighbor itself. At the same time, R4 has no (*,G) state for the group 224.10.10.10 as it is not on the shared tree.

```
Rack1R5#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

(*, 224.10.10.10), 00:03:36/00:02:52, RP 150.1.5.5, flags: S
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse, 00:03:36/00:02:52

Rack1R4#show ip mroute 224.10.10.10
Group 224.10.10.10 not found
```

Note

When you start pinging from R6 and check mroute states in R4, you will notice that (*,G) state for the group 224.10.10.10 is prune, as there are no receivers for this group below R4. At the same time, there is an (S,G) state toward R6 that enables traffic flow from R6 down to SW4.

```
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

(*, 224.10.10.10), 00:00:05/stopped, RP 150.1.5.5, flags: SP
  Incoming interface: Serial0/1, RPF nbr 155.1.45.5
  Outgoing interface list: Null

(155.1.146.6, 224.10.10.10), 00:00:05/00:02:54, flags: T
  Incoming interface: FastEthernet0/1, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/1, Forward/Sparse, 00:00:05/00:03:24
```

Note

When you look at the mroutes in R5, notice that there is (*,224.10.10.10) state representing the shared tree and (155.1.146.6,224.10.10.10) state representing the SPT built by the RP. This SPT is used to connect the multicast source to the shared tree.

```
Rack1R5#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

(*, 224.10.10.10), 00:54:20/stopped, RP 150.1.5.5, flags: S
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse, 00:54:20/00:02:42

(155.1.146.6, 224.10.10.10), 00:00:11/00:02:56, flags: T
  Incoming interface: Serial0/1, RPF nbr 155.1.45.4
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse, 00:00:11/00:02:48
```

Note

When you check mroute states in SW2 and SW4 you will only see (*,G) entries. This is because the SPT switchover rate is 128Kbps and we haven't exceeded it yet. Thus SW4 never initialized SPT-switchover toward the source.

```
Rack1SW2#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
       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

(*, 224.10.10.10), 00:54:31/00:03:27, RP 150.1.5.5, flags: S
  Incoming interface: Vlan58, RPF nbr 155.1.58.5
  Outgoing interface list:
    Port-channel1, Forward/Sparse, 00:54:31/00:03:05
<snip>

Rack1SW4#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
       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

(*, 224.10.10.10), 00:55:15/00:02:58, RP 150.1.5.5, flags: SCL
  Incoming interface: Port-channel1, RPF nbr 155.1.108.8
  Outgoing interface list:
    Vlan10, Forward/Sparse, 00:55:14/00:02:54, H
<snip>
```

Note

In order to see SPT switchover, disable the threshold setting in R4 and repeat multicast testing again.

```
SW4:
no ip pim spt-threshold 128
```

```
Rack1R6#ping 224.10.10.10 repeat 1000

Type escape sequence to abort.
Sending 1000, 100-byte ICMP Echos to 224.10.10.10, timeout is 2
seconds:

Reply to request 0 from 155.1.108.10, 49 ms
Reply to request 1 from 155.1.108.10, 32 ms
```

Note

Now you can see the SPT entry in the multicast routing table of SW2. In our case, STP and shared tree overlaps (follow the same paths), but in general case they would differ.

```
Rack1SW2#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
       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

(*, 224.10.10.10), 01:03:15/stopped, RP 150.1.5.5, flags: S
  Incoming interface: Vlan58, RPF nbr 155.1.58.5
  Outgoing interface list:
    Port-channel1, Forward/Sparse, 01:03:15/00:03:13

(155.1.146.6, 224.10.10.10), 00:00:00/00:03:29, flags:
  Incoming interface: Vlan58, RPF nbr 155.1.58.5
  Outgoing interface list:
    Port-channel1, Forward/Sparse, 00:00:00/00:03:29
```

8.4 PIM Sparse-Dense Mode

- Remove the previous PIM Sparse Mode configuration and replace it with Sparse-Dense-Mode on all interfaces.
- Modify the static RP assignments so that R5 is only used as the RP for groups in the range 224.0.0.0/8.
- Configure SW4's VLAN 10 interface to join the groups 224.30.30.30 and 239.0.0.1, and ensure that SW3 can send multicast packets to this segment for both groups.

Configuration

Note

PIM Sparse-Dense mode is a hybrid of SM and DM modes of operation. However, it does not define any extension to PIM protocol. Rather, when you apply the command **`ip pim sparse-dense-mode`** to an interface, the router will forward traffic for both sparse and dense multicast groups out of this interface. Here "sparse" multicast group is the one that has RP defined. This mode of operations is useful when you have a range of groups that you want to be delivered using the simple dense-mode. As we will see later, AutoRP groups are good example of dense-mode groups needed to be flooded along with sparse-groups.

Most of the times you don't have to use PIM SM/DM mode interfaces, unless you are using Cisco proprietary Auto-RP protocol. The problem with PIM SM/DM is that any group that does not have RP defines is treated like dense-mode group. Consider the situation where some routers lose the RP information for their sparse-mode groups. For example, routers haven't received Auto-RP announces for some time and expired old information. In this situation, the routers will switch all groups to dense-mode and start flooding the network with multicast traffic. In order to prevent this behavior, you should either use PIM SM only along with standard RP information dissemination protocol, such as BSR, or issue the command **`no ip dm-fallback`** on all PIM SM/DM routers. This will prevent the DM fallback behavior and only allow forwarding sparse-mode groups.

R4:
```
ip access-list standard SPARSE_GROUPS
 permit 224.0.0.0 0.255.255.255
!
ip multicast-routing
ip pim rp-address 150.1.5.5 SPARSE_GROUPS
!
interface FastEthernet 0/1
 ip pim sparse-dense-mode
!
interface Serial 0/1
 ip pim sparse-dense-mode
```

R5:
```
ip access-list standard SPARSE_GROUPS
 permit 224.0.0.0 0.255.255.255
!
ip multicast-routing
ip pim rp-address 150.1.5.5 SPARSE_GROUPS
!
interface Serial 0/1
 ip pim sparse-dense-mode
!
interface FastEthernet 0/0
 ip pim sparse-dense-mode

!
! Needed to fix RPF check for R5’s Loopback0
!
interface Loopback0
 ip ospf network point-to-point
```

R6:
```
ip access-list standard SPARSE_GROUPS
 permit 224.0.0.0 0.255.255.255
!
ip multicast-routing
ip pim rp-address 150.1.5.5 SPARSE_GROUPS
!
interface FastEthernet 0/0.146
 ip pim sparse-dense-mode
```

SW2:
```
ip multicast-routing distributed
ip access-list standard SPARSE_GROUPS
 permit 224.0.0.0 0.255.255.255

!
ip pim rp-address 150.1.5.5 SPARSE_GROUPS
!
interface Vlan 58
 ip pim sparse-dense-mode
!
interface Port-Channel1
 ip pim sparse-dense-mode
```

SW4:
```
ip access-list standard SPARSE_GROUPS
 permit 224.0.0.0 0.255.255.255

!
ip multicast-routing
ip pim rp-address 150.1.5.5 SPARSE_GROUPS
!
interface Port-Channel1
 ip pim sparse-dense-mode
!
interface Vlan 10
 ip igmp join-group 224.10.10.10
 ip igmp join-group 239.0.0.1
 ip pim sparse-dense-mode
```

Verification

Note

First, ping the group that has no RP from R6. Notice that all routers on the path to the receiver create (S,G) dense state. There is corresponding (*,G) state as well, but it has RP value of 0.0.0.0 and has all PIM-enabled interfaces in OIL.

```
Rack1R6#ping 239.0.0.1 repeat 100

Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 239.0.0.1, timeout is 2 seconds:

Reply to request 0 from 155.1.108.10, 37 ms
Reply to request 1 from 155.1.108.10, 32 ms

Rack1R4#show ip mroute 239.0.0.1
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

(*, 239.0.0.1), 00:00:10/stopped, RP 0.0.0.0, flags: D
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/1, Forward/Sparse-Dense, 00:00:10/00:00:00
    FastEthernet0/1, Forward/Sparse-Dense, 00:00:10/00:00:00

(155.1.146.6, 239.0.0.1), 00:00:10/00:02:55, flags: T
  Incoming interface: FastEthernet0/1, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/1, Forward/Sparse-Dense, 00:00:12/00:00:00
```

```
Rack1R5#show ip mroute 239.0.0.1
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

(*, 239.0.0.1), 00:00:26/stopped, RP 0.0.0.0, flags: D
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse-Dense, 00:00:26/00:00:00
    Serial0/1, Forward/Sparse-Dense, 00:00:26/00:00:00

(155.1.146.6, 239.0.0.1), 00:00:26/00:02:53, flags: T
  Incoming interface: Serial0/1, RPF nbr 155.1.45.4
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse-Dense, 00:00:27/00:00:00

Rack1SW2#show ip mroute 239.0.0.1
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
       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

(*, 239.0.0.1), 00:00:42/stopped, RP 0.0.0.0, flags: D
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Port-channel1, Forward/Sparse-Dense, 00:00:42/00:00:00
    Vlan58, Forward/Sparse-Dense, 00:00:42/00:00:00

(155.1.146.6, 239.0.0.1), 00:00:42/00:02:53, flags: T
  Incoming interface: Vlan58, RPF nbr 155.1.58.5
  Outgoing interface list:
    Port-channel1, Forward/Sparse-Dense, 00:00:42/00:00:00
```

Note

Now ping a group that has RP configured. Notice that (*,G) now represents the shared tree, and used to forward the first packets from the sources. SPT is built by the RP and the router connected to the receiver.

```
Rack1R5#show ip mroute 224.10.10.10
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

(*, 224.10.10.10), 02:20:25/stopped, RP 150.1.5.5, flags: S
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse-Dense, 02:20:25/00:02:41

(155.1.146.6, 224.10.10.10), 00:00:10/00:03:21, flags: T
  Incoming interface: Serial0/1, RPF nbr 155.1.45.4
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse-Dense, 00:00:10/00:03:19

Rack1SW2#show ip mroute 224.10.10.10
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
       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

(*, 224.10.10.10), 02:29:57/stopped, RP 150.1.5.5, flags: S
  Incoming interface: Vlan58, RPF nbr 155.1.58.5
  Outgoing interface list:
    Port-channel1, Forward/Sparse-Dense, 02:29:57/00:03:12

(155.1.146.6, 224.10.10.10), 00:00:03/00:03:29, flags: T
  Incoming interface: Vlan58, RPF nbr 155.1.58.5
  Outgoing interface list:
    Port-channel1, Forward/Sparse-Dense, 00:00:03/00:03:26
```

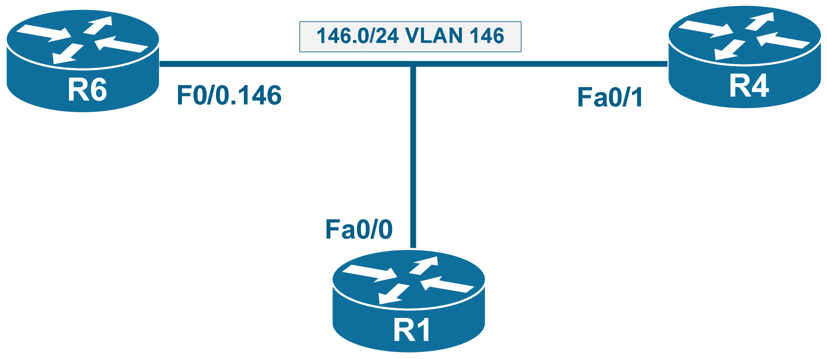
8.5 PIM Assert

- Enable multicast routing in R1 and configure PIM sparse-dense mode on Ethernet interface of R1.
- Configure PIM sparse-dense mode on the connection between R1 and R5.
- Use the same RP assignment that was configured in other routers.
- Ensure R1 is always elected via PIM to flood the shared VLAN146 segment.

Configuration

Note

If multiple multicast routers share a single segment, then all of them could flood this segment with the same multicast traffic.



For example, if both R1 and R4 receive the same multicast flow from their upstream neighbors, they will both send it to R6. From R6's point of view, both flows are the same with respect to RPF validation. Thus, traffic duplication may occur.

In order to avoid this situation, only one router is allowed to flood traffic on the shared segment. When a router detects that someone is sending traffic for the same source IP/destination group on the segment where it has active (S,G) state for the same group/source, it immediately originates a PIM Assert message. This message contains the source IP, the group group and the path cost to the source. The path cost is a touple (AD, Metric), where AD is the administrative distance of the routing protocol used to look up the source IP, and Metric is the same protocol's metric to reach the source. Router with the best (lowest) AD value on the segment wins the assertion. If ADs are equal, metric is used as tie-breaker. If both metrics are the same, the router with the highest IP wins. If another router on the segment receives the Assert message and determines that it loses the assertion, it will remove the (S,G) state on its interface and stop flooding traffic. However, if it sees itself as the winner, it will emit a superior PIM Assert message to inform the other router that it should stop flooding the traffic for this (S,G) pair.

Note that PIM Assert procedure might be dangerous on NBMA interfaces. PIM continues to treat those interfaces as fully broadcast networks, even though not all nodes are capable of hearing each-other's broadcast messages. Imagine a hub-and-spoke Frame-Relay topology. If both the hub and the spoke start flooding the segment with the same multicast traffic, PIM Assert will occur. If for some reason the spoke would win, then the hub will stop sending its multicast traffic. However, all traffic coming from the spoke to the hub is NOT relayed back to other spokes sharing the same segment, based on the RPF rule. Thus, all other spokes will effectively stop receiving the multicast traffic. To avoid this situation, either use PIM NBMA mode (described in separate task) or make sure that the hub always wins the PIM Assert procedure.

In our scenario, R1 and R4 run different IGPs. This is not a well-designed multicast solution, as you may want all routers to be under the same IGP. In our case, R4 would be the assert winner as EIGRP has better AD (90) than OSPF (110). However, by adjusting the AD of OSPF in R1, we artificially make R1 the assert winner on the shared segment.

R1:
```
access-list 10 permit 224.0.0.0 0.255.255.255
!
ip multicast-routing
ip pim rp-address 150.1.5.5 10
!
interface FastEthernet0/0
 ip pim sparse-dense-mode
!
interface Serial 0/0.1
 ip pim sparse-dense-mode
!
router ospf 1
 distance 80
```

R5:
```
interface Serial 0/0
 ip pim sparse-dense-mode
```

Verification

Note

Join the group 239.6.6.6 in R6 and ping this IP address from SW4. The group will use simple dense-mode forwarding, and R1 will be elected as the assert winner on VLAN146 segment.

```
R6:
interface FastEthernet 0/0.146
 ip igmp join-group 239.6.6.6

Rack1SW4#ping 239.6.6.6 repeat 1000

Type escape sequence to abort.
Sending 1000, 100-byte ICMP Echos to 239.6.6.6, timeout is 2 seconds:

Reply to request 0 from 155.1.146.6, 40 ms
Reply to request 0 from 155.1.146.6, 92 ms..
Rack6AS>1
```

Note

The mroute entry in R1 is marked with “A” flag, which means “Assert Winner”.

```
Rack1R1#show ip mroute 239.6.6.6
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

(*, 239.6.6.6), 00:02:05/stopped, RP 0.0.0.0, flags: DC
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse-Dense, 00:02:05/00:00:00
    Serial0/0.1, Forward/Sparse-Dense, 00:02:05/00:00:00

(155.1.108.10, 239.6.6.6), 00:00:22/00:02:46, flags: T
  Incoming interface: Serial0/0.1, RPF nbr 155.1.0.5
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse-Dense, 00:00:24/00:00:00, A
```

Note

If you would enable the following debugging in R1 before sending pings from SW4, you may catch PIM Assert message exchange between R1 and R4. Notice that R1 wins due to the better AD.

```
Rack1R1#debug ip pim

PIM(0): Received v2 Assert on FastEthernet0/0 from 155.1.146.4
PIM(0): Assert metric to source 155.1.108.10 is [90/2174976]
PIM(0): We win, our metric [80/20]
PIM(0): (155.1.108.10/32, 239.6.6.6) oif FastEthernet0/0 in Forward
state
<snip>
PIM(0): Send v2 Assert on FastEthernet0/0 for 239.6.6.6, source
155.1.108.10, metric [80/20]
PIM(0): Assert metric to source 155.1.108.10 is [80/20]
PIM(0): We win, our metric [80/20]
PIM(0): (155.1.108.10/32, 239.6.6.6) oif FastEthernet0/0 in Forward
state
```

Note

R4 prunes the interface where it loses the assert procedure from OIL for the group 239.6.6.6

```
Rack1R4#show ip mroute 239.6.6.6
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

(*, 239.6.6.6), 00:07:12/stopped, RP 0.0.0.0, flags: DC
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/1, Forward/Sparse-Dense, 00:07:12/00:00:00
    FastEthernet0/1, Forward/Sparse-Dense, 00:07:12/00:00:00

(155.1.108.10, 239.6.6.6), 00:05:27/00:00:39, flags: PT
  Incoming interface: Serial0/1, RPF nbr 155.1.45.5
  Outgoing interface list:
    FastEthernet0/1, Prune/Sparse-Dense, 00:02:26/00:00:33
```

8.6 PIM Accept RP

- Ensure that SW2 and R5 will only accept (*,G) joins toward R5's Loopback0 for groups 224.10.10.10 and 224.110.110.110

Configuration

Note

This is a security feature used by IOS routers to prevent unwanted RPs or groups to become active in the PIM SM multicast domain. When you configure **`ip pim accept-rp {rp-address | auto-rp} [access-list]`** the local router will only access (*,G) Join/Prune messages towards the RP-address specified in the command. Additionally, if the access-list parameter is specified, the router will only accept Join/Prune messages for groups matching this access-list. Note that regular SPT joins are not affected by this configuration.

In order to implement thorough security using this feature, you must configure every router on all potential paths to the RP, or simply configure your RP. This will allow illegal Joins across the network, but they will eventually be dropped at the RP.

```
R5:
ip access-list standard ALLOWED_GROUPS
 permit 224.10.10.10
 permit 224.110.110.110
!
ip pim accept-rp 150.1.5.5 ALLOWED_GROUPS

SW2:
ip access-list standard ALLOWED_GROUPS
 permit 224.10.10.10
 permit 224.110.110.110
!
ip pim accept-rp 150.1.5.5 ALLOWED_GROUPS
```

Verification

Note

Join SW4’s VLAN10 interface to the groups 224.10.10.10 and 224.11.11.11. Observe the console output on **SW2:**

```
Rack1SW2#
%PIM-6-INVALID_RP_JOIN: Received (*, 224.11.11.11) Join from 0.0.0.0
for invalid RP 150.1.5.5

%PIM-6-INVALID_RP_JOIN: Received (*, 224.0.1.40) Join from 0.0.0.0 for
invalid RP 150.1.5.5
```

Note

Notice that SW2 rejected joins toward R5’s Loopback0 for invalid groups. If you wonder what is group 224.0.1.40, it’s Auto-RP discovery group, which all routers join by default. Since we happened to define RP for group range covering this particular one, every router attempts to join the shared tree for it.

Using the **`show ip mroute`** command you may observe that only group 224.10.10.10 has mroute state corresponding to the shared tree.

```
Rack1SW2#show ip mroute 224.10.10.10
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
       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

(*, 224.10.10.10), 01:32:57/00:03:10, RP 150.1.5.5, flags: S
  Incoming interface: Vlan58, RPF nbr 155.1.58.5
  Outgoing interface list:
    Port-channel1, Forward/Sparse-Dense, 01:32:57/00:03:10

Rack1SW2#show ip mroute 224.11.11.11
Group 224.11.11.11 not found
```

8.7 PIM DR Election

- Ensure that R1 is responsible for multicast source registration with RP on VLAN146 segment.

Configuration

Note

PIM Designated Router (DR) is elected on every multiple-access segment, i.e. every segment where multiple routers share the same medium/subnet. Election process is based on the highest priority and highest IP address – the router with numerically higher value wins the election. The process is pre-emptive and every new router with better priority will preempt the previous DR.

One purpose of DR is signaling multicast delivery trees using PIM messages when it sees interested receivers on the segment by means of IGMP. Another purpose is registering active sources on the segment with the regional RP. When the DR hears multicast packets on the segment, it will check that the destination group has an RP. If that's the case, the data packets are encapsulated into special PIM Register messages and sent to the RP. The RP will start forwarding them down the shared tree, if there are any active subscribers. At the same time, the RP will build a shortest-path tree towards the DP and send PIM Register-Stop message to the DR to inform it that regular forwarding may start now. After this, the multicast traffic is delivered over the SPT.

Note that PIM Register messages are subject to RPF check, as usual. If the Register message is received on non-RPF interface, the check would fail.

R1:
```
interface FastEthernet0/0
 ip pim dr-priority 100
```

Verification

```
Rack1R1#show ip pim interface fastEthernet 0/0 detail
FastEthernet0/0 is up, line protocol is up
  Internet address is 155.1.146.1/24
  Multicast switching: fast
  Multicast packets in/out: 4624/811
  Multicast TTL threshold: 0
  PIM: enabled
    PIM version: 2, mode: sparse-dense
    PIM DR: 155.1.146.1 (this system)
    PIM neighbor count: 2
    PIM Hello/Query interval: 30 seconds
    PIM Hello packets in/out: 603/304
    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
```

8.8 PIM Accept Register

- Configure R5 so that the only source allowed on VLAN146 segment is R6.
- Your configuration should not affect sources in any other segments.

