MULTICAST VPN

Some Multicast profiles explained

ABSTRACT

In depth description of some of the most common multicast vpn profile and configurations for Cisco routers.

Andreetta,R,Riccardo,JBP11 R

26 vpn profiles

Contents

Multicast vpn 2

Profile 0, Default MDT - GRE - PIM C-Mcast Signaling 5

Profile 1, Default MDT - MLDP MP2MP PIM C-Mcast Signaling 8

Profile 2, Partitioned MDT - MLDP MP2MP - PIM C-Mcast Signaling 9

Profile 3, Default MDT – GRE, BGP-AD, PIM C-Mcast Signaling 10

Profile 4 Partitioned MDT, MLDP MP2MP, BGP-AD, PIM C-mcast Signaling 12

Profile 5 Partitioned MDT, MLDP P2MP, BGP-AD, PIM C-mcast Signaling 12

Profile 6, VRF MLDP - In-Band Signaling 12

Summary for profiles 0, 1 and 6/7 17

BGP C-Mcast Signaling 17

PIM SM as customer multicast protocol 19

PIM vs BGP signaling 22

Customer BIDIR-PIM 22

Assert multicast mechanism 27

What is PIM assert mechanism ? by Cisco TAC 28

Assert mechanism on default mdt 39

Assert Mechanism on the Default MDT 39

Assert Mechanism with Data MDTs 44

Partitioned MDT 48

Duplicated traffic problem 49

Solution 54

Profile 10: VRF Static - P2MP TE - BGP-AD 57

Profile 11: Default MDT GRE, BGP-AD, BGP C-Mcast Signaling 57

Profile 12: Default MDT, MLDP - P2MP, BGP-AD, BGP C-Mcast Signaling 60

Troubleshooting commands 61

Troubleshooting Multicast Routing 62

Abstract 62

Common Reasons for Multicast Problems 62

PIM SM Troubleshooting Routine 63

Understanding the *mtrace* command 64

Resolving RPF failures in control plane 67

Troubleshooting Auto-RP 67

Troubleshooting PIM BSR 69

Troubleshooting Multicast Data Plane 70

Decoding the *show ip mroute* command output 71

Summary... 74

Copyright disclaimer: I’ve found most of the content of this document on the internet, on blogs, Cisco communities, Cisco live presentations, “You tube” videos. The most useful and detailed resource in my opinion is the following one:

https://www.youtube.com/channel/UC2xDUkd_PuxDvhl4zQz67Aw

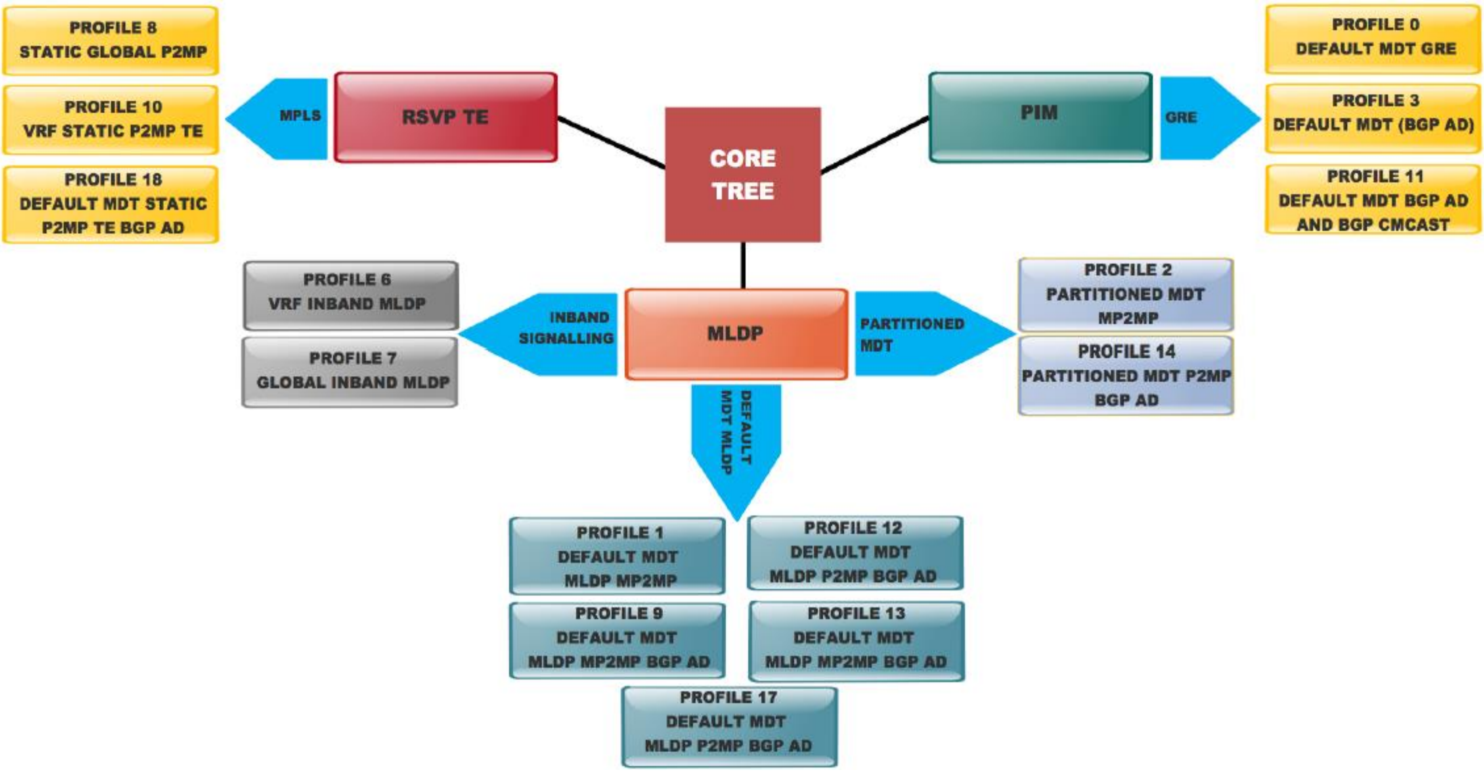
.. with 19 MVPN Videos of ‘Decoding Packets’ channel. Multicast is NOT the only treated subject. This document is not for dummies, the prerequisite is already having a quite detailed multicast knowledge, and tries to better explain some of the 26 (!!!!!) profiles that Cisco supports on some of its devices.

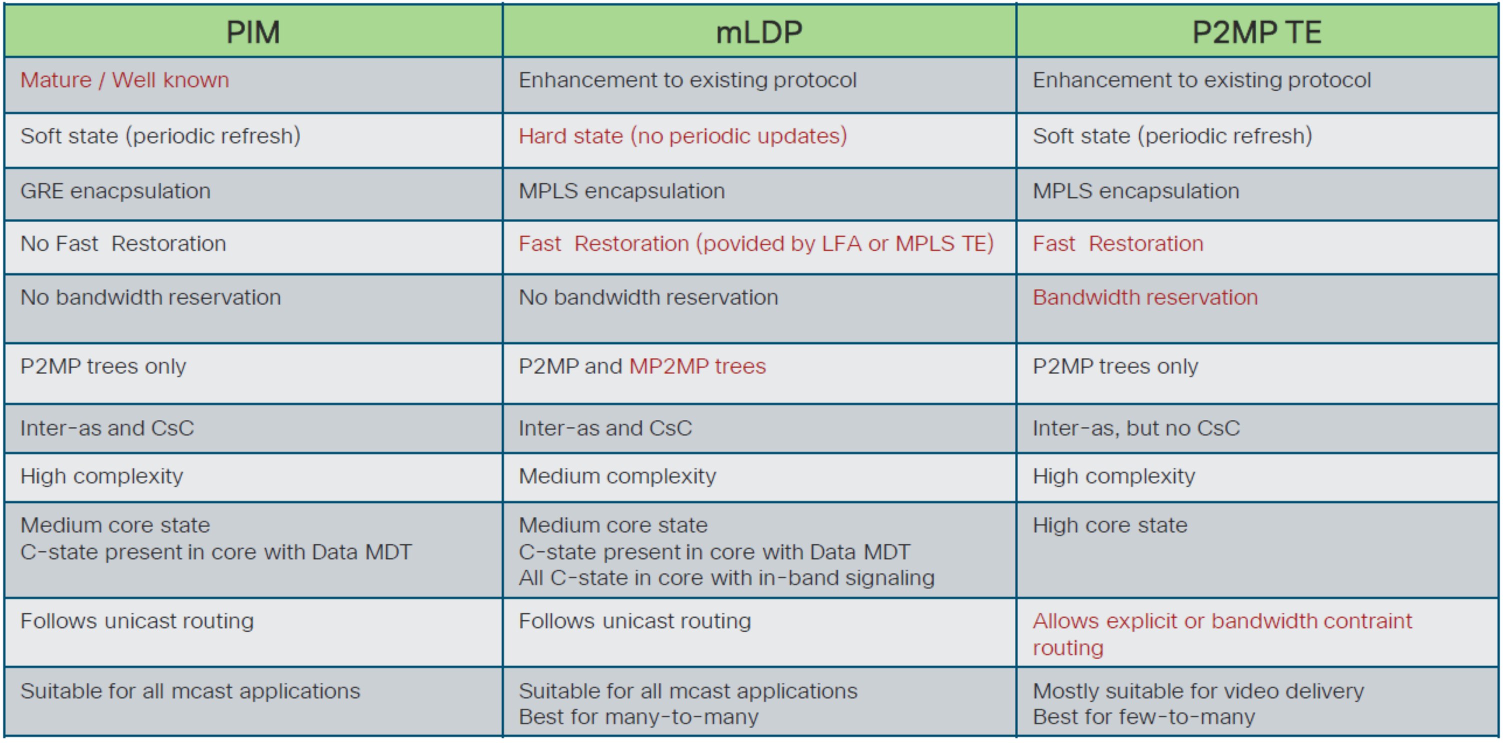
Multicast vpn

Multicast is in my opinion the most complex protocol in the networking area, and transporting multicast traffic in an mpls backbone leads the complexity to an even higher level.

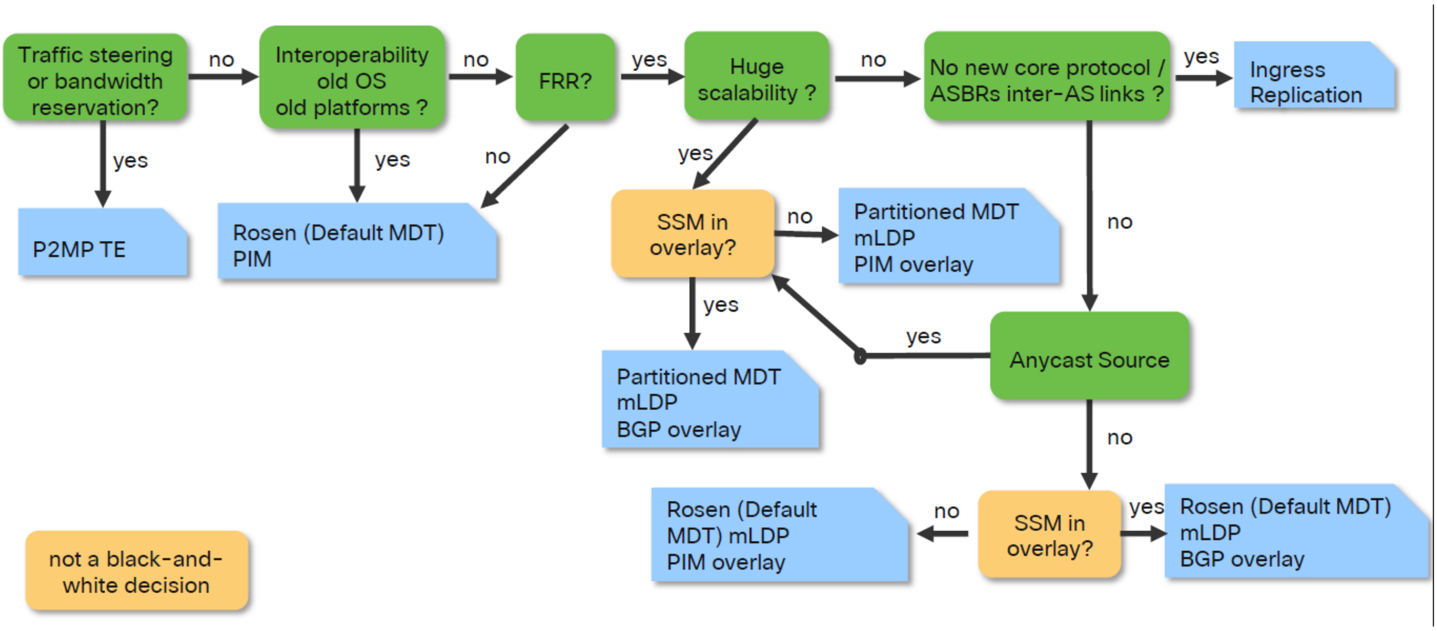
20 years old Rosen GRE is still a valid option on the field, supported by everyone and compatible between all vendors. New implementations will for sure use **multicast ldp**, even though network core using Segment Routing could not be so happy about keeping the multicast version of ldp still configured in the backbone. For this reason a new protocol called ‘BIER’ is under proposal, but to be honest I don’t know if it’s going on successfully, and some vendors are supporting and using it.

https://www.ciscolive.com/c/dam/r/ciscolive/us/docs/2018/pdf/BRKIPM-2249.pdf





PIM	mLDP	P2MP TE
Mature / Well known	Enhancement to existing protocol	Enhancement to existing protocol
Soft state (periodic refresh)	Hard state (no periodic updates)	Soft state (periodic refresh)
GRE enacpsulation	MPLS encapsulation	MPLS encapsulation
No Fast Restoration	Fast Restoration (povided by LFA or MPLS TE)	Fast Restoration
No bandwidth reservation	No bandwidth reservation	Bandwidth reservation
P2MP trees only	P2MP and MP2MP trees	P2MP trees only
Inter-as and CsC	Inter-as and CsC	Inter-as, but no CsC
High complexity	Medium complexity	High complexity
Medium core state C-state present in core with Data MDT	Medium core state C-state present in core with Data MDT All C-state in core with in-band signaling	High core state
Follows unicast routing	Follows unicast routing	Allows explicit or bandwidth contraint routing
Suitable for all mcast applications	Suitable for all mcast applications Best for many-to-many	Mostly suitable for video delivery Best for few-to-many


Traffic steering or bandwidth reservation?
no
yes
P2MP TE
Interoperability old OS old platforms ?
no
yes
Rosen (Default MDT) PIM
FRR?
no
yes
Huge scalability ?
no
yes
SSM in overlay?
no
Partitioned MDT mLDP PIM overlay
yes
Partitioned MDT mLDP BGP overlay
No new core protocol / ASBRs inter-AS links ?
yes
Ingress Replication
no
Anycast Source
yes
no
SSM in overlay?
no
Rosen (Default MDT) mLDP PIM overlay
yes
Rosen (Default MDT) mLDP BGP overlay
not a black-and-white decision

Regarding multicast vpn configurations, there are **26 possible profiles**. Here we try to describe the most important characteristics for everyone of them.

Profile 0 Default MDT - GRE - PIM C-Mcast Signaling
Profile 1 Default MDT - MLDP MP2MP PIM C-Mcast Signaling
Profile 2 Partitioned MDT - MLDP MP2MP - PIM C-Mcast Signaling
Profile 3 Default MDT - GRE - BGP-AD - PIM C-Mcast Signaling
Profile 4 Partitioned MDT - MLDP MP2MP - BGP-AD - PIM C-Mcast Signaling
Profile 5 Partitioned MDT - MLDP P2MP - BGP-AD - PIM C-Mcast Signaling
Profile 6 VRF MLDP - In-Band Signaling

Profile 7 Global MLDP In-band Signaling
Profile 8 Global Static - P2MP-TE
Profile 9 Default MDT - MLDP - MP2MP - BGP-AD - PIM C-Mcast Signaling
Profile 10 VRF Static - P2MP TE - BGP-AD
Profile 11 Default MDT - GRE - BGP-AD - BGP C-Mcast Signaling
Profile 12 Default MDT - MLDP - P2MP - BGP-AD - BGP C-Mcast Signaling
Profile 13 Default MDT - MLDP - MP2MP - BGP-AD - BGP C-Mcast Signaling
Profile 14 Partitioned MDT - MLDP P2MP - BGP-AD - BGP C-Mcast Signaling
Profile 15 Partitioned MDT - MLDP MP2MP - BGP-AD - BGP C-Mcast Signaling
Profile 16 Default MDT Static - P2MP TE - BGP-AD - BGP C-Mcast Signaling
Profile 17 Default MDT - MLDP - P2MP - BGP-AD - PIM C-Mcast Signaling
Profile 18 Default Static MDT - P2MP TE - BGP-AD - PIM C-Mcast Signaling
Profile 19 Default MDT - IR - BGP-AD - PIM C-Mcast Signaling
Profile 20 Default MDT - P2MP-TE - BGP-AD - PIM - C-Mcast Signaling
Profile 21 Default MDT - IR - BGP-AD - BGP - C-Mcast Signaling
Profile 22 Default MDT - P2MP-TE - BGP-AD BGP - C-Mcast Signaling
Profile 23 Partitioned MDT - IR - BGP-AD - PIM C-Mcast Signaling
Profile 24 Partitioned MDT - P2MP-TE - BGP-AD - PIM C-Mcast Signaling
Profile 25 Partitioned MDT - IR - BGP-AD - BGP C-Mcast Signaling
Profile 26 Partitioned MDT - P2MP TE - BGP-AD - BGP C-Mcast Signaling

Some configurations are ‘IOS-XR specific’. Here a few advises:

```
! global routing table multicast enable
multicast-routing
 address-family ipv4
  mdt mldp in-band-signaling ipv4      ← only for profile 7

! other commands inside the vrf
multicast-routing
 vrf one
  address-family ipv4
   mdt mldp in-band-signaling ipv4
   mdt partitioned mldp ipv4 p2mp (bidir)
   mdt partitioned mldp ipv4 mp2mp (bidir)
   mdt partitioned ingress-replication
   mdt mldp in-band-signaling ipv4          ← only for profile 6
   mdt default mldp ipv4 <root>
   mdt default mldp p2mp (partitioned)(bidir)
   mdt default ingress-replication
   mdt default <ipv4-group>
   mdt default (ipv4) <ipv4-group> partitioned
   mdt data <ipv4-group/length>
   mdt data <max nr of data groups> (threshold)
   mdt static p2mp-te tunnel-te <0-65535>
   mdt static tunnel-mte <0-65535>
 !
!
```

Always specify the mdt tree in the core, enable multicast for loopback 0, and use it to build up the core default tree:

```
multicast-routing
 address-family ipv4
  interface Loopback0           ← always enable loopback0 belonging to the CORE
   enable
 !
 mdt source Loopback0
 mdt mldp in-band-signaling ipv4
 rate-per-route
 interface all enable
 accounting per-prefix
 !
 vrf one
  address-family ipv4
   mdt source Loopback0                        ← always to be configured
   mdt default mldp ipv4 10.1.100.7            ← core tree based on mldp
   rate-per-route
   interface all enable
   accounting per-prefix
  !
 !
!
```

It is possible for multiple MDTs or core trees to be configured and signaled. In order to specify the core tree that the multicast traffic should take, a Reverse Path Forwarding (RPF) policy should be configured. This is done with a route-policy. The egress Provider Edge (PE) then initiates the core tree based on the RPF policy. Use the rpf topology route-policy route-policy-name command in order to complete this action. This is the route-policy that is applied under the section for the router Protocol Independent Multicast (PIM). In the route-policy, you can optionally set the core tree after you specify an IF-statement:

```
RP/0/3/CPU0:Router(config-rpl)#set core-tree ?
ingress-replication-default Ingress Replication Default MDT core
ingress-replication-partitioned Ingress Replication Partitioned MDT core
mldp-default MLDP Default MDT core
mldp-inband MLDP Inband core
mldp-partitioned-mp2mp MLDP Partitioned MP2MP MDT core
mldp-partitioned-p2mp MLDP Partitioned P2MP MDT core
p2mp-te-default P2MP TE Default MDT core
p2mp-te-partitioned P2MP TE Partitioned MDT core
parameter Identifier specified in the format: '$'
followed by alphanumeric characters
pim-default PIM Default MDT core
```

Profile 0, Default MDT - GRE - PIM C-Mcast Signaling

This is the oldest but still supported approach: the ‘Rosen GRE’ solution. With this solution, **PIM** is also **used in the core** of the Service Provider to build on ‘overlay’ tree. Customer’s multicast traffic received in the vrf is encapsulated through a new logically created GRE interface using the multicast configured address (232.100.1.1 in the example below), customer signaling is also encapsulated in the same tree to reach all PE devices. Once the PEs dynamically learn about the other sources/PEs in the core (every PE sends register messages to the RP, which is the place where sources and receivers meet each other at the beginning), they will switch over the (PE loop0, 232.100.1.1) group, thus in the core also the P routers will have to maintain a few states, proportional to the number of PEs participating into multicast for that particular vrf. The logical interface created in the SP core **works like a LAN**, where all the PEs will see each other as neighbors.

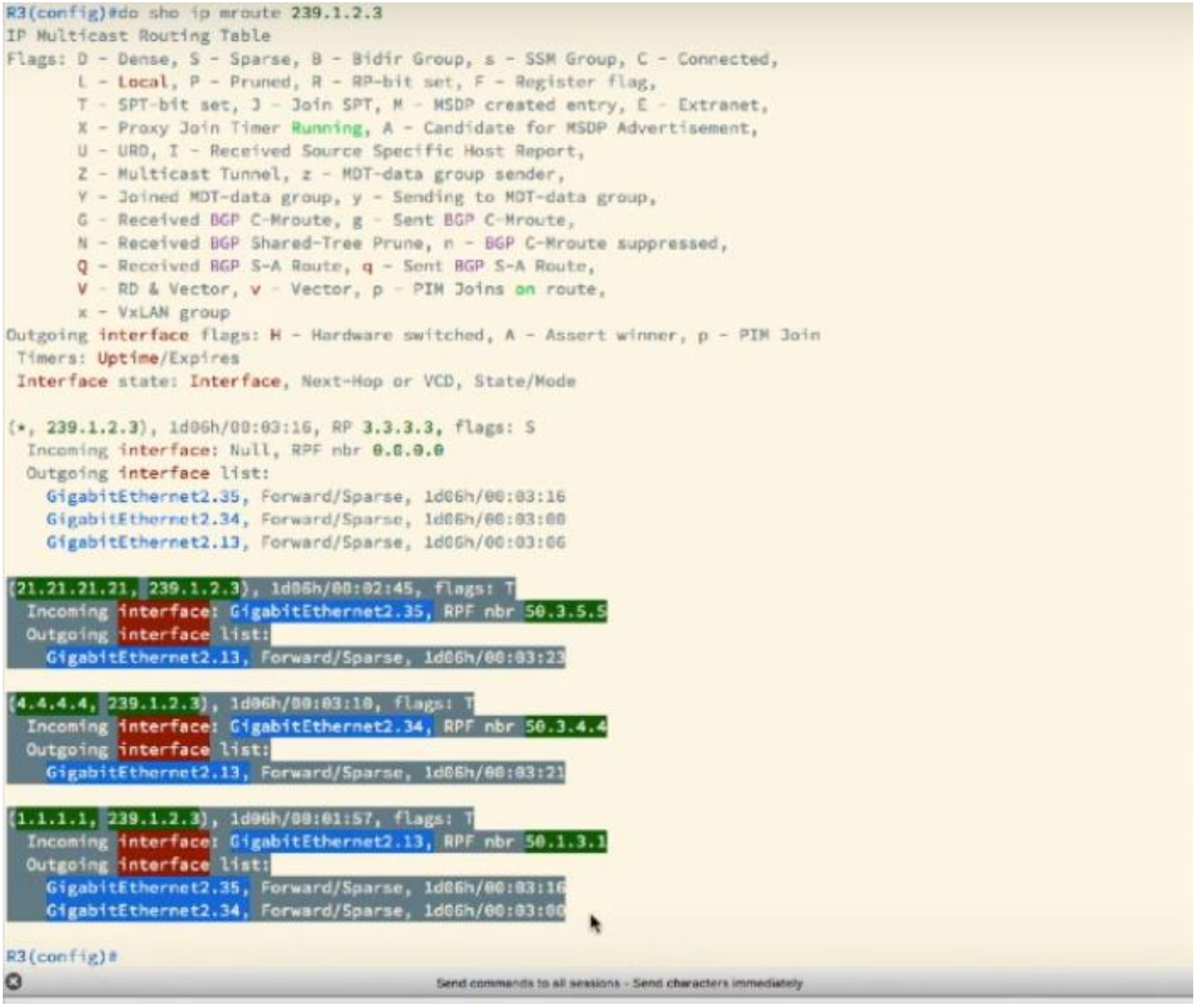
To reduce the number of states, the tree could be configured for **BIDIR pim** so that (S, G) groups would no more be maintained, but routing could be suboptimal in this case. This approach looks like the profile 1 solution, where a MP2MP lsp is built through multicast ldp. The problem of finding a bidir tree solves something similar to STP, finding a common ‘root’ that reaches all remote nodes/leafs.

In case **Source Specific Multicast** instead is used in the core, **no RP** is necessary, but bgp auto-discovery will need to be used to discover the other PEs participating into multicast for that vrf. See profile 3 for more information.

A problem with the described approach (common to all the above approaches), is that unnecessary data traffic is sent to some PEs even though they don’t have a receiver behind them. In case there are high bandwidth sources (it’s configurable what this means), PEs can optionally join **(PEx, 232.x.y.z)** ‘data’ groups, where PEx is the bgp next-hop in the specific vrf to reach the high bandwidth’s source ip address. This avoids wasting bandwidth and helps optimizing network usage and rtt.

```
route-policy rpf-for-one
 set core-tree pim-default
end-policy
!
multicast-routing
 address-family ipv4
  interface Loopback0
   enable
  !
 interface GigabitEthernet0/0/0/3 <<< Multicast is enabled for global context intf
  enable
 !
 mdt source Loopback0
 !
 vrf one
  address-family ipv4
   mdt source Loopback0
    mdt data 232.100.100.0/24           ← data mdt trees
    mdt default ipv4 232.100.1.1        ← build a PIM overlay in the core
   rate-per-route
   interface all enable
   accounting per-prefix
 !
!

! in case an RP is configured on all PEs
ip pim rp-address 11.11.11.11

! In case an RP is NOT configured, SSM must be used. In this case, probably bgp auto
! discovery MUST also be configured, otherwise PEs do not know which are the other PEs
! that has to be joined in the core PIM. In this case every PE would do a join for the
! 232.100.1.1 multicast group toward every other PE participating for that specific vrf
! thus building up multiple P2MP tress, practically working as a unique MP2MP bidir tree.
! This would happen in profile 3.
```

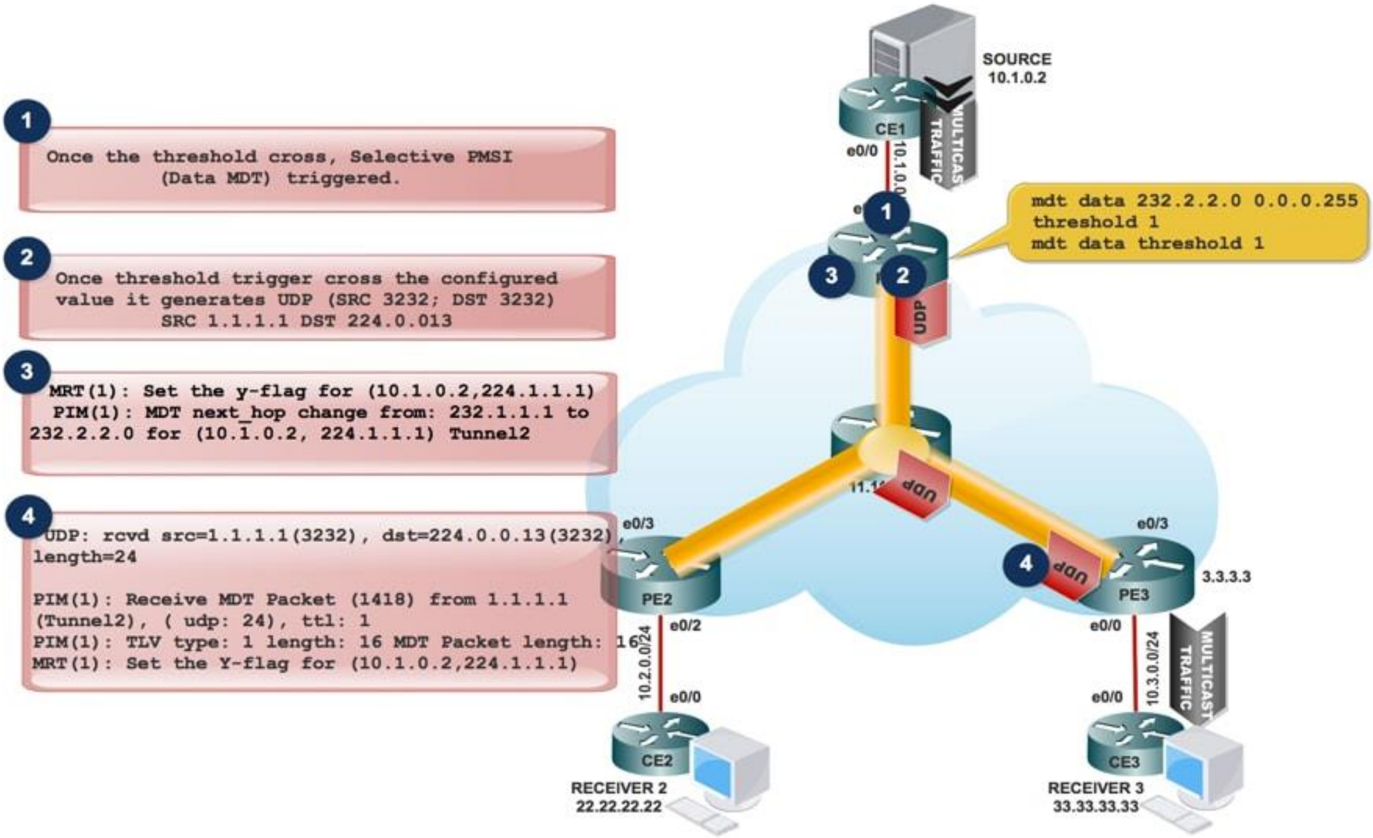
NOTE: Cisco in this example used a SSM reserved multicast group instead of an ASM multicast group chosen from some other multicast space address. This is probably not correct.



```
R3(config)#do sho ip mroute 239.1.2.3
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, E - Extranet,
       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,
       G - Received BGP C-Mroute, g - Sent BGP C-Mroute,
       N - Received BGP Shared-Tree Prune, n - BGP C-Mroute suppressed,
       Q - Received BGP S-A Route, q - Sent BGP S-A Route,
       V - RD & Vector, v - Vector, p - PIM Joins on route,
       x - VxLAN group
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.1.2.3), 1d06h/00:03:16, RP 3.3.3.3, flags: S
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    GigabitEthernet2.35, Forward/Sparse, 1d06h/00:03:16
    GigabitEthernet2.34, Forward/Sparse, 1d06h/00:03:00
    GigabitEthernet2.13, Forward/Sparse, 1d06h/00:03:06

(21.21.21.21, 239.1.2.3), 1d06h/00:02:45, flags: T
  Incoming interface: GigabitEthernet2.35, RPF nbr 50.3.5.5
  Outgoing interface list:
    GigabitEthernet2.13, Forward/Sparse, 1d06h/00:03:23

(4.4.4.4, 239.1.2.3), 1d06h/00:03:10, flags: T
  Incoming interface: GigabitEthernet2.34, RPF nbr 50.3.4.4
  Outgoing interface list:
    GigabitEthernet2.13, Forward/Sparse, 1d06h/00:03:21

(1.1.1.1, 239.1.2.3), 1d06h/00:01:57, flags: T
  Incoming interface: GigabitEthernet2.13, RPF nbr 50.1.3.1
  Outgoing interface list:
    GigabitEthernet2.35, Forward/Sparse, 1d06h/00:03:16
    GigabitEthernet2.34, Forward/Sparse, 1d06h/00:03:00

R3(config)#
```

The choice of joining directly the high bandwidth source only, using a new multicast group for this purpose, depends on the configured bandwidth thresholds that could even be ‘0’ (always switch) of ‘infinite’ (never switch). The PE near the source sends an encapsulated UDP packet to 224.0.013 (all PIM neighbors), thus communicating the old default mdt and the new value, and setting an ‘Y’ flag to ask for re-joining to the new data tree.

```
MRT(1): Set the y-flag for (10.1.0.2, 224.1.1.1)

PIM(1): MDT next_hop change from: 232.1.1.1 to 232.2.2.0 for (10.1.0.2, 224.1.1.1)
Tunnel2
```



```
1
Once the threshold cross, Selective PMSI
(Data MDT) triggered.
```

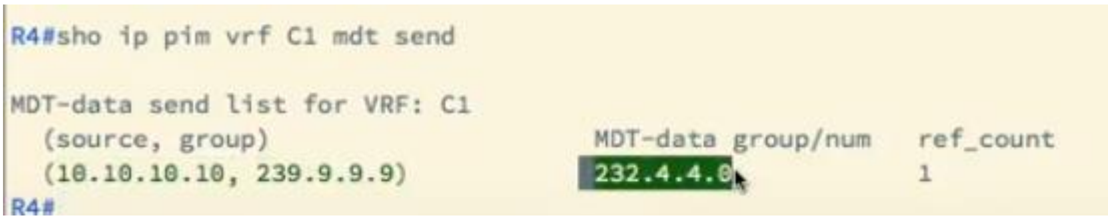
```
2
Once threshold trigger cross the configured
value it generates UDP (SRC 3232; DST 3232)
SRC 1.1.1.1 DST 224.0.013
```

```
3
MRT(1): Set the y-flag for (10.1.0.2,224.1.1.1)
PIM(1): MDT next_hop change from: 232.1.1.1 to
232.2.2.0 for (10.1.0.2, 224.1.1.1) Tunnel2
```

```
4
UDP: rcvd src=1.1.1.1(3232), dst=224.0.0.13(3232),
length=24

PIM(1): Receive MDT Packet (1418) from 1.1.1.1
(Tunnel2), ( udp: 24), ttl: 1
PIM(1): TLV type: 1 length: 16 MDT Packet length: 16
MRT(1): Set the Y-flag for (10.1.0.2,224.1.1.1)
```

```
mdt data 232.2.2.0 0.0.0.255
threshold 1
mdt data threshold 1
```

PE3 joins the new group while PE2 doesn't, in this way traffic will not be sent also to PE2. This kind of PIM messages can be seen using the following command.



```
R4#sho ip pim vrf C1 mdt send

MDT-data send list for VRF: C1
  (source, group)                     MDT-data group/num   ref_count
  (10.10.10.10, 239.9.9.9)            232.4.4.0            1
R4#
```

Profile 1, Default MDT - MLDP MP2MP PIM C-Mcast Signaling

This is very similar to profile 0, with the only difference that the default mdt is built using **multicast ldp** (or **mldp** from now on). Of course the RP would be managed with a 'phantom RP' (e.g. one router has a /32 loopback, another one a /31 thus providing node redundancy).

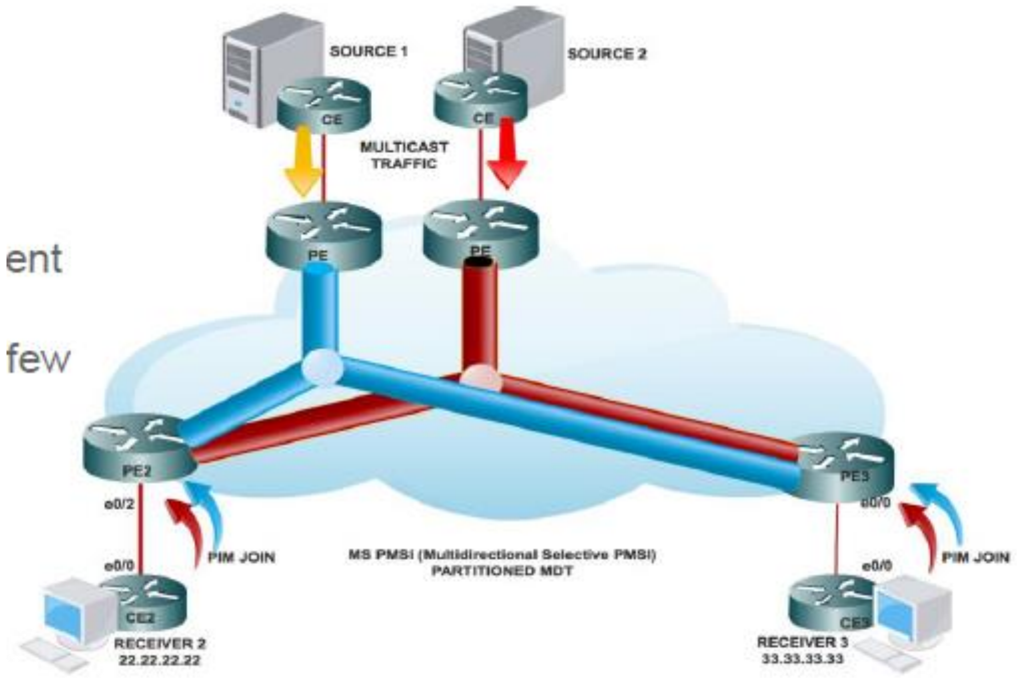
```
route-policy rpf-for-one
 set core-tree mldp-default               ← rpf policy to be used with the default tree
end-policy

vrf one
 vpn id 1:1                 ← not really necessary unless mpvn address-family is used ???
  address-family ipv4 unicast
   import route-target
    1:1
   !
   export route-target
    1:1
   !
!
```

```
router pim
 vrf one
  address-family ipv4
   rpf topology route-policy rpf-for-one
   !
   interface GigabitEthernet0/1/0/0
     enable
   !
!
multicast-routing
 vrf one
  address-family ipv4
   mdt source Loopback0
    mdt default mldp ipv4 10.1.100.1          ← mp2mp ldp tree
    mdt data 100                              ← number of max data tree for this vrf
   rate-per-route
   interface all enable
  !
  accounting per-prefix
 !
 !
!
mpls ldp
 mldp
  logging notifications
  address-family ipv4
 !
!
```

Profile 2, Partitioned MDT - MLDP MP2MP - PIM C-Mcast Signaling

Profile 2 is not currently supported in the Cisco IOS, and MLDP does not support Partitioned MDT with Multipoint-to-Multipoint (MP2MP) trees. In IOS-XR this is apparently supported, follows the configuration hereafter. The advantage of this approach is that in case there are a few sources behind a small number of PEs, a new dedicated mldp tree could be built on demand from the receiver.

```
vrf one
 address-family ipv4 unicast
  import route-target
   1:1
  !
  export route-target
   1:1
  !
!
route-policy rpf-for-one
 set core-tree mldp-partitioned-mp2mp
end-policy
!
router pim
 vrf one
  address-family ipv4
   rpf topology route-policy rpf-for-one
  !
  interface GigabitEthernet0/1/0/0
   enable
  !
 !
!
multicast-routing
 vrf one
  address-family ipv4
   mdt source Loopback0
```



Note: The data MDTs are optional. If data MDTs are configured, then BGP-AD must be configured as well (in this case data mdt subnets would be advertised through bgp). If not, this results in an error pop up when you attempt to commit this configuration. With data MDTs configured, this becomes **profile 4**, since BGP-AD must also be configured.

Profile 3, Default MDT – GRE, BGP-AD, PIM C-Mcast Signaling

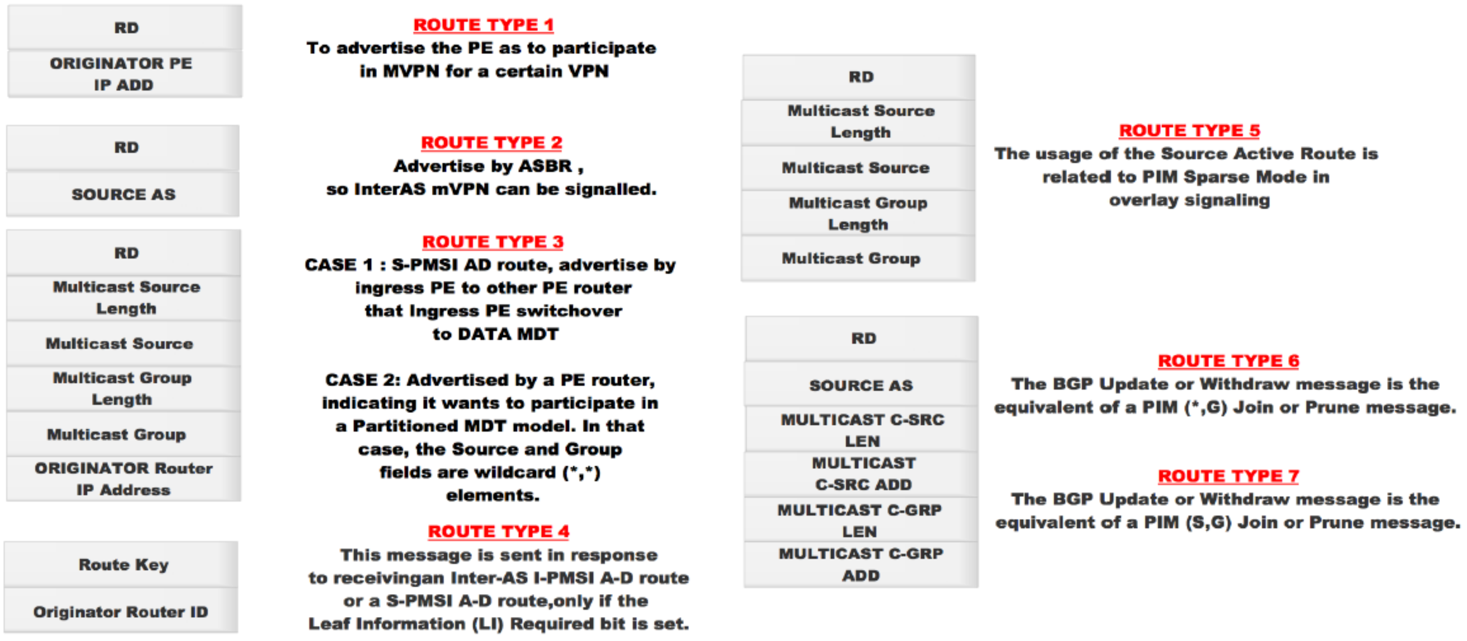
It is similar to profile 0, but **bgp auto discovery** is used to build up a list of SSM trees in the core. Through bgp and the usage of a specific **mvpn address-family** (just very similar to the 'mdt' one), for example PE1, PE2 and PE7 discover that they are participating to multicast for vrf 'one', and they send in the core an SSM join for all the other PEs. For example PE1 sends a join for (PE2 Loo0, 232.1.1.1) and (PE3 Loo0, 232.1.1.1). In this way there will be N trees of type P2MP (N being the number of PEs participating to multicast for that specific vrf) through which every PE can reach all the others. All the related information are decoded into the bgp update, for example the route type 1, the tunnel type '3' for P2MP.
Beware that the mvpn address-family needs to be activated:

- globally
- for the internal Border Gateway Protocol (iBGP) peers
- for the VRF

```
router bgp 1
 mvpn                            ← for profile 6 and 2 also this one is needed
 address-family ipv4 mvpn
 !
 neighbor x.y.z.t                ← other PE/RR
  address-family ipv4 mvpn
 !
 vrf <name>
  address-family ipv4 mvpn
!
```

(see also Cisco documentation for this)

```
vrf definition one
 rd 1:1
 !
 address-family ipv4
  mdt auto-discovery pim                ← this is to enable bgp AD
  mdt default 232.1.1.1
  route-target export 1:1
  route-target import 1:1
 exit-address-family
!

ip multicast-routing vrf one
!
interface Loopback0
 ip address 10.100.1.1 255.255.255.255
 ip pim sparse-mode
!
interface Ethernet2/0
 vrf forwarding one
 ip address 10.2.1.1 255.255.255.0
 ip pim sparse-mode
!
router bgp 1
 mvpn                  ← needed
 bgp log-neighbor-changes
 neighbor 10.100.1.7 remote-as 1
 neighbor 10.100.1.7 update-source Loopback0
!
 address-family ipv4 mvpn
  neighbor 10.100.1.7 activate
  neighbor 10.100.1.7 send-community extended
 exit-address-family
 !
 address-family vpnv4
  neighbor 10.100.1.7 activate
  neighbor 10.100.1.7 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf one
  redistribute connected
  neighbor 10.2.1.8 remote-as 65001
  neighbor 10.2.1.8 activate

 exit-address-family
```



RD
ORIGINATOR PE IP ADD

ROUTE TYPE 1
To advertise the PE as to participate in MVPN for a certain VPN

RD
SOURCE AS

ROUTE TYPE 2
Advertise by ASBR ,
so InterAS mVPN can be signalled.