Configuration

Note

This feature is configured on PIM SM Rendezvous Point and controls sources that are allowed to register with the RP. Remember that registration is performed by the PIM DR router, and every register message contains the original multicast packet with the IP address of the multicast source and group destination address. If the RP denies the registration, it sends PIM Register-Stop to the DR immediately and never builds the SPT towards the source.

The command to enable PIM Register message filtering is **`ip pim accept-register {list <Extended-ACL> | route-map <Route-Map>}`**. The basis of filtering is extended ACL in the format:

```
ip access-list REGISTER_FILTER
 permit ip <source-ip> <source-wildcard> <group-address>
<group-wildcard>
```

matching both sources and destination groups at the same time. For example, to permit the source 155.1.146.1 to send traffic to any group via the RP, use the entry `permit ip host 155.1.146.1 224.0.0.0 15.255.255.255`. Notice that if the RP is the DR for some segment, this filtering will not work.

For our scenario, we create an access-list used with “negative” filtering in the route-map. The access-list permits all sources on VLAN146 with except to R6. Later, all those sources are denied by the router-map. Everything else is permitted.

R5:
```
ip access-list extended VLAN146_SOURCES
 deny ip host 155.1.146.6 any
 permit ip 155.1.146.0 0.0.0.255 any
!
route-map ACCEPT_REGISTER deny 10
 match ip address VLAN146_SOURCES
!
route-map ACCEPT_REGISTER permit 100
!
ip pim accept-register route-map ACCEPT_REGISTER
```

Verification

Note

First of all, we join SW4 to group 224.10.10.10 and try pinging it from R6. Since the IP address of R6 is permitted to register with the RP, everything goes smoothly.

```
SW4:
interface Vlan 10
 ip igmp join-group 224.10.10.10

Rack1R6#ping 224.10.10.10 repeat 100

Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 224.10.10.10, timeout is 2 seconds:

Reply to request 0 from 155.1.108.10, 64 ms
Reply to request 1 from 155.1.108.10, 32 ms
Reply to request 2 from 155.1.108.10, 32 ms
Reply to request 3 from 155.1.108.10, 32 ms
```

Note

Now change the IP address of R6's VLAN146 interface as follows:

```
R6:
interface FastEthernet 0/0.146
 ip address 155.1.146.66 255.255.255.0
```

And enable PIM debugging in R1 (the DR) using the command **`debug ip pim`**. Notice that R1 receives Register-Stop message immediately after sending the first Register packet. However, this time the RP does not build an SPT toward the source.

```
Rack1R6#ping 224.10.10.10 repeat 100

Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 224.10.10.10, timeout is 2 seconds:
.
Rack6AS>1
[Resuming connection 1 to r1 ... ]

Rack1R1#show logging
<snip>

PIM(0): Send v2 Register to 150.1.5.5 for 155.1.146.66, group
224.10.10.10
PIM(0): Received v2 Register-Stop on Serial0/0.1 from 150.1.5.5
PIM(0):   for source 155.1.146.66, group 224.10.10.10
PIM(0): Clear Registering flag to 150.1.5.5 for (155.1.146.66/32,
224.10.10.10)
```

Note

You may discover more on R5 – the RP. Notice the console message that warns you about illegitimate source. Also, there is no SPT entry toward the new IP address of R6 in the mroute table of R5. Thus, the source is not connected to the shared tree.

```
Rack1R5#
%PIM-4-INVALID_SRC_REG: Received Register from 155.1.0.1 for
(155.1.146.66, 224.10.10.10), not willing to be RP

Rack1R5#show ip mroute 224.10.10.10
IP Multicast Routing Table
<snip>

(*, 224.10.10.10), 05:39:10/00:02:52, RP 150.1.5.5, flags: S
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse-Dense, 05:39:10/00:02:52

(155.1.146.6, 224.10.10.10), 00:02:34/00:01:07, flags: T
  Incoming interface: Serial0/1, RPF nbr 155.1.45.4
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse-Dense, 00:02:34/00:02:52
```

Note

At the same time, when you source the traffic flow off a different IP address, everything works fine.

```
Rack1R1#ping 224.10.10.10

Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 224.10.10.10, timeout is 2 seconds:

Reply to request 0 from 155.1.108.10, 60 ms
```

8.9 Multicast Tunneling

- Without configuring multicast routing on the transit nodes, make sure that receivers at SW3 may receive multicast feeds from R1.
- Do not configure any additional IGP adjacencies to accomplish this task.

Configuration

Note

Tunneling is a common technique used with multicast technologies in order to traverse non-multicast capable network. This greatly simplifies configuration, but has major drawback of “hiding” the real network topology and thus preventing effective multicast replication. When configuring a tunnel, you have an option of enabling or disabling the IGP on the tunnel interface. If you plan to use the tunnel for multicast traffic delivery, it is obligatory to enable PIM on the same interface. Based on these two factors, you may easily run into RPF failure issues.

1) If no IGP is running on the tunnel, then you will need to provide some static mroutes to correct the RPF checks.
2) If IGP is running on the tunnel interface, the cost of traversing the tunnel might be higher than the cost of traversing the underlying network. It is generally not the best idea to run the same IGP on the underlying network and on the tunnel. You may need to adjust metrics or provide static mroute fix-ups.

In our situation, we don’t run any IGP on the tunnel. Thus we provide static mroute fix-up to allow the endpoints perform RPF checks correctly. Also, we need to replicate the static RP configuration, to allow SW3 receive traffic for sparse groups.

R1:
```
interface Tunnel 0
 ip unnumbered Loopback0
 tunnel source Loopback0
 tunnel destination 150.1.9.9
 ip pim sparse-dense-mode
!
router ospf 1
 passive-interface Tunnel0
```

SW3:
```
ip multicast-routing
!
interface Tunnel 0
 ip unnumbered Loopback0
 tunnel source Loopback0
 tunnel destination 150.1.1.1
 ip pim sparse-dense-mode
!
router ospf 1
 passive-interface Tunnel0
!
ip mroute 0.0.0.0 0.0.0.0 tunnel 0
!
ip access-list standard SPARSE_GROUPS
 permit 224.0.0.0 0.255.255.255
!
ip pim rp-address 150.1.5.5 SPARSE_GROUPS
```

Verification

Note

To verify your configuration, Join SW3's interface VLAN9 to the multicast group 224.10.10.10 and ping this group from R1. Note that enabling PIM on the interface is mandatory, as otherwise IOS will not forward multicast out of this interface.

```
SW3:
interface Vlan 9
 ip igmp join-group 224.10.10.10
 ip pim sparse-dense-mode
```

Notice that all RPF information in SW3 is condensed in one single default mroute. It overrides all checks using the unicast routing table and allows SW3 receiving multicast successfully.

```
Rack1SW3#show ip mroute 224.10.10.10
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
       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

(*, 224.10.10.10), 00:01:24/00:02:41, RP 150.1.5.5, flags: SJCL
  Incoming interface: Tunnel0, RPF nbr 150.1.1.1, Mroute
  Outgoing interface list:
    Vlan9, Forward/Sparse-Dense, 00:01:24/00:02:41
```

```
Rack1R1#show ip mroute 224.10.10.10
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

(*, 224.10.10.10), 00:01:35/00:02:54, RP 150.1.5.5, flags: S
  Incoming interface: Serial0/0.1, RPF nbr 155.1.0.5
  Outgoing interface list:
    Tunnel0, Forward/Sparse-Dense, 00:01:35/00:02:54

Rack1R1#ping 224.10.10.10

Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 224.10.10.10, timeout is 2 seconds:

Reply to request 0 from 150.1.9.9, 64 ms
Reply to request 0 from 155.1.108.10, 104 ms
Reply to request 0 from 150.1.9.9, 88 ms
```

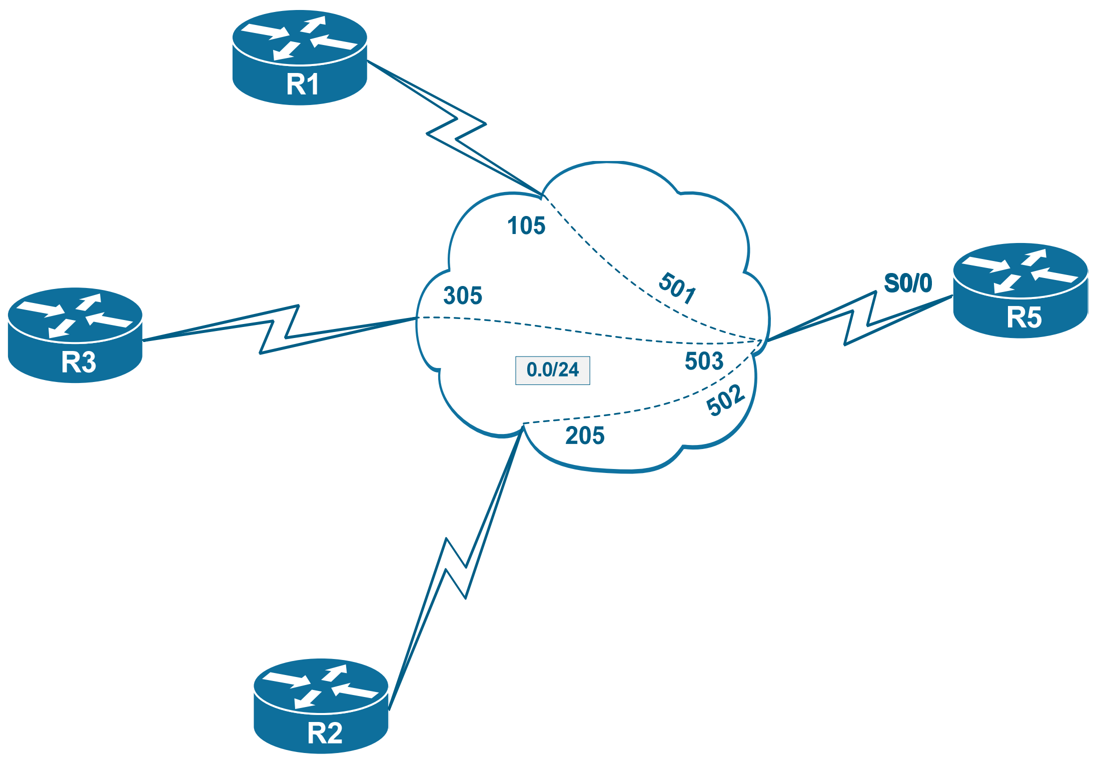
8.10 PIM NBMA Mode

- Configure R3 to join the multicast group 224.110.110.110 and make sure R6 can ping it.

Configuration

Note

PIM NBMA mode is an extension to PIM protocol operations on WAN interfaces, particularly for Frame-Relay and ATM which utilize the concept of virtual-circuits.



Look at the figure above. Firstly, imagine that R5 is forwarding multicast traffic down to R1, R2 and R3. By default, all multicast packets are treated as broadcast on the Frame-Relay segment, and queued in the special broadcast queue. This queue is a priority queue, used to store the routing protocol updates and L2 keepalives. Packets from this queue are replicated and sent over all active VCs terminating on the interface. This is called “pseudo-broadcast” and performed at router’s CPU level. Needless to say, that this procedure is very CPU intensive when multicast flow is intensive. Not to mention that it may also block critical routing updates on the interface.

Secondly, PIM treats the Frame-Relay interface as a “broadcast” media. That is, it assumes that all multicast packets received on this interface are heard by all members connected to the cloud. However, Frame-Relay is non-broadcast technology, and it only works like broadcast in case of fully meshed topology. If you look at the figure above, you will see that hub-and-spoke topology does not satisfy this requirement. When R1 sends a multicast packet, only R5 receives it. Based on the RPF rule, it will never send the packet back out of the same interface. Thus, any multicast packets sent from spoke will never reach other spokes. If you want to resolve this issue, you have to configure sub-interfaces in R5 for all spokes.

The other issue that comes from NBMA nature of Frame-Relay technology is that spokes don't hear each other's PIM messages. For example, if R1, R2 and R3 all have multicast subscribers, they all build SPT trees across R5. At R5, all the trees converge to a single (S,G) state. Now suppose that R1 loses its last receiver, and sends PIM Prune message to R5. Since R5 assumes that all other routers also heard this message, it declares that the shared segment is free of any receivers. After this, R5 prunes the (S,G) state and leaves R2 and R3 without multicast traffic. Both R2 and R3 never had a chance to override the PIM Prune message sent from R1, as they never heard it.

PIM NBMA mode resolves all of the above issues. First of all, it works only with PIM SM (sparse mode groups), as it relies on PIM Join message. Secondly, it treats the "monolithic" Frame-Relay interface as a collection of point-to-point links connected to other nodes. It does so by tracking the PIM Join messages received from those neighbors. For every neighbor that sends the Join message the router creates a separate (S,G) state bound to this neighbor's IP address. Now this state emulates a point-to-point link connected to this neighbor. All PIM Prune messages received from this neighbor will only affect its own (S,G) states, leaving other neighbors' states intact. Also, any multicast traffic received from this neighbor is treated like it has been received from the point-to-point link. Thus, the RPF procedure will allow forwarding the multicast packets back out of the same interface to all other interested neighbors. Finally, PIM NBMA mode replaces the ineffective pseudo-broadcast replication on all PVCs with fast-switched packet distribution only to the subscribed neighbors. This allows saving router CPU resources and WAN bandwidth.

Notice that you may enable NBMA mode on the PIM SM/DM interface, even though this is not the recommended configuration. It will only work for groups signaled via sparse mode.

R3:
```
ip multicast-routing
!
ip access-list standard SPARSE_GROUPS
 permit 224.0.0.0 0.255.255.255
!
ip pim rp-address 150.1.5.5 SPARSE_GROUPS
!
interface Serial 1/0.1
 ip pim sparse-dense-mode
!
interface Loopback0
 ip pim sparse-dense-mode
 ip igmp join-group 224.110.110.110
```

R5:
```
interface Serial 0/0
 ip pim nbma-mode
```

Verification

Note

First, try pinging the group from R6 before you apply the NBMA mode configuration. The operation will fail, and you may see that OIL for the group on R5 is empty (Pruned, as the mode is SM/DM).

```
Rack1R6#ping 224.110.110.110 repeat 1000

Type escape sequence to abort.
Sending 1000, 100-byte ICMP Echos to 224.110.110.110, timeout is 2
seconds:
...

Rack1R3#show ip mroute 224.110.110.110
IP Multicast Routing Table
<snip>

(*, 224.110.110.110), 00:00:27/00:02:57, RP 150.1.5.5, flags: SJCL
  Incoming interface: Serial1/0.1, RPF nbr 155.1.0.5
  Outgoing interface list:
    Loopback0, Forward/Sparse-Dense, 00:00:27/00:02:57

Rack1R5#show ip mroute 224.110.110.110
IP Multicast Routing Table
<snip>
(*, 224.110.110.110), 00:00:55/stopped, RP 150.1.5.5, flags: SP
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/0, Prune/Sparse-Dense, 00:00:44/00:00:00

(155.1.146.6, 224.110.110.110), 00:00:38/00:02:21, flags: PT
  Incoming interface: Serial0/1, RPF nbr 155.1.45.4
  Outgoing interface list:
    Serial0/0, Prune/Sparse-Dense, 00:00:38/00:02:21
```

Note

After you enable PIM NBMA mode on the hub-and-spoke interface, notice the change in mroute states. Now there are NBMA IP addresses next to the interface name, tracking the state of sparse neighbors. Since the interface is now treated as array of point-to-point links, the split-horizon rule is no longer in effect. Pings from R6 now succeed.

```
Rack1R5#show ip mroute 224.110.110.110
IP Multicast Routing Table
<snip>

(*, 224.110.110.110), 00:02:58/00:03:28, RP 150.1.5.5, flags: S
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/0, 155.1.0.3, Forward/Sparse-Dense, 00:00:01/00:03:28

(155.1.146.6, 224.110.110.110), 00:02:41/00:02:54, flags: T
  Incoming interface: Serial0/1, RPF nbr 155.1.45.4
  Outgoing interface list:
    Serial0/0, 155.1.0.3, Forward/Sparse-Dense, 00:00:01/00:03:28

Rack1R6#ping 224.110.110.110 repeat 1000

Type escape sequence to abort.
Sending 1000, 100-byte ICMP Echos to 224.110.110.110, timeout is 2
seconds:

Reply to request 0 from 155.1.0.3, 65 ms
Reply to request 1 from 155.1.0.3, 60 ms
Reply to request 2 from 155.1.0.3, 60 ms
Reply to request 3 from 155.1.0.3, 60 ms
```

8.11 Auto-RP

- Remove all static RP settings and previous Accept-RP/Register configuration.
- Using Cisco proprietary technology, configure so that R5 advertises itself as the RP for all multicast groups.

Configuration

Note

Auto-RP was the first protocol to automatically distribute RP information across the multicast domain when using PIMv1. Auto-RP is Cisco proprietary protocol, and it was later "replaced" by standard-based Bootstrap Router (BSR) protocol with the advent of PIMv2. However, all Cisco routers still support Auto-RP along with PIMv2 protocol.

Auto-RP defines two new concepts – Candidate RP (cRP) and Mapping Agents (MA). cRP is any router that is willing to become an RP. You configure this router using the command **`ip pim send-rp-announce <Interface> scope <TTL> [group-list <Std-ACL>] [interval <seconds>]`**. The router will start sending UDP packets to the IP/Port 224.0.1.39/496 with the list of groups serviced by this particular RP. The cRP announces are originated every 60 seconds by default, with the Time-to-Live field in the IP headers set to **`<TTL>`** – a form of administrative scoping. The interface that you use in this command must be enabled for PIM and its IP address will be used as the RP's IP address. The list of group is defined using the **`<Std-ACL>`** access-list. You configure this list using standard access-list syntax, e.g.

```
access-list GROUPS permit 239.0.0.0 0.0.0.255
access-list GROUPS permit 232.0.2.0 0.0.0.255
access-list GROUPS deny 224.0.1.50 0.0.0.0
```

The Auto-RP code will convert wildcard masks into the prefix-lengths, so you cannot use discontinuous masks. Next, the "deny" statements are interpreted as special type of *negative* group announcement. This means that all groups matching this range should be treated as "dense" mode groups. We will discuss how the routers interpret the mapping entries a bit later.

The cRP announces are flooded across the network and reach special routers called Mapping Agents. You configure these routers using the command **`ip pim send-rp-discovery <Interface> scope <TTL> interval <Seconds>`**. Agents listen on the standard address 224.0.1.39

and collect all announcements from candidate RPs. After that, every mapping agent compiles a resulting list of Group to RP mappings and start sending “RP discovery” messages to the special multicast address 224.0.1.40 port 496. The discovery messages contain an amalgam of all information learned by mapping agents. Notice that if there are multiple MAs in the network, they will hear each other, and the all of them, with except to the one with the highest IP address, will cease sending discoveries.

When building a discovery message, MA would follow a few simple rules:

1) If there are two announces with the same group range but different RPs, select the announcement with the highest RP IP address.
2) If there are two announcements, where one group is a subset of another, but the RPs are different, send them both.
3) All other announcements are grouped together without any conflicts resolution.

All regular routers join the multicast group 224.0.1.40 and listen to the discovery messages. Based on their contents, they populate their Auto-RP cache and learn about Group-to-RP mappings. The cache contains both “negative” and “positive” entries. When looking for an RP, Auto-RP code will first scan through negative entries. If a match is found, the group is considered to be dense. Note that RP information for the negative entries is effectively ignored. If the group is not found in the “negative” list, the code looks up in the positive list. Since every group in the list is bound to a particular RP, there could be conflicts when multiple RPs try to service overlapping group ranges. The receiving router uses the longest-match rule to resolve all conflicts: if there are multiple matches, only the one with the longest prefix length is selected.

Note that “negative” statements could be defined at any cRP and affect ALL routers in the multicast region. For example if a single group list would contain **`deny any`** statement, then all groups will be treated as dense, even if there are “positive” entries.

The last thing to discuss about Auto-RP is how the multicast groups 224.0.1.39 and 224.0.1.40 are propagated across the network. Since there is no explicit RP information for these groups, they must use dense mode forwarding. This requires the use of **`pim sparse-dense-mode`** on all interfaces within the multicast domain. As it has been discussed before, this is not the safest thing to do in a large-scale network. You may want to use the **`no ip dm-fallback`** global command in such situations or use the Auto-RP Listener feature, discussed in a separate task.

Note that you may use PIM SM mode along with Auto-RP if you define static

RP value for the Auto-RP groups (224.0.1.39 and 224.0.1.40). This will require you to use the **`override`** option when defining the static RP. By default, Auto-RP announces override the statically configured RP. If you want them to persist, use the **`override`** keyword along with the **`ip pim rp-address`** command.

You may ask one question – why there is a need for Mapping Agent? Could candidate RPs just broadcast themselves to all routers and the latter learn/elect best RPs directly? This is possible, but might result in different routers electing different RPs for the same group ranges. Some routers may miss announcements of a particular candidate RP due to network outages or RP failures. Therefore, cRP information should be collected in one single point before being disseminated to the multicast routers.

R1, R3, R4, R5, R6, SW2, SW4:
```
no ip pim rp-address 150.1.5.5
```

R5:
```
no ip pim accept-register route-map ACCEPT_REGISTER
no ip pim accept-rp 150.1.5.5 ALLOWED_GROUPS
!
interface Loopback0
 ip pim sparse-dense-mode
!
ip pim send-rp-announce Loopback 0 scope 10
ip pim send-rp-discovery loopback 0 scope 10
```

SW2:
```
no ip pim accept-rp 150.1.5.5 ALLOWED_GROUPS
```

Verification

Note

Initial verification includes checking for RP to group mapping. In our case, there is just on RP, thus all groups map to it. Make sure you performed the verification on all routers, as some of them might reject discovery announces due to RPF issues.

```
Rack1R5#show ip pim rp mapping
PIM Group-to-RP Mappings
This system is an RP (Auto-RP)
This system is an RP-mapping agent (Loopback0)

Group(s) 224.0.0.0/4
  RP 150.1.5.5 (?), v2v1
    Info source: 150.1.5.5 (?), elected via Auto-RP
         Uptime: 00:00:34, expires: 00:02:24
```

Note

If you wonder what the (?) sign means, it is for the hostname of the RP. It will show you the hostname if the router has DNS names lookup configured and the IP address of the RP resolves to a name.

8.12 Auto-RP - Multiple Candidate RPs

- Configure SW2 and SW4 as RPs for group ranges 224.0.0.0-231.255.255.255 and 232.0.0.0-239.255.255.255 respectively.
- In case of one RP failure, the other one should provide backup for its groups.
- The group 224.110.110.110 should be always switched in dense-mode.

Configuration

Note

As discussed in the previous task, Auto-RP MAs amalgamate information learned from multiple candidate RPs to advertise an RP discovery message. You may want to use multiple RPs in cases when you want load-balancing or both.

1) If your goal is load balancing, try to configure the group-mapping access-lists so that every RP services a range of groups.
2) If you aim at redundancy, make both RPs service the same group ranges. The one with the highest IP address will be selected by the MA.
3) If you want to achieve both load-balancing and redundancy, you may map RP1 to a specific group range, say 224.0.0.0-231.255.255.255 and permit 224.0.0.0 15.255.255.255 in the end of the respective ACL. Use the entry to permit the range 232.0.0.0-239.255.255.255 for RP2 along with the entry 224.0.0.0 15.255.255.255 in the end. This will ensure that RP1 and RP2 are used for specific group ranges, while RP1 is used for the rest of the groups when RP2 fails and vice versa. This is based on the longest match selection criteria used by the multicast routers and the fact that for overlapping ranges MA will only advertise the RP with the highest IP address.

Notice that the mapping list is compiled by the MA, but still the final RP selection is performed by the multicast router. Longest match criteria is an effective rule to allow for unique RP selection along with providing redundancy.

R5:
```
no ip pim send-rp-announce Loopback 0 scope 10
```

SW2:
```
ip access-list standard SW2_GROUPS
 permit 224.0.0.0 7.255.255.255
 permit 224.0.0.0 15.255.255.255
 deny 224.110.110.110
!
interface Loopback0
 ip pim sparse-dense-mode
!
ip pim send-rp-announce Loopback 0 scope 10 group-list SW2_GROUPS
```

SW4:
```
no ip access-list standard SW4_GROUPS
ip access-list standard SW4_GROUPS
 permit 232.0.0.0 7.255.255.255
 permit 224.0.0.0 15.255.255.255
 deny 224.110.110.110
!
interface Loopback0
 ip pim sparse-dense-mode
!
ip pim send-rp-announce Loopback 0 scope 10 group-list SW4_GROUPS
```

Verification

Note

Look at RP mappings in R5. Notice how the MA elects the best candidate RP for a given range. The group 224.110.110.110 is negatively cached, and thus is always processes in dens-mode.

```
Rack1R5#show ip pim rp mapping
PIM Group-to-RP Mappings
This system is an RP-mapping agent (Loopback0)

Group(s) 224.0.0.0/5
  RP 150.1.8.8 (?), v2v1
    Info source: 150.1.8.8 (?), elected via Auto-RP
         Uptime: 00:01:15, expires: 00:02:42
Group(s) 224.0.0.0/4
  RP 150.1.10.10 (?), v2v1
    Info source: 150.1.10.10 (?), elected via Auto-RP
         Uptime: 00:01:05, expires: 00:02:51
  RP 150.1.8.8 (?), v2v1
    Info source: 150.1.8.8 (?), via Auto-RP
         Uptime: 00:01:15, expires: 00:02:42
Group(s) (-)224.110.110.110/32
  RP 150.1.10.10 (?), v2v1
    Info source: 150.1.10.10 (?), elected via Auto-RP
         Uptime: 00:01:05, expires: 00:02:53
  RP 150.1.8.8 (?), v2v1
    Info source: 150.1.8.8 (?), via Auto-RP
         Uptime: 00:01:15, expires: 00:02:45
Group(s) 232.0.0.0/5
  RP 150.1.10.10 (?), v2v1
    Info source: 150.1.10.10 (?), elected via Auto-RP
         Uptime: 00:01:05, expires: 00:02:51
```

Note

Check the Auto-RP cache of R6. Notice that it only has single RP for every range. SW4 is elected as the RP for all ranges with except to 224.0.0.0/5.

```
Rack1R6#show ip pim rp mapping
PIM Group-to-RP Mappings

Group(s) 224.0.0.0/5
  RP 150.1.8.8 (?), v2v1
    Info source: 150.1.5.5 (?), elected via Auto-RP
         Uptime: 00:02:16, expires: 00:02:49
Group(s) 224.0.0.0/4
  RP 150.1.10.10 (?), v2v1
    Info source: 150.1.5.5 (?), elected via Auto-RP
         Uptime: 00:02:06, expires: 00:02:49
Group(s) (-)224.110.110.110/32
  RP 150.1.10.10 (?), v2v1
    Info source: 150.1.5.5 (?), elected via Auto-RP
         Uptime: 00:02:06, expires: 00:02:50
Group(s) 232.0.0.0/5
  RP 150.1.10.10 (?), v2v1
    Info source: 150.1.5.5 (?), elected via Auto-RP
         Uptime: 00:05:06, expires: 00:02:54
```

Note

Now you may check if the group 224.110.110.110 is forwarded using PIM Dense mode. Recall that R3 has joined this group before, and ping it from SW2. However, make sure you temporality shut down R1’s Frame-Relay interface. Otherwise, R1 will send PIM Prune message to R5 and R5 will remove the whole interface from OIL for the group. You cannot override this behavior with PIM DM, unless you use subinterfaces.

R1:
```
interface Serial 0/0
 shutdown
```

Notice that the group has no RP in the mroute output, just as it is supposed to be for the dense group.

```
Rack1SW2#ping 224.110.110.110 repeat 100

Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 224.110.110.110, timeout is 2
seconds:

Reply to request 0 from 155.1.0.3, 75 ms
Reply to request 1 from 155.1.0.3, 67 ms
Reply to request 2 from 155.1.0.3, 67 ms
Reply to request 3 from 155.1.0.3, 67 ms
Reply to request 4 from 155.1.0.3, 68 ms
Reply to request 5 from 155.1.0.3, 67 ms
Reply to request 6 from 155.1.0.3, 67 ms

Rack1R5#show ip mroute 224.110.110.110
IP Multicast Routing Table
<snip>

(*, 224.110.110.110), 00:00:24/stopped, RP 0.0.0.0, flags: D
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/0, Forward/Sparse-Dense, 00:00:24/00:00:00
    Serial0/1, Forward/Sparse-Dense, 00:00:24/00:00:00
    FastEthernet0/0, Forward/Sparse-Dense, 00:00:24/00:00:00

(155.1.58.8, 224.110.110.110), 00:00:24/00:02:51, flags: T
  Incoming interface: FastEthernet0/0, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/0, Forward/Sparse-Dense, 00:00:25/00:00:00
    Serial0/1, Forward/Sparse-Dense, 00:00:25/00:00:00
```

8.13 Auto-RP - Filtering Candidate RPs

- Configure R5 to filter announces for the group 224.110.110.110 received from SW4.

Configuration

Note

Auto-RP mapping agent allows for filtering the incoming RP announces sent by RP candidates. This type of filtering applies only to Auto-RP announces received by listening to 224.0.1.39 multicast IP address and thus is only effective on Mapping Agents. You configure a filtering statement using the following syntax: **`ip pim rp-announce-filter {group-list <access-list> | rp-list <access-list> [group-list <access-list>]}`**. All access-lists are either numbered or named standard ACLs. RP announces are inspected and if a match is found for the group and the RP IP address, the action is taken based on the "permit" or "deny" statement.

If you omit the **`rp-list`** keyword, then any announces containing groups matching the **`group-list`** are matched. If you omit the **`group-list`** parameter, then all updates from the RPs in **`rp-list`** are matched.

```
R5:
ip access-list standard RP_LIST
 permit 150.1.10.10
!
ip access-list standard GROUP_LIST
 deny 224.110.110.110
 permit any
!
ip pim rp-announce-filter rp-list RP_LIST group-list GROUP_LIST
```

Verification

Note

Use the command **`debug ip pim auto-rp`** on the MA to monitor the updates that are filtered.

```
Rack1R5#debug ip pim auto-rp
PIM Auto-RP debugging is on
Auto-RP(0): Received RP-announce, from 150.1.10.10, RP_cnt 1, ht 181
Auto-RP(0): Filtered -224.110.110.110/32 for RP 150.1.10.10
Auto-RP(0): Update (232.0.0.0/5, RP:150.1.10.10), PIMv2 v1
Auto-RP(0): Update (224.0.0.0/4, RP:150.1.10.10), PIMv2 v1
Auto-RP(0): Received RP-announce, from 150.1.10.10, RP_cnt 1, ht 181
Auto-RP(0): Filtered -224.110.110.110/32 for RP 150.1.10.10
Auto-RP(0): Update (232.0.0.0/5, RP:150.1.10.10), PIMv2 v1
Auto-RP(0): Update (224.0.0.0/4, RP:150.1.10.10), PIMv2 v1
```

Note

Check that the Auto-RP cache in MA does not have the entry for 224.110.110.110 mapped to the RP.

```
Rack1R5#show ip pim rp mapping
PIM Group-to-RP Mappings
This system is an RP-mapping agent (Loopback0)

Group(s) 224.0.0.0/5
  RP 150.1.8.8 (?), v2v1
    Info source: 150.1.8.8 (?), elected via Auto-RP
         Uptime: 00:07:14, expires: 00:02:44
Group(s) 224.0.0.0/4
  RP 150.1.10.10 (?), v2v1
    Info source: 150.1.10.10 (?), elected via Auto-RP
         Uptime: 00:07:04, expires: 00:02:55
  RP 150.1.8.8 (?), v2v1
    Info source: 150.1.8.8 (?), via Auto-RP
         Uptime: 00:07:14, expires: 00:02:41
Group(s) (-)224.110.110.110/32
  RP 150.1.8.8 (?), v2v1
    Info source: 150.1.8.8 (?), elected via Auto-RP
         Uptime: 00:07:14, expires: 00:02:44
Group(s) 232.0.0.0/5
  RP 150.1.10.10 (?), v2v1
    Info source: 150.1.10.10 (?), elected via Auto-RP
         Uptime: 00:07:04, expires: 00:02:55
```

8.14 Auto-RP Listener

- Switch from PIM SM/DM on all interfaces to PIM SM.
- Ensure all routers still hear Auto-RP announces.

Configuration

Note

Auto-RP listener is another solution to allow using Auto-RP without risking any groups falling back to dense mode forwarding. This feature works in tandem with PIM sparse mode enabled on all interfaces (not PIM SM/DM). However, two Auto-RP multicast groups 224.0.1.39 and 224.0.1.40 are flooded in dense mode. No other groups are allowed to use dense mode and thus dangerous flooding fallback is eliminated.

```
R1:
interface FastEthernet 0/0
 ip pim sparse-mode
!
interface Serial 0/0.1
 ip pim sparse-mode
!
ip pim autorp listener

R3:
interface Serial 1/0.1
 ip pim sparse-mode
!
ip pim autorp listener

R4:
interface FastEthernet 0/1
 ip pim sparse-mode
!
interface Serial 0/1
 ip pim sparse-mode
!
ip pim autorp listener

R5:
interface FastEthernet 0/0
 ip pim sparse-mode
!
interface Serial 0/0
 ip pim sparse-mode
!
interface Serial 0/1
 ip pim sparse-mode
!
```

```
ip pim autorp listener
```

R6:

```
interface FastEthernet 0/0.146
 ip pim sparse-mode
!
ip pim autorp listener
```

SW2:

```
interface Vlan 58
 ip pim sparse-mode
!
interface Port-Channel 1
 ip pim sparse-mode
!
ip pim autorp listener
```

SW4:

```
interface Port-Channel 1
 ip pim sparse-mode
!
ip pim autorp listener
```

Verification

Note

Make sure that all transit interfaces are configured for PIM SM. Repeat the following operation on all routers and pay attention to Ver/Mode field.

```
Rack1R5#show ip pim interface

Address          Interface                Ver/   Nbr    Query  DR
DR
                                          Mode   Count  Intvl  Prior
155.1.45.5       Serial0/1                v2/S   1      30     1
0.0.0.0
155.1.58.5       FastEthernet0/0          v2/S   1      30     1
155.1.58.8
155.1.0.5        Serial0/0                v2/S   2      30     1
155.1.0.5
150.1.5.5        Loopback0                v2/SD  0      30     1
150.1.5.5

Rack1R1#show ip pim autorp
AutoRP Information:
  AutoRP is enabled.
  AutoRP groups over sparse mode interface is enabled

PIM AutoRP Statistics: Sent/Received
  RP Announce: 0/0, RP Discovery: 0/41
```

Note

Ensure that all routers are still capable of learning Auto-RP information.

```
Rack1R6#show ip pim rp mapping
PIM Group-to-RP Mappings

Group(s) 224.0.0.0/5
  RP 150.1.8.8 (?), v2v1
    Info source: 150.1.5.5 (?), elected via Auto-RP
         Uptime: 00:07:37, expires: 00:02:45
Group(s) 224.0.0.0/4
  RP 150.1.10.10 (?), v2v1
    Info source: 150.1.5.5 (?), elected via Auto-RP
         Uptime: 00:25:33, expires: 00:02:46
Group(s) (-)224.110.110.110/32
  RP 150.1.8.8 (?), v2v1
    Info source: 150.1.5.5 (?), elected via Auto-RP
         Uptime: 00:21:33, expires: 00:02:47
Group(s) 232.0.0.0/5
  RP 150.1.10.10 (?), v2v1
    Info source: 150.1.5.5 (?), elected via Auto-RP
         Uptime: 00:09:43, expires: 00:02:46
```

Note

Check that Auto-RP groups are still forwarded without any RP information. The output shows Forward/Sparse, by the actual forwarding uses dense mode.

```
Rack1R1#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), 00:20:40/stopped, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Tunnel0, Forward/Sparse-Dense, 00:20:40/00:00:00
    Serial0/0.1, Forward/Sparse, 00:20:40/00:00:00
    FastEthernet0/0, Forward/Sparse, 00:20:40/00:00:00

(150.1.5.5, 224.0.1.40), 00:19:46/00:02:40, flags: LT
  Incoming interface: Serial0/0.1, RPF nbr 155.1.0.5
  Outgoing interface list:
    Tunnel0, Forward/Sparse-Dense, 00:19:47/00:00:00
    FastEthernet0/0, Forward/Sparse, 00:19:47/00:00:00, A
```

8.15 Auto-RP and RP/MA Placement

- Make R3 another MA that preempts the existing MA in R5.
- Provide a solution for R1 to hear Auto-RP discovery messages from R3.

Configuration

Note

As discussed previously, PIM by default treats NBMA interfaces like if they are broadcast-capable. That is, it assumes that all neighbors on a WAN cloud can hear multicast packets sent by any neighbor. However, in non-fully meshed topologies such as hub-and-spoke networks, this is not true. When a spoke sends multicast over the NBMA segment, only hub can hear it. The hub will not forward multicast back to other spokes, based on the split-horizon rule. This problem could be solved using PIM NBMA mode, but this only works with PIM Sparse mode.

Now the real issue is that Auto-RP uses dense mode for RP information dissemination, and PIN NBMA solution would not work. If a candidate RP or a Mapping Agent is placed behind the spoke node, then RP information could be lost. Imagine if a MA is located behind the spoke node. Then any Auto-RP discovery messages will only reach the hub node, but not the other spokes. This could also be the case, if RP and MA are both placed behind NBMA spokes. Therefore, when designing your multicast network, take care and place MA behind the hub. RPs could still be located at spokes, as long as announces reach the hub.

Other solutions include the use of sub-interfaces in the hub router or creating tunnels between the hub and the spokes. Both solutions effectively break the split-horizon rule, by receiving and sending multicast packets on different interfaces. Notice that you will need static mroutes if you don’t run any IGP across the tunnel interfaces.

R1:
```
interface Loopback0
 ip pim sparse-mode
!
ip pim send-rp-discovery loopback 0 scope 10
!
interface tunnel 1
 tunnel source Loopback0
 tunnel destination 150.1.3.3
 ip unnumbered Loopback0
 ip pim sparse-mode
!
router ospf 1
 passive-interface Tunnel1
```

R3:
```
interface tunnel 1
 tunnel source Loopback0
 tunnel destination 150.1.1.1
 ip unnumbered Loopback0
 ip pim sparse-mode
!
! Static mroute is needed to allow R3 hearing
! Auto-RP discoveries from R1.
!
ip mroute 150.1.1.1 255.255.255.255 Tunnel 1
!
router ospf 1
 passive-interface Tunnel1
```

R5:
```
no ip pim send-rp-discovery loopback 0 scope 10
```

Verification

Note

First, make sure that R1 is fully functional as MA. It should receive announces from both SW2 and SW4. Check the multicast routing table for groups 224.0.1.39 for this and ensure the RPF neighbor is R5.

```
Rack1R1#show ip mroute 224.0.1.39
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

(*, 224.0.1.39), 00:02:34/stopped, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Loopback0, Forward/Sparse, 00:02:34/00:00:00
    Tunnel0, Forward/Sparse-Dense, 00:02:34/00:00:00
    Serial0/0.1, Forward/Sparse, 00:02:34/00:00:00
    FastEthernet0/0, Forward/Sparse, 00:02:34/00:00:00

(150.1.8.8, 224.0.1.39), 00:02:08/00:00:56, flags: LT
  Incoming interface: Serial0/0.1, RPF nbr 155.1.0.5
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse, 00:02:08/00:00:00
    Tunnel0, Forward/Sparse-Dense, 00:02:08/00:00:00
    Loopback0, Forward/Sparse, 00:02:08/00:00:00

(150.1.10.10, 224.0.1.39), 00:02:18/00:00:46, flags: LT
  Incoming interface: Serial0/0.1, RPF nbr 155.1.0.5
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse, 00:02:18/00:00:00, A
    Tunnel0, Forward/Sparse-Dense, 00:02:20/00:00:00
    Loopback0, Forward/Sparse, 00:02:20/00:00:00
```

Note

Check that the Auto-RP cache is populated with the announces from SW2 and SW4.

```
Rack1R1#sh ip pim rp mapping
PIM Group-to-RP Mappings
This system is an RP-mapping agent (Loopback0)

Group(s) 224.0.0.0/5
  RP 150.1.8.8 (?), v2v1
    Info source: 150.1.8.8 (?), elected via Auto-RP
         Uptime: 00:02:13, expires: 00:00:45
Group(s) 224.0.0.0/4
  RP 150.1.10.10 (?), v2v1
    Info source: 150.1.10.10 (?), elected via Auto-RP
         Uptime: 00:02:24, expires: 00:00:36
  RP 150.1.8.8 (?), v2v1
    Info source: 150.1.8.8 (?), via Auto-RP
         Uptime: 00:02:13, expires: 00:00:43
Group(s) (-)224.110.110.110/32
  RP 150.1.10.10 (?), v2v1
    Info source: 150.1.10.10 (?), elected via Auto-RP
         Uptime: 00:02:24, expires: 00:00:34
  RP 150.1.8.8 (?), v2v1
    Info source: 150.1.8.8 (?), via Auto-RP
         Uptime: 00:02:13, expires: 00:00:46
Group(s) 232.0.0.0/5
  RP 150.1.10.10 (?), v2v1
    Info source: 150.1.10.10 (?), elected via Auto-RP
         Uptime: 00:02:24, expires: 00:00:33
```

Note

Next, verify that R5 can hear Auto-RP discoveries from R1. Notice that the OIL for the group 224.0.1.40 does not contain the Frame-Relay interface, and thus R3 cannot hear the discovery messages relayed via R5.

```
Rack1R5#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), 00:13:04/stopped, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/0, Forward/Sparse, 00:13:04/00:00:00
    Loopback0, Forward/Sparse-Dense, 00:13:04/00:00:00
    Serial0/1, Forward/Sparse, 00:13:04/00:00:00
    FastEthernet0/0, Forward/Sparse, 00:13:04/00:00:00

(150.1.1.1, 224.0.1.40), 00:02:30/00:02:23, flags: LT
  Incoming interface: Serial0/0, RPF nbr 155.1.0.1
  Outgoing interface list:
    Serial0/1, Forward/Sparse, 00:02:30/00:00:00
    Loopback0, Forward/Sparse-Dense, 00:02:30/00:00:00
    FastEthernet0/0, Forward/Sparse, 00:02:30/00:00:00
```

Note

Check the Auto-RP information in R3 after you have implemented the tunnel workaround. Notice that this workaround will not work without the static mroute. The tunnel is used only to deliver the Auto-RP discoveries to R3, while regular multicast would take paths across the physical links.

```
Rack1R3#show ip pim rp mapping
PIM Group-to-RP Mappings

Group(s) 224.0.0.0/5
  RP 150.1.8.8 (?), v2v1
    Info source: 150.1.1.1 (?), elected via Auto-RP
         Uptime: 00:00:23, expires: 00:02:33
Group(s) 224.0.0.0/4
  RP 150.1.10.10 (?), v2v1
    Info source: 150.1.1.1 (?), elected via Auto-RP
         Uptime: 00:00:23, expires: 00:02:33
Group(s) (-)224.110.110.110/32
  RP 150.1.10.10 (?), v2v1
    Info source: 150.1.1.1 (?), elected via Auto-RP
         Uptime: 00:00:23, expires: 00:02:33
Group(s) 232.0.0.0/5
  RP 150.1.10.10 (?), v2v1
    Info source: 150.1.1.1 (?), elected via Auto-RP
         Uptime: 00:00:23, expires: 00:02:36

Rack1R3#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), 01:11:20/stopped, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Tunnel1, Forward/Sparse, 00:05:59/00:00:00
    Loopback0, Forward/Sparse, 00:43:53/00:00:00
    Serial1/0.1, Forward/Sparse, 01:11:20/00:00:00

(150.1.1.1, 224.0.1.40), 00:02:28/00:02:38, flags: LT
  Incoming interface: Tunnel1, RPF nbr 150.1.1.1, Mroute
  Outgoing interface list:
    Serial1/0.1, Forward/Sparse, 00:02:30/00:00:00
    Loopback0, Forward/Sparse, 00:02:30/00:00:00
```

8.16 Filtering Auto-RP Messages

- Configure R1 so that SW2 and SW4 cannot hear the Auto-RP discovery messages.