RD
Multicast Source Length
Multicast Source
Multicast Group Length
Multicast Group
ORIGINATOR Router IP Address

ROUTE TYPE 3
CASE 1 : S-PMSI AD route, advertise by ingress PE to other PE router that Ingress PE switchover to DATA MDT

CASE 2: Advertised by a PE router, indicating it wants to participate in a Partitioned MDT model. In that case, the Source and Group fields are wildcard (*,*) elements.

Route Key
Originator Router ID

ROUTE TYPE 4
This message is sent in response to receivingan Inter-AS I-PMSI A-D route or a S-PMSI A-D route,only if the Leaf Information (LI) Required bit is set.

RD
Multicast Source Length
Multicast Source
Multicast Group Length
Multicast Group

ROUTE TYPE 5
The usage of the Source Active Route is related to PIM Sparse Mode in overlay signaling

RD
SOURCE AS
MULTICAST C-SRC LEN
MULTICAST C-SRC ADD
MULTICAST C-GRP LEN
MULTICAST C-GRP ADD

ROUTE TYPE 6
The BGP Update or Withdraw message is the equivalent of a PIM (*,G) Join or Prune message.

ROUTE TYPE 7
The BGP Update or Withdraw message is the equivalent of a PIM (S,G) Join or Prune message.

Profile 4 Partitioned MDT, MLDP MP2MP, BGP-AD, PIM C-mcast Signaling

Profile 4 is not currently supported in the Cisco IOS, and MLDP does not support Partitioned MDT with MP2MP. This profile is only supported on IOS XR software.

Profile 5 Partitioned MDT, MLDP P2MP, BGP-AD, PIM C-mcast Signaling

Profile 5 is not currently supported in the Cisco IOS, and PIM signaling is not supported over Partitioned MDT. This profile is only supported on IOS XR software.

This profile is very similar to **profile 3**, except that mldp is used inside the core instead of a native pim tree based on GRE encapsulation. Thus there will be bgp auto-discovery to dynamically learn all the PEs belonging to the same multicast vrf, and each of them will instantiate a p2mp lsp ONLY toward the PEs behind which there is a source or the RP. In this way there could be potentially a couple of mdt trees, and no default mdt.

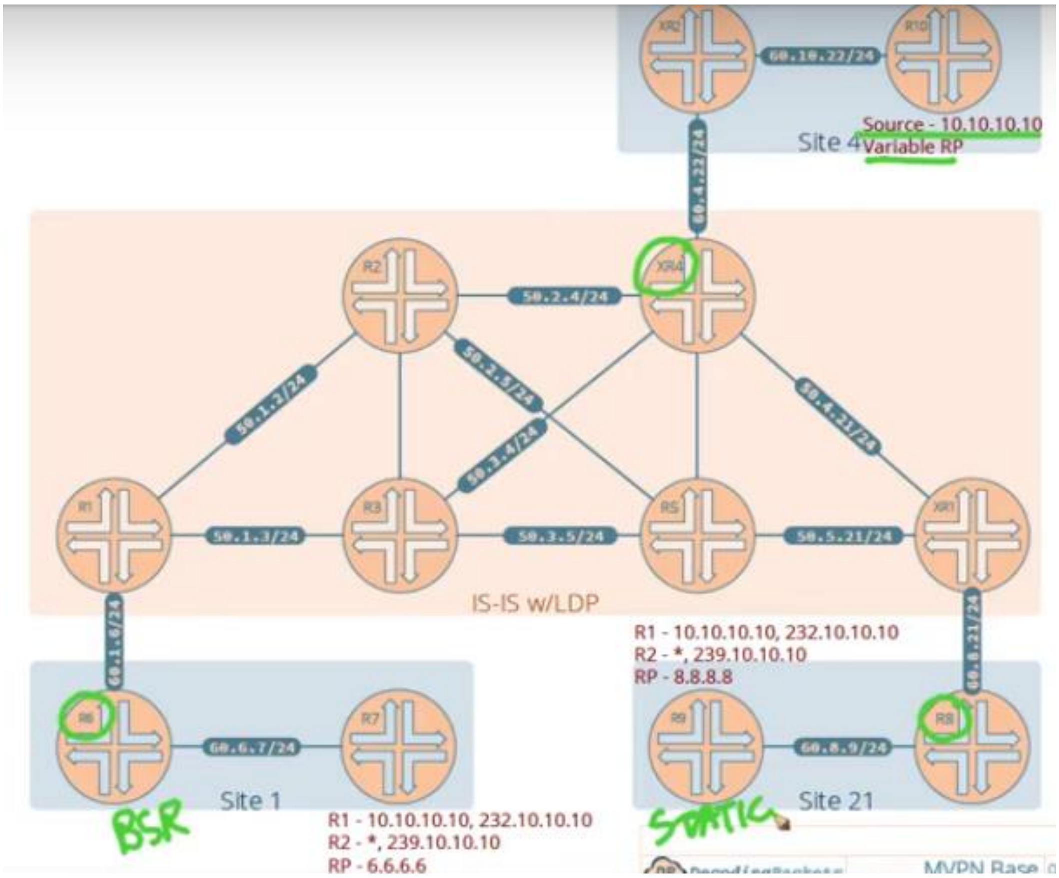
```
vrf one
 address-family ipv4 unicast
  import route-target
   1:1
  !
  export route-target
   1:1
 !
!
route-policy rpf-for-one
 set core-tree mldp-partitioned-p2mp
end-policy
!
router pim
 vrf one
  address-family ipv4
   rpf topology route-policy rpf-for-one
  !
  interface GigabitEthernet0/1/0/0            ← toward the CE
   enable
  !
 !
!
multicast-routing
 vrf one
  address-family ipv4
   mdt source Loopback0
    mdt partitioned mldp ipv4 p2mp
    mdt data 100
   rate-per-route
   interface all enable
  bgp auto-discovery mldp
  !
  accounting per-prefix
 !
!
mpls ldp
 mldp
  logging notifications
  address-family ipv4
 !
!
```

Profile 6, VRF MLDP - In-Band Signaling

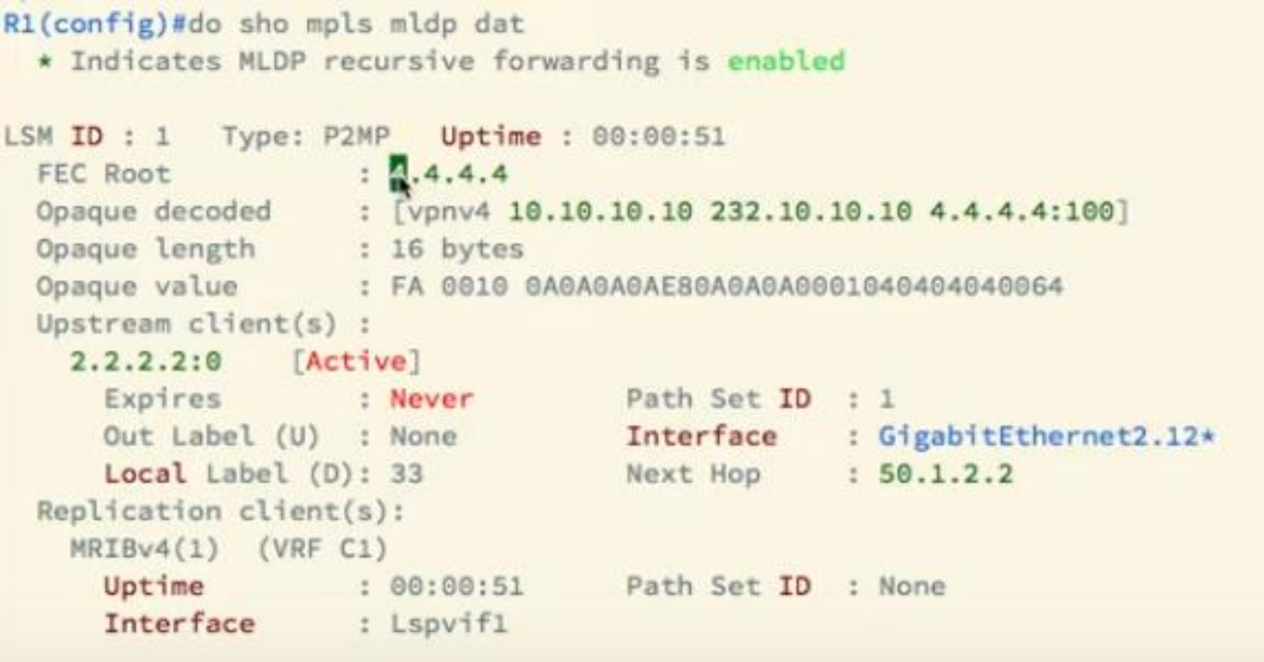
This profile uses m-ldp to distribute the necessary signaling information between PEs, in every vrf where multicast is enabled. Information about joined/pruned (*,G) or (S,G) groups are inserted into the **opaque value** of the LSP multicast tree. **PIM** is **NOT** needed anymore **in the CORE** as on overlay signaling protocol.

Beware that Cisco doesn’t support ASM for this profile (Any Source Multicast), even though with IOS-XR you can ‘hack it’.

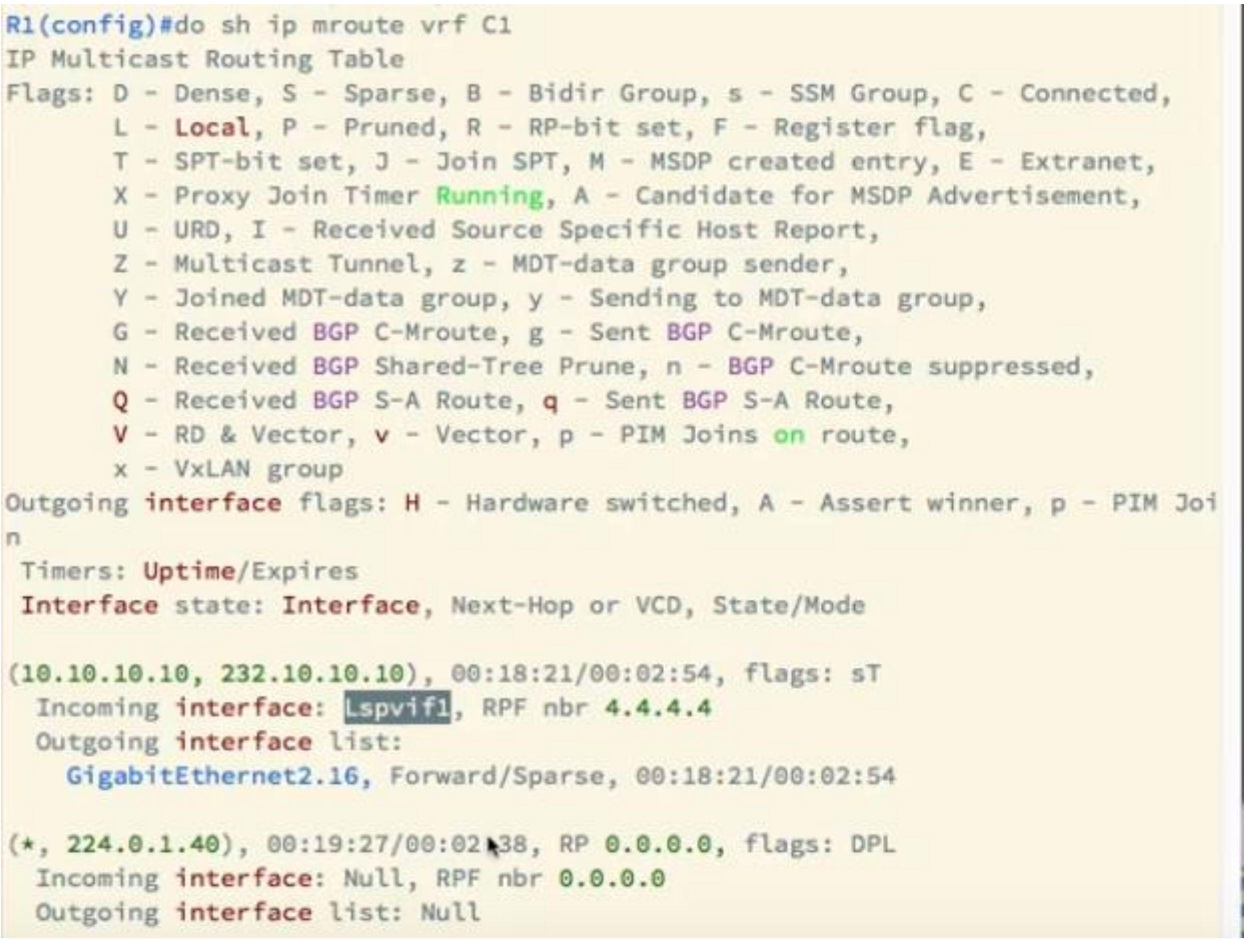
```
multicast-routing
 vrf one
  address-family ipv4
   mdt source loopback0
   mdt mldp in-band-signaling ipv4
```



In this test there is one RP in everyone of the three sites, both IOS XE and IOS-XR softare are used. Remote routers statically join the (*, G) and (S, G) groups. Once the mldp inband signaling is enabled on R1, a P2MP tree is created toward XR4 (4.4.4.4) since this is the bgp next-hop in the vrf to reach 10.10.10.10 (R10 loopback). This is the output on R1:

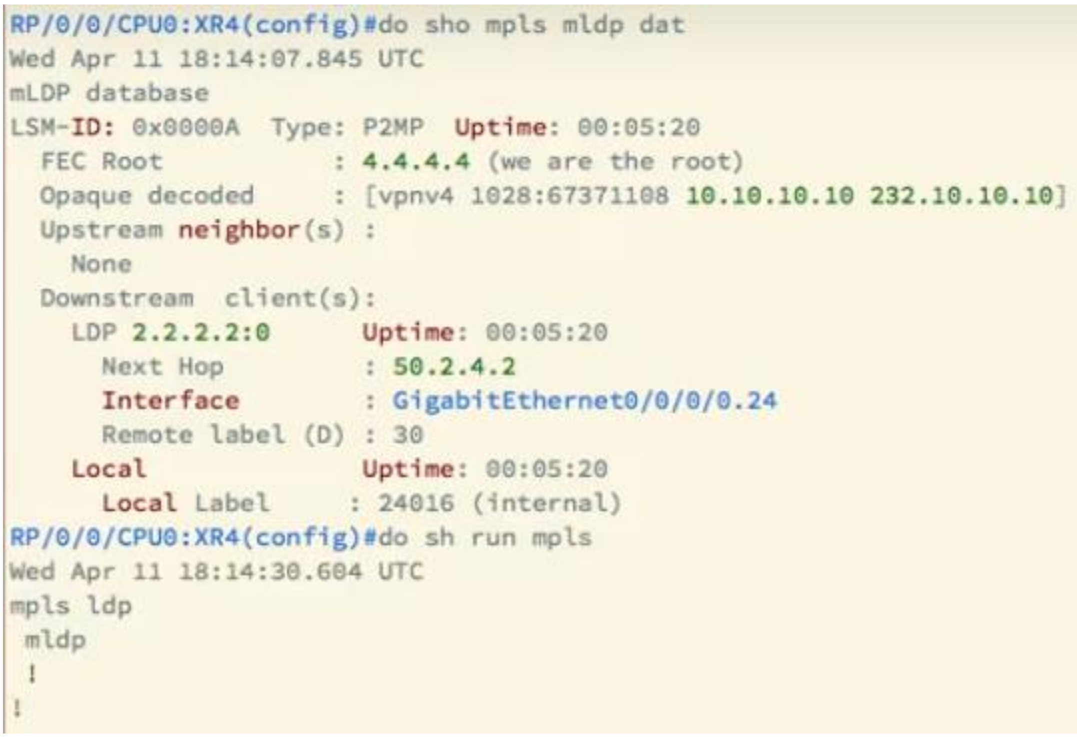


```
R1(config)#do sho mpls mldp dat
  * Indicates MLDP recursive forwarding is enabled

LSM ID : 1   Type: P2MP   Uptime : 00:00:51
  FEC Root           : 4.4.4.4
  Opaque decoded     : [vpnv4 10.10.10.10 232.10.10.10 4.4.4.4:100]
  Opaque length      : 16 bytes
  Opaque value       : FA 0010 0A0A0A0AE80A0A0A0001040404040064
  Upstream client(s) :
    2.2.2.2:0    [Active]
      Expires          : Never         Path Set ID  : 1
      Out Label (U)    : None          Interface    : GigabitEthernet2.12*
      Local Label (D): 33              Next Hop     : 50.1.2.2
  Replication client(s):
    MRIBv4(1)  (VRF C1)
      Uptime           : 00:00:51      Path Set ID  : None
      Interface        : Lspvif1
```



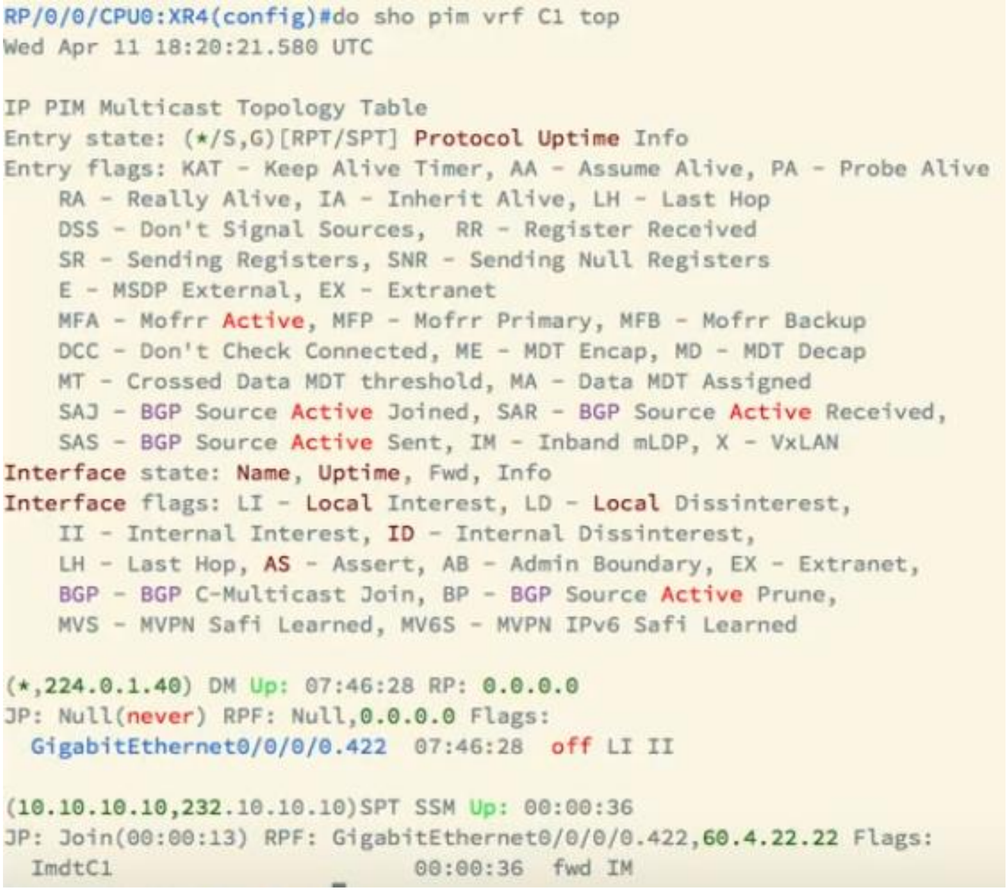
```
R1(config)#do sh ip mroute vrf C1
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry, E - Extranet,
       X - Proxy Join 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,
       G - Received BGP C-Mroute, g - Sent BGP C-Mroute,
       N - Received BGP Shared-Tree Prune, n - BGP C-Mroute suppressed,
       Q - Received BGP S-A Route, q - Sent BGP S-A Route,
       V - RD & Vector, v - Vector, p - PIM Joins on route,
       x - VxLAN group
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Joi
n
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(10.10.10.10, 232.10.10.10), 00:18:21/00:02:54, flags: sT
  Incoming interface: Lspvif1, RPF nbr 4.4.4.4
  Outgoing interface list:
    GigabitEthernet2.16, Forward/Sparse, 00:18:21/00:02:54

(*, 224.0.1.40), 00:19:27/00:02:38, RP 0.0.0.0, flags: DPL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list: Null
```

In the opaque value there is the address-family where the source is seen (vpnv4), the source we are joining, the multicast group to be listened, and the route-distinguisher related to the vpnv4 next-hop.



```
RP/0/0/CPU0:XR4(config)#do sho mpls mldp dat
Wed Apr 11 18:14:07.845 UTC
mLDP database
LSM-ID: 0x0000A  Type: P2MP  Uptime: 00:05:20
  FEC Root            : 4.4.4.4 (we are the root)
  Opaque decoded      : [vpnv4 1028:67371108 10.10.10.10 232.10.10.10]
  Upstream neighbor(s) :
    None
  Downstream  client(s):
    LDP 2.2.2.2:0       Uptime: 00:05:20
      Next Hop          : 50.2.4.2
      Interface         : GigabitEthernet0/0/0/0.24
      Remote label (D) : 30
    Local               Uptime: 00:05:20
      Local Label      : 24016 (internal)
RP/0/0/CPU0:XR4(config)#do sh run mpls
Wed Apr 11 18:14:30.604 UTC
mpls ldp
 mldp
 !
!
```