Configuration

Note

Auto-RP uses special dense-mode groups to disseminate RP information, and the information is flooded across the multicast domain. The only way to control the scope of the information is by setting the TTL to the value that roughly represents the diameter of the multicast domain. However, the problem is that this does not enforce their strict administrative limits on the information scope, and per-link flooding is not controlled. The other way to filter Auto-RP announces is to use the **`ip multicast boundary`** command, discussed in the separate task.

R1:

```
ip pim send-rp-discovery Loopback0 scope 2
```

Verification

Note

Check that SW2 still has entries in the Auto-RP cache while SW4 not.

```
Rack1SW4#show ip pim rp ma
PIM Group-to-RP Mappings
This system is an RP (Auto-RP)

Rack1SW4#
```

```
Rack1SW2#show ip pim rp ma
PIM Group-to-RP Mappings
This system is an RP (Auto-RP)

Group(s) 224.0.0.0/5
  RP 150.1.8.8 (?), v2v1
    Info source: 150.1.1.1 (?), elected via Auto-RP
         Uptime: 00:32:32, expires: 00:01:58
Group(s) 224.0.0.0/4
  RP 150.1.10.10 (?), v2v1
    Info source: 150.1.1.1 (?), elected via Auto-RP
         Uptime: 00:31:43, expires: 00:02:02
Group(s) (-)224.110.110.110/32
  RP 150.1.10.10 (?), v2v1
    Info source: 150.1.1.1 (?), elected via Auto-RP
         Uptime: 00:31:43, expires: 00:01:59
Group(s) 232.0.0.0/5
  RP 150.1.10.10 (?), v2v1
    Info source: 150.1.1.1 (?), elected via Auto-RP
         Uptime: 00:31:43, expires: 00:02:01
Rack1SW2#
```

8.17 Multicast Boundary

- Configure R5's connection to VLAN58 so that traffic to the group range 232.0.0.0/5 cannot reach SW2.
- Filter the Auto-RP messages to remove the information about this group range.

Configuration

Note

Multicast boundary feature allows for setting administrative borders for multicast traffic. This feature applies filtering to both the control plane traffic (IGMP, PIM, AutoRP) and data plane (installing multicast route states out of the configured interface). It is much more flexible than using Auto-RP TTL scoping and allows applying finer and granular access control. Using this feature you may contaminate multicast traffic within the boundaries of your administrative domain, without relying on TTL-based filtering.

When you apply the command **`ip multicast boundary <access-list> [filter-autorp]`** to an interface, the following filtering rules apply.

1) If the access-list is a standard ACL, than any ingress IGMP or PIM messages are inspected to see if the group being joined or tree being built has a match in the access-list. This might be an (S,G) or (*,G) join. Additionally, the interface is used as outgoing interface to forward a group G only if the group matches the access-list.

2) If the access-list is an extended ACL, then it specifies both multicast sources and groups, using the format **`permit ip <src-ip> <src-wildcard> <group-address> <group-mask>.`** Any incoming PIM/IGMP messages are inspected, and if both the source and group are matched, they are permitted. At the same time, the interface could be used as outgoing for multicast traffic sourced off the IP addresses matching the extended access-list with the group matching the same access-list entry. If you want to match only (*,G) shared tree signaling, use the source IP address of 0.0.0.0 – this affects PIM Join/Prune messages.

Keep in mind that unicast PIM Register messages are not affected by the multicast-boundary configuration, and must be filtered using the respective feature.

If you have specified the **`filter-autorp`** keyword, then the router will inspect any Auto-RP messages (announces or discovery) and filter away those not matching the access-list. Note that the access-list must be a *standard* ACL if you use Auto-RP filtering. In order for Auto-RP group range to be permitted, the whole range must be covered by permit statements in the access-list. If any of the range's group is not permitted, the whole range is removed from the advertisement.

Starting with version 12.3(17)T you may use **`in`** or **`out`** options to the multicast boundary command (does not work with the **`filter-autorp`** option though). When applied as ingress, the command affect control plane traffic – IGMP/PIM Joins and Auto-RP messages. When configured as egress filter, it will control the interface being added to the OIL for multicast groups, allowing only the groups permitted by the access-list.

```
R5:
ip access-list standard PERMITTED_GROUPS
 deny 232.0.0.0 7.255.255.255
 permit any
!
interface FastEthernet 0/0
 ip multicast boundary PERMITTED_GROUPS filter-autorp
```

Verification

Note

Notice the following console message on R5. Because the range 224.0.0.0-239.255.255.255 advertised by both SW2 and SW4 *overlaps* with the denied group range, it is also filtered from Auto-RP announces, in addition to the explicitly denied range. This results in the following RP to group mapping in the MA.

```
Rack1R5#
%AUTORP-4-OVERLAP: AutoRP Announcement packet, group 224.0.0.0 with
mask 240.0.0.0 removed because of multicast boundary for 232.0.0.0 with
mask 248.0.0.0

Rack1R1#show ip pim rp mapping
PIM Group-to-RP Mappings
This system is an RP-mapping agent (Loopback0)

Group(s) 224.0.0.0/5
  RP 150.1.8.8 (?), v2v1
    Info source: 150.1.8.8 (?), elected via Auto-RP
         Uptime: 01:34:10, expires: 00:02:46
Group(s) (-)224.110.110.110/32
  RP 150.1.10.10 (?), v2v1
    Info source: 150.1.10.10 (?), elected via Auto-RP
         Uptime: 01:33:20, expires: 00:02:34
  RP 150.1.8.8 (?), v2v1
    Info source: 150.1.8.8 (?), via Auto-RP
         Uptime: 01:34:10, expires: 00:02:46
```

Note

Based on the above, SW2 won’t be even able to join the shared trees for the denied groups, as it does not know the RP for them.

8.18 PIM Bootstrap Router

- Remove the Auto-RP configuration in SW2, SW4 and R1.
- Remove the multicast boundary in R5.
- Configure R5 to advertise itself as the RP for all multicast groups using the standards-based protocol.

Configuration

Note

Bootstrap router or BSR is standard-based solution available with PIMv2, which performs the same function as Auto-RP, i.e. disseminates RP information. Both protocols use the concept of candidate RP. However, BSR does not use any dense-mode groups to distribute RP-to-Group mapping information. Instead, the information is flooded using PIM message, on hop-by-hop basis. That is, when a router receives a candidate RP announce inside a PIM message, it applies RPF check, validating that announce was received on the interface that is on the shortest path to the RP. If the RPF check succeeds, the message is flooded out of all PIM-enabled interfaces.

In order to configure a candidate RP, use the command **`ip pim rp-candidate <PIM-Enabled-Interface> [group-list <Standard-ACL>] [interval <Seconds>] [priority <0-255>]`**. If you omit all arguments, the router will start advertising itself as the RP for all groups. You may specify the list of groups using the **`group-list`** argument. All entries in this list are “positive”, if you compare that to Auto-RP, and you cannot use “negative” groups. Priority value is used when the routers select the best RP for a given group, and lower values are preferred. The default priority is zero (which is against standard, which specifies 192) and is the highest possible value. You may rarely want to change the priority value for a candidate RP, possibly only in cases when you want to gracefully take the RP out of service.

Now, the part where BSR differs from Auto-RP is the bootstrap router itself. This router performs the role similar to Auto-RP MA, by listening to candidate RP announcements. However, unlike Auto-RP MA, BSR does not elect the best RP for every group range. Instead of this, for every group range it builds a set of candidate RPs, including all routers that advertised their willingness to service this group range. This is called group range to RP set mapping.

The resulting array of group range to RP set mappings is distributed by the BSR using PIM messages and the same flooding procedure described above. The command to configure a router as a BSR is **`ip pim bsr-candidate <Interface-Name> [hash-mask-length] [priority]`**. Ignore the

`hash-mask-length` parameter for a moment, and notice the priority field. By default, the priority of zero is advertised in all BSR messages. The higher is the priority value, the more preferred is the BSR. The IP address of the interface used to source the BSR messages is used as a tie-breaker in case if two priorities match – the higher IP is preferred. If there are multiple BSRs, they all listen to other potential BSR messages. If a BSR hears a message with higher priority or IP address, it immediately stops its own BSR advertisements. This process ensures unique BSR in the domain while maintaining some redundancy.

The bootstrap messages are received by every multicast router and used to populate their RP cache. Note that it's up to the routers to select the best matching RP from the sets advertised by the BSR router. In order to facilitate RP load-balancing, routers may use special hash function to select the best RP from the set that services the same group range.

R1:
```
no ip pim send-rp-discovery Loopback 0
```

R5:
```
ip pim rp-candidate Loopback0
ip pim bsr-candidate Loopback0
!
interface FastEthernet 0/0
 no ip multicast boundary PERMITTED_GROUPS filter-autorp
```

SW2:
```
no ip pim send-rp-announce Loopback 0
```

SW4:
```
no ip pim send-rp-announce Loopback 0
```

Verification

Note

Enable PIM BSR debugs in R5 to see how BSR messages are originated. Notice that R5 sends both candidate RP and BSR router messages. After that, check RP mappings on all multicast routers to ensure they all received BSR information.

```
Rack1R5#debug ip pim bsr
PIM-BSR debugging is on
Rack1R5#
PIM-BSR(0): RP-set for 224.0.0.0/4
PIM-BSR(0):   RP(1) 150.1.5.5, holdtime 150 sec priority 0
PIM-BSR(0): Bootstrap message for 150.1.5.5 originated
PIM-BSR(0): Build v2 Candidate-RP advertisement for 150.1.5.5 priority
0, holdtime 150
PIM-BSR(0):  Candidate RP's group prefix 224.0.0.0/4
PIM-BSR(0): Send Candidate RP Advertisement to 150.1.5.5
PIM-BSR(0):  RP 150.1.5.5, 1 Group Prefixes, Priority 0, Holdtime 150

Rack1R1#show ip pim bsr-router
PIMv2 Bootstrap information
  BSR address: 150.1.5.5 (?)
  Uptime:      00:21:04, BSR Priority: 0, Hash mask length: 32
  Expires:     00:01:35

Rack1R1#show ip pim rp mapping
PIM Group-to-RP Mappings

Group(s) 224.0.0.0/4
  RP 150.1.5.5 (?), v2
    Info source: 150.1.5.5 (?), via bootstrap, priority 0, holdtime 150
         Uptime: 00:05:23, expires: 00:02:06

...

Rack1R6#show ip pim rp mapping
PIM Group-to-RP Mappings

Group(s) 224.0.0.0/4
  RP 150.1.5.5 (?), v2
    Info source: 150.1.5.5 (?), via bootstrap, priority 0, holdtime 150
         Uptime: 00:06:34, expires: 00:01:52
```

Note

Always remember that BSR messages are subject to RPF checks. If your router does not receive BSR information, enable the command **`debug ip pim bsr`** and see if the locally received BSR packets are dropped due to RPF checks. For example, temporarily configure a static route in R1 for R5’s Loopback0 to make R1 think it’s reachable via **R6:**

```
R1:
ip route 150.1.5.5 255.255.255.255 155.1.146.6
```

```
Rack1R1#
PIM-BSR(0): bootstrap from non-RPF neighbor 155.1.146.6
PIM-BSR(0): bootstrap on non-RPF path Tunnel1
PIM-BSR(0): bootstrap on non-RPF path Serial0/0.1

Rack1R1#show ip pim rp mapping
PIM Group-to-RP Mappings

Rack1R1#
```

8.19 BSR - Multiple RP Candidates

- Configure SW2 and SW4 to advertise themselves as RP candidates for all multicast groups.
- R5 should distribute this information and instruct all routers to load-balance multicast groups between the two RPs.
- Use the maximum possible hash mask length to evenly distribute the load across the RPs.

Configuration

Note

As mentioned in the previous task, BSR protocol distributes multicast group ranges along with the RP-Set information. This allows the multicast routers to effectively load-balance among multiple RPs using special procedure. This procedure is deterministic, and ensures that for a given group ALL multicast routers will select the same RP. This is needed to make sure a source for a given group will not register to an RP that is not selected for this group. Here is how the procedure it works.

Input: Group Address (G), RP-Set (R1, R2… Rn), Mask (distributed by BSR).

1) Among the routers in RP-Set, select those that have numerically lowest priority values. By default all cRPs advertise priority of zero, so they all are eligible. You may adjust the priority and take some of the RPs out of service gracefully.

2) For every RP IP address, calculate the hash function value:

value1 = Hash(G&Mask, R1), value2 = Hash(G&Mask, R2) … valueN = Hash(G&Mask,Rn).

Notice that Group IP address is ANDed with the Mask value. Mask value is propagated by the BSR using the **`hash-mask-length`** parameter. Thus, only first **`hash-mask-length`** bits of the Group are used to calculate the hash value. The default value is zero – i.e. the group IP address is ignored when computing the hash value and all groups map to the same RP.

3) Select the RP with the highest hash function value for a given group. In case if the value are the same, select the RP with the highest IP address.

Using this “pseudo-random” selection procedure, the whole multicast address space is partitioned among different RPs. Every RP will get approximately

2^[32-**`hash-mask-length]`** groups assigned, provided that there is enough RPs to evenly distribute the load.

BSR protocol implements automatic failover procedure. If any of the RPs would fail, the BSR will exempt it from bootstrap messages and the automatic failover will occur. The failover delay is based on the RP/BSR advertisement intervals. Naturally, if there are redundant BSRs, then in case of the primary failure, the secondary would revive and take its role.

In this task, odd/even groups are distributed among two RPs, both covering the same group ranges, based on the hash mask length of 31 bits. This ensures the load is evenly distributed among the two RPs.

R5:
```
no ip pim rp-candidate Loopback0
ip pim bsr-candidate Loopback0 31
```

SW2:
```
ip pim rp-candidate Loopback0
```

SW4:
```
ip pim rp-candidate Loopback0
```

Verification

Note

You may quickly find which RP will be selected for a given group using the show **`ip pim rp-hash`** command. For example:

```
Rack1R4#show ip pim rp-hash 239.1.1.1
  RP 150.1.10.10 (?), v2
    Info source: 150.1.5.5 (?), via bootstrap, priority 0, holdtime 202
         Uptime: 00:05:03, expires: 00:02:51
  PIMv2 Hash Value (mask 255.255.255.254)
    RP 150.1.10.10, via bootstrap, priority 0, hash value 989207280
    RP 150.1.8.8, via bootstrap, priority 0, hash value 718054422

Rack1R4#show ip pim rp-hash 239.1.1.2
  RP 150.1.8.8 (?), v2
    Info source: 150.1.5.5 (?), via bootstrap, priority 0, holdtime 205
         Uptime: 00:05:07, expires: 00:02:55
  PIMv2 Hash Value (mask 255.255.255.254)
    RP 150.1.10.10, via bootstrap, priority 0, hash value 1093093598
    RP 150.1.8.8, via bootstrap, priority 0, hash value 1364246456
```

Note

Notice the RP selected for the group and the hash function values.

8.20 Filtering BSR Messages

- Configure your network so that R6 does not learn any RP information,

Configuration

Note

As you may recall, filtering Auto-RP messages requires applying the administrative multicast boundary. Filtering BSR messages is much easier, as those are carried inside PIM. When you apply the command **`ip pim bsr-border`** on a link, the BSR messages are not flooded or received on this link. This means that any RP or BSR advertisements are filtered, effectively isolating two domains of RP information exchange.

R1:
```
interface FastEthernet0/0
 ip pim bsr-border
```

R4:
```
interface FastEthernet0/1
 ip pim bsr-border
```

Verification

Note

Ensure that R6 did not learn any RP information via BSR.

```
Rack1R6#show ip pim rp mapping
PIM Group-to-RP Mappings

Rack1R6#
```

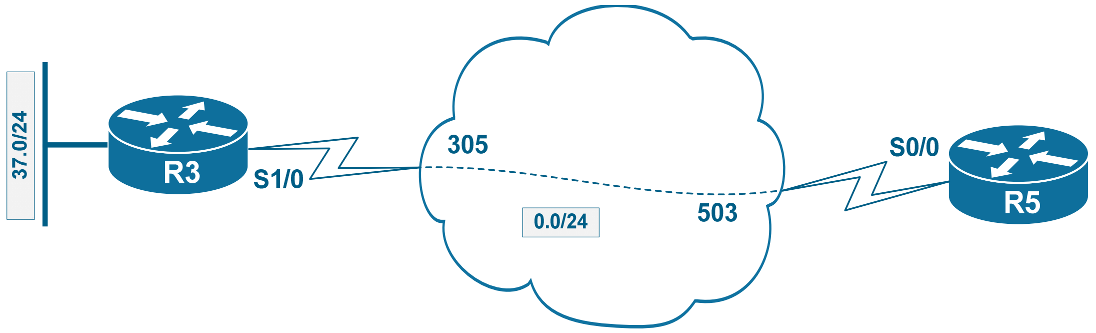
8.21 Stub Multicast Routing & IGMP Helper

- Remove the tunnel connecting R1 and R3. Configure R3 as the stub multicast router with SW1 emulating a host on the stub segment.

Configuration

Note

Stub multicast routing is useful in situations when you have small remote site connected to the centralized WAN cloud across low-bandwidth links and using the low-end routers. Such remote site never performs any transit function, and may suffer from resource starvation due to PIM DM flood-and-prune behavior, or PIM SM RP announces, RP cache growth and mroute states proliferation.



Look at the figure. Assume R5 is the central router and R3 is the stub remote router with directly connected receivers. If both routers are running PIM DM, then periodic flood-and-prune behavior would overwhelm the WAN link. If both routers run PIM SM, then R3 will have to accept RP discovery messages, build its own RP cache and maintains states for all multicast groups joined by local receivers, thus draining out small router's resources.

Stub multicast router allows R3 to be exempted from PIM and IGMP messages processing. Instead of that, R3 is configured to forward all IGMP messages received on its client-facing interface to R5. Thus, R3 never creates any group states. The command to achieve this is **`ip igmp helper-address <Upstream-IP>`**. As a result, R5 will track all IGMP states for client on the segment connected to R3.

Next, R3 is configured with PIM DM on both uplink (Frame-Relay in our case) and client-facing interfaces. This will allow the router to flood ANY multicast received from its upstream down to clients, based on default PIM DM behavior, as there is no downstream router to prune the tree.

Finally, R5 is configured with PIM enabled on its downstream interface, but with special neighbor filter applied, to make sure the upstream and the downstream routers never form PIM adjacency. This allows the upstream router flooding multicast traffic downstream, as IOS will not send a multicast packet out of non-PIM interface. At the same time, the downstream router will not be able to prune any group using PIM signaling. The command to filter PIM neighbors is **`ip pim neighbor-filter <ACL>`** where the standard ACL permits or denies particular neighbors.

The net effect is that the upstream router (R5) builds multicast distribution tree on behalf of the downstream router (R3) by the virtue of IGMP proxy function performed by the downstream. At the same time, no excessive load is being put on the stub router or the stub link, effectively saving bandwidth and router resources. Effectively, R5 performs all multicast routing functions for R3, and the latter is only used as “dumb” packet forwarder.