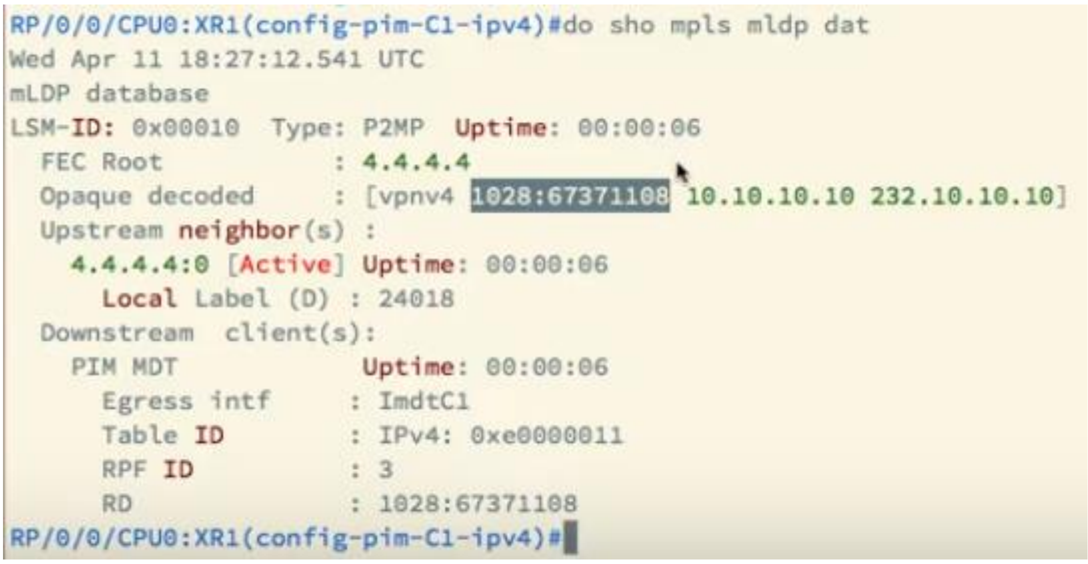
Beware that on XR4 the opaque value shows the decoded information in a slightly different way, less readable.



```
RP/0/0/CPU0:XR4(config)#do sho pim vrf C1 top
Wed Apr 11 18:20:21.580 UTC

IP PIM Multicast Topology Table
Entry state: (*/S,G)[RPT/SPT] Protocol Uptime Info
Entry flags: KAT - Keep Alive Timer, AA - Assume Alive, PA - Probe Alive
    RA - Really Alive, IA - Inherit Alive, LH - Last Hop
    DSS - Don't Signal Sources,  RR - Register Received
    SR - Sending Registers, SNR - Sending Null Registers
    E - MSDP External, EX - Extranet
    MFA - Mofrr Active, MFP - Mofrr Primary, MFB - Mofrr Backup
    DCC - Don't Check Connected, ME - MDT Encap, MD - MDT Decap
    MT - Crossed Data MDT threshold, MA - Data MDT Assigned
    SAJ - BGP Source Active Joined, SAR - BGP Source Active Received,
    SAS - BGP Source Active Sent, IM - Inband mLDP, X - VxLAN
Interface state: Name, Uptime, Fwd, Info
Interface flags: LI - Local Interest, LD - Local Dissinterest,
    II - Internal Interest, ID - Internal Dissinterest,
    LH - Last Hop, AS - Assert, AB - Admin Boundary, EX - Extranet,
    BGP - BGP C-Multicast Join, BP - BGP Source Active Prune,
    MVS - MVPN Safi Learned, MV6S - MVPN IPv6 Safi Learned

(*,224.0.1.40) DM Up: 07:46:28 RP: 0.0.0.0
JP: Null(never) RPF: Null,0.0.0.0 Flags:
  GigabitEthernet0/0/0/0.422  07:46:28  off LI II

(10.10.10.10,232.10.10.10)SPT SSM Up: 00:00:36
JP: Join(00:00:13) RPF: GigabitEthernet0/0/0/0.422,60.4.22.22 Flags:
  ImdtC1                      00:00:36   fwd  IM
```

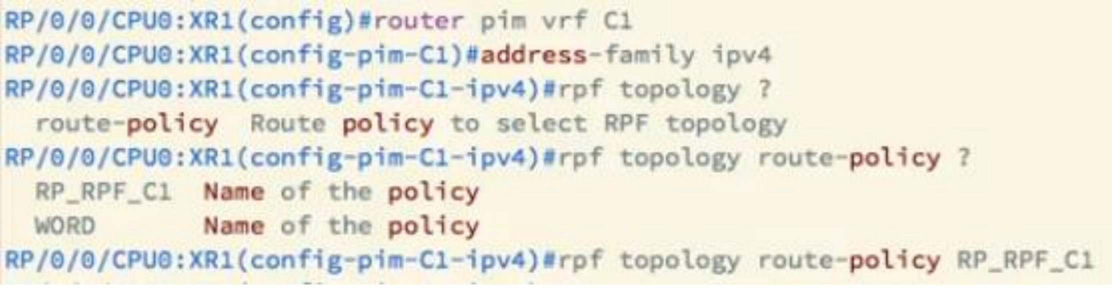
ImdtC1 interface is the 'OIL' (Outgoing Interface List) for this (S,G) group : it's the logical interface in the SP core network. XR software also shows the vrf name with it.

Once we add also XR1 to the core tree:



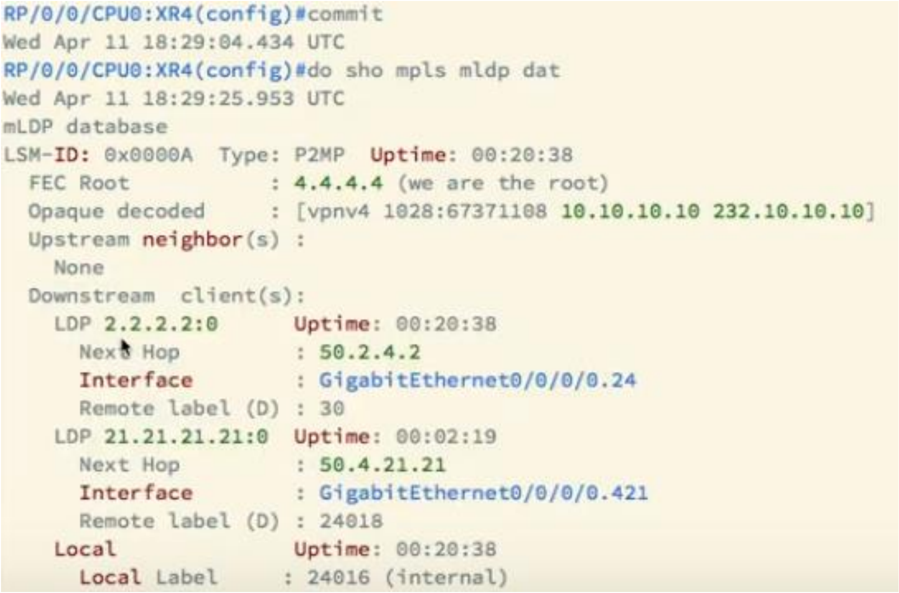
```
RP/0/0/CPU0:XR1(config-pim-C1-ipv4)#do sho mpls mldp dat
Wed Apr 11 18:27:12.541 UTC
mLDP database
LSM-ID: 0x00010  Type: P2MP  Uptime: 00:00:06
  FEC Root              : 4.4.4.4
  Opaque decoded        : [vpnv4 1028:67371108 10.10.10.10 232.10.10.10]
  Upstream neighbor(s) :
    4.4.4.4:0 [Active] Uptime: 00:00:06
      Local Label (D) : 24018
  Downstream  client(s):
    PIM MDT             Uptime: 00:00:06
      Egress intf       : ImdtC1
      Table ID          : IPv4: 0xe0000011
      RPF ID            : 3
      RD                : 1028:67371108
RP/0/0/CPU0:XR1(config-pim-C1-ipv4)#
```

In case on the remote PE there is a **NULL RPF** interface entry, remember to configure the core tree route-policy:



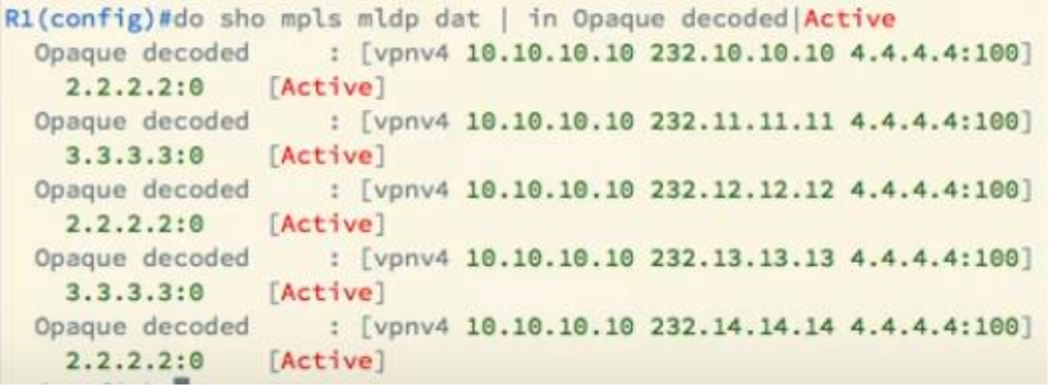
```
RP/0/0/CPU0:XR1(config)#router pim vrf C1
RP/0/0/CPU0:XR1(config-pim-C1)#address-family ipv4
RP/0/0/CPU0:XR1(config-pim-C1-ipv4)#rpf topology ?
  route-policy  Route policy to select RPF topology
RP/0/0/CPU0:XR1(config-pim-C1-ipv4)#rpf topology route-policy ?
  RP_RPF_C1   Name of the policy
  WORD        Name of the policy
RP/0/0/CPU0:XR1(config-pim-C1-ipv4)#rpf topology route-policy RP_RPF_C1
```

On R4 traffic gets ‘copied’ toward two nodes now, both R2 and XR1:



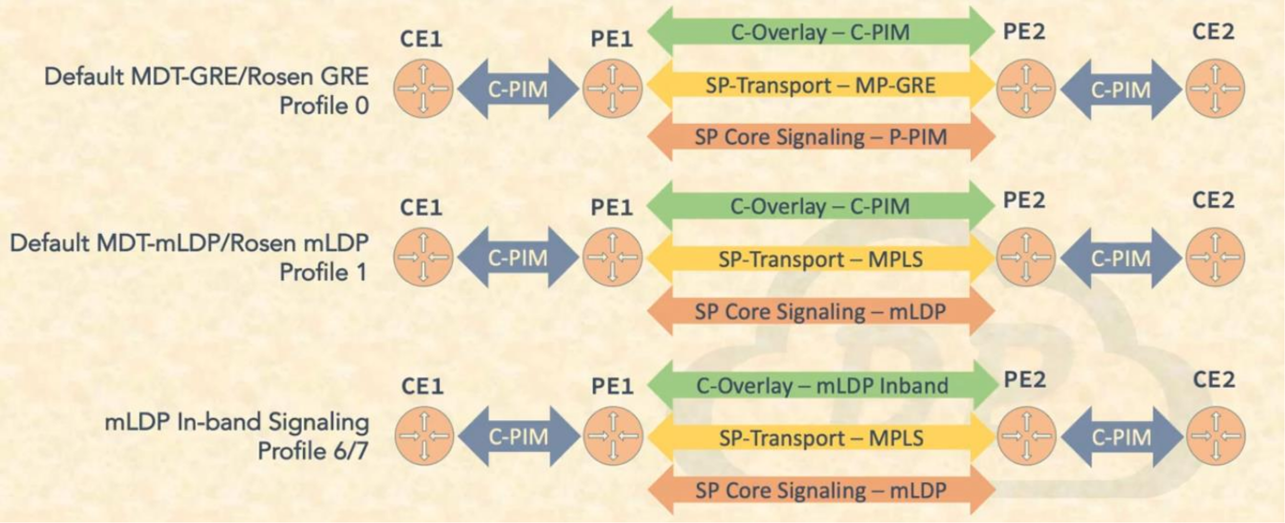
```
RP/0/0/CPU0:XR4(config)#commit
Wed Apr 11 18:29:04.434 UTC
RP/0/0/CPU0:XR4(config)#do sho mpls mldp dat
Wed Apr 11 18:29:25.953 UTC
mLDP database
LSM-ID: 0x0000A  Type: P2MP  Uptime: 00:20:38
  FEC Root              : 4.4.4.4 (we are the root)
  Opaque decoded        : [vpnv4 1028:67371108 10.10.10.10 232.10.10.10]
  Upstream neighbor(s) :
    None
  Downstream  client(s):
    LDP 2.2.2.2:0       Uptime: 00:20:38
      Next Hop          : 50.2.4.2
      Interface         : GigabitEthernet0/0/0/0.24
      Remote label (D) : 30
    LDP 21.21.21.21:0  Uptime: 00:02:19
      Next Hop          : 50.4.21.21
      Interface         : GigabitEthernet0/0/0/0.421
      Remote label (D) : 24018
    Local               Uptime: 00:20:38
      Local Label      : 24016 (internal)
```

For every single (S, G) group there will be a signaled LSP. In case multipath is available, LSPs should also be balanced in the core. This can potentially pose scaling problems in case there are MANY different sources in customer’s networks, on P/PE nodes of the service provider.



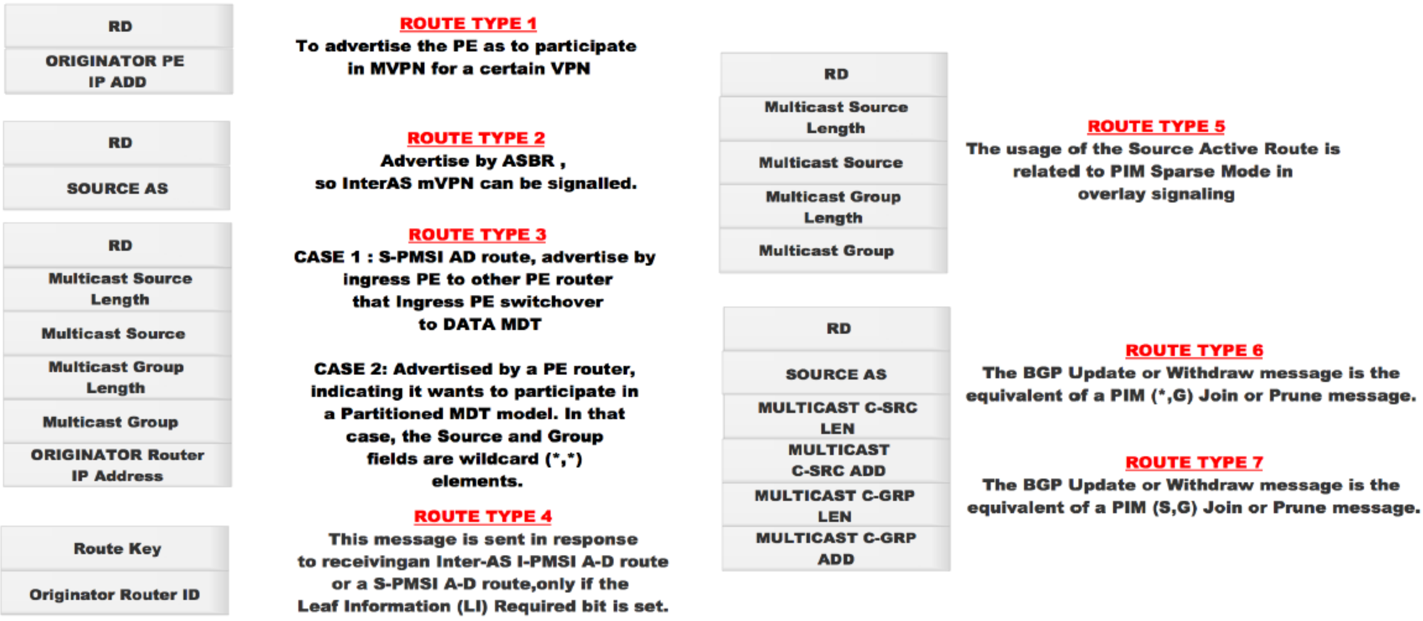
```
R1(config)#do sho mpls mldp dat | in Opaque decoded|Active
  Opaque decoded       : [vpnv4 10.10.10.10 232.10.10.10 4.4.4.4:100]
    2.2.2.2:0     [Active]
  Opaque decoded       : [vpnv4 10.10.10.10 232.11.11.11 4.4.4.4:100]
    3.3.3.3:0     [Active]
  Opaque decoded       : [vpnv4 10.10.10.10 232.12.12.12 4.4.4.4:100]
    2.2.2.2:0     [Active]
  Opaque decoded       : [vpnv4 10.10.10.10 232.13.13.13 4.4.4.4:100]
    3.3.3.3:0     [Active]
  Opaque decoded       : [vpnv4 10.10.10.10 232.14.14.14 4.4.4.4:100]
    2.2.2.2:0     [Active]
```

Summary for profiles 0, 1 and 6/7



BGP C-Mcast Signaling

With this configuration all customer's PIM messages are translated into bgp messages and sent to all the other PEs. Many different route types have been defined and are effectively used for this purpose.



RD
ORIGINATOR PE IP ADD

ROUTE TYPE 1
To advertise the PE as to participate in MVPN for a certain VPN

RD
SOURCE AS

ROUTE TYPE 2
Advertise by ASBR , so InterAS mVPN can be signalled.

RD
Multicast Source Length
Multicast Source
Multicast Group Length
Multicast Group
ORIGINATOR Router IP Address

ROUTE TYPE 3
CASE 1 : S-PMSI AD route, advertise by ingress PE to other PE router that Ingress PE switchover to DATA MDT

CASE 2: Advertised by a PE router, indicating it wants to participate in a Partitioned MDT model. In that case, the Source and Group fields are wildcard (*,*) elements.

Route Key
Originator Router ID

ROUTE TYPE 4
This message is sent in response to receivingan Inter-AS I-PMSI A-D route or a S-PMSI A-D route,only if the Leaf Information (LI) Required bit is set.

RD
Multicast Source Length
Multicast Source
Multicast Group Length
Multicast Group

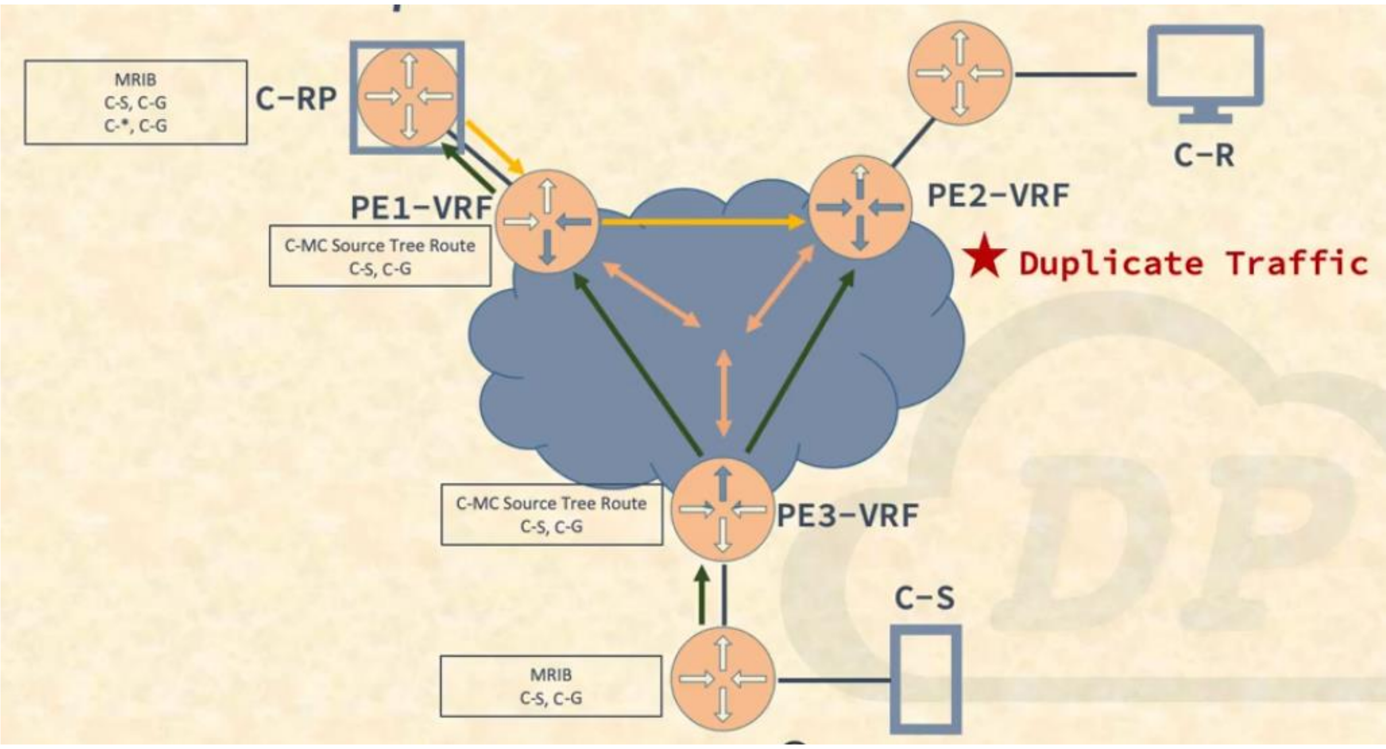
ROUTE TYPE 5
The usage of the Source Active Route is related to PIM Sparse Mode in overlay signaling

RD
SOURCE AS
MULTICAST C-SRC LEN
MULTICAST C-SRC ADD
MULTICAST C-GRP LEN
MULTICAST C-GRP ADD

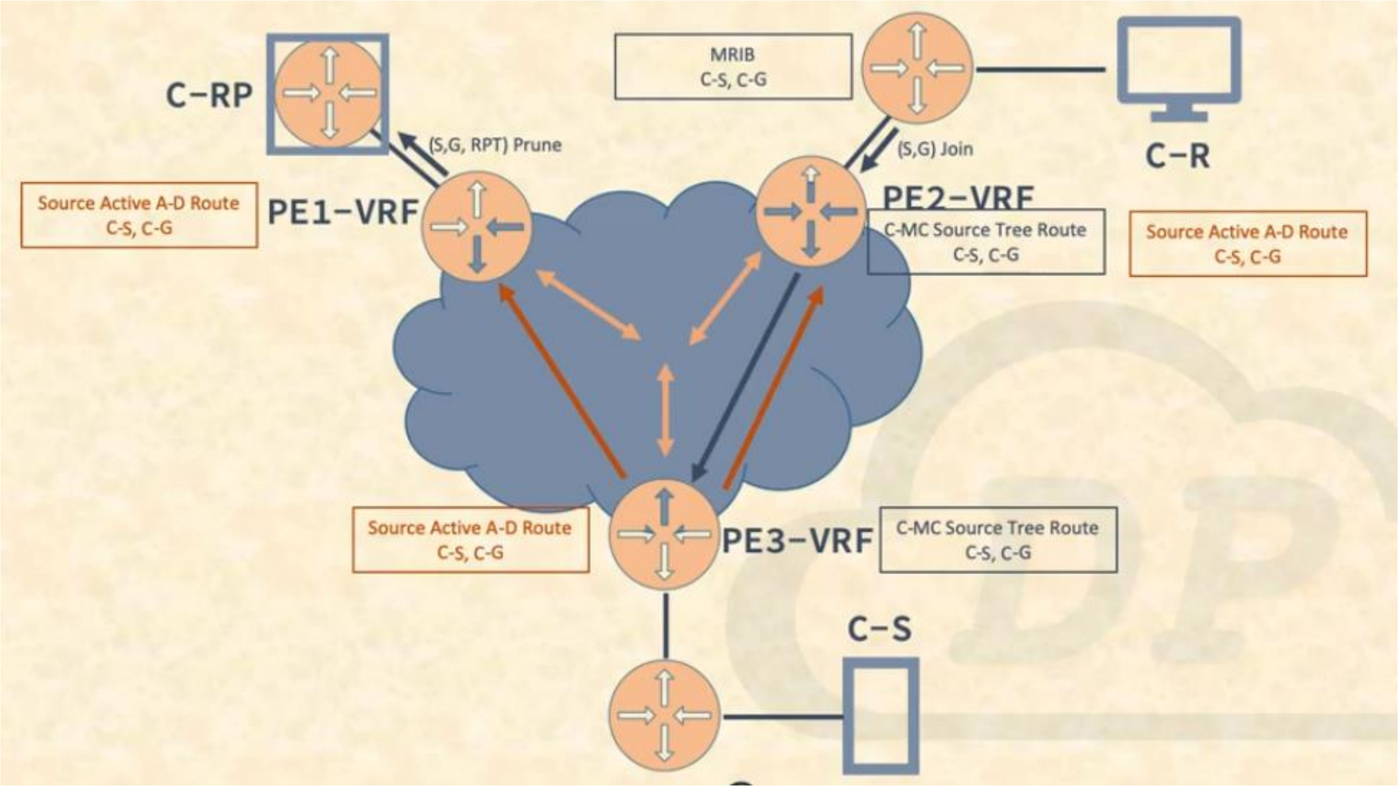
ROUTE TYPE 6
The BGP Update or Withdraw message is the equivalent of a PIM (*,G) Join or Prune message.

ROUTE TYPE 7
The BGP Update or Withdraw message is the equivalent of a PIM (S,G) Join or Prune message.

For example one very peculiar and specific case that has been addressed and solved, is the “source active auto discovery” message, as depicted in the following picture. In case the RP is on a third site respect to the source and the receiver, there is traffic duplication due to mpls provider. This scenario happens even in case traffic switches to the data tree in the core, because it could take time before the customers nodes switch to the SPT.



To avoid this, a source active for multicast group G is shared between all PEs through bgp, and the PE in front of the RP sends an (S, G, RPT) Prune message.



PIM SM as customer multicast protocol

PIM-SM process is more complicated compared to PIM-SSM due to the Source and Receivers initially meeting at the RP and then switching over to SPT. **The placement of C-RP is always very important**. If the C-Receivers and C-RP are at different customer sites then things can get a little bit more complicated. So there are two ways to handle PIM-SM:

a) PE has the knowledge about the active C-Sources: There are two ways a PE can know about C-Sources:

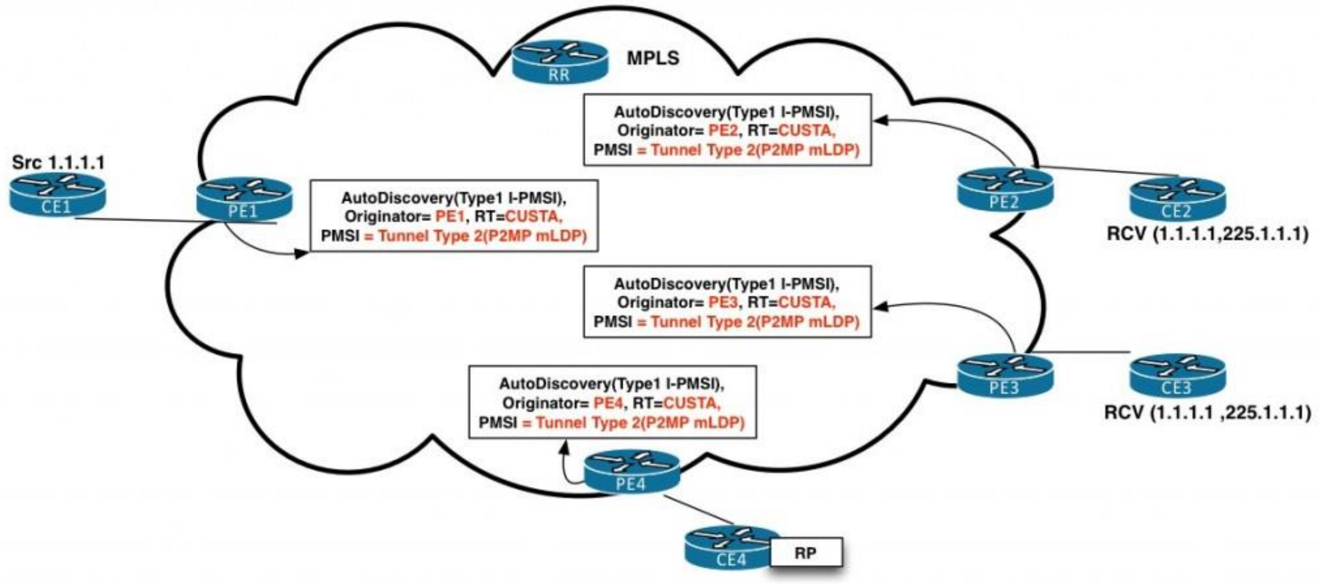
- One of the PE routers acts as a fully functional customer RP(C-RP) for that MVPN.
- An MSDP session established between the PE router and the customer RP(C-RP) to convey information about multicast sources.

When a PE using any of the above methods learns of a new multicast source within that MVPN, it constructs a **Source Active A-D route**. This route is sent to all other PEs that have one or more sites of that MVPN connected to them. The downstream PEs with (*, C-G) receivers construct (C-S, C-G) Source Tree Join routes towards the upstream PE. From a Provider perspective this approach is simple, but from a Customer perspective there is a problem with both approaches. i.e. – **SP is interfering with the Customer domain**.

b) Signaling (*, C-G) join state via BGP: If a PE has downstream (*, C-G) receivers, it will send a Shared Tree Join route to the upstream PE en route to the C-RP. We will cover option b as it's more complicated and probably more interesting. In Fig.7, C-Source is located behind PE1 and C-RP is located at a different site then source. C-Receivers are behind PE2 and PE3.

Auto-Discovery and MI-PMSI P-Tunnel:

Auto-Discovery and MI-PMSI setup process will be similar to what we saw for PIM-SSM. After this step every PE would know about other relevant PEs and the MI-PMSI (Default MDT) tunnel will be setup. At this point any C-Multicast control plane traffic like RP advertisements (Auto-RP, BSR) will flow over MI-PMSI. This will allow PEs to learn about the RP.


MPLS
RR
AutoDiscovery(Type1 I-PMSI), Originator= PE2, RT=CUSTA, PMSI = Tunnel Type 2(P2MP mLDP)
Src 1.1.1.1
CE1
PE1
AutoDiscovery(Type1 I-PMSI), Originator= PE1, RT=CUSTA, PMSI = Tunnel Type 2(P2MP mLDP)
PE2
CE2
RCV (1.1.1.1,225.1.1.1)
AutoDiscovery(Type1 I-PMSI), Originator= PE3, RT=CUSTA, PMSI = Tunnel Type 2(P2MP mLDP)
PE3
CE3
RCV (1.1.1.1 ,225.1.1.1)
AutoDiscovery(Type1 I-PMSI), Originator= PE4, RT=CUSTA, PMSI = Tunnel Type 2(P2MP mLDP)
PE4
CE4
RP

C-MCAST Signaling

When the PEs receives a (C-*, C-G) join, they generate a Shared tree join BGP route towards PE where the RP is located. In Fig.8, when PE2 and PE3 receive a PIM join for (* ,225.1.1.1) they perform a lookup for the RP address which is behind PE4. PE2 and PE3 generate a BGP shared tree join (Route Type 6) with the Route Target set to PE4. PE4 receives the route and forwards the PIM Join (* , 225.1.1.1) to RP. At this point RP is aware that there are interested receivers behind PE4.

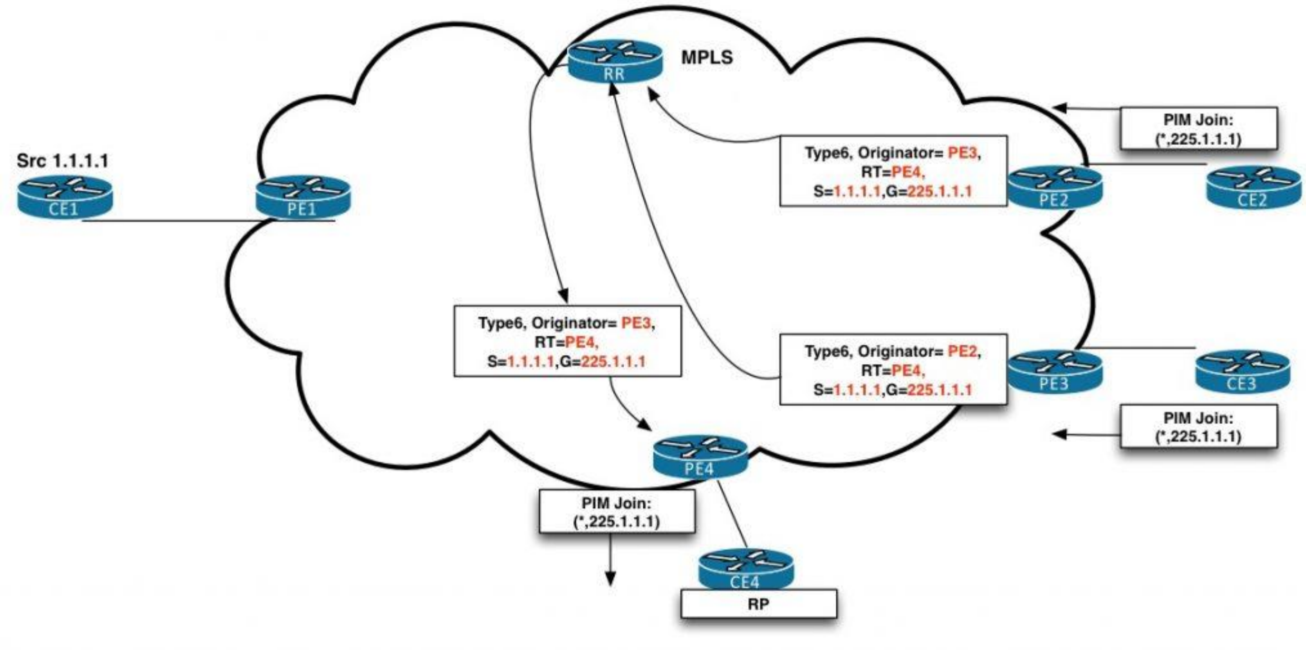


Fig.8

In Fig.9, let's examine the steps involved once the source begins forwarding multicast traffic.

(1) When source (1.1.1.1) starts sending traffic to G=225.1.1.1,
(2) The first hop router CE1 informs the RP of the source's existence by sending a (C-S, C-G) register to the RP. Once the RP receives the PIM Register message, it generates a
(3)PIM Join (1.1.1.1, 225.1.1.1) towards CE1.
(4)PE4 generates Source Tree Join BGP route towards PE1 after receiving the PIM join from RP.
(5)When PE1 receives source tree join route it sends a PIM Join (1.1.1.1, 225.1.1.1) to CE1.
(6)PE1 also generates a Source Active A-D route as a result of receiving a Source Tree Join C-multicast route from PE4 for (1.1.1.1, 225.1.1.1) which is propagated to all the PEs of the MVPN. Type 5 BGP Source Active A-D routes are only applicable to PIM-SM. A PE will only generate a Source Active Route when it (PE1) creates a (C-S,C-G) state as a result of receiving a C-multicast route for (C-S,C-G) from some other PE (PE4). It also requires that the C-G(225.1.1.1) group is an ASM group, and the PE(PE1) that creates the state MUST originate a Source Active A-D route.

In Fig.10, PE2 and PE3 generate a Source Tree Join (1.1.1.1, 225.1.1.1) targeted towards PE1 in response to the Type 5 Source Active A-D route. PE2 and PE3 generate the Source Tree Joins because they see a "match" to the Source Active A-D route. A match is considered when the (C-S, C-G) Source Active A-D route matches a given (C-*,C-G) entry, and if the C-G is same and the PE has originated a Shared Tree Join C-multicast route for the same C-G. At this point PE2 and PE3 will start receiving traffic directly from PE1.

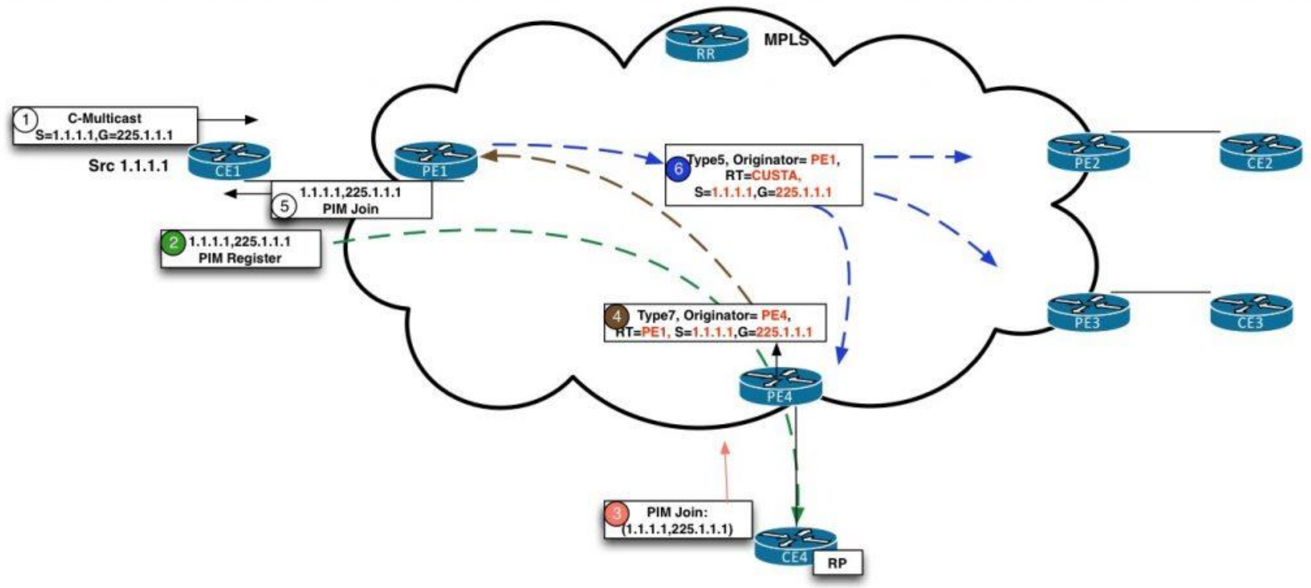

MPLS
RR
1 C-Multicast
S=1.1.1.1,G=225.1.1.1
Src 1.1.1.1
CE1
PE1
1.1.1.1,225.1.1.1
5 PIM Join
2 1.1.1.1,225.1.1.1
PIM Register
6 Type5, Originator= PE1,
RT=CUSTA,
S=1.1.1.1,G=225.1.1.1
PE2
CE2
PE3
CE3
4 Type7, Originator= PE4,
RT=PE1, S=1.1.1.1,G=225.1.1.1
PE4
3 PIM Join:
(1.1.1.1,225.1.1.1)
CE4
RP

Fig.9

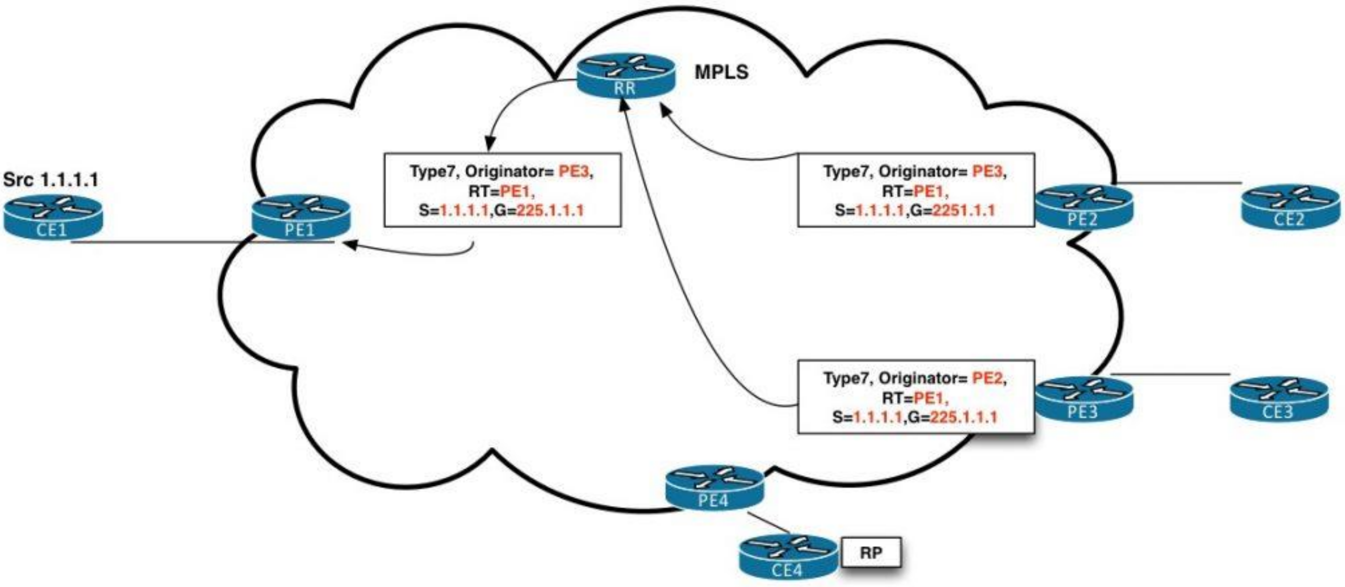

RR
MPLS
Src 1.1.1.1
CE1
PE1
Type7, Originator= PE3,
RT=PE1,
S=1.1.1.1,G=225.1.1.1
Type7, Originator= PE3,
RT=PE1,
S=1.1.1.1,G=2251.1.1
PE2
CE2
Type7, Originator= PE2,
RT=PE1,
S=1.1.1.1,G=225.1.1.1
PE3
CE3
PE4
CE4
RP

Fig.10

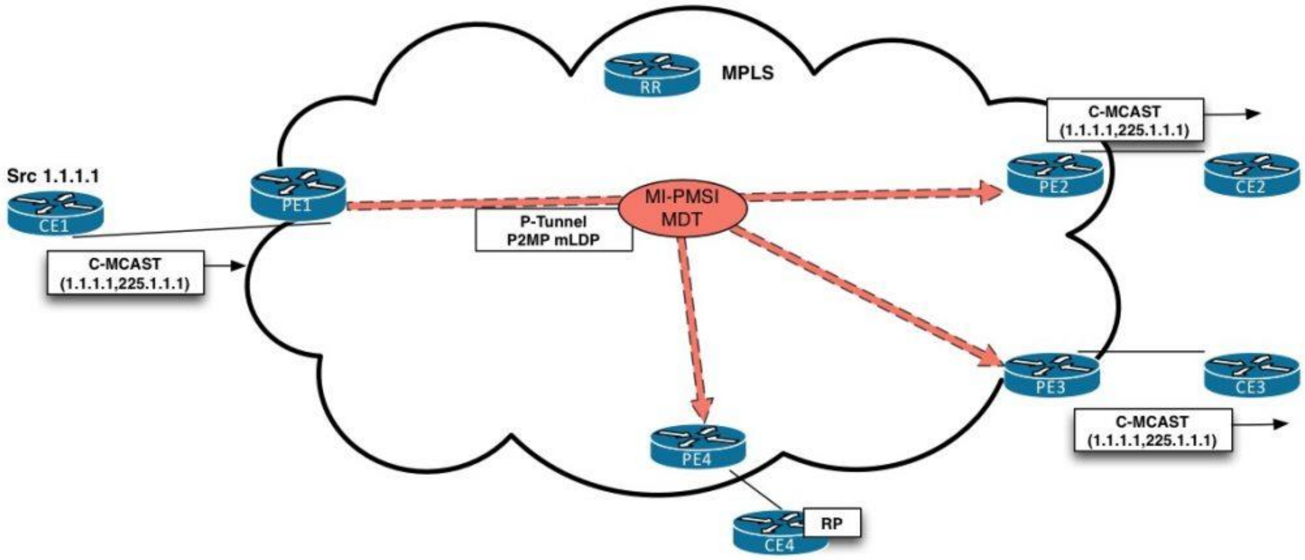
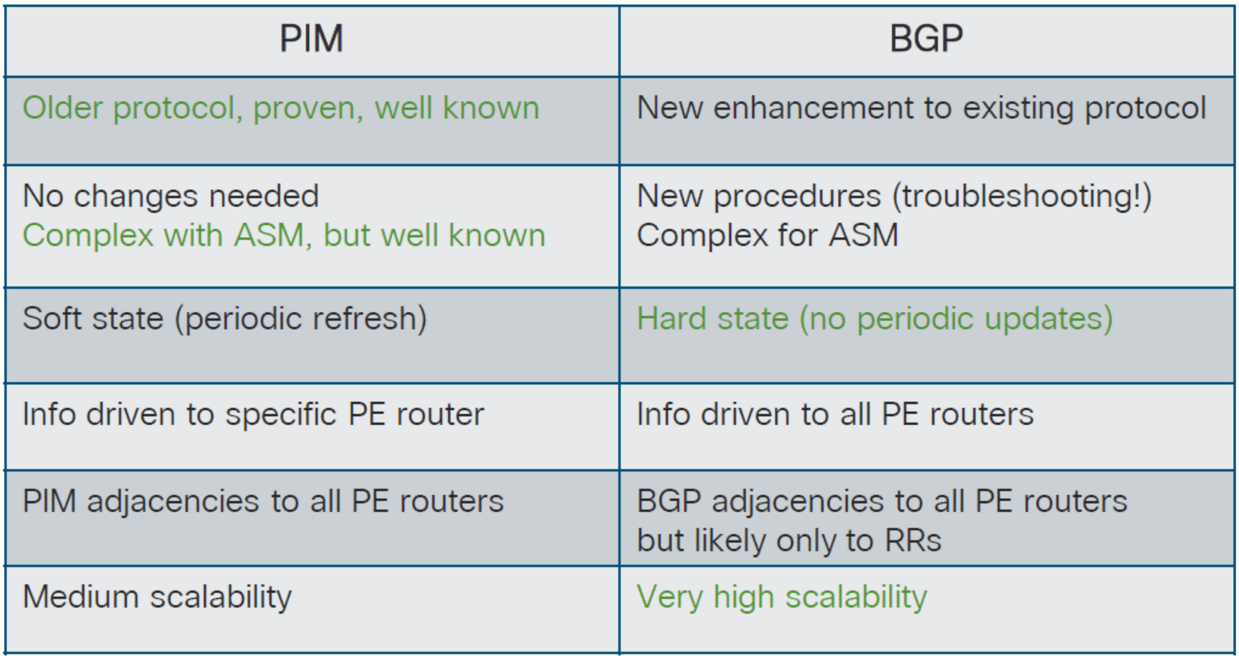


Fig.11

As you can see, the procedure for handling PIM-SM is definitely more complicated compared to PIM-SSM. Note that there may be a possibility to have (C-S,C-G) packets being sent at the same time on the PMSI by both the local PE connected to the Source and the PE connected to the site that contains C-RP. This would result in transient unnecessary traffic on the provider backbone. However, **no duplicates will reach the customer hosts subscribed to C-G as the downstream PEs will drop the packet**.