R3:
```
no interface Tunnel 1
!
interface FastEthernet0/0
 ip igmp helper-address 155.1.0.5
 ip pim dense-mode
!
interface Serial 1/0.1
 ip pim dense-mode
```

R5:
```
no access-list 33
access-list 33 deny 155.1.0.3
access-list 33 permit any
!
interface Serial 0/0
 ip pim sparse-mode
 ip pim neighbor-filter 33
```

Note

We enabled PIM on SW1’s interface connected to R3 to activate multicast processing on this interface. Neighbor filter is used to avoid establishing PIM neighbors with R3.

```
SW1:
ip multicast-routing distributed
!
access-list 7 deny any
!
interface FastEthernet 0/3
 ip igmp join-group 239.1.1.7
 ip pim dense-mode
 ip pim neighbor-filter 7
```

Verification

Note

Check that R5 sees the group 239.1.1.7 as directly connected on its Frame-Relay interface. Next, ping the group from SW2.

```
Rack1R5#show ip igmp groups
IGMP Connected Group Membership
Group Address    Interface                Uptime    Expires   Last
Reporter   Group Accounted
239.1.1.7        Serial0/0                00:00:15  00:02:44  155.1.0.3
<snip>

Rack1SW2#ping 239.1.1.7 repeat 100

Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 239.1.1.7, timeout is 2 seconds:
.
Reply to request 1 from 155.1.37.7, 67 ms
Reply to request 2 from 155.1.37.7, 67 ms
Reply to request 3 from 155.1.37.7, 67 ms
```

Note

Check multicast route states in R5 and R3. Notice that R5 uses sparse-mode SPT to forward traffic down to R3. In turn, R3 simply floods the traffic using dense mode forwarding to the directly connected source.

```
Rack1R5#show ip mroute 239.1.1.7
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

(*, 239.1.1.7), 00:07:21/stopped, RP 150.1.8.8, flags: SJC
  Incoming interface: FastEthernet0/0, RPF nbr 155.1.58.8
  Outgoing interface list:
    Serial0/0, Forward/Sparse, 00:02:15/00:02:42

(155.1.58.8, 239.1.1.7), 00:00:15/00:02:58, flags: JT
  Incoming interface: FastEthernet0/0, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/0, Forward/Sparse, 00:00:15/00:02:44

Rack1R5#
```

```
Rack1R3#show ip mroute 239.1.1.7
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

(*, 239.1.1.7), 00:14:41/stopped, RP 150.1.8.8, flags: SJC
  Incoming interface: FastEthernet0/0, RPF nbr 155.1.37.7
  Outgoing interface list:
    Serial1/0.1, Forward/Dense, 00:14:41/00:00:00

(155.1.58.8, 239.1.1.7), 00:00:27/00:02:54, flags: JT
  Incoming interface: Serial1/0.1, RPF nbr 155.1.0.5
  Outgoing interface list:
    FastEthernet0/0, Forward/Dense, 00:00:27/00:00:00
```

8.22 IGMP Filtering

- Only permit R3 to accept IGMP joins for groups in range 239.1.1.0/24 on its connection to SW1.
- Limit the number of concurrent IGMP states on the interface to 10.

Configuration

Note

IGMP is the protocol used by multicast receivers their willingness to listen to a particular multicast group. When a host wants to join a multicast group, it sends an IGMP report message to the multicast address heard by all routers on the segment. This report contains the multicast group that the host wants to join. The multicast router may control groups allowed to be joined by the receivers. When you apply the command **`ip igmp access-group <ACL>`** to an interface, the router will filter all attempts to join groups not matching the access-list. Recall that the similar goal could be accomplished using the command **`ip multicast boundary`**, but **`ip igmp access-group`** is commonly used on the interfaces facing the receivers. Notice that you can use either standard or extended access-lists with this command.

If you use a standard access-list, your configuration applies to IGMP v1, v2 and v3 receivers. The hosts are allowed to listen to the channels (multicast groups) matching an entry in the access-list. If you use an extended access-list, then you may also selectively filter IGMPv3 reports. IGMPv3 allows receivers to join explicit sources along with the multicast group. That is, every IGMPv3 report contains the list of groups along with the multicast sources that the receiver wants to listen. The access-list entry will have the format **`permit ip <src-ip> <src-mask> <group-ip> <group-mask>`**. If you want to filter joins to any source, use the 0.0.0.0 255.255.255.255 wildcard pair. However, if you want to filter IGMPv2 or v1 joins, that don't support explicit source specification, use the host IP address of **`0.0.0.0`** for the source.

Another useful feature is limiting the number of mroute states created for the interface as a result of IGMP reports. The same command **`ip igmp limit <N>`** could be applied globally and per-interface at the same time. In the first case, it limits the aggregate number of multicast groups joined by directly connected receivers on all multicast interfaces. When applied per-interface, it limits the number of different multicast groups that could be joined on this particular interface.

R3:
```
ip access-list standard IGMP_FILTER
```

```
 permit 239.1.1.0 0.0.0.255
!
interface FastEthernet0/0
 ip igmp access-group IGMP_FILTER
 ip igmp limit 10
```

Verification

Note

Start by checking the IGMP settings on R3's interface. Notice that the maximum number of allowed IGMP states is 10.

```
Rack1R3#show ip igmp interface fastEthernet 0/0
FastEthernet0/0 is up, line protocol is up
  Internet address is 155.1.37.3/24
  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 IGMP_FILTER
  IGMP activity: 6 joins, 5 leaves
  Interface IGMP State Limit : 1 active out of 10 max
  Multicast routing is enabled on interface
  Multicast TTL threshold is 0
  Multicast designated router (DR) is 155.1.37.7
  IGMP querying router is 155.1.37.3 (this system)
  IGMP helper address is 155.1.0.5
  No multicast groups joined by this system
```

Note

Now configure SW1 to join a different group, e.g. 239.2.2.7.

```
SW1:
interface FastEthernet 0/3
 ip igmp join-group 239.2.2.7
```

Check that there is still just one IGMP state create in **R3:**

```
Rack1R3#show ip igmp groups
IGMP Connected Group Membership
Group Address    Interface              Uptime    Expires   Last
Reporter   Group Accounted
239.1.1.7        FastEthernet0/0        00:03:09  00:02:56
155.1.37.7      Ac
224.0.1.40       Loopback0              04:57:32  00:01:57  150.1.3.3
```

```
Rack1R3#show ip igmp interface fastEthernet 0/0
FastEthernet0/0 is up, line protocol is up
  Internet address is 155.1.37.3/24
  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 IGMP_FILTER
  IGMP activity: 6 joins, 5 leaves
  Interface IGMP State Limit : 1 active out of 10 max
  Multicast routing is enabled on interface
  Multicast TTL threshold is 0
  Multicast designated router (DR) is 155.1.37.7
  IGMP querying router is 155.1.37.3 (this system)
  IGMP helper address is 155.1.0.5
  No multicast groups joined by this system
```

8.23 IGMP Timers

- Configure the designated IGMP querier on VLAN146 segment so that failed multicast traffic receivers are detected and removed within 60 seconds.
- Every active receiver should respond to general IGMP queries within 4 seconds.
- Designated querier failures should be detected 3 times faster than by default.
- Since there is just one receiver on R3's connection to SW1, configure the router to remove group states immediately, as soon as IGMP leave report is received.

Configuration

Note

IGMP reports are sent asynchronously, and thus might be missed by the router. Therefore, one of the multicast-capable routers sharing the same segment is elected as the designated IGMP querier, and periodically sends IGMP Membership query to all hosts on the segment. Every host receiving the Membership query should respond with an IGMP report containing all joined groups. The designated querier is the router with the lowest IP address on the segment, if the running IGMP version is 2 or 3. Notice that the PIM DR on the segment is elected based on the highest IP address by default, and thus some functional decoupling and load-balancing may occur if there is more than one router on the segment. In addition to sending the periodic membership queries, the designated querier also builds PIM SM shared trees, on behalf of the receivers signaling multicast group membership.

Periodic queries are sent at the interval defined by the interface-level command **`ip igmp query-interval <seconds>.`** If a non-designated multicast router does not hear the membership queries for more than the interval defined by the command **`ip igmp querier-timeout <seconds>`** it will attempt to become a designated querier itself, assuming that the old one failed. If the latter command is not configured on the interface, the querier timeout equals twice the query-interval configured on the same interface. The default query-interval value is 60 seconds (though RFC recommends 125 seconds).

When IGMPv1 is configured on the interface explicitly, the router times out group state if the is no report for this group during 3 consecutive membership queries (180 seconds by default). This might result in very high leave latency, as IGMPv1 have no explicit group leave message. IGMPv2 significantly enhances the procedure of leaving a multicast group as detailed below:

1) Every membership query message contains a special timer value, defined by the command **`ip igmp query-max-response-time [time-in-seconds].`** Every host on the segment is supposed to send an IGMP report during the time-window defined by this interval. All hosts count to the interval value and randomly fire the IGMP report. If a host hears another report for the same group, it suppresses its own message. Thus, excessive report flooding is avoided. If there is no report for a given group during the query-max-response interval, the router removes mroute state for this group.

2) Every host, willing to leave a group, may send out a special IGMP Leave report for this particular group. As soon as the router receives a leave report, it generates a special IGMP group specific query for the multicast channel in question. This query is needed to check if there are any other hosts on the segment that still need this group. The number of special queries generated is defined by the command **`ip igmp last-member-query-count <N>`** which is default to 2. The Queries are sent at the intervals specified by the command **`ip igmp last-member-query-interval <milliseconds>`** which is 1000ms by default. If there is no answer to the special queries, the group state is removed from mroute table.

3) If there is just one receiver for a particular group on the segment, e.g. if there is just one host connected to the router, the leave latency could be further reduced by configuring the **`ip igmp immediate-leave group-list <access-list>`** command. If there is an IGMP leave message received on the interface for a group matching the access-list, the mroute state is immediately removed, without any further delays. The access-list is a standard ACL that defines the groups eligible for this feature.

Notice that the explicit leave feature is also available for IGMPv3, with the extension to the multicast group source. That is, a host may leave a particular sender for the group, while continuing listening to the other senders.

For our scenario, R1 is the designated querier for the segment. Since R4 has the next lowest IP address, it is the backup querier. In real life, you would configure every router on the segment with the same settings, but the task's goal is to check your understanding or querier election. For the immediate leave feature we configure only the groups permitted by the IGMP access-group. You may configure it for all groups, as the resulting effect would be the same.

R1:
```
interface FastEthernet0/0
  ip igmp query-interval 20
  ip igmp query-max-response-time 4
```

R3:
```
ip access-list standard IMMEDIATE_LEAVE
 permit 239.1.1.0 0.0.0.255
!
interface FastEthernet 0/0
 ip igmp immediate-leave group-list IMMEDIATE_LEAVE
```

R4:
```
interface FastEthernet0/1
 ip igmp querier-timeout 60
```

Verification

Note

Verify timers using **`the show igmp interface`** command.

```
Rack1R1#show ip igmp interface fastEthernet 0/0
FastEthernet0/0 is up, line protocol is up
  Internet address is 155.1.146.1/24
  IGMP is enabled on interface
  Current IGMP host version is 2
  Current IGMP router version is 2
  IGMP query interval is 20 seconds
  IGMP querier timeout is 40 seconds
  IGMP max query response time is 4 seconds
  Last member query count is 2
  Last member query response interval is 1000 ms
  Inbound IGMP access group is not set
  IGMP activity: 4 joins, 1 leaves
  Multicast routing is enabled on interface
  Multicast TTL threshold is 0
  Multicast designated router (DR) is 155.1.146.1 (this system)
  IGMP querying router is 155.1.146.1 (this system)

Rack1R4#show ip igmp interface fastEthernet 0/1
FastEthernet0/1 is up, line protocol is up
  Internet address is 155.1.146.4/24
  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 60 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: 5 joins, 2 leaves
  Multicast routing is enabled on interface
  Multicast TTL threshold is 0
  Multicast designated router (DR) is 155.1.146.1
  IGMP querying router is 155.1.146.1
  No multicast groups joined by this system
```

Note

Now enable IGMP debugging in R3 and unsubscribe SW1 from the group 239.1.1.7. Notice that the group state is removed immediately, without any additional last member queries.

```
SW1:
interface FastEthernet 0/3
 no ip igmp join-group 239.1.1.7

Rack1R3#debgu ip igmp
IGMP(0): Received Leave from 155.1.37.7 (FastEthernet0/0) for 239.1.1.7
IGMP(0): Leave group 239.1.1.7 immediately on FastEthernet0/0
IGMP_ACL(0): Group 239.1.1.7 unaccounted on FastEthernet0/0
IGMP(0): Send v2 general Query on FastEthernet0/0
IGMP(0): Deleting 239.1.1.7 on FastEthernet0/0
IGMP(0): Helper Leave for group 239.1.1.7 out Serial1/0.1
IGMP(0): Received v2 Query on Serial1/0.1 from 155.1.0.5
IGMP(0): Previous querier timed out, v2 querier for Serial1/0.1 is this
systemv2 querier for Serial1/0.1 is this system.
IGMP(0): Received v2 Query on Serial1/0.1 from 155.1.0.5
IGMP(0): Received v2 Query on Serial1/0.1 from 155.1.0.5
IGMP(0): Previous querier timed out, v2 querier for Serial1/0.1 is this
systemv2 querier for Serial1/0.1 is this system.
IGMP(0): Received v2 Report on FastEthernet0/0 from 155.1.37.7 for
239.2.2.7
IGMP(*): Group 239.2.2.7 access denied on FastEthernet0/0
```

8.24 Multicast Helper Map

- When R6 sends broadcast UDP packets on port 5000, those packets should be transported across the network and broadcasted on VLAN37 segment.
- Use the group 239.1.1.100 to accomplish this task and use the DNS broadcasts for testing.
- You may use static mroutes if needed to accomplish the task.

Configuration

Note

The purpose of the feature is to allow forwarding of broadcast traffic across multicast capable network. Generally, you need a single Layer2 domain between two nodes to let them hear each other's broadcast packets. However, broadcast UDP packets could be relayed between two subnets using special IOS feature know as helper-address. There are two variations of this feature:

1) Unicast helper (**`ip helper-address`**), which converts the broadcast destination address to the fixed unicast IP address. Most often this feature is used with DHCP to forward requests to the server.
2) Multicast helper (**`ip multicast helper-map`**), which converts the broadcast destination to a fixed multicast address.

Multicast helper-map feature allows scalable forwarding of broadcast traffic between disjoint segments. This is often need to support legacy applications like stock tickers that use broadcast to deliver information to multiple sources simultaneously. To configure multicast helper, follow the steps outlined below:

Step 1:
Set up a multicast network between the segments that should exchange broadcast packets. You should select a group to deliver the broadcast packets and decide which PIM mode to use. If you chose PIM SM, make sure the group you chose maps to an RP. Make sure multicast works, by joining an interface in the egress router (closest to the broadcast receiver) to the selected group and pinging this group from the ingress router (closest to the broadcast source).

Step 2:
Enabling broadcast forwarding in the ingress router – the one directly connected to the source. If there are multiple sources, you have to configure all respective routers. Use the command **`ip forward-protocol udp`**

`<port-number>` to enable forwarding of broadcast UDP packets sent to the specified port-number.

Step 3:
Configuring multicast helper-map in the ingress routers to redirect broadcast packets to the selected multicast address. The syntax for this interface-level command is **`ip multicast helper-map broadcast <mcast-address> <ACL>`**. The access-list controls which broadcast packets are eligible to be converted into the multicast. For example, if you want to forward UDP packets destined to port 5000 use the access-list similar to the following: **`access-list 100 permit udp any any eq 5000`**. Note that the same UDP port must be enabled for broadcast forwarding at **Step 2**. All broadcast traffic received on the configured interface and matching the access-list is converted and sent to the specified multicast address. If the group is in sparse mode, the ingress router will register the source with the RP, per the usual procedure.

Step 4:
Enabling broadcast forwarding in the egress router, i.e. the router directly connected to the destination subnet. Use the same command that you used at Step 2 **`ip forward-protocol udp <port-number>`** to accomplish this. Next, enable multicast helper map in the egress router on all interfaces that may *receive* multicast traffic. Note that you should not configure the multicast-helper on the interface connected to the destination. Use the command **`ip multicast helper-map <mcast-group> <directed-broadcast-IP> <ACL>`** where mcast-group is the same group you used at **Step 3** and directed-broadcast-IP is the broadcast subnet IP address on the segment that receives the broadcast traffic.

Step 5:
Enable directed broadcasts on the interface connected to the receiving segment using the command **`ip directed-broadcast`**. This is needed to successfully send broadcasts out of this segment. By default, the broadcasts are sent to the address 255.255.255.255 irrespective of the **`directed-broadcast-IP`** configured at **Step 4**. If you want to use a different address, put the command **`ip broadcast-address <IP>`** on the same segment.

To test your configuration, you need a broadcast packet source. You may use the IP SLA command to generate UDP packets to a segment broadcast address, but this might not work on some platforms/IOS versions. If that's the case, you may use either of the following tricks:

1) Enable DNS names resolution but do not configure a DNS server. After this, the router will broadcast for any DNS name entered in the command line using the address 255.255.255.255 out of all interfaces. You will need to

adjust the port in your access-lists to forward this broadcast traffic.

2) Configure extended **`traceroute`** command parameters to trace to the broadcast destination, starting off the port number that is covered by your ACL.

☠ Pitfall

Our particular scenario poses some caveats.

1) You need to select the proper ingress router. The group 239.1.1.100 is to be transported in the sparse mode. Thus, the source should be first registered with the RP. The only router allowed to register sources is the DR, which is R1. Thus, we configure R1 to generate multicast packets.

2)The next issue, is that R3 is stub multicast router. Thus, it does not exchange any PIM messages with R5 and it won't join group 239.1.1.100 once you configure the multicast helper on the ingress interface. To resolve this issue, SW1 should be configured with static IGMP join for group 239.1.1.100 to let R5 know that about that.

3) If you loaded our default configuration, then you will see that R3 has free equal-cost routes to reach VLAN146 segment. This will cause RPF failures, unless you configure **`ip multicast multipath`** in R3, allowing R3 to perform RPF across the equal-cost paths.

4) Finally, R5 will prefer path to VLAN146 segment across the Serial connection to R4. Thus, in order to allow R5, which acts on behalf of R3, to join the SPT towards the VLAN146 segment, you need a static mroute in R5, to make R1 the RPF neighbor for VLAN146 subnet.

Also notice that the scenario requires PIM NBMA mode on R5's Frame-Relay interface to work properly.

R1:
```
ip forward-protocol udp 5000
!
ip access-list extended TRAFFIC
 permit udp any any eq 5000
 permit udp any any eq 53
!
interface FastEthernet 0/0
 ip multicast helper-map broadcast 239.1.1.100 TRAFFIC
```

R3:
```
ip forward-protocol udp 5000
ip multicast multipath
!
ip access-list extended TRAFFIC
 permit udp any any eq 5000
 permit udp any any eq 53
!
interface FastEthernet 0/0
 ip directed-broadcast
 ip broadcast-address 155.1.37.255
!
interface Serial 1/0.1
 ip multicast helper-map 239.1.1.100 155.1.37.255 TRAFFIC
```

R5:
```
ip mroute 155.1.146.0 255.255.255.0 155.1.0.1
```

Verification

Note

For verification we are going to use DNS broadcasts sent from R6. Configure R6 to resolve DNS names, but do not provide any DNS server:

```
R6:
ip domain lookup
```

Enable some debugging and start traffic flow in **R6:**

```
Rack1R1#debug ip mpacket
Rack1R1#conf t
Rack1R1(config)#access-list 100 permit udp any any eq 53
Rack1R1#debug ip packet detail 100

Rack1R3#debug ip mpacket
Rack1R3#conf t
Rack1R3(config)#access-list 100 permit udp any any eq 53
Rack1R3#debug ip packet detail 100

Rack1R6#dddd
Translating "dddd"...domain server (255.255.255.255)
```

Note

R1 accepts the broadcasts and converts them to the multicast packets. Initially, the SPT is not built and some packets are lost, and the OIL for (S,G) is empty. When the SPT is finally finished, everything works smoothly.

```
Rack1R1#
IP: tableid=0, s=155.1.146.6 (FastEthernet0/0), d=155.1.146.255
(FastEthernet0/0), routed via RIB
IP: s=155.1.146.6 (FastEthernet0/0), d=155.1.146.255 (FastEthernet0/0),
len 50, rcvd 3
    UDP src=54670, dst=53
IP(0): s=155.1.146.6 (FastEthernet0/0) d=239.1.1.100 id=0, ttl=254,
prot=17, len=64(50), mroute OIL null
IP: tableid=0, s=155.1.146.6 (FastEthernet0/0), d=155.1.146.255
(FastEthernet0/0), routed via RIB
IP: s=155.1.146.6 (FastEthernet0/0), d=155.1.146.255 (FastEthernet0/0),
len 50, rcvd 3
    UDP src=54670, dst=53
IP(0): s=155.1.146.6 (FastEthernet0/0) d=239.1.1.100 (Serial0/0.1)
id=1, ttl=254, prot=17, len=50(50), mforward
```

Note

R3 receives the multicast packets and sends them as broadcasts to SW1.

```
Rack1R3#
IP(0): s=155.1.146.6 (Serial1/0.1) d=239.1.1.100 (FastEthernet0/0)
id=0, ttl=252, prot=17, len=50(50), mforward
IP: tableid=0, s=155.1.146.6 (Serial1/0.1), d=155.1.37.255
(FastEthernet0/0), routed via RIB
```

Note

Check mroute states in R1, R5 and R5 to ensure the traffic follows the SPT.

```
Rack1R1#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), 00:00:13/stopped, RP 150.1.8.8, flags: SPF
  Incoming interface: Serial0/0.1, RPF nbr 155.1.0.5
  Outgoing interface list: Null

(155.1.146.6, 239.1.1.100), 00:00:13/00:03:23, flags: FT
  Incoming interface: FastEthernet0/0, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/0.1, Forward/Sparse, 00:00:13/00:03:16

Rack1R5#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), 00:37:47/stopped, RP 150.1.8.8, flags: SJC
  Incoming interface: FastEthernet0/0, RPF nbr 155.1.58.8
  Outgoing interface list:
    Serial0/0, 155.1.0.3, Forward/Sparse, 00:37:47/00:02:05

(155.1.146.6, 239.1.1.100), 00:00:23/00:02:54, flags: T
  Incoming interface: Serial0/0, RPF nbr 155.1.0.1, Mroute
  Outgoing interface list:
    Serial0/0, 155.1.0.3, Forward/Sparse, 00:00:24/00:02:59
```

```
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), 00:39:01/stopped, RP 150.1.8.8, flags: SJCL
  Incoming interface: FastEthernet0/0, RPF nbr 155.1.37.7
  Outgoing interface list:
    Serial1/0.1, Forward/Dense, 00:39:01/00:00:00

(155.1.146.6, 239.1.1.100), 00:00:32/00:02:53, flags: LJT
  Incoming interface: Serial1/0.1, RPF nbr 155.1.0.5
  Outgoing interface list:
    FastEthernet0/0, Forward/Dense, 00:00:32/00:00:00
```

8.25 Multicast Rate Limiting

- Configure R3 to limit the aggregate rate of the multicast traffic leaving VLAN37 interface to 1Mbps.
- Any source sending to the group IP 239.1.1.7 should be limited to 128Kbps.
- Multicast feed from R6 to the IP address 239.1.1.100 should be limited to 256Kbps.