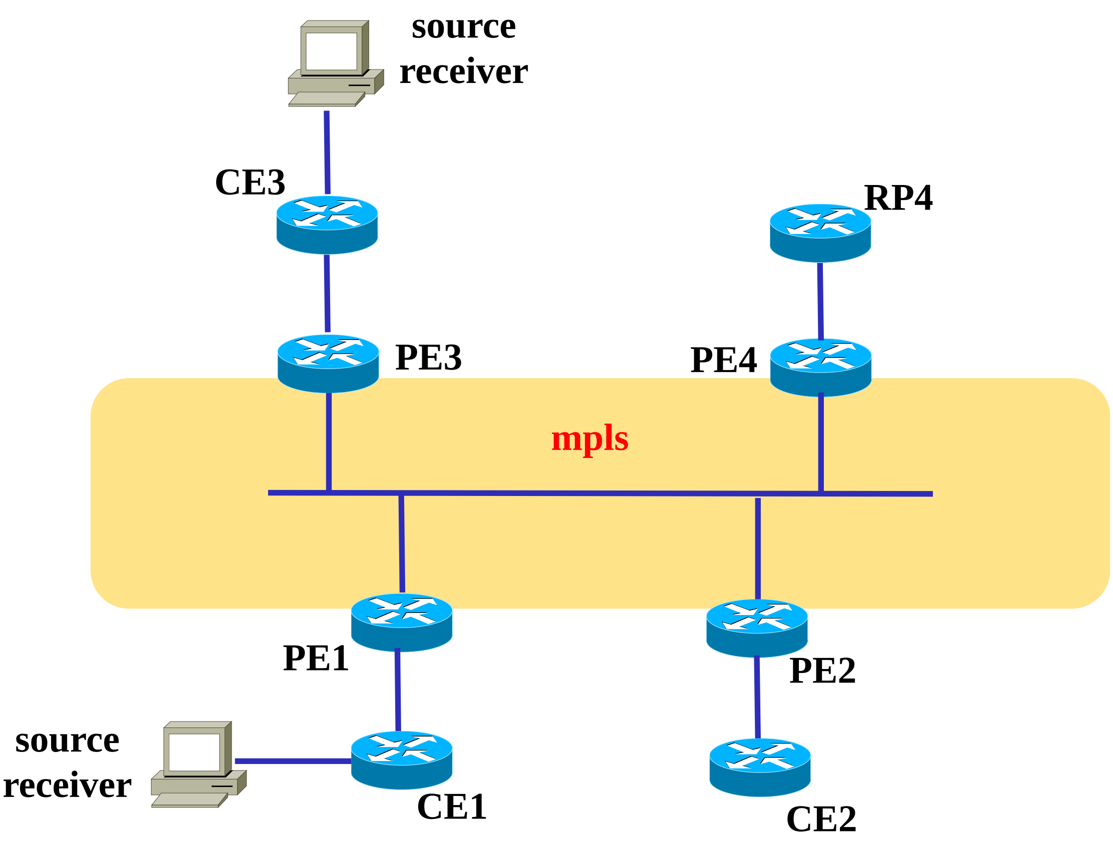
PIM vs BGP signaling



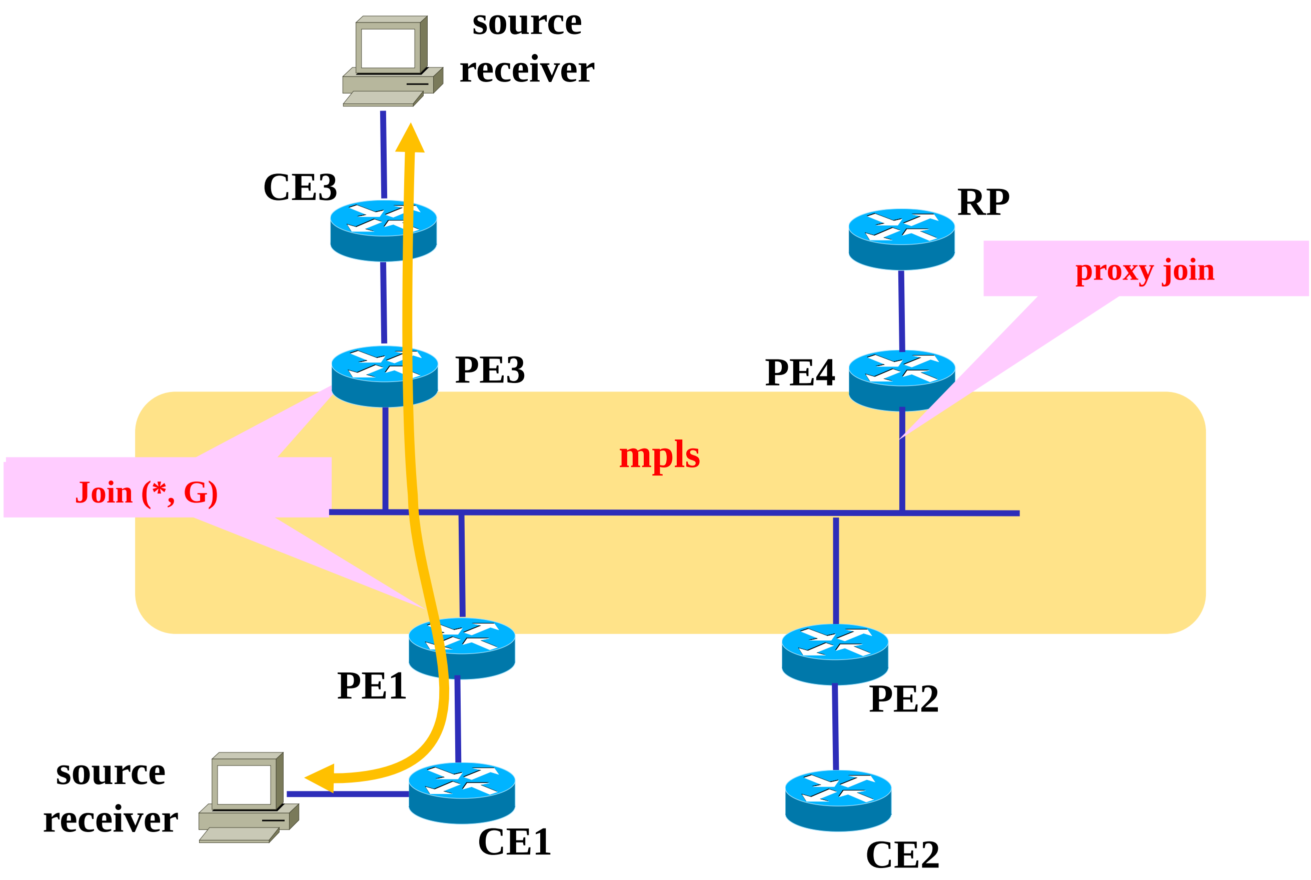
PIM	BGP
Older protocol, proven, well known	New enhancement to existing protocol
No changes needed Complex with ASM, but well known	New procedures (troubleshooting!) Complex for ASM
Soft state (periodic refresh)	Hard state (no periodic updates)
Info driven to specific PE router	Info driven to all PE routers
PIM adjacencies to all PE routers	BGP adjacencies to all PE routers but likely only to RRs
Medium scalability	Very high scalability

Customer BIDIR-PIM

Probably something similar happens in case **BIDIR-PIM** is used by the customer: in that case an RP is something mandatory. But it doesn't make ANY sense to make the traffic **necessarily** flow through it, when there is an mpls network in between. More specifically, when a (*,G) pim join is received by a PE, he knows there is a source/receiver behind it. A 'source active' (*, G) message is sent toward all the other PEs, so that they can also join the mdt in case there are sources/receivers behind them, and traffic doesn't flow through the RP, wherever it is: worst case would be a site where you have just the RP but no source/receivers, the PE would just send a join toward the RP, but data traffic would NOT flow through it.

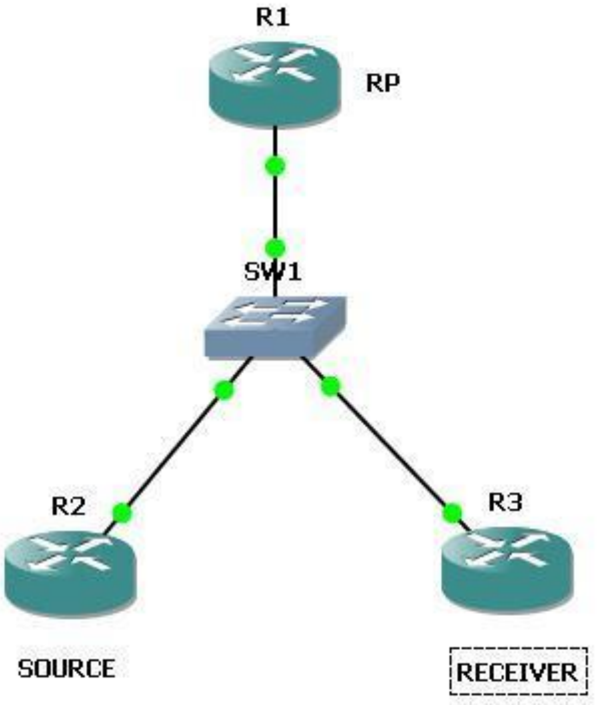


You are in the above situation in which there is a shared 'logical' LAN in the mpls backbone. Until there are sources behind the PEs, they will continue sending (*, G) join pim packets in the LAN, PE2 will not send any join, while PE4 will send 'proxy join' (unless there are source also behind the RP).



```
PE3#show ip mroute vrf one 225.1.1.1
IP Multicast Routing Table
Flags: D -Dense, S -Sparse, B -Bidir Group, s -SSM Group, C -Connected,
…
X -Proxy Join Timer Running, A -Candidate for MSDP Advertisement,
U -URD, I -Received Source Specific Host Report,
…
…
 (*, 225.1.1.1), 00:32:14/stopped, RP 10.100.1.7, flags: SG
Incoming interface: Ethernet0/0, RPF nbr 10.2.3.7
Outgoing interface list:
Lspvif1, Forward/Sparse, 00:29:41/stopped          ← mpls core interface
(10.2.2.6, 225.1.1.1), 00:10:36/00:00:09, flags: PTXg
Incoming interface: Lspvif1, RPF nbr 10.100.1.2
Outgoing interface list: Null
```

Very often without knowing it in PIM sparse networks we encounter the situation that the RP of a network is a RP on a stick. Take a look at the drawing to get a better understanding of the situation here.



Problem here is that the incoming traffic of the source-tree is going out the same interface down the shared tree. The PIM SM rule, that says “incoming and outgoing interface for a specific stream may not be the same due to loop prevention”, prevents the usage of this. Let’s take a look what happens.

R2 registers the source in the RP with the PIM register message:

```
R2#ping 239.100.100.100 rep 100 so fa0/0

Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 239.100.100.100, timeout is 2 seconds:
Packet sent with a source address of 172.16.0.2
….
```

```
R1#sh ip mroute 239.100.100.100
IP Multicast Routing Table
…
(*, 239.100.100.100), 00:02:31/stopped, RP 172.16.0.1, flags: SP      ← proxy join
Incoming interface: Null, RPF nbr 0.0.0.0
Outgoing interface list: Null

(172.16.0.2, 239.100.100.100), 00:02:31/00:02:51, flags: PT           ← proxy join
Incoming interface: FastEthernet0/0, RPF nbr 172.16.0.2
Outgoing interface list: Null
```

R3 joins the shared tree by becoming a receiver in the group 239.100.100.100. But we cannot see any change in the mroute table of the RP:

```
R1#sh ip mroute 239.100.100.100
…
```

```
(*, 239.100.100.100), 00:05:04/stopped, RP 172.16.0.1, flags: SP
Incoming interface: Null, RPF nbr 0.0.0.0
Outgoing interface list: Null

(172.16.0.2, 239.100.100.100), 00:05:04/00:02:58, flags: PT
Incoming interface: FastEthernet0/0, RPF nbr 172.16.0.2
Outgoing interface list: Null
```

Logically for a short period of time the mroute output should look like this:

```
R1#sh ip mroute 239.100.100.100
…
(*, 239.100.100.100), 00:05:04/stopped, RP 172.16.0.1, flags: SP
Incoming interface: Null, RPF nbr 0.0.0.0
Outgoing interface list: Null

(172.16.0.2, 239.100.100.100), 00:05:04/00:02:58, flags: PT
Incoming interface: FastEthernet0/0, RPF nbr 172.16.0.2
Outgoing interface list: FastEthernet0/0
```

This situation will never occur because the router internally knows whats going on here. The traffic succeeds.

```
R2#ping 239.100.100.100 rep 1000 so fa0/0

Type escape sequence to abort.
Sending 1000, 100-byte ICMP Echos to 239.100.100.100, timeout is 2 seconds:
Packet sent with a source address of 172.16.0.2
…………………………………………………….
Reply to request 567 from 172.16.0.3, 68 ms
Reply to request 568 from 172.16.0.3, 72 ms
Reply to request 569 from 172.16.0.3, 32 ms
Reply to request 570 from 172.16.0.3, 92 ms
Reply to request 571 from 172.16.0.3, 128 ms
Reply to request 572 from 172.16.0.3, 116 ms
Reply to request 573 from 172.16.0.3, 40 ms
Reply to request 574 from 172.16.0.3, 64 ms
Reply to request 575 from 172.16.0.3, 56 ms
Reply to request 576 from 172.16.0.3, 72 ms
```

The RP does not delete the entry. It sends periodic S,G join (instead of prunes, which is the normal behaviour in a case where the incoming and outgoing interfaces are equal) messages to keep the stream alive, although the OIL is “null”. **This feature is called proxy-join timer**.
Once the RP receives a packet that matches the proxy join timer rules, the timer is started and the router recognizes in which position he is and that he does not have to drop the multicast traffic. So multicast forwarding can occur.
The proxy join timer feature I will more dig into in a different post where I will take a deeper look into one legged rendezvous-points that have a turnaround router in front of them.

Assert multicast mechanism

A pretty important topic that is very easy to overlook when studying multicast is the **PIM Assert Mechanism**. In Figure 1, R1 and R4 have a route to the source 150.1.5.5 (the multicast source), and share a multi-access connection to R6. R6's FastEthernet0/0 interface has joined the multicast group 239.6.6.6. Both R1 and R4 are receiving copies of the same multicast packets from the source (illustrated by the yellow arrows), but it's not very efficient for both routers to forward the packets onto the same network segment. This would result in duplicate traffic and a waste of bandwidth and processing power.

To stop this duplication of shared traffic, PIM routers connected to a shared segment will elect a single forwarder for that particular segment. Since PIM does not have its own routing protocol that could be used to determine the best path to send data across, it relies on a special process called the PIM Assert Mechanism to make this determination.

This mechanism tells a router that when it receives a multicast packet from a particular source on an interface that is already listed in its own Outbound Interface List (OIL) for the same (S,G) pair, that it needs to send an Assert Message. Assert Messages contain the metric of the unicast route to the source in question, the Administrative Distance of the protocol that discovered the route, the multicast source and the group itself, and are used to elect what is called the PIM Forwarder.

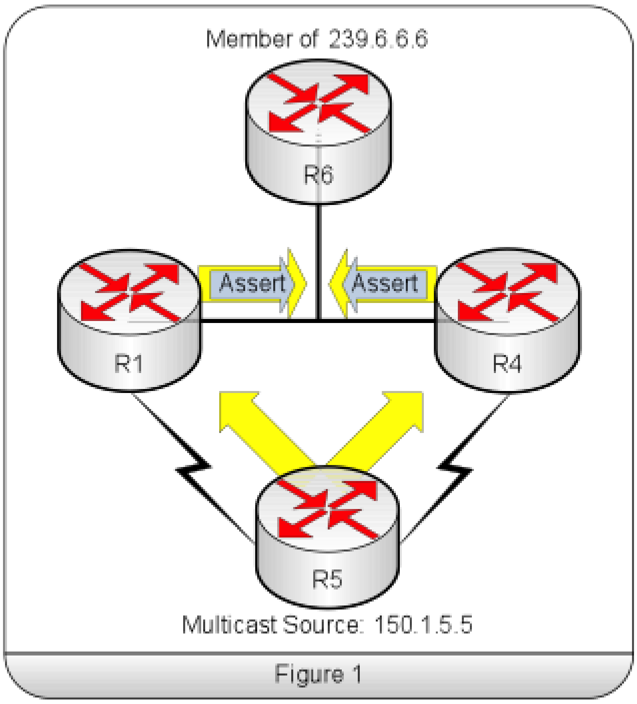


Figure 1

In the scenario in figure 1, both R1 and R4 will send the same multicast stream to R6. This means they will put their VLAN 146 interfaces into the OIL for the (S,G) pair (150.1.5.5, 239.6.6.6) and because this is a LAN segment each device will see each the others stream. This condition, each router producing duplicate packets on the segment, will trigger the Assert Mechanism.

These **Assert Messages are used to elect the PIM forwarder** using the following three rules:

1. The router generating an Assert with the lowest Administrative distance is elected the forwarder. The AD would only differ if the routes to R5 where from different routing protocols. If the Administrative Distances are the same then we move to step 2.
2. The best unicast routing metric is used to break a tie if the Administrative Distances are the same. The combination of AD and the unicast routing metric is referred to as a "tuple". If metrics are the same them we move on to step 3.
3. The device with the highest IP Address will be elected as the PIM Forwarder.

When a device is elected to be the PIM Forwarder it will continue to send the multicast stream while the other device stops forwarding that group's traffic. Furthermore, the “Assert Loser” will prune its physical interface connected to the shared media.

Using the following show commands we can see the outcome of the election. R4 shows that it won the election by displaying the “A” associated with the interface that is forwarding multicasts, and R1 prunes its interface exactly as we discussed.

```
R4#show ip mroute 239.6.6.6
<output omitted for clarity>
(150.1.5.5, 239.6.6.6), 00:01:04/00:01:12, flags: T
Incoming interface: Serial0/1/0, RPF nbr 155.1.45.5
Outgoing interface list
FastEthernet0/0, Forward/Sparse-Dense, 00:00:39/00:00:00, A      ← Assert, Forward

R1#show ip mroute 239.6.6.6
<output omitted for clarity>
(150.1.5.5, 239.6.6.6), 00:01:15/00:01:24, flags: PT
Incoming interface: Serial0/0.1, RPF nbr 155.1.0.5
Outgoing interface list:
FastEthernet0/0, Prune/Sparse-Dense, 00:01:27/00:01:32     ← Pruned interface
```

Do not confuse the PIM forwarder with the Designated Router (the latter being the elected router in a LAN to forward traffic to local LAN receivers), the PIM forwarder’s job is simply to forward multicast traffic onto a shared segment. We will cover the Designated Router in another blog post.

What is PIM assert mechanism ? by Cisco TAC

When there are multiple PIM enabled routers on a shared segment, it is possible that these routers encounter duplicate multicast traffic. This might be the case because two or more routers on the same shared segment might have a valid (S,G) or (*,G) entry which populates the outgoing interface towards the shared segment for the same source IP/destination group. PIM assert mechanism is used to detect and eliminate duplication of multicast traffic on a shared segment. It is important to note that this mechanism does not prevent duplication of occurring, instead it uses duplication of multicast traffic as a trigger in order to activate this mechanism which elects a single forwarder for this stream. When you have duplication of multicast traffic on a shared segment, you can assume that there are multiple routers that send the same (S,G) or (*,G) on a shared segment. If you elect one router to forward this stream effectively, it eliminates duplication.

PIM leverages on PIM assert messages which are triggered when you receive a multicast packet on the Outgoing Interface List (OIL). These assert messages contain metrics which are then used to calculate who will become assert winner. Downstream routers on the LAN also receive PIM assert messages. These messages are then used by downstream devices to send appropriate Join/Prune messages to the upstream router that won the assert ellection.

Scenario 1. LHR Rationale

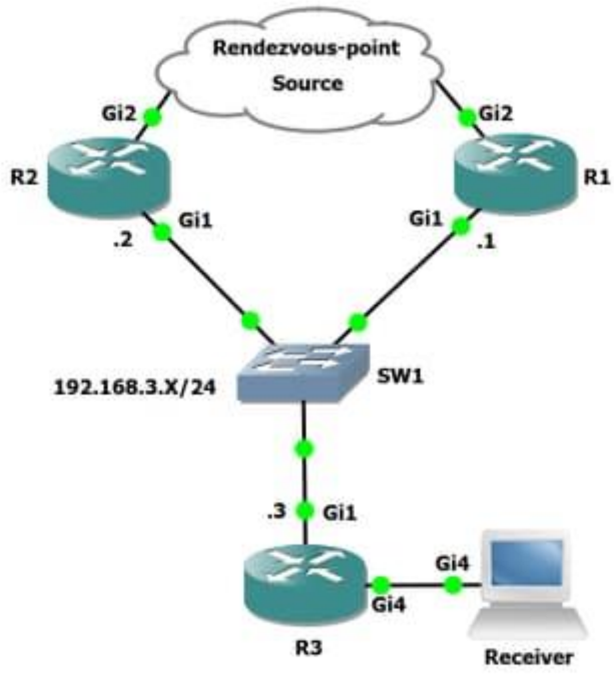


Figure 1.

In the network diagram, R3 is the Last Hop Router (LHR), R3 connects to both R2 and R1 via a shared segment.

When you receive an Internet Group Management Protocol (IGMP) report from the receiver, R3 checks who the RPF neighbor towards the RP is. In the topology, R1 is the RPF neighbor towards the RP, hence R3 sends a (*,G) join towards R1. Once R1 pulls down the stream (assume the group is active) R3 sends an (S,G) join towards the source and pulls the source tree down. R2 is the RPF neighbor towards the source tree which means R3 will send the (S,G) join towards R2. R3 has the same RPF interface towards both RP and the source. Here you can see the R3 mroute table for group 239.1.1.1.

```
R3#show ip mroute
IP Multicast Routing Table
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode
(*, 239.1.1.1), 00:00:55/stopped, RP 192.168.0.100, flags: SJC
  Incoming interface: GigabitEthernet1, RPF nbr 192.168.3.1
  Outgoing interface list:
    GigabitEthernet4, Forward/Sparse, 00:00:55/00:02:04

(10.0.0.2, 239.1.1.1), 00:00:52/00:02:07, flags: JT
  Incoming interface: GigabitEthernet1, RPF nbr 192.168.3.2, Mroute
  Outgoing interface list:
    GigabitEthernet4, Forward/Sparse, 00:00:52/00:02:07

(*, 224.0.1.40), 00:01:22/00:02:09, RP 192.168.0.100, flags: SJPCL
  Incoming interface: GigabitEthernet1, RPF nbr 192.168.3.1
```

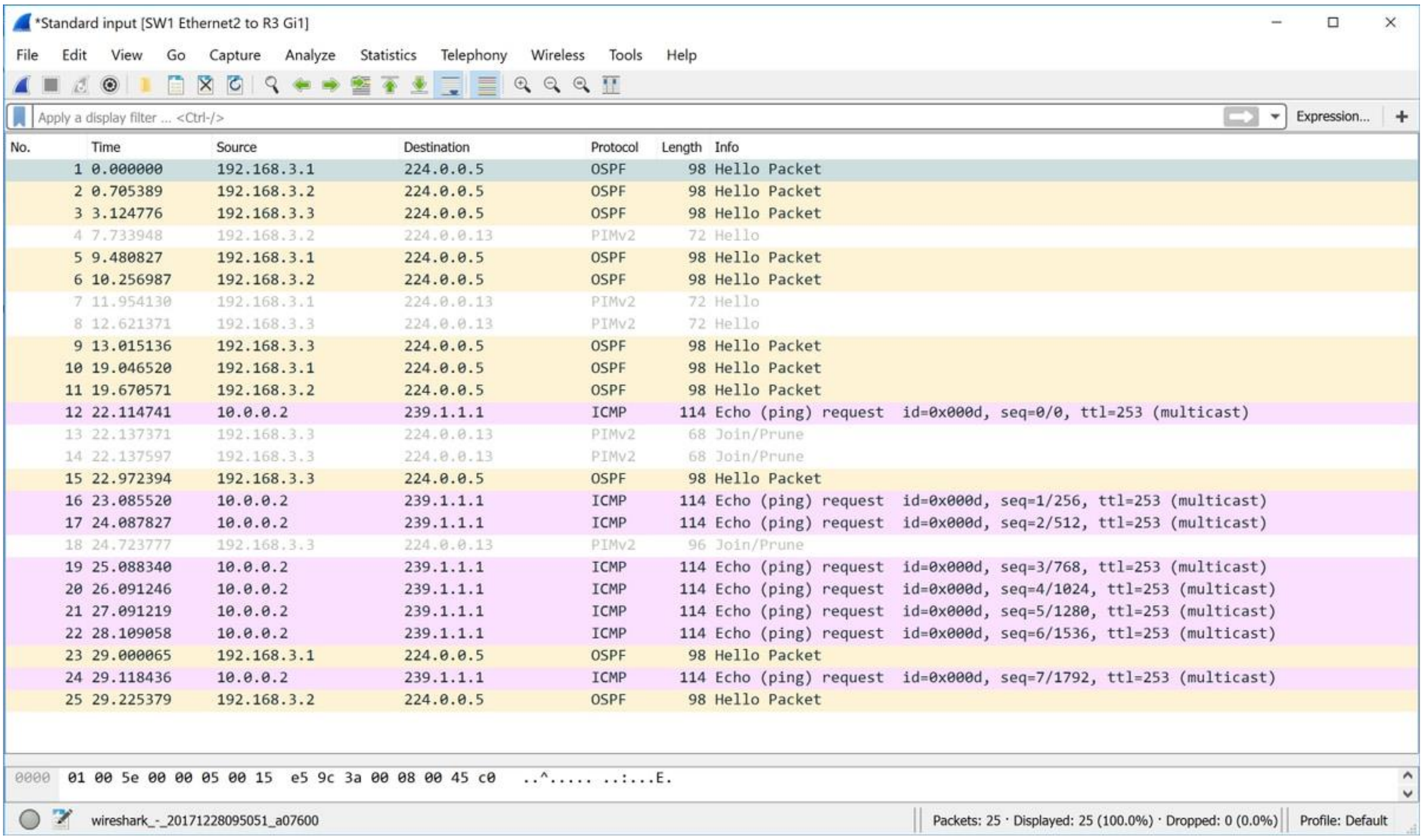
So as you can see on R3 the (*,G) RPF neighbor is 192.168.3.1 and the RPF neighbor towards the (S,G) is 192.168.3.2. Now, this should result in both R1 and R2 to have a valid OIL towards R1. Let’s have a look at these entries:

```
R1#show ip mroute
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.1.1.1), 00:15:02/00:02:33, RP 192.168.0.100, flags: S
  Incoming interface: GigabitEthernet2, RPF nbr 192.168.5.2
  Outgoing interface list:
    GigabitEthernet1, Forward/Sparse, 00:15:02/00:02:33

(10.0.0.2, 239.1.1.1), 00:13:24/00:02:33, flags: PR
  Incoming interface: GigabitEthernet2, RPF nbr 192.168.5.2
  Outgoing interface list: Null

(*, 224.0.1.40), 00:29:17/00:02:51, RP 192.168.0.100, flags: SJCL
  Incoming interface: GigabitEthernet2, RPF nbr 192.168.5.2
  Outgoing interface list:
    GigabitEthernet1, Forward/Sparse, 00:16:06/00:02:51
  Outgoing interface list: Null
```

```
R2#show ip mroute
IP Multicast Routing Table
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.1.1.1), 00:08:00/stopped, RP 192.168.0.100, flags: SP
  Incoming interface: GigabitEthernet2, RPF nbr 192.168.4.1
  Outgoing interface list: Null

(10.0.0.2, 239.1.1.1), 00:00:03/00:02:56, flags: T
  Incoming interface: GigabitEthernet2, RPF nbr 192.168.4.1
  Outgoing interface list:
    GigabitEthernet1, Forward/Sparse, 00:00:03/00:03:26

(*, 224.0.1.40), 01:37:30/00:02:22, RP 192.168.0.100, flags: SJPL
  Incoming interface: GigabitEthernet2, RPF nbr 192.168.4.1
```

As mentioned before, assert can be triggered when there are two upstream routers which have a valid OIL populated on a shared segment. Since both R1 and R2 have a valid OIL, check if there is an assert mechanism in packet capture.

This packet capture was captured on R3 interface Gi1 towards SW1.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.3.1	224.0.0.5	OSPF	98	Hello Packet
2	0.705389	192.168.3.2	224.0.0.5	OSPF	98	Hello Packet
3	3.124776	192.168.3.3	224.0.0.5	OSPF	98	Hello Packet
4	7.733948	192.168.3.2	224.0.0.13	PIMv2	72	Hello
5	9.480827	192.168.3.1	224.0.0.5	OSPF	98	Hello Packet
6	10.256987	192.168.3.2	224.0.0.5	OSPF	98	Hello Packet
7	11.954130	192.168.3.1	224.0.0.13	PIMv2	72	Hello
8	12.621371	192.168.3.3	224.0.0.13	PIMv2	72	Hello
9	13.015136	192.168.3.3	224.0.0.5	OSPF	98	Hello Packet
10	19.046520	192.168.3.1	224.0.0.5	OSPF	98	Hello Packet
11	19.670571	192.168.3.2	224.0.0.5	OSPF	98	Hello Packet
12	22.114741	10.0.0.2	239.1.1.1	ICMP	114	Echo (ping) request id=0x000d, seq=0/0, ttl=253 (multicast)
13	22.137371	192.168.3.3	224.0.0.13	PIMv2	68	Join/Prune
14	22.137597	192.168.3.3	224.0.0.13	PIMv2	68	Join/Prune
15	22.972394	192.168.3.3	224.0.0.5	OSPF	98	Hello Packet
16	23.085520	10.0.0.2	239.1.1.1	ICMP	114	Echo (ping) request id=0x000d, seq=1/256, ttl=253 (multicast)
17	24.087827	10.0.0.2	239.1.1.1	ICMP	114	Echo (ping) request id=0x000d, seq=2/512, ttl=253 (multicast)
18	24.723777	192.168.3.3	224.0.0.13	PIMv2	96	Join/Prune
19	25.088340	10.0.0.2	239.1.1.1	ICMP	114	Echo (ping) request id=0x000d, seq=3/768, ttl=253 (multicast)
20	26.091246	10.0.0.2	239.1.1.1	ICMP	114	Echo (ping) request id=0x000d, seq=4/1024, ttl=253 (multicast)
21	27.091219	10.0.0.2	239.1.1.1	ICMP	114	Echo (ping) request id=0x000d, seq=5/1280, ttl=253 (multicast)
22	28.109058	10.0.0.2	239.1.1.1	ICMP	114	Echo (ping) request id=0x000d, seq=6/1536, ttl=253 (multicast)
23	29.000065	192.168.3.1	224.0.0.5	OSPF	98	Hello Packet
24	29.118436	10.0.0.2	239.1.1.1	ICMP	114	Echo (ping) request id=0x000d, seq=7/1792, ttl=253 (multicast)
25	29.225379	192.168.3.2	224.0.0.5	OSPF	98	Hello Packet

```
0000  01 00 5e 00 00 05 00 15  e5 9c 3a 00 08 00 45 c0   ..^..... ..:...E.
```

wireshark_-_20171228095051_a07600

Packets: 25 · Displayed: 25 (100.0%) · Dropped: 0 (0.0%)

Profile: Default

In this packet capture, you do not see any assert packets even though there are all the prerequisites to create duplication on the shared segment between R1, R2, and R3. Why do you not see any PIM assert packets when the (S,G) stream was activated?

It seems that RFC 7761 might hold the answer to these questions.

```
4.2.2.  Setting and Clearing the (S,G) SPTbit

   Basically, Update_SPTbit(S,G,iif) will set the SPTbit if we have the
   appropriate (S,G) join state, and if the packet arrived on the
   correct upstream interface for S, and if one or more of the following
   conditions apply:

   1.  The source is directly connected, in which case the switch to the
       SPT is a no-op.

   2.  The RPF interface to S is different from the RPF interface to the
       RP.  The packet arrived on RPF_interface(S), and so the SPT must
       have been completed.

   3.  No one wants the packet on the RP tree.

   4.  RPF'(S,G) == RPF'(*,G).  In this case, the router will never be
       able to tell if the SPT has been completed, so it should just
       switch immediately.  The RPF'(S,G) != NULL check ensures that the
       SPTbit is set only if the RPF neighbor towards S is valid.
```

In the case where the RPF interface is the same for the RP and for S, but RPF'(S,G) and RPF'(*,G) differ, we wait for an Assert(S,G), which indicates that the upstream router with (S,G) state believes the SPT has been completed.

The (S,G) SPTbit is used to distinguish whether to forward on (*,G) or on (S,G) state. When you switch from the RP tree to the source tree, there is a transition period when data arrives due to upstream (*,G) state while upstream (S,G) state is established, at that time, the router should continue to forward only on (*,G) state. This prevents temporary black holes that would be caused by sending a Prune(S,G,rpt) before the upstream (S,G) state has finished being established.

Although it seems that the scenario can correlate to the last point mentioned above. In the case where the RPF interface is the same for the RP and for S,
but RPF'(S,G) and RPF'(*,G) differ, we wait for an Assert(S,G), which indicates that the upstream router with (S,G) state believes the SPT has been completed.

For assert to be triggered, the router must receive a duplicate packet on its already populated OIL for the same source IP/destination group on the segment. R3 is also a LHR, which means it is designated to switch from (*,G) towards SPT (S,G) when a packet is received from (*,G).

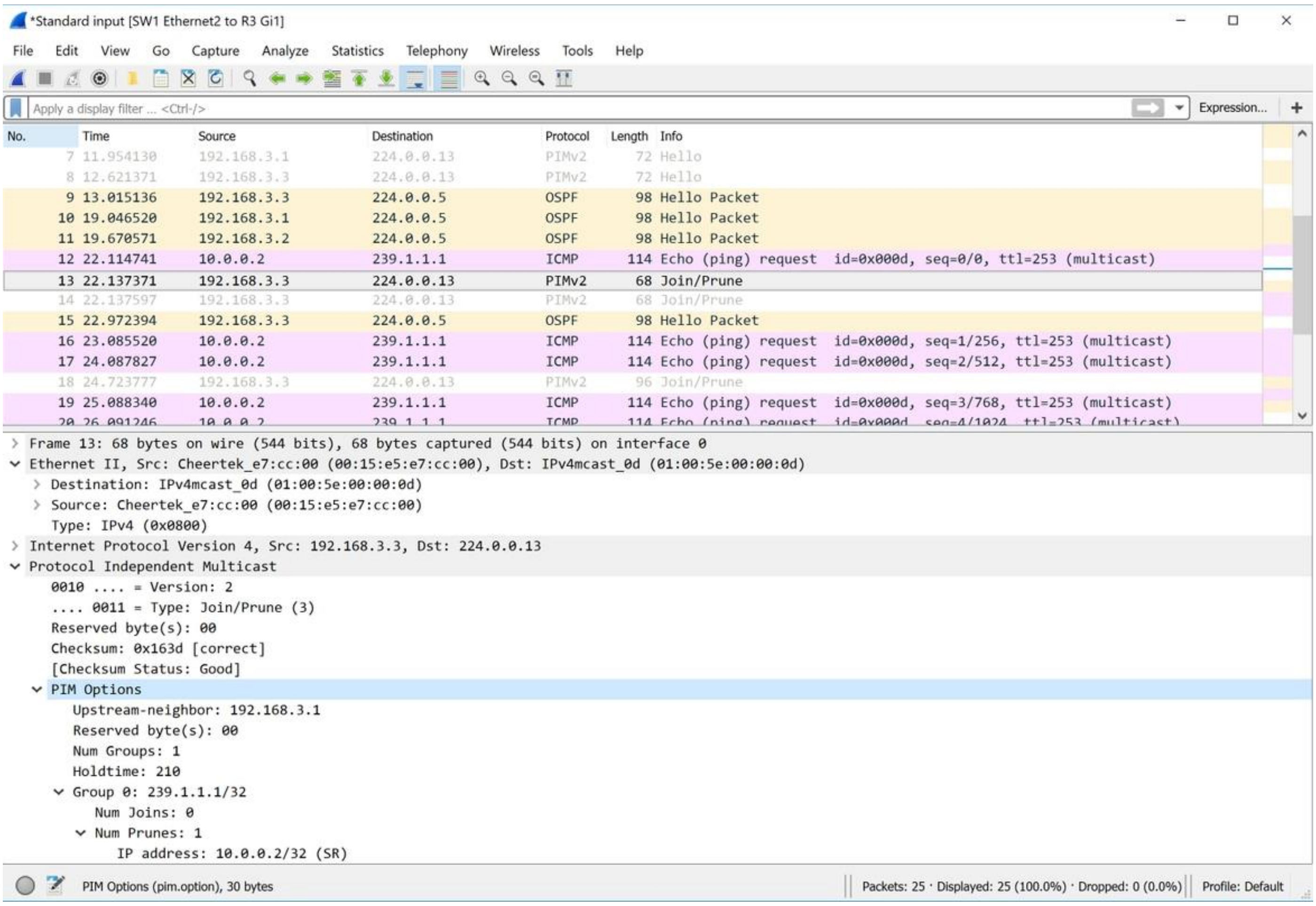
In the packet capture we observe that no asserts are triggered. Although we do see a prune sent immediately after the first ICMP echo is received.

As you can see, once the first Internet Control Message Protocol (ICMP) request packet is received on R3 interface G1, a (*,G) SR-bit prune is sent towards upstream neighbor 192.168.3.1. This prunes (*,G) for the specific source defined.

You can see these flags also set: (SR):

```
The S flag: indicates that the multicast group is a sparse mode group.
The R flag: The R flag is the RP-bit flag and indicates that the information in
the (S, G) entry is applicable to the shared tree.
```

In the second PIM packet No. 14, you can see that R3 tries to join the (S,G) tree.



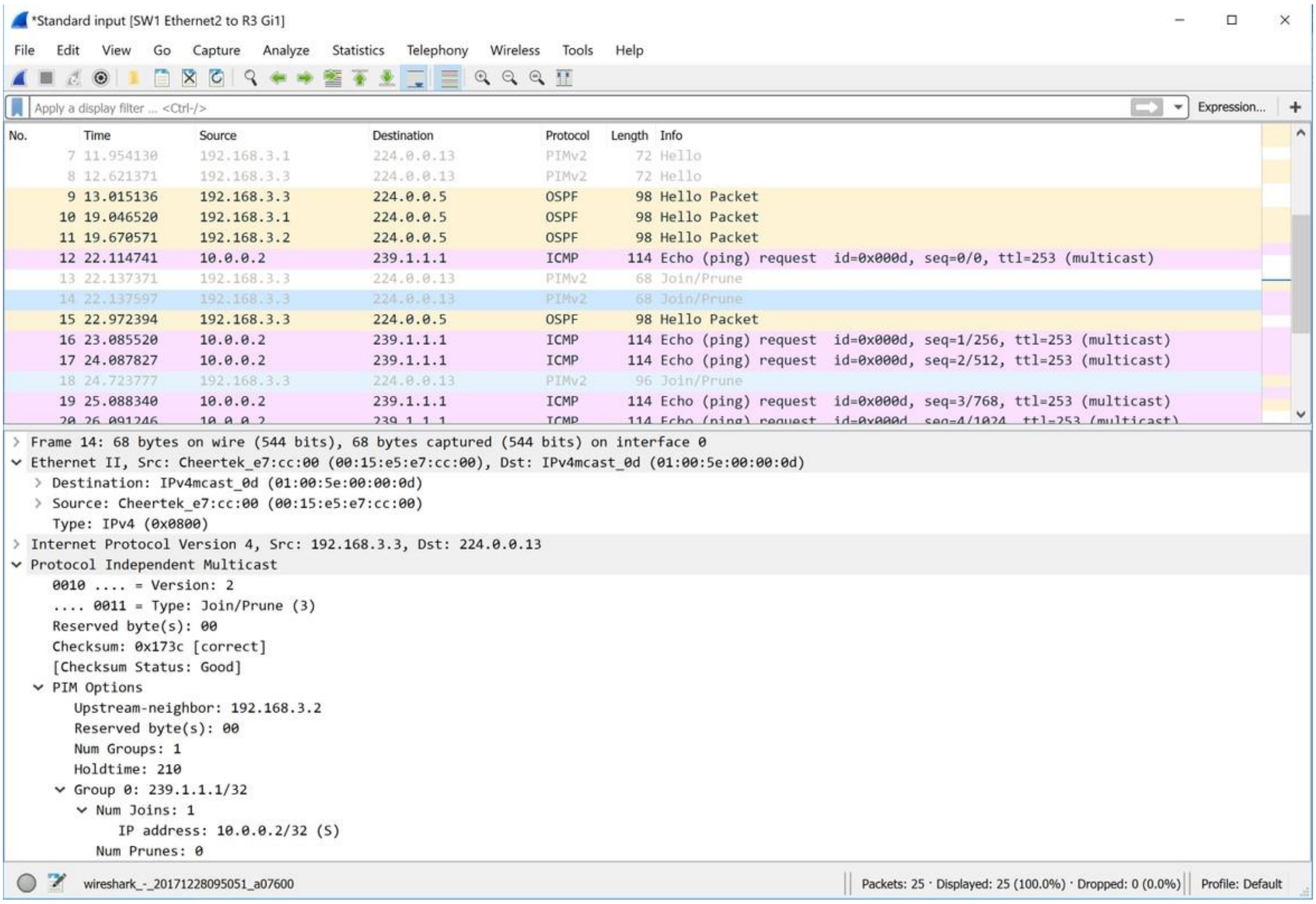
```
No.  Time       Source       Destination  Protocol Length Info
   7 11.954130  192.168.3.1  224.0.0.13   PIMv2      72 Hello
   8 12.621371  192.168.3.3  224.0.0.13   PIMv2      72 Hello
   9 13.015136  192.168.3.3  224.0.0.5    OSPF       98 Hello Packet
  10 19.046520  192.168.3.1  224.0.0.5    OSPF       98 Hello Packet
  11 19.670571  192.168.3.2  224.0.0.5    OSPF       98 Hello Packet
  12 22.114741  10.0.0.2     239.1.1.1    ICMP      114 Echo (ping) request  id=0x000d, seq=0/0, ttl=253 (multicast)
  13 22.137371  192.168.3.3  224.0.0.13   PIMv2      68 Join/Prune
  14 22.137597  192.168.3.3  224.0.0.13   PIMv2      68 Join/Prune
  15 22.972394  192.168.3.3  224.0.0.5    OSPF       98 Hello Packet
  16 23.085520  10.0.0.2     239.1.1.1    ICMP      114 Echo (ping) request  id=0x000d, seq=1/256, ttl=253 (multicast)
  17 24.087827  10.0.0.2     239.1.1.1    ICMP      114 Echo (ping) request  id=0x000d, seq=2/512, ttl=253 (multicast)
  18 24.723777  192.168.3.3  224.0.0.13   PIMv2      96 Join/Prune
  19 25.088340  10.0.0.2     239.1.1.1    ICMP      114 Echo (ping) request  id=0x000d, seq=3/768, ttl=253 (multicast)
  20 26.091246  10.0.0.2     239.1.1.1    ICMP      114 Echo (ping) request  id=0x000d, seq=4/1024, ttl=253 (multicast)
```

```
> Frame 13: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface 0
v Ethernet II, Src: Cheertek_e7:cc:00 (00:15:e5:e7:cc:00), Dst: IPv4mcast_0d (01:00:5e:00:00:0d)
  > Destination: IPv4mcast_0d (01:00:5e:00:00:0d)
  > Source: Cheertek_e7:cc:00 (00:15:e5:e7:cc:00)
    Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 192.168.3.3, Dst: 224.0.0.13
v Protocol Independent Multicast
    0010 .... = Version: 2
    .... 0011 = Type: Join/Prune (3)
    Reserved byte(s): 00
    Checksum: 0x163d [correct]
    [Checksum Status: Good]
  v PIM Options
      Upstream-neighbor: 192.168.3.1
      Reserved byte(s): 00
      Num Groups: 1
      Holdtime: 210
    v Group 0: 239.1.1.1/32
        Num Joins: 0
      v Num Prunes: 1
          IP address: 10.0.0.2/32 (SR)
```

PIM Options (pim.option), 30 bytes

Packets: 25 · Displayed: 25 (100.0%) · Dropped: 0 (0.0%)

Profile: Default



*Standard input [SW1 Ethernet2 to R3 Gi1]

```
No.    Time        Source         Destination    Protocol  Length Info
     7 11.954130   192.168.3.1    224.0.0.13     PIMv2       72 Hello
     8 12.621371   192.168.3.3    224.0.0.13     PIMv2       72 Hello
     9 13.015136   192.168.3.3    224.0.0.5      OSPF        98 Hello Packet
    10 19.046520   192.168.3.1    224.0.0.5      OSPF        98 Hello Packet
    11 19.670571   192.168.3.2    224.0.0.5      OSPF        98 Hello Packet
    12 22.114741   10.0.0.2       239.1.1.1      ICMP       114 Echo (ping) request  id=0x000d, seq=0/0, ttl=253 (multicast)
    13 22.137371   192.168.3.3    224.0.0.13     PIMv2       68 Join/Prune
    14 22.137597   192.168.3.3    224.0.0.13     PIMv2       68 Join/Prune
    15 22.972394   192.168.3.3    224.0.0.5      OSPF        98 Hello Packet
    16 23.085520   10.0.0.2       239.1.1.1      ICMP       114 Echo (ping) request  id=0x000d, seq=1/256, ttl=253 (multicast)
    17 24.087827   10.0.0.2       239.1.1.1      ICMP       114 Echo (ping) request  id=0x000d, seq=2/512, ttl=253 (multicast)
    18 24.723777   192.168.3.3    224.0.0.13     PIMv2       96 Join/Prune
    19 25.088340   10.0.0.2       239.1.1.1      ICMP       114 Echo (ping) request  id=0x000d, seq=3/768, ttl=253 (multicast)
    20 26.091246   10.0.0.2       239.1.1.1      ICMP       114 Echo (ping) request  id=0x000d, seq=4/1024, ttl=253 (multicast)
```

```
> Frame 14: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface 0
v Ethernet II, Src: Cheertek_e7:cc:00 (00:15:e5:e7:cc:00), Dst: IPv4mcast_0d (01:00:5e:00:00:0d)
  > Destination: IPv4mcast_0d (01:00:5e:00:00:0d)
  > Source: Cheertek_e7:cc:00 (00:15:e5:e7:cc:00)
    Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 192.168.3.3, Dst: 224.0.0.13
v Protocol Independent Multicast
    0010 .... = Version: 2
    .... 0011 = Type: Join/Prune (3)
    Reserved byte(s): 00
    Checksum: 0x173c [correct]
    [Checksum Status: Good]
  v PIM Options
      Upstream-neighbor: 192.168.3.2
      Reserved byte(s): 00
      Num Groups: 1
      Holdtime: 210
    v Group 0: 239.1.1.1/32
      v Num Joins: 1
            IP address: 10.0.0.2/32 (S)
        Num Prunes: 0
```

wireshark_-_20171228095051_a07600 | Packets: 25 · Displayed: 25 (100.0%) · Dropped: 0 (0.0%) | Profile: Default

It is observed that once the first data plane is received, packet R3 prunes the (*,G) and builds the (S,G). This is the reason why you do not see PIM assert packets. This certain scenario is in effect when you have a LHR who has same RPF interface for (S,G) and (*,G). Although this behaviour may slightly differ from RFC 7761, it should not cause any issues.

Scenario 2. Assert Path Selection

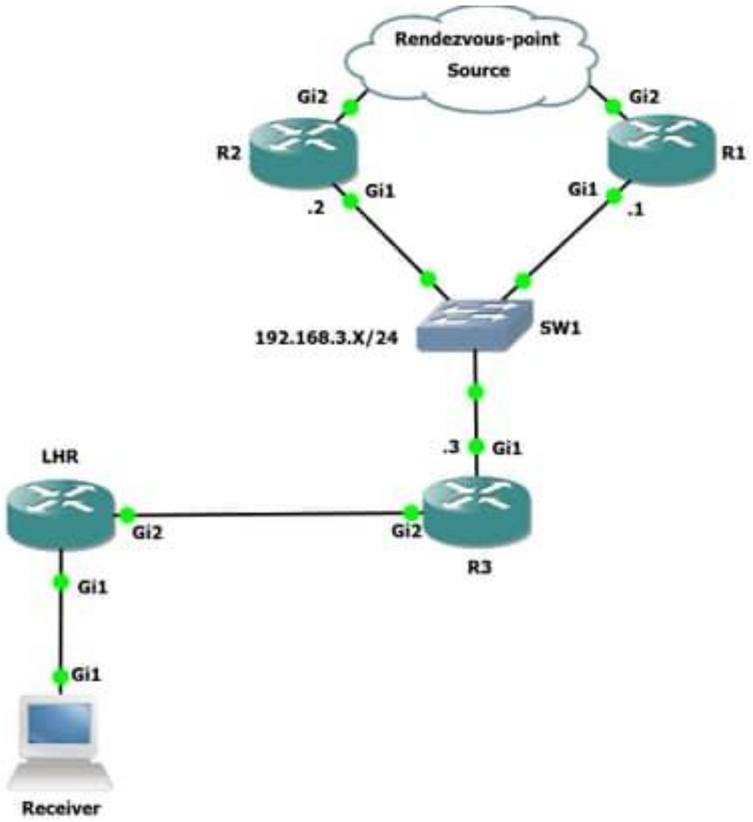


Figure 2.

In this topology, there is another router connected on R3 which is the LHR. The LHR connects directly to the receiver. The source and RP are both higher than R2 and R1. The RPF neighbor towards the RP is R1, the RPF neighbor towards the source is R2.

Let's check the RPF neighbor for both the source and RP.

Here you see the RPF neighbor towards the RP: 192.168.0.100 is 192.168.3.1.

```
R3#show ip rpf 192.168.0.100
RPF information for ? (192.168.0.100)
  RPF interface: GigabitEthernet1
  RPF neighbor: ? (192.168.3.1)
  RPF route/mask: 192.168.0.100/32
  RPF type: unicast (ospf 1)
  Doing distance-preferred lookups across tables
  RPF topology: ipv4 multicast base, originated from ipv4 unicast base
```

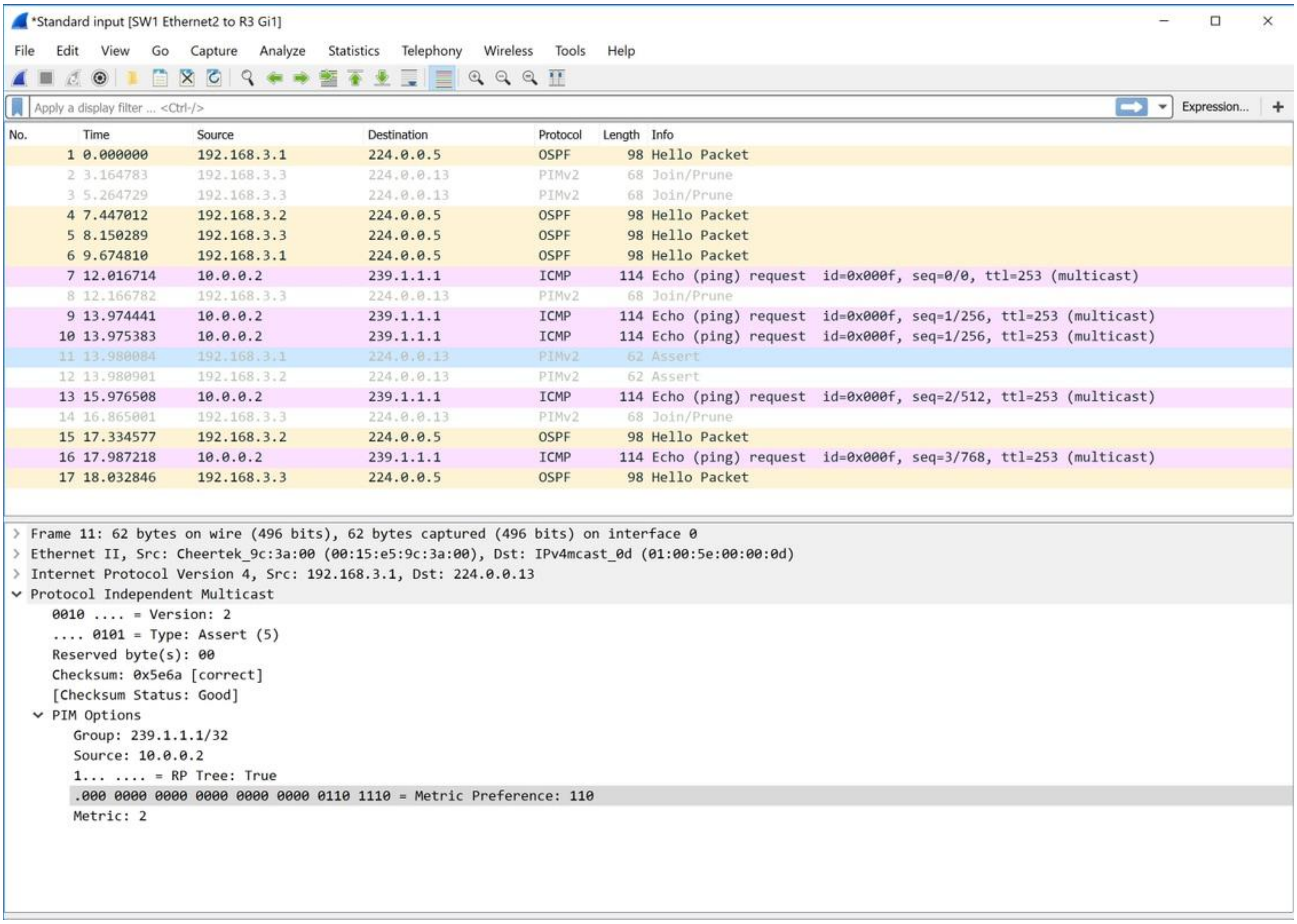
Here you see the RPF neighbor towards the Source: 10.0.0.2 is 192.168.3.2.

```
R3# show ip rpf 10.0.0.2
RPF information for ? (10.0.0.2)
  RPF interface: GigabitEthernet1
  RPF neighbor: ? (192.168.3.2)
  RPF route/mask: 10.0.0.0/24
  RPF type: unicast (ospf 1)
  Doing distance-preferred lookups across tables
  RPF topology: ipv4 multicast base, originated from ipv4 unicast base
```

Before you activate the source, lets take a look at the mroute table on R3, as you can see there is already (*,G) for the group 239.1.1.1. This is because the receiver connected to LHR has already requested for the specified group.

```
R3#show ip mroute
IP Multicast Routing Table
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.1.1.1), 00:00:57/00:02:32, RP 192.168.0.100, flags: S
  Incoming interface: GigabitEthernet1, RPF nbr 192.168.3.1
  Outgoing interface list:
    GigabitEthernet2, Forward/Sparse, 00:00:57/00:02:32

(*, 224.0.1.40), 00:11:24/00:02:41, RP 192.168.0.100, flags: SJCL
  Incoming interface: GigabitEthernet1, RPF nbr 192.168.3.1
  Outgoing interface list:
    GigabitEthernet2, Forward/Sparse, 00:02:02/00:02:41
```

Now, activate the source and capture packets on R3 interface Gi1.

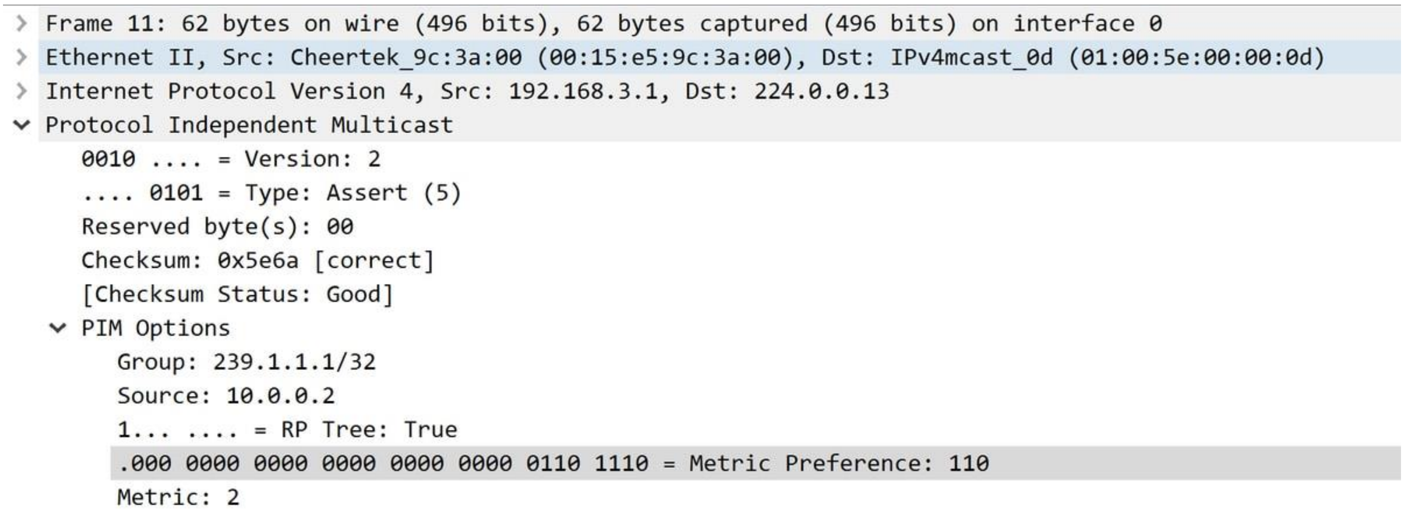

*Standard input [SW1 Ethernet2 to R3 Gi1]
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

```
No.     Time          Source          Destination     Protocol  Length  Info
      1 0.000000      192.168.3.1     224.0.0.5       OSPF        98 Hello Packet
      2 3.164783      192.168.3.3     224.0.0.13      PIMv2       68 Join/Prune
      3 5.264729      192.168.3.3     224.0.0.13      PIMv2       68 Join/Prune
      4 7.447012      192.168.3.2     224.0.0.5       OSPF        98 Hello Packet
      5 8.150289      192.168.3.3     224.0.0.5       OSPF        98 Hello Packet
      6 9.674810      192.168.3.1     224.0.0.5       OSPF        98 Hello Packet
      7 12.016714     10.0.0.2        239.1.1.1       ICMP       114 Echo (ping) request  id=0x000f, seq=0/0, ttl=253 (multicast)
      8 12.166782     192.168.3.3     224.0.0.13      PIMv2       68 Join/Prune
      9 13.974441     10.0.0.2        239.1.1.1       ICMP       114 Echo (ping) request  id=0x000f, seq=1/256, ttl=253 (multicast)
     10 13.975383     10.0.0.2        239.1.1.1       ICMP       114 Echo (ping) request  id=0x000f, seq=1/256, ttl=253 (multicast)
     11 13.980084     192.168.3.1     224.0.0.13      PIMv2       62 Assert
     12 13.980901     192.168.3.2     224.0.0.13      PIMv2       62 Assert
     13 15.976508     10.0.0.2        239.1.1.1       ICMP       114 Echo (ping) request  id=0x000f, seq=2/512, ttl=253 (multicast)
     14 16.865001     192.168.3.3     224.0.0.13      PIMv2       68 Join/Prune
     15 17.334577     192.168.3.2     224.0.0.5       OSPF        98 Hello Packet
     16 17.987218     10.0.0.2        239.1.1.1       ICMP       114 Echo (ping) request  id=0x000f, seq=3/768, ttl=253 (multicast)
     17 18.032846     192.168.3.3     224.0.0.5       OSPF        98 Hello Packet
```

```
> Frame 11: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface 0
> Ethernet II, Src: Cheertek_9c:3a:00 (00:15:e5:9c:3a:00), Dst: IPv4mcast_0d (01:00:5e:00:00:0d)
> Internet Protocol Version 4, Src: 192.168.3.1, Dst: 224.0.0.13
v Protocol Independent Multicast
    0010 .... = Version: 2
    .... 0101 = Type: Assert (5)
    Reserved byte(s): 00
    Checksum: 0x5e6a [correct]
    [Checksum Status: Good]
  v PIM Options
      Group: 239.1.1.1/32
      Source: 10.0.0.2
      1... .... = RP Tree: True
      .000 0000 0000 0000 0000 0000 0110 1110 = Metric Preference: 110
      Metric: 2
```

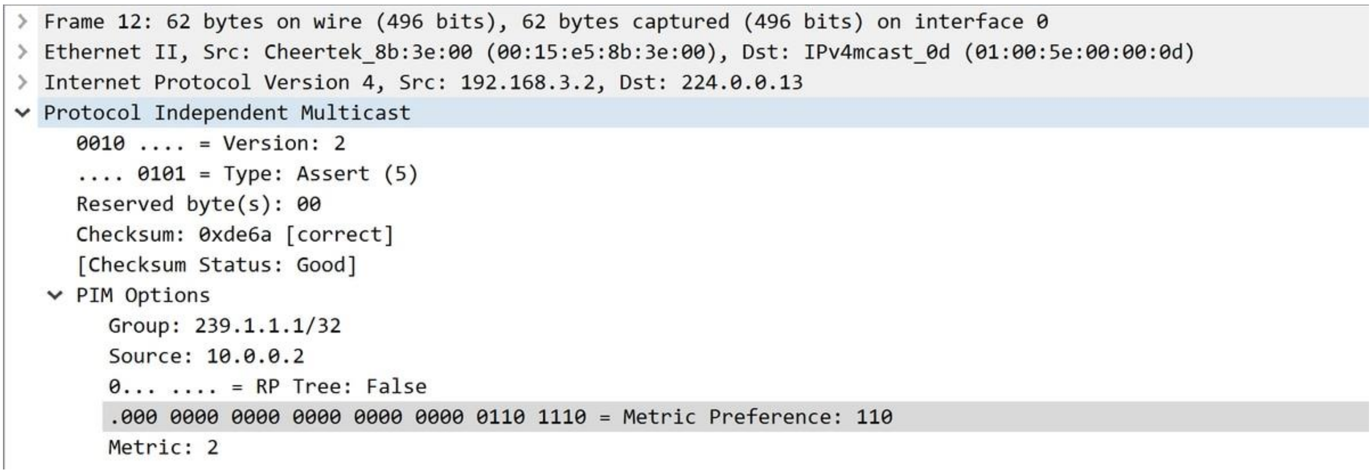
As you can see in this packet capture, PIM asserts packets are already present.

Frame 11:



```
> Frame 11: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface 0
> Ethernet II, Src: Cheertek_9c:3a:00 (00:15:e5:9c:3a:00), Dst: IPv4mcast_0d (01:00:5e:00:00:0d)
> Internet Protocol Version 4, Src: 192.168.3.1, Dst: 224.0.0.13
v Protocol Independent Multicast
    0010 .... = Version: 2
    .... 0101 = Type: Assert (5)
    Reserved byte(s): 00
    Checksum: 0x5e6a [correct]
    [Checksum Status: Good]
  v PIM Options
      Group: 239.1.1.1/32
      Source: 10.0.0.2
      1... .... = RP Tree: True
      .000 0000 0000 0000 0000 0000 0110 1110 = Metric Preference: 110
      Metric: 2
```

Frame 12:



```
> Frame 12: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface 0
> Ethernet II, Src: Cheertek_8b:3e:00 (00:15:e5:8b:3e:00), Dst: IPv4mcast_0d (01:00:5e:00:00:0d)
> Internet Protocol Version 4, Src: 192.168.3.2, Dst: 224.0.0.13
v Protocol Independent Multicast
    0010 .... = Version: 2
    .... 0101 = Type: Assert (5)
    Reserved byte(s): 00
    Checksum: 0xde6a [correct]
    [Checksum Status: Good]
  v PIM Options
      Group: 239.1.1.1/32
      Source: 10.0.0.2
      0... .... = RP Tree: False
      .000 0000 0000 0000 0000 0000 0110 1110 = Metric Preference: 110
      Metric: 2
```

When you look at these packets, you should be able to determine who is the assert winner. Now let's take a look at the PIM assert forwarder selection.

The metric preference is the Administrative Distance (AD). This refers to the administrative distance of the routing protocol installing the route in the routing table, which is used to look up the source IP address and the Metric is the cost of the route.

There are also other attributes which are used to determine who is the assert winner. You can see these details in RFC 7761.

Abstract from RFC 7761 Section 4.6.3.

```
4.6.3.  Assert Metrics

   Assert metrics are defined as:

     struct assert_metric {
       rpt_bit_flag;
       metric_preference;
       route_metric;
       ip_address;
     };
   When comparing assert_metrics, the rpt_bit_flag, metric_preference,
   and route_metric fields are compared in order, where the first lower
   value wins.  If all fields are equal, the primary IP address of the
   router that sourced the Assert message is used as a tie-breaker, with
   the highest IP address winning.
```

With the use of these fields defined and path selection, you can determine who the assert winner will be in this scenario. If you take a look at the assert packets again, you can see that metric preference is not compared since the decision is made on the very first selection criteria which is rpt_bit_flag.

In this scenario, comparing R1 and R2 is compared. Both routers send assert messages which were seen previously and once both devices see each other’s assert messages they can compare metrics between each other in order to determine who the winner is.

Since R2 sends an assert message with the RP tree: False which has a value of 0, it is indeed lower than what R1 sent with a RP tree: True which has a value of 1. RP tree bit is set to either 0 or 1.

RP tree bit when set to 1 means that you are currently on the shared tree; the RPT bit cleared indicates that the sender of the assert had (S,G) forwarding state on an interface.

As (S,G) asserts have priority over (*,G) asserts, R2 should be the assert winner. Transition to the "I am Assert Winner" state. As mentioned in the earlier statement in RFC 7761, the lower value is more preferred.

Let's have a look at both R1 and R2 in order to see who the assert winner is.

```
R2#show ip mroute
IP Multicast Routing Table
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.1.1.1), 00:42:52/stopped, RP 192.168.0.100, flags: SP
  Incoming interface: GigabitEthernet2, RPF nbr 192.168.4.1
  Outgoing interface list: Null

(10.0.0.2, 239.1.1.1), 00:42:52/00:01:40, flags: T
  Incoming interface: GigabitEthernet2, RPF nbr 192.168.4.1
  Outgoing interface list:
    GigabitEthernet1, Forward/Sparse, 00:42:52/00:03:07, A

(*, 224.0.1.40), 00:43:23/00:02:25, RP 192.168.0.100, flags: SJPL
  Incoming interface: GigabitEthernet2, RPF nbr 192.168.4.1
  Outgoing interface list: Null
```

In this output, you can see that the (S,G) on R2 has the A flag set on the OIL which indicates that it is the assert winner. Here on R1, you do not have an OIL on the (S,G) and the P flag is set which means the particular (S,G) has been pruned off in this case: it is not the assert winner.

Note: When assert is present on a shared segment, downstream neighbors send Join(*,G) and Join(S,G) periodic messages to the appropriate RPF neighbor, i.e., the RPF neighbor as modified by the assert process. They are not always sent to the RPF neighbor as indicated by the MRIB.

```
R1#show ip mroute
IP Multicast Routing Table
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.1.1.1), 00:44:32/00:03:09, RP 192.168.0.100, flags: S
  Incoming interface: GigabitEthernet2, RPF nbr 192.168.5.2
  Outgoing interface list:
    GigabitEthernet1, Forward/Sparse, 00:44:32/00:03:09, A

(10.0.0.2, 239.1.1.1), 00:44:19/00:03