Configuration

Note

You may use the regular QoS policing and rate-limiting commands to control multicast traffic flow. However, there is a special command for multicast traffic rate control that has its unique features. The command is **`ip multicast rate-limit {in | out} [group-list <acl>] [source-list <acl>] [<limit>]`** where **`limit`** is specified in Kilobits per second. The **`in`** and **`out`** keywords control the ingress or egress traffic respectively. If you omit group-list and source-list, the limit will apply to aggregate multicast traffic rate, for all groups. However, if you omit the speed limit parameter, the command will discard any multicast traffic matching both the group and source-list, if those are specified. When applied without any parameters, this command will simply drop all multicast traffic.

This command behaves differently when configured with group-list (and possible source-list). When you designate the groups to be rate limited, the limit will apply to each source/group pair *individually*. That is, if you configure the command to limit the rate of traffic destined to the group 239.0.0.1 to 128Kbs and there are 3 independent sources, then the limit will apply to every source, resulting in 3x128 Kbps of aggregate rate.

You may combine the group-specific limits with the aggregate limit on the same interface. This way, you will apply per-flow rate-limits and at the same time control the aggregate rate. For example:

```
ip multicast rate-limit out group-list 100 128
ip multicast rate-limit out 512
```

Also, you may apply multiple group-specific multicast rate-limiting commands. In this case, the particular (S,G) pair is limited based on the first match in the configured access-lists. Notice that this restriction makes the order of the statements particularly important! If you put the aggregate limit in the beginning of the list, it will match everything and other entries will never be matched.

R3:
```
ip access-list standard GROUP7
 permit 239.1.1.7
!
ip access-list standard GROUP100
 permit 239.1.1.100
!
interface FastEthernet0/0
 ip multicast rate-limit out group-list GROUP7 128
 ip multicast rate-limit out group-list GROUP100 256
 ip multicast rate-limit out 1000
```

Verification

Note

To verify the limits, source some multicast to groups 239.1.1.7 and 239.1.1.100. Then check the mroute states in R3. Notice that every group limit appears next to the respective (S,G) entry:

```
Rack1R1#ping 239.1.1.7 repeat 100

Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 239.1.1.7, timeout is 2 seconds:

Reply to request 0 from 155.1.37.7, 128 ms
Reply to request 0 from 155.1.37.7, 128 ms

Rack1R6#dddd
Translating "dddd"...domain server (255.255.255.255)

Rack1R3#show ip mroute
IP Multicast Routing Table
<snip>

(*, 239.1.1.100), 01:17:18/stopped, RP 150.1.8.8, flags: SJCL
  Incoming interface: FastEthernet0/0, RPF nbr 155.1.37.7
  Outgoing interface list:
    Serial1/0.1, Forward/Dense, 01:17:18/00:00:00

(155.1.146.6, 239.1.1.100), 00:00:04/00:02:57, flags: LJT
  Incoming interface: Serial1/0.1, RPF nbr 155.1.0.5
  Outgoing interface list:
    FastEthernet0/0, Forward/Dense, 00:00:04/00:00:00, limit 256 kbps

(*, 239.1.1.7), 02:08:57/stopped, RP 150.1.8.8, flags: SJCF
  Incoming interface: FastEthernet0/0, RPF nbr 155.1.37.7
  Outgoing interface list:
    Serial1/0.1, Forward/Dense, 02:08:57/00:00:00

(150.1.1.1, 239.1.1.7), 00:02:38/00:02:59, flags: JT
  Incoming interface: Serial1/0.1, RPF nbr 155.1.0.5
  Outgoing interface list:
    FastEthernet0/0, Forward/Dense, 00:02:38/00:00:00, limit 128 kbps

(155.1.0.1, 239.1.1.7), 00:02:38/00:02:59, flags: FT
  Incoming interface: Serial1/0.1, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Dense, 00:02:38/00:00:00, limit 128 kbps
```

```
(155.1.146.1, 239.1.1.7), 00:02:38/00:00:22, flags: JT
  Incoming interface: Null, RPF nbr 155.1.13.1
  Outgoing interface list:
    FastEthernet0/0, Forward/Dense, 00:02:38/00:00:00, limit 128 kbps
    Serial1/0.1, Forward/Dense, 00:02:38/00:00:00, A

(*, 224.110.110.110), 02:14:36/00:02:31, RP 150.1.10.10, flags: SJCL
  Incoming interface: FastEthernet0/0, RPF nbr 155.1.37.7
  Outgoing interface list:
    Loopback0, Forward/Sparse-Dense, 02:14:37/00:02:30

(*, 224.0.1.40), 02:20:07/00:02:36, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Dense, 02:08:58/00:00:00
    Serial1/0.1, Forward/Dense, 02:20:07/00:00:00
```

8.26 Bidirectional PIM

- The group range 238.0.0.0/8 is used for network video-conferencing with many participants.
- Configure the network so that this group uses single shared tree rooted in R5 for multicast traffic delivery.

Configuration

Note

Bidirectional PIM or PIM BiDir is special extension to PIM SM concept that uses only the shared tree for multicast distribution. This mode of operations is useful in situations where most receivers are also senders at the same time. For example, this might be the case when you run videoconference. In such situations, in addition to joining the shared tree rooted at the RP, every receiver needs to join the shortest-path multicast distribution tree rooted at every other participant. If the number of participants is significant, the amount of multicast route states in the core of the network will grow at quadric rate.

The special feature of PIM SM shared and shortest trees is that they are unidirectional – traffic passes down from the root to the leaves of the tree. PIM BiDir uses single distribution tree rooted at the RP for all sources and receivers at the same time. If there are multiple RPs, there could be many BiDir trees. Unlike the classic tree, traffic may flow up and down the tree. When a source sends multicast packets, they first flow up to the root of the tree (the RP) and then down to all receivers.

To build the bi-directional tree, PIM elects special designated forwarded (DF) on every link in the network. DF is elected based on the rules similar to the ones used with PIM Assert procedure – i.e., the router on the link that has shortest metric to reach the RP is selected as the DF for this link. Notice that a single router might be DF on one link and non-DF on another. After the elections, DF routers are the only ones that are allowed to forward traffic upstream the bi-directional tree, toward the RP. Every router in the multicast domain creates (*,G) state for each BiDir group, with the OIL built based on PIM Join messages received from its neighbors. This is the “downstream” portion of the BiDir tree. Any packet received on valid RPF interface is forwarded based on the OIL. At the same time, DF will forward a copy of packet toward the RP - upstream the shared tree - provided that the packet is not received on the interface pointing to the RP.

Notice that PIM BiDir does not use source registration procedure, via PIM Register/Register-Stop messages. Every source connected to a PIM BiDir capable router may start sending at any time, and the packets will flow upwards to the RP. After reaching the RP, packets are either dropped, if there are no receivers for this group (i.e. the OIL for this (*,G) state is empty) or forwarded down the BiDir tree. There is no way for RP to signal the source to stop sending traffic even if there are no receivers.

Configuring PIM BiDir is relatively simple. You just need to enable BiDir PIM in all multicast routers using the command **`ip pim bidir-enable`** and designate particular RP/Group combinations as bi-directional. You can do this in a number of ways.

1) Using static RP configuration with the command **`ip pim rp-address <IP> <ACL> bidir`**.

2) Using BSR or Auto-RP for RP information dissemination you may flag particular groups/RP combination as bi-directional using the following syntax:

Auto-RP:
`ip pim send-rp-announce <interface> scope <TTL> group-list <ACL> bidir`

Auto-RP:
`ip pim rp-candidate <interface> group-list <ACL> bidir`

This is all you need to enable bi-directional PIM. However, remember to enable bi-directional mode on all routers in your network, or might end up with routing loops.

```
R1, R3, R4, SW2, SW4:
ip pim bidir-enable

R5:
ip pim bidir-enable
ip access-list standard GROUP238
 permit 238.0.0.0 0.255.255.255
!
ip pim rp-candidate Loopback 0 group-list GROUP238 bidir
```

Verification

Note

To verify, join R1 and SW4 to the bi-directional group 238.1.1.1.

```
R1:
interface FastEthernet 0/0
 ip igmp join-group 238.1.1.1

SW4:
interface Vlan 10
 ip igmp join-group 238.1.1.1
```

and then ping from R5.

```
Rack1R5#ping 238.1.1.1 repeat 100

Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 238.1.1.1, timeout is 2 seconds:

Reply to request 0 from 155.1.108.10, 4 ms
Reply to request 0 from 155.1.0.1, 188 ms
Reply to request 0 from 155.1.0.1, 104 ms
Reply to request 0 from 155.1.108.10, 8 ms
Reply to request 1 from 155.1.108.10, 4 ms
Reply to request 1 from 155.1.0.1, 189 ms
Reply to request 1 from 155.1.0.1, 104 ms
Reply to request 1 from 155.1.108.10, 8 ms
```

Note

Check mroute states in all routers. Notice that some interfaces are marked as Bidir-Upstream – those are used to send packets upwards to the root of the tree. The root of the tree (the RP) has not upstream interface.

```
Rack1R1#show ip mroute 238.1.1.1
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

(*, 238.1.1.1), 00:08:15/00:02:59, RP 150.1.5.5, flags: BCL
  Bidir-Upstream: Serial0/0.1, RPF nbr 155.1.0.5
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse, 00:08:15/00:02:40
    Serial0/0.1, Bidir-Upstream/Sparse, 00:08:15/00:00:00

Rack1R5#show ip mroute 238.1.1.1
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

(*, 238.1.1.1), 00:26:24/00:03:05, RP 150.1.5.5, flags: B
  Bidir-Upstream: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/0, 155.1.0.1, Forward/Sparse, 00:12:58/00:00:00
    FastEthernet0/0, Forward/Sparse, 00:26:24/00:02:53
```

```
Rack1SW2#show ip mroute 238.1.1.1
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
       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

(*, 238.1.1.1), 00:20:36/00:02:39, RP 150.1.5.5, flags: B
  Bidir-Upstream: Vlan58, RPF nbr 155.1.58.5
  Outgoing interface list:
    Port-channel1, Forward/Sparse, 00:20:36/00:02:39
    Vlan58, Bidir-Upstream/Sparse, 00:20:36/00:00:00

Rack1SW4#show ip mroute 238.1.1.1
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
       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

(*, 238.1.1.1), 00:20:54/00:02:54, RP 150.1.5.5, flags: BCL
  Bidir-Upstream: Port-channel1, RPF nbr 155.1.108.8
  Outgoing interface list:
    Port-channel1, Bidir-Upstream/Sparse, 00:20:54/00:00:00
    Vlan10, Forward/Sparse-Dense, 00:20:53/00:02:54, H
```

8.27 Source Specific Multicast

- Using the default multicast group range enable PIM SSM functionality in your network.
- R6 should join the multicast feed (150.1.10.10,232.6.6.6).

Configuration

Note

Classic multicast delivery technologies use IGMPv2 and PIM DM/SM and are know as Any Source Multicast or ASM. That is, receivers agree to accept traffic from any source. This is why Rendezvous Points are actually needed in PIM SM - to allow receivers discover new sources. The core of PIM SSM protocol is the use of IGMPv3 signaling by clients. This client-side protocol allows receivers specifying sources that want to listen to explicitly. That is, the host may explicitly ask to join group G at source S. PIM SSM accompanies IGMPv3 in building shortest-path trees (SPT) only, towards the sources. There are *no* shared trees in PIM SSM and no RPs used. Thus, there is no need to use auxiliary protocols like BSR or AutoRP to distribute RP information either. Notice that source discovery is outside the scope of PIM SSM and IGMPv3, and should be accompanied via some other means, like global directory services.

Configuring PIM SSM is relatively straight-forward, since it uses regular PIM messages. You just need to specify the range of groups that are using SSM signaling with the command **`ip pim ssm range {default|range <Standard-ACL>}`**. The **`default`** keyword means that range 232.0.0.0/8 will be used for SSM. For the groups in the SSM range, no shared trees are allowed and (*,G) Joins are dropped.

The second step in configuring PIM SSM is enabling IGMPv3 on the interfaces connected to the receivers capable of using this protocol. Without IGMPv3 there is no use of PIM SSM, as sources are now selected by the receivers explicitly, to allow building shortest-path trees.

R1, R3, R4, R5, SW2, SW4:
```
ip pim ssm default
```

R1:
```
interface FastEthernet 0/0
 ip igmp version 3
```

R6:
```
interface FastEthernet 0/0.146
 ip igmp version 3
 ip igmp join 232.6.6.6 source 150.1.10.10
```

Verification

Note

PIM SSM is generally easier to configure than ASM, as it does not require complicated RP infrastructure. All you need to check is the SPT toward the explicit source.

```
Rack1R1#show ip igmp groups 232.6.6.6 detail

Flags: L - Local, U - User, SG - Static Group, VG - Virtual Group,
       SS - Static Source, VS - Virtual Source,
       Ac - Group accounted towards access control limit

Interface:      FastEthernet0/0
Group:          232.6.6.6
Flags:          SSM
Uptime:         00:26:06
Group mode:     INCLUDE
Last reporter:  155.1.146.6
Group source list: (C - Cisco Src Report, U - URD, R - Remote, S -
Static,
                    V - Virtual, M - SSM Mapping, L - Local,
                    Ac - Channel accounted towards access control
limit)
  Source Address   Uptime    v3 Exp   CSR Exp   Fwd  Flags
  150.1.10.10      00:26:06  00:02:54  stopped   Yes  R

Rack1R1#show ip mroute 232.6.6.6 150.1.10.10
IP Multicast Routing Table
<snip>

(150.1.10.10, 232.6.6.6), 00:26:09/00:02:26, flags: sTI
  Incoming interface: Serial0/0.1, RPF nbr 155.1.0.5
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse, 00:26:09/00:02:26

Rack1R5#show ip mroute 232.6.6.6 150.1.10.10
IP Multicast Routing Table
<snip>

(150.1.10.10, 232.6.6.6), 00:25:39/00:02:38, flags: sT
  Incoming interface: FastEthernet0/0, RPF nbr 155.1.58.8
  Outgoing interface list:
    Serial0/0, 155.1.0.1, Forward/Sparse, 00:25:39/00:02:38

Rack1SW2#show ip mroute 232.6.6.6
IP Multicast Routing Table
<snip>

(150.1.10.10, 232.6.6.6), 00:26:11/00:03:04, flags: sT
  Incoming interface: Port-channel1, RPF nbr 155.1.108.10
  Outgoing interface list:
    Vlan58, Forward/Sparse, 00:26:11/00:03:04

Rack1SW4#show ip mroute 232.6.6.6
IP Multicast Routing Table
<snip>

(150.1.10.10, 232.6.6.6), 00:26:46/00:02:31, flags: sT
  Incoming interface: Loopback0, RPF nbr 0.0.0.0
  Outgoing interface list:
    Port-channel1, Forward/Sparse, 00:26:46/00:02:31

Rack1SW4#ping 232.6.6.6

Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 232.6.6.6, timeout is 2 seconds:

Reply to request 0 from 155.1.146.6, 48 ms
```

8.28 DVMRP Interoperability

- Configure R1 for this task. Enable multicast routing and configure PIM dense mode on VLAN146 and Loopack0 interfaces. Create a DVMRP tunnel sourced off the Loopback0 interface with (non-existent) destination 204.12.X.100 (where X is your rack number). Advertise VLAN146 subnet to DVMRP backbone with the offset value of 3. Configure the use of DVMRP routes for RPF check by PIM on VLAN146 interface.

Configuration

Note

DVMRP or Distance-Vector Multicast Routing Protocol is defined in RFC 1075. This protocol was implemented in UNIX *mrouted* daemon and was the first to gain more or less widespread adoption. DVMRP is based on RIP routing protocol, and uses IGMPv1 messages to carry its routing information. For many years, DVMRP was used as the core routing protocol of MBONE – experimental set of multicast-capable networks, used to facilitate multicast testing. However, DVMRP never was a scalable protocol, and there most enterprises use PIM as the standard based multicast routing protocol nowadays.

Similar to RIP, DVMRP propagates distance-vector information, but routing updates carry subnets describing multicast sources and metrics to reach them. The metric used by DVMRP is the same hop count used by RIP. When a router receives a DVMRP update, it extracts the subnets contained there along with their metrics and stores them in a separate multicast routing table. This table is used to perform RPF checks only, not to route packets based on their destination addresses.

DVMRP implements TRPB – Truncated Reverse Path Broadcasting, which is another name for Constrained RPF. With respect to multicast flooding DVMRP is very much similar to PIM Dense Mode - traffic is flooded using RPF checks and then routers with no subscribers send “prune” messages upstream toward the source.

Cisco IOS routers do not implement complete DVMRP stack and rely on PIM for multicast routing and signaling. However, Cisco router might be configured to border a DVMRP domain and receive DVMRP routing updates. IOS routers are capable of storing DVMRP information and using it to perform RPF checks on packets received from DVMRP cloud. At the same time, the routers will generate DVMRP updates to cover sources in PIM cloud and let DVMRP systems receive multicast feeds. IOS routers may generate DVMRP prune and graft messages in response to the respective PIM messages.

Use the command **`ip dvmrp ineroperability`** to configure the IOS router for interoperation with DVMRP systems. This will allow the router to accept/send DVMRP updates and populate the multicast route cache. In order to use this information for a particular interface, enter the **`ip dvmrp unicast-routing`** interface-level command. This will make router prefer cached DVMRP information over unicast routes for RPF checks. Like RIP, DVMRP performs automatic summarization when crossing major subnet boundaries. You may disable this behavior, using the command **`no ip dvmrp auto-summary`**.

By default, local router will only advertise directly connected subnets in DVMRP updates. If you want to advertise more information, use the interface-level command **`ip dvmrp metric <hops> [list <access-list>] [protocol <process-id>]`** to redistribute static subnets or IGP networks to all DVMRP neighbors on the interface. If you omit the protocol specification, the command will only advertise connected routes. If you want to filter out certain updates, use the metric value of zero to remove the matching subnets from DVMRP updates.

Another thing you might want to configure is DVMRP tunnel. DVMRP tunnels are supported in IOS routers to connect to remote DVMRP cloud over non-mulitcast network. Notice that you cannot configure DMVRP tunnel between two IOS routers, and your configuration is always uni-directional. DMVRP tunnels are commonly used to link your multicast network with the MBONE. Use the following syntax to configure DVMRP tunnel:

```
interface tunnel 0
 ip unnumbered Loopback0
 ip pim dense-mode
 tunnel source Loopback0
 tunnel destination <IP in MBONE>
 tunnel mode dvmrp
```

Notice that PIM is enabled on the interface, to allow multicast feeds to flow to the MBONE, even though no real PIM adjacencies are established over the tunnel.

R4:
```
ip dvmrp interoperability
!
access-list 40 permit 155.1.0.0 0.0.255.255
!
interface FastEthernet0/1
 ip dvmrp metric 1 list 40 eigrp 100
```

Verification

Note

There is no way to fully verify DVMRP interoperability unless you have a real DVMRP router. However, you may check if the router actually generates DVMRP updates using the debug commands.

```
Rack1R4#debug ip dvmrp detail
DVMRP(0): Building Report for FastEthernet0/1
DVMRP(0): Report 155.1.146.0/24, metric 32
DVMRP(0): Report 155.1.10.0/24, metric 1
DVMRP(0): Report 155.1.8.0/24, metric 1
DVMRP(0): Report 155.1.5.0/24, metric 1
DVMRP(0): Report 155.1.58.0/24, metric 1
DVMRP(0): Report 155.1.45.0/24, metric 1
DVMRP(0): Report 155.1.67.0/24, metric 32
DVMRP(0): Report 155.1.108.0/24, metric 1
DVMRP(0): Report 150.1.6.0/24, metric 32
DVMRP(0): Report 150.1.5.0/24, metric 1
DVMRP(0): Report 150.1.10.0/24, metric 1
DVMRP(0): Report 150.1.8.0/24, metric 1
DVMRP(0): Delay Report on FastEthernet0/1
DVMRP(0): 12 unicast, 0 MBGP, 0 DVMRP routes advertised
DVMRP(0): Send Report on FastEthernet0/1 to 224.0.0.4
```

8.29 Multicast BGP Extension

- Enable multicast exchange between AS 100 and AS200 on both peering links.
- Multicast traffic should be preferably routed across the Frame-Relay link, using the Ethernet link as backup.

Configuration

Note

Multicast BGP extension is commonly needed when you plan to exchange multicast traffic between two different administrative domains, i.e. different autonomous systems. To achieve this goal, you need to fulfill the following tasks

1) Enable PIM between the two domains, to allow signaling of shared and shortest-path trees between them.

1.1) PIM SM is most often used for multicast traffic exchange between different domains. Each domain usually has its own set of RPs, and thus you should prevent BSR/Auto-RP information from leaking between the domains.

1.2) Exchange information on active sources between RPs in every domain. Since the RPs are separated, one domain cannot easily learn about the sources in another domain. As we'll see later, special protocol called MSDP is used for this purpose.

2) In order to facilitate multicast traffic forwarding, you need to exchange information on routes towards the multicast sources in each domain, to allow routers performing RPF checks correctly. PIM uses the unicast routing table to perform RPF checks, and thus it may use routes learned via either IGP or BGP. It is common to use BGP to exchange routing information.

In some cases, you may want to apply different policies to unicast routes exchanged via BGP and to the information about multicast sources. This is possible thanks to Multi-Protocol BGP extension. Using a special address family, you may exchange prefixes under the "multicast" address-family, and apply different policy to this information. These prefixes are interpreted in the same was the **`mroute`** command information – they are used for RPF checks on the router that receives them. That is, if a prefix is learned via multicast BGP extension, it is assumed to have RPF neighbor towards the next-hop IP address found in the update. If needed, BGP performs recursive routing lookup for the next hop via the IGP routing table to find the immediate RPF neighbor. Unlike the **`mroute`** command, which is purely local, the information is propagated via BGP to every neighbor configured for multicast address family.

Using separate policies for multicast inter-domain RPF information allows you for example the use of different inter-domain links for unicast and multicast traffic. Or you may selectively filter out certain multicast sources from another domain, while leaving unicast routes intact.

Our tasks requires us enabling BGP multicast extension in all BGP routers. Notice the use of peer-group under the multicast address family. This is needed to propagate multicast RPF information through both domains, as route-reflection is configured separately per address family. Notice the use of AS-PATH prepending to designate the primary path. Multicast prefixes are subject to the same best-path selection procedure, and thus you may use the same methods of path manipulation you used with unicast prefixes. Finally, PIM is activated on the links connecting two autonomous systems. PIM BSR border is used to stop BSR information from leaking to AS 100.

R3:
```
router bgp 100
 address-family ipv4 multicast
 neighbor 155.1.0.5 activate
 redistribute ospf 1
 neighbor 150.1.7.7 activate
 neighbor 150.1.7.7 next-hop-self
!
interface Serial 1/0.1
 ip pim sparse-mode
```

R4:
```
route-map PREPEND
 set as-path prepend 200 200 200
!
router bgp 200
 address-family ipv4 multicast
 neighbor 155.1.146.6 activate
 redistribute eigrp 100
 neighbor 155.1.146.6 route-map PREPEND out
 neighbor 150.1.5.5 activate
!
interface FastEthernet 0/1
 ip pim sparse-mode
 ip pim bsr-border
```

R5:
```
router bgp 200
 address-family ipv4 multicast
 neighbor 155.1.0.3 activate
 redistribute eigrp 100
 neighbor IBGP route-reflector-client
 neighbor 150.1.4.4 peer-group IBGP
 neighbor 150.1.8.8 peer-group IBGP
 neighbor 150.1.10.10 peer-group IBGP
 neighbor IBGP next-hop-self
!
interface Serial 0/0
 ip pim sparse-mode
 ip pim bsr-border
```

R6:
```
route-map PREPEND
 set as-path prepend 100 100 100
!
router bgp 100
 address-family ipv4 multicast
 neighbor 155.1.146.4 activate
 redistribute ospf 1
 neighbor 155.1.146.4 route-map PREPEND out
 neighbor 150.1.7.7 activate
 neighbor 150.1.7.7 next-hop-self
!
interface FastEthernet 0/0.146
 ip pim sparse-mode
```

SW1:
```
router bgp 100
 address-family ipv4 multicast
 neighbor IBGP route-reflector-client
 neighbor 150.1.3.3 peer-group IBGP
 neighbor 150.1.6.6 peer-group IBGP
 neighbor 150.1.9.9 peer-group IBGP
```

SW2:
```
router bgp 200
 address-family ipv4 multicast
 neighbor 150.1.5.5 activate
```

SW3:
```
router bgp 100
 address-family ipv4 multicast
 neighbor 150.1.7.7 activate
```

SW4:
```
router bgp 200
 address-family ipv4 multicast
 neighbor 150.1.5.5 activate
```

Verification

Note

Use the regular show BGP commands to check that multicast address family is activated between the routers. Repeat it on every BGP router to make sure you didn't miss anything.

```
Rack1R6#show ip bgp ipv4 multicast summary
BGP router identifier 150.1.6.6, local AS number 100
BGP table version is 104, main routing table version 104
19 network entries using 2223 bytes of memory
19 path entries using 912 bytes of memory
24/11 BGP path/bestpath attribute entries using 2976 bytes of memory
1 BGP rrinfo entries using 24 bytes of memory
2 BGP AS-PATH entries using 48 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 6183 total bytes of memory
BGP activity 103/64 prefixes, 351/295 paths, scan interval 60 secs

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
150.1.7.7       4   100     362     454      104    0    0 00:09:36
0
155.1.146.4     4   200     294     258      104    0    0 01:10:05
10
```

Note

Next, check BGP tables on the border routers to make sure that best paths toward the multicast prefixes are across the Frame-Relay cloud:

```
Rack1R4#show ip bgp ipv4 multicast regexp 100$
BGP table version is 91, 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
*  150.1.3.0/24     155.1.146.6              3             0 100 100 100 100 ?
*>i                 150.1.5.5                0    100      0 100 ?
*> 150.1.6.0/24     155.1.146.6              0             0 100 100 100 100 ?
*>i150.1.6.6/32     150.1.5.5                3    100      0 100 ?
*>i150.1.7.0/24     150.1.5.5                2    100      0 100 ?
*                   155.1.146.6              2             0 100 100 100 100 ?
*>i150.1.9.9/32     150.1.5.5                3    100      0 100 ?
*                   155.1.146.6              3             0 100 100 100 100 ?
*>i155.1.7.0/24     150.1.5.5                2    100      0 100 ?
*                   155.1.146.6              2             0 100 100 100 100 ?
*>i155.1.9.0/24     150.1.5.5                3    100      0 100 ?
*                   155.1.146.6              3             0 100 100 100 100 ?
```

```
*>i155.1.37.0/24    150.1.5.5                0    100      0 100 ?
*                   155.1.146.6              2             0 100 100 100 100 ?
*>i155.1.67.0/24    150.1.5.5                2    100      0 100 ?
*                   155.1.146.6              0             0 100 100 100 100 ?
*>i155.1.79.0/24    150.1.5.5                2    100      0 100 ?
   Network          Next Hop            Metric LocPrf Weight Path
*                   155.1.146.6              2             0 100 100 100 100 ?
```

Check the best paths in the other AS as well.

```
Rack1R6#show ip bgp ipv4 multicast regexp 200$
BGP table version is 115, 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
*>i150.1.4.0/24     150.1.3.3          2297856    100      0 200 ?
*                   155.1.146.4              0             0 200 200 200 200 ?
*>i150.1.5.0/24     150.1.3.3                0    100      0 200 ?
*                   155.1.146.4        2297856             0 200 200 200 200 ?
*>i150.1.8.0/24     150.1.3.3           156160    100      0 200 ?
*                   155.1.146.4        2300416             0 200 200 200 200 ?
*>i150.1.10.0/24    150.1.3.3           158720    100      0 200 ?
*                   155.1.146.4        2302976             0 200 200 200 200 ?
*>i155.1.0.0/24     150.1.3.3                0    100      0 200 ?
*                   155.1.146.4              0             0 200 200 200 200 ?
*>i155.1.8.0/24     150.1.3.3            28416    100      0 200 ?
*                   155.1.146.4        2172672             0 200 200 200 200 ?
*>i155.1.10.0/24    150.1.3.3            30976    100      0 200 ?
*                   155.1.146.4        2175232             0 200 200 200 200 ?
*>i155.1.45.0/24    150.1.3.3                0    100      0 200 ?
*                   155.1.146.4              0             0 200 200 200 200 ?
*>i155.1.58.0/24    150.1.3.3                0    100      0 200 ?
   Network          Next Hop            Metric LocPrf Weight Path
*                   155.1.146.4        2172416             0 200 200 200 200 ?
*>i155.1.108.0/24   150.1.3.3            30720    100      0 200 ?
*                   155.1.146.4        2174976             0 200 200 200 200 ?
```

8.30 MSDP

- Configure the devices per the "Multicast BGP" scenario. Change PIM dense-mode on all links where it's configured to PIM sparse-mode.
- Configure R5 as the RP for AS 200 and SW1 as the RP for AS 100. Use the BSR method to distribute RP information, and configure BSR border on the link between R3 and SW1. Create an MSDP peering session between SW1 and R5 sourcing it off the Loopback0 interfaces.

Configuration

Note

When implementing inter-domain multicast using PIM SM, each domain usually has its own RPs. In order to allow sources and receivers from different domains to locate each other, RPs need to exchange the information on their local active sources. After this information is exchanged between the RPs, all routers that joined the respective shared trees may build shortest-path trees towards the actual sources.

MSDP or Multicast Source Discovery Protocol is used to exchange multicast sources information between the RPs. It is configured as a TCP connection between the two RPs, and used to exchange the so-called Source Active (SA) messages. Note that all MSDP peerings are configured manually, using the command **`ip msdp peer`** at both endpoints. When a source in one PIM SM domain starts sending the multicast traffic, the respective DR will start registration process with the local RP. When the local RP receives the PIM Register message, it replicates it to all of its MSDP neighbors as SA message. The SA message contains the IP address of the multicast source as well as the destination group and the IP address of the RP sending the SA message. The latter is know as MSDP ID, could be changed using the command **`ip msdp originator-id`**.

When any RP receives new SA message, it checks if there are local receivers that joined the shared tree for the encapsulated group. If there are any, the message is forwarded down the tree, allowing the receivers to learn about the sources in another domain. After that, the receivers might join the SPT towards the source in the other domain. This is only possible if the source IP address is learned via BGP or some other inter-domain route exchange procedure. Till the moment there is active source in a domain, the respective RP will forward periodic SA messages with empty payload to refresh the active state for this group/source in all other domains.

You may connect RP using MSDP in arbitrary meshed topology, even looped. In order to prevent the SA messages from cycling the topology, every MSDP peer forwards SA messages only after they pass the RPF check. The RPF check is performed based on the RP IP address (originator-ID) inside the message and the IP address of the MSDP peer that relayed the message. If the MSDP peer is on the shortest path towards the originating RP, the message is accepted, otherwise it is dropped. This RPF check needs full routing information from other domains to know routes to other RPs. If you have a stub multicast domain, lacking full BGP information, you may use the command **`ip msdp default-peer`** to identify the upstream RP that forwards SA messages. RPF checks are not applied to default peers and all SA messages are accepted.

Our scenario is a bit tricky, as it has two RPs in AS 200. However, the BSR protocol ensures that all routers in AS 200 will select the same RP for a given group. Thus you only need to peer SW1 with SW2 and R5 via MSDP, but there is no need to peer SW2 with R5 via MSDP.

R5:
```
ip msdp peer 150.1.7.7 connect-source Loopback 0
ip msdp peer 150.1.7.7 remote-as 100
```

SW2:
```
ip msdp peer 150.1.7.7 connect-source Loopback 0
ip msdp peer 150.1.7.7 remote-as 100
```

SW1:
```
ip msdp peer 150.1.5.5 remote-as 200
ip msdp peer 150.1.5.5 connect-source Loopback 0
ip msdp peer 150.1.8.8 remote-as 200
ip msdp peer 150.1.8.8 connect-source Loopback 0
```

Verification

Note

Let’s select two multicast groups that maps to different RPs in AS 200. For example 239.1.1.1 and 239.1.1.2:

```
Rack1R4#show ip pim rp-hash 239.1.1.1
  RP 150.1.5.5 (?), v2
    Info source: 150.1.10.10 (?), via bootstrap, priority 0, holdtime
90
         Uptime: 00:08:12, expires: 00:01:17
  PIMv2 Hash Value (mask 255.255.255.254)
    RP 150.1.5.5, via bootstrap, priority 0, hash value 1362971077
    RP 150.1.8.8, via bootstrap, priority 0, hash value 718054422

Rack1R4#show ip pim rp-hash 239.1.1.2
  RP 150.1.8.8 (?), v2
    Info source: 150.1.10.10 (?), via bootstrap, priority 0, holdtime
202
         Uptime: 00:08:14, expires: 00:03:03
  PIMv2 Hash Value (mask 255.255.255.254)
    RP 150.1.5.5, via bootstrap, priority 0, hash value 443334807
    RP 150.1.8.8, via bootstrap, priority 0, hash value 1364246456
```

Note

Now configure speakers in both systems to join these groups.

```
R4:
interface Loopback0
 ip pim sparse-mode
 ip igmp join-group 239.1.1.1
 ip igmp join-group 239.1.1.2

SW3:
interface Loopback0
 ip pim sparse-mode
 ip igmp join-group 239.1.1.1
 ip igmp join-group 239.1.1.2
```

Note

Initially, every router joins the shared tree in its own domain.

```
Rack1SW3#show ip mroute 239.1.1.1
IP Multicast Routing Table
<snip>

(*, 239.1.1.1), 00:19:49/00:02:22, RP 150.1.7.7, flags: SJCL
  Incoming interface: Vlan79, RPF nbr 155.1.79.7
  Outgoing interface list:
    Loopback0, Forward/Sparse, 00:17:45/00:02:22

Rack1SW3#show ip mroute 239.1.1.2
IP Multicast Routing Table
<snip>

(*, 239.1.1.2), 00:02:07/00:02:24, RP 150.1.7.7, flags: SJCL
  Incoming interface: Vlan79, RPF nbr 155.1.79.7
  Outgoing interface list:
    Loopback0, Forward/Sparse, 00:02:07/00:02:24

Rack1R4#show ip mroute 239.1.1.1
IP Multicast Routing Table
<snip>

(*, 239.1.1.1), 00:12:37/00:02:38, RP 150.1.5.5, flags: SJCL
  Incoming interface: Serial0/1, RPF nbr 155.1.45.5
  Outgoing interface list:
    Loopback0, Forward/Sparse, 00:12:37/00:02:38

Rack1R4#show ip mroute 239.1.1.2
IP Multicast Routing Table
<snip>

(*, 239.1.1.2), 00:03:25/00:02:28, RP 150.1.8.8, flags: SJCL
  Incoming interface: Serial0/1, RPF nbr 155.1.45.5
  Outgoing interface list:
    Loopback0, Forward/Sparse, 00:03:25/00:02:28
```

Note

Enable MSDP debugging in SW1 and start pinging group 239.1.1.1 from **SW1:**

```
Rack1SW1#debug ip msdp de
Rack1SW1#debug ip msdp detail
MSDP Detail debugging is on

Rack1SW4#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 155.1.45.4, 36 ms
Reply to request 0 from 155.1.79.9, 208 ms
Reply to request 0 from 155.1.79.9, 144 ms
Reply to request 0 from 155.1.79.9, 112 ms
Reply to request 0 from 155.1.45.4, 88 ms
Reply to request 0 from 155.1.45.4, 60 ms
Reply to request 1 from 155.1.45.4, 36 ms
Reply to request 1 from 155.1.79.9, 172 ms
Reply to request 1 from 155.1.79.9, 104 ms
Reply to request 1 from 155.1.79.9, 80 ms
Reply to request 1 from 155.1.45.4, 64 ms
Reply to request 1 from 155.1.45.4, 52 ms
Reply to request 2 from 155.1.45.4, 36 ms
Reply to request 2 from 155.1.79.9, 104 ms
```

Note

Notice that SW1 received Source Active messages for the sources in SW4. Since SW4 uses all of its PIM-enabled interfaces to source multicast, there are multiple SA messages for every registered source. The actual source is registered with the RP in AS 200.

```
Rack1SW1#
MSDP(0): Received 120-byte TCP segment from 150.1.5.5
MSDP(0): Append 120 bytes to 0-byte msg 63 from 150.1.5.5, qs 1
MSDP(0): WAVL Insert SA Source 155.1.10.10 Group 239.1.1.1 RP 150.1.5.5
Successful
MSDP(0): Forward decapsulated SA data for (155.1.10.10, 239.1.1.1) on
Vlan79
MSDP(0): Received 120-byte TCP segment from 150.1.5.5
MSDP(0): Append 120 bytes to 0-byte msg 64 from 150.1.5.5, qs 1
MSDP(0): WAVL Insert SA Source 155.1.108.10 Group 239.1.1.1 RP
150.1.5.5 Successful
MSDP(0): Forward decapsulated SA data for (155.1.108.10, 239.1.1.1) on
Vlan79
MSDP(0): Received 120-byte TCP segment from 150.1.5.5
MSDP(0): Append 120 bytes to 0-byte msg 65 from 150.1.5.5, qs 1
MSDP(0): WAVL Insert SA Source 150.1.10.10 Group 239.1.1.1 RP 150.1.5.5
Successful
MSDP(0): Forward decapsulated SA data for (150.1.10.10, 239.1.1.1) on
Vlan79
MSDP(0): Received 3-byte TCP segment from 150.1.5.5
```

Note

Now check that SW3 has joined the SPTs towards the sources in different AS. Notice that RPF information for these sources is taken from MBGP updates, not the unicast routing table.

```
Rack1SW3#show ip mroute 239.1.1.1
IP Multicast Routing Table
<snip>

(*, 239.1.1.1), 00:25:31/stopped, RP 150.1.7.7, flags: SJCL
  Incoming interface: Vlan79, RPF nbr 155.1.79.7
  Outgoing interface list:
    Loopback0, Forward/Sparse, 00:23:27/00:02:41

(150.1.10.10, 239.1.1.1), 00:03:09/00:02:58, flags: LJT
  Incoming interface: Vlan79, RPF nbr 155.1.79.7, Mbgp
  Outgoing interface list:
    Loopback0, Forward/Sparse, 00:03:09/00:02:41

(155.1.10.10, 239.1.1.1), 00:03:09/00:02:56, flags: LJT
  Incoming interface: Vlan79, RPF nbr 155.1.79.7, Mbgp
  Outgoing interface list:
    Loopback0, Forward/Sparse, 00:03:09/00:02:40

(155.1.108.10, 239.1.1.1), 00:03:09/00:02:57, flags: LJT
  Incoming interface: Vlan79, RPF nbr 155.1.79.7, Mbgp
  Outgoing interface list:
    Loopback0, Forward/Sparse, 00:03:09/00:02:40
```

Note

Now make sure the SPTs are built across the Frame-Relay link, as this is the preferred path for multicast traffic. Use the **`show ip mroute`** command to accomplish this.

```
Rack1SW1#show ip mroute 239.1.1.1
IP Multicast Routing Table
<snip>

(*, 239.1.1.1), 00:27:33/00:02:39, RP 150.1.7.7, flags: S
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Vlan79, Forward/Sparse, 00:25:29/00:02:39

(150.1.10.10, 239.1.1.1), 00:05:11/00:03:21, flags: MT
  Incoming interface: FastEthernet0/3, RPF nbr 155.1.37.3, Mbgp
  Outgoing interface list:
    Vlan79, Forward/Sparse, 00:05:11/00:02:39

(155.1.10.10, 239.1.1.1), 00:05:11/00:02:48, flags: MT
  Incoming interface: FastEthernet0/3, RPF nbr 155.1.37.3, Mbgp
  Outgoing interface list:
    Vlan79, Forward/Sparse, 00:05:11/00:02:38

(155.1.108.10, 239.1.1.1), 00:05:11/00:03:29, flags: MT
  Incoming interface: FastEthernet0/3, RPF nbr 155.1.37.3, Mbgp
  Outgoing interface list:
    Vlan79, Forward/Sparse, 00:05:11/00:02:38

Rack1R3#show ip mroute 239.1.1.1
IP Multicast Routing Table
<snip>

(*, 239.1.1.1), 00:26:55/stopped, RP 150.1.7.7, flags: SP
  Incoming interface: FastEthernet0/0, RPF nbr 155.1.37.7
  Outgoing interface list: Null

(150.1.10.10, 239.1.1.1), 00:05:21/00:03:26, flags: T
  Incoming interface: Serial1/0.1, RPF nbr 155.1.0.5, Mbgp
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse, 00:05:21/00:02:33

(155.1.10.10, 239.1.1.1), 00:05:22/00:01:15, flags: T
  Incoming interface: Serial1/0.1, RPF nbr 155.1.0.5, Mbgp
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse, 00:05:22/00:02:32

(155.1.108.10, 239.1.1.1), 00:05:22/00:03:24, flags: T
  Incoming interface: Serial1/0.1, RPF nbr 155.1.0.5, Mbgp
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse, 00:05:22/00:02:32
```

Note

You may also use the **`mtrace`** command, to trace the multicast delivery tree from the leaf to the root. The first parameter is the source address and the second parameter is the destination group. This command queries the neighbors for the upstream multicast path and tells you the method used for RPF check at every router. Notice that inside AS100 the RPF checks are performed using MBGP.

```
Rack1SW1#mtrace 150.1.10.10 239.1.1.1
Type escape sequence to abort.
Mtrace from 150.1.10.10 to 155.1.37.7 via group 239.1.1.1
From source (?) to destination (?)
Querying full reverse path...
 0  155.1.37.7
-1  155.1.37.7 PIM/MBGP Reached RP/Core [150.1.10.0/24]
-2  155.1.37.3 PIM/MBGP  [150.1.10.0/24]
-3  155.1.0.5 [AS 200] PIM Reached RP/Core [150.1.10.0/24]
-4  155.1.58.8 [AS 200] PIM  [150.1.10.0/24]
-5  155.1.108.10 [AS 200] PIM  [150.1.10.0/24]
```

Note

You may now repeat the tests for the group 239.1.1.2 and see that it works fine as well.

```
Rack1SW4#ping 239.1.1.2 repeat 1000

Type escape sequence to abort.
Sending 1000, 100-byte ICMP Echos to 239.1.1.2, timeout is 2 seconds:

Reply to request 0 from 155.1.45.4, 36 ms
Reply to request 0 from 155.1.79.9, 140 ms
Reply to request 0 from 155.1.79.9, 104 ms
Reply to request 0 from 155.1.45.4, 84 ms
Reply to request 0 from 155.1.45.4, 56 ms
Reply to request 1 from 155.1.45.4, 36 ms
Reply to request 1 from 155.1.79.9, 244 ms
Reply to request 1 from 155.1.79.9, 176 ms
Reply to request 1 from 155.1.79.9, 120 ms
Reply to request 1 from 155.1.45.4, 88 ms
Reply to request 1 from 155.1.79.9, 80 ms
Reply to request 1 from 155.1.45.4, 76 ms
Reply to request 1 from 155.1.45.4, 60 ms
```

```
Rack1SW3#mtrace 150.1.10.10 239.1.1.2
Type escape sequence to abort.
Mtrace from 150.1.10.10 to 155.1.79.9 via group 239.1.1.2
From source (?) to destination (?)
Querying full reverse path...
 0  155.1.79.9
-1  155.1.79.9 PIM/MBGP  [150.1.10.0/24]
-2  155.1.79.7 PIM/MBGP Reached RP/Core [150.1.10.0/24]
-3  155.1.37.3 PIM/MBGP  [150.1.10.0/24]
-4  155.1.0.5 [AS 200] PIM  [150.1.10.0/24]
-5  155.1.58.8 [AS 200] PIM Reached RP/Core [150.1.10.0/24]
-6  155.1.108.10 [AS 200] PIM  [150.1.10.0/24]
```

8.31 Anycast RP

- Instead of relying on BSR protocol to distribute load between the RPs in AS 200, implement a solution that hides both RPs behind the same RP IP address 150.1.100.100.
- Every RP should inform the other one of the active sources registered with them.

Configuration

Note

Anycast RP is a special RP redundancy scenario, which allows using redundant RPs sharing the *same* IP address. Here Anycast means that group of RPs use the same IP address used by all multicast routers in the domain to build shared trees. However, the PIM Joins are being sent to the closest RP, based on the unicast routing table. Thus, different routers might join shared trees rooted at different RPs. At the same time, different DRs will pick up different physical RPs based on the anycast address to register their local sources.

In order to maintain consistent sources information, MSDP sessions should be configured between the RPs. This will ensure that all routers joining different RPs will still have full information about all potential sources in the domain. Thus, the following are the guidelines to configure AnycastRP

1) Use the same IP address on all routers as the candidate RP IP address. Propagate this information via BSR or Autor-RP.
2) Using different IP addresses on every router, source MSDP sessions and link all candidate RPs in a mesh. Note that you might need to manually specify MSDP originator ID to be different on every RP, or the MSDP sessions won’t come up.

Anycast RP is purely intra-domain solution, and does not deal with inter-domain multicast. Thus, it is a good example of using the inter-domain technology inside a single multicast region to achieve RP redundancy above the scheme used by PIM BSR.

In our scenario, we mix intra-domain MSDP with inter-domain connections. That is, a domain with AnycastRP peers another domain with regular RP. This results in looped MSDP topology, which would successfully work due to MSDP RPF checks.

R5:
```
interface Loopback100
 ip address 150.1.100.100 255.255.255.255
!
router eigrp 100
 network 150.1.100.100 0.0.0.0
!
ip msdp originator-id Loopback 0
ip msdp peer 150.1.8.8 connect-source Loopback 0
ip pim rp-candidate Loopback100
```

SW2:
```
interface Loopback100
 ip address 150.1.100.100 255.255.255.255
!
router eigrp 100
 network 150.1.100.100 0.0.0.0
!
ip msdp originator-id Loopback 0
ip msdp peer 150.1.5.5 connect-source Loopback 0
ip pim rp-candidate Loopback100
```

Verification

Note

First, join receivers in R4 and SW3 to the multicast group 239.1.1.1. Check that R4 actually uses the Anycast RP IP address as its RP.

```
R4:
interface Loopback0
 ip pim sparse-mode
 ip igmp join-group 239.1.1.1

SW3:
interface Loopback0
 ip pim sparse-mode
 ip igmp join-group 239.1.1.1

Rack1R4#show ip mroute 239.1.1.1
IP Multicast Routing Table
<snip>

(*, 239.1.1.1), 00:39:32/00:02:38, RP 150.1.100.100, flags: SJCL
  Incoming interface: Serial0/1, RPF nbr 155.1.45.5
  Outgoing interface list:
    Loopback0, Forward/Sparse, 00:39:32/00:02:38
```

Note

Next source multicast from SW4 and confirm that it actually reaches the receivers.

```
Rack1SW4#ping 239.1.1.2 repeat 1000

Type escape sequence to abort.
Sending 1000, 100-byte ICMP Echos to 239.1.1.2, timeout is 2 seconds:

Reply to request 0 from 155.1.45.4, 36 ms
Reply to request 0 from 155.1.79.9, 140 ms
Reply to request 0 from 155.1.79.9, 104 ms
Reply to request 0 from 155.1.45.4, 84 ms
Reply to request 0 from 155.1.45.4, 56 ms
Reply to request 1 from 155.1.45.4, 36 ms
```

Note

Look at the SA caches of R5 and SW1. Both of them should have been updated by SW2, as SW2 is the closest RP to the source (SW4) and the source registers with it.

```
Rack1R5#show ip msdp sa-cache
MSDP Source-Active Cache - 3 entries
(150.1.10.10, 239.1.1.1), RP 150.1.8.8, MBGP/AS 200, 00:00:26/00:05:34, Peer
150.1.8.8
(155.1.10.10, 239.1.1.1), RP 150.1.8.8, MBGP/AS 200, 00:00:26/00:05:34, Peer
150.1.8.8
(155.1.108.10, 239.1.1.1), RP 150.1.8.8, MBGP/AS 200, 00:00:25/00:05:34, Peer
150.1.8.8

Rack1SW1#show ip msdp sa-cache
MSDP Source-Active Cache - 3 entries
(150.1.10.10, 239.1.1.1), RP 150.1.8.8, MBGP/AS 200, 00:00:53/00:05:33, Peer
150.1.8.8
(155.1.10.10, 239.1.1.1), RP 150.1.8.8, MBGP/AS 200, 00:00:53/00:05:33, Peer
150.1.8.8
(155.1.108.10, 239.1.1.1), RP 150.1.8.8, MBGP/AS 200, 00:00:53/00:05:33, Peer
150.1.8.8
```

Note

Now source traffic from AS 100 and make sure both receivers are able to hear it. Check the SA caches of SW2 and R5 after that, to confirm that SW1 has actually updated them.

```
Rack1R6#ping 239.1.1.1 repeat 100

Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 239.1.1.1, timeout is 2 seconds:

Reply to request 0 from 155.1.79.9, 8 ms
Reply to request 0 from 155.1.45.4, 69 ms
Reply to request 0 from 155.1.146.4, 8 ms
Reply to request 1 from 155.1.79.9, 4 ms
Reply to request 1 from 155.1.45.4, 48 ms
Reply to request 1 from 155.1.146.4, 4 ms
Reply to request 2 from 155.1.79.9, 4 ms
Reply to request 2 from 155.1.45.4, 44 ms
Reply to request 2 from 155.1.146.4, 4 ms
Reply to request 3 from 155.1.79.9, 4 ms
Reply to request 3 from 155.1.45.4, 44 ms
Reply to request 3 from 155.1.146.4, 4 ms
```

```
Rack1R5#show ip msdp sa-cache
MSDP Source-Active Cache - 4 entries
(150.1.10.10, 239.1.1.1), RP 150.1.8.8, MBGP/AS 200, 00:09:41/00:05:51, Peer
150.1.8.8
(155.1.10.10, 239.1.1.1), RP 150.1.8.8, MBGP/AS 200, 00:09:41/00:05:51, Peer
150.1.8.8
(155.1.67.6, 239.1.1.1), RP 150.1.7.7, MBGP/AS 100, 00:00:11/00:05:48,
Peer 150.1.7.7
(155.1.108.10, 239.1.1.1), RP 150.1.8.8, MBGP/AS 200, 00:09:40/00:05:51, Peer
150.1.8.8

Rack1SW2#show ip msdp sa-cache
MSDP Source-Active Cache - 1 entries
(155.1.67.6, 239.1.1.1), RP 150.1.7.7, MBGP/AS 100, 00:00:16/00:05:43, Peer
150.1.7.7
```

8.32 Catalyst IGMP Snooping

- Configure the devices per the "Basic IP Addressing" scenario. Configure multicast routing on R1 and enable PIM dense mode on VLAN146 interface of R1.
- Configure R4 and R6 to join a multicast group on their VLAN146 interfaces. Enable IGMP snooping for VLAN146 on the switches interconnecting R1, R4 and R6.
- Start sending the multicast packets from R1 to the multicast group joined by R4 and R6. Remove IGMP join from R6 and see how this changes the switches mac-address tables.

Configuration

Note

Multicast poses special difficulties in switched Ethernet networks. By default, IP Multicast addresses are mapped to Ethernet multicast MAC addresses per the following procedure:

1) Multicast Ethernet MAC addresses range start at 01:00:5E:00:00:00 through 01:00:5E:7F:FF:FF. The highest bit of the 4th byte in these addresses is fixed at 1, and thus only low-order 23 bits could vary. This allocation is historical.

2) Based on this range, the low-order 23 bits of the multicast IP address are mapped into the low-order 23 bits of the MAC address.

3) The high order 4 bits of the Layer 3 IP address is fixed to 1110 to indicate the Class D address space between 224.0.0.0 through 239.255.255.255. Thus, 28 bits remain for IP Multicast group numbering.

Therefore, there are 2^5 multicast groups mapped to the same multicast MAC address. However, this is only a part of the puzzle. As you remember, Ethernet switches flood multicast traffic out of all ports by default. In heavily loaded networks, this might result in excessive bandwidth usage. In order to overcome this issue, switches might listen to IGMP and PIM messages received on Layer2 ports. Based on the information snooped from these messages, switches may selectively prune some ports from unneeded multicast traffic. This procedure is called IGMP snooping and allows for selective multicast flooding in switched networks.

Notice that in order to work effectively, IGMP snooping must be implemented in Layer3 capable switches. This is because Laye2 devices cannot distinguish IGMP and PIM packets from any other multicast frames, and must process all multicast traffic at CPU level. Thus, layer 2 switches performance might be severely impacted by intense multicast flows.

IGMP snooping is enabled by default in the Catalyst multilayer switches. Snooping is enabled globally and on per-VLAN basis. If you want to disable IGMP snooping globally on all VLANs, use the command **`no ip igmp snooping`**. Use the command **`no ip igmp snooping vlan <VLAN-ID>`** to disable it for a single VLAN. Generally, switches automatically discover switchports connected to multicast-capable routers by listening to PIM and DVMRP messages, and flood all multicast groups on such ports. If you want to statically configure a port as connected to a multicast router, use the command **`ip igmp snooping vlan <vlan-id> mrouter interface <interface-id>`**.

If your switch has just one host connected to every layer2 switch-port, you may want to enable the IGMP Snooping Immediate Leave feature using the command **`ip igmp snooping vlan <vlan-id> immediate-leave`**. When you enable this feature, the switch will immediately remove a port from the flood list for a given group once it receives the IGMPv2 Leave message on this port to the particular group.

SW1:
```
ip igmp snooping vlan 146 immediate-leave
```

Verification

Note

First, check IGMP Snooping general settings for VLAN146. Notice that IGMPv2 immediate leave is enabled for this VLAN.

```
Rack1SW1#show ip igmp snooping vlan 146
Global IGMP Snooping configuration:
-----------------------------------
IGMP snooping              : Enabled
IGMPv3 snooping (minimal)  : Enabled
Report suppression         : Enabled
TCN solicit query          : Disabled
TCN flood query count      : 2
Last Member Query Interval : 1000

Vlan 146:
--------
IGMP snooping                         : Enabled
IGMPv2 immediate leave                : Enabled
Explicit host tracking                : Enabled
Multicast router learning mode        : pim-dvmrp
Last Member Query Interval            : 1000
CGMP interoperability mode            : IGMP_ONLY
```

Note

Now join R1's VLAN146 interface to the IGMP group 239.1.1.100 and check that IGMP snooping actually processes the IGMP packets. From the commands input, you can see that there are two ports in the flood list for the group – one for the receiver, and the other for the multicast router (R4).

```
R1:
interface FastEthernet 0/0
 ip igmp join-group 239.1.1.100

Rack1SW1#show ip igmp snooping groups vlan 146
Vlan      Group            Type        Version     Port List
-----------------------------------------------------------------
146       239.1.1.100      igmp        v2          Fa0/1, Fa0/19

Rack1SW1#show ip igmp snooping mrouter vlan 146
Vlan    ports
----    -----
 146    Fa0/19(dynamic)

Rack1SW4#show ip igmp snooping groups vlan 146
Vlan      Group            Version     Port List
-------------------------------------------------------------
```

```
146       239.1.1.100       v2          Fa0/13

Rack1SW4#show ip igmp snooping mrouter vlan 146
Vlan    ports
----    -----
 146    Fa0/4(dynamic), Fa0/18(dynamic)
```

Note

Multicast router information is learned via PIM messages. There are two multicast routers on VLAN 146 – R4 and R6. Confirm that R4 actually receives the IGMP report from R1.

```
Rack1R4#show ip igmp groups
IGMP Connected Group Membership
Group Address    Interface                Uptime    Expires   Last
Reporter   Group Accounted
239.1.1.100      FastEthernet0/1          00:15:59  00:02:20
155.1.146.1
<snip>
```

8.33 Catalyst Multicast VLAN Registration

- Configure SW1 so that multicast traffic sent by R1 on VLAN146 is received by R5 on VLAN58.
- Use the configuration that allows R1 dynamically tracking sources in other VLANs.

Configuration

Note

Multicast VLAN registration (MVR) is a special type of multicast traffic delivery suited to access layer of metro Ethernet networks, especially for ring topologies. In these networks, it is common to allocate a VLAN based on network geographic region. In this configuration, when you send a video multicast feed to receivers into multiple VLANs, the feed will be replicated for every VLAN across the ring topology, causing bandwidth over-utilization.

MVR uses single dedicated VLAN across the whole ring to deliver multicast feed to all receivers. The actual receivers reside in different VLANs, but the switches intercept their IGMP Join requests and pull the multicast feed from the MVR VLAN to the receiver VLAN. Thus, MVR allows multicast traffic to cross VLAN boundaries based on client IGMP reports. This function is similar to IGMP snooping, but they work independently. You may have both features enable in the switch, but MVR will only inspect IGMP reports for groups explicitly configured to be supported by MVR.

There are two modes for MVR feature: dynamic and compatible. Before we look into the difference, notice that that MVR and multicast routing are mutually exclusive. When MVR is enabled, switch performs like Layer2 device with respect to multicast traffic, and multicast-routing should remain disabled. However, the switch still inspects IGMP messages and may forward them to the port where multicast router is connected or not. Back to the modes - the first mode is called dynamic. This mode allows the multicast router connected to the switch ring to listen to the IGMP messages and dynamically create respective mroute states. That is, IGMP join messages from the clients are forwarded to the multicast source port, allowing stopping broadcast of unneeded channels .The other mode, called compatible (the default in switches), inspects the IGMP messages but does not forward them to the multicast router ports. This was the default mode of operation of MVR feature in 3500XL switches. If you use compatible mode, make sure you configured static IGMP joins in the multicast router to feed the multicast streams down. Default mode assumes that the source constantly sends down all multicast channels, and does not process client IGMP joins.

To perform MVR configuration, follow the steps below:

Step 1:
Enable MVR globally using the commands **`mvr`** and **`mvr group <mcast-group> <count>.`** This will enable MVR feature for the particular group or for the number of groups specified by **`<count>`** argument and starting at **`<mcast-group>`** address. Note that you cannot have more than 256 address configured for MVR, and they should never alias. Here aliasing means that two multicast IP addresses map to the same multicast MAC address (see the previous task). For example addresses 228.1.1.1 and 230.1.1.1 would alias to the same MAC address and would be rejected.

Step 2:
Define MVR VLAN – the VLAN that spans multiple switches and carries the actual multicast traffic feed. This VLAN should be allowed on all trunks to permit multicast delivery. The command is **`mvr vlan <vlan-id>`**. You may also define MVR mode using the command **`mvr {dynamic|compatible}`**

Step 3:
Configure source and receiver interfaces, using the interface-mode command **`mvr type {source|receiver}`**. Additionally, you may configure the port for immediate leave, using the command **`mvr immediate`**. When this command is enabled, any single IGMP Leave would cause the switch to prune the port from receiving multicast feeds. This is good for the ports that have only one host connected.

Step 4:
Optionally, configure static group joins using the command **`mvr vlan <vlan-id> group <ip-address>`** on receiver ports. This is similar to static join command configure in the router, but allows pulling multicast traffic from the MVR VLAN. Another optional command is **`mvr querytime <1/10 of second>`**. This command is similar to **`the igmp query-max-response-time`** configured in routers. The switch passively listens to IGMP general queries sent from multicast routers, and then starts the **`querytime`** timer, waiting for client replies. If no replies are received during the interval, the switch prunes the port from the list of output ports for the group.

Notice that you cannot configure trunk ports as MVR receivers.

SW1:

```
no ip multicast-routing distributed
mvr
mvr group 239.1.1.100
mvr mode dynamic
!
interface FastEthernet 0/1
 mvr type source
!
interface FastEthernet 0/5
 mvr type receiver
```

Verification

Note

Check MVR settings in SW1. Since the mode is dynamic, R1 should receive IGMP Joins from R5.

```
Rack1SW1#show mvr
MVR Running: TRUE
MVR multicast VLAN: 146
MVR Max Multicast Groups: 256
MVR Current multicast groups: 1
MVR Global query response time: 5 (tenths of sec)
MVR Mode: dynamic

Rack1SW1#show mvr interface
Port      Type       Status           Immediate Leave
----      ----       ------           ---------------
Fa0/1     SOURCE     ACTIVE/UP        DISABLED
Fa0/5     RECEIVER   ACTIVE/UP        DISABLED

Rack1SW1#show mvr members
MVR Group IP    Status          Members
------------    ------          -------
239.001.001.100 ACTIVE          Fa0/5(d)

Rack1R1#show ip igmp groups
IGMP Connected Group Membership
Group Address    Interface              Uptime    Expires   Last
Reporter   Group Accounted
239.1.1.100      FastEthernet0/0        00:18:04  00:02:56
155.1.58.5
224.0.1.40       FastEthernet0/0        00:29:42  00:02:50
155.1.146.1
```

Note

Now source the multicast feed from R1 and check that R5 actually receives the multicast packets. Notice that you may need to disable multicast routing cache in R5 for this.

```
Rack1R1#ping 239.1.1.100 repeat 100

Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 239.1.1.100, timeout is 2 seconds:
...

Rack1R5#debug ip mpacket
IP multicast packets debugging is on
Rack1R5#
IP(0): s=155.1.146.1 (FastEthernet0/0) d=239.1.1.100 id=89, ttl=253,
prot=1, len=114(100), RPF lookup failed for source or RP
Rack1R5#
IP(0): s=155.1.146.1 (FastEthernet0/0) d=239.1.1.100 id=90, ttl=253,
prot=1, len=114(100), RPF lookup failed for source or RP
Rack1R5#
IP(0): s=155.1.146.1 (FastEthernet0/0) d=239.1.1.100 id=91, ttl=253,
prot=1, len=114(100), RPF lookup failed for source or RP
```

8.34 Catalyst IGMP Profiles

- Configure SW4 to permit only the IGMP reports for groups in ranges 232.0.0.0/8 and 239.0.0.0/8 from R4.

Configuration

Note

Catalyst switches allow filtering of IGMP messages sent by directly connected hosts to multicast routers. This feature is similar to **`ip igmp access-group`** used in routers, but applies to transit IGMP messages. The functionality is accomplished by using IGMP profiles. IGMP profiles have global numbers, and every profile defines a list of ranges plus the accompanying operation – "permit" or "deny" (default). In the first case, the switch permits IGMP reports for the groups specified by the range and denies everything else. In the second case, the switch denies the group range configured and permits everything else. To configure the profile, use the command

```
ip igmp profile <global-number>
 range low-address1 [high-address1]
 range low-address2 [high-address2]
 ...
 [permit|deny]
```

The profile applies ingress to layer 2 ports only using the interface-level command **`ip igmp filter <number>`** and affects all IGMP reports sent by hosts connected to the particular port.

SW4:
```
ip igmp profile 1
  permit
  range 232.0.0.0 232.255.255.255
  range 239.0.0.0 239.255.255.255
!
interface FastEthernet 0/4
 ip igmp filter 1
```

Verification

Note

Join R4's VLAN146 interface to a couple of groups. One group should match the profile criteria, while other should not. After this, verify what IGMP groups are seen in the switch, using IGMP snooping function.

```
R4:
interface FastEthernet0/1
 ip igmp join-group 239.4.4.4
 ip igmp join-group 230.4.4.4

Rack1SW4#show ip igmp snooping groups vlan 146
Vlan      Group             Version     Port List
-----------------------------------------------------------
146       224.0.1.40        v2          Fa0/13
146       239.4.4.4         v2          Fa0/4
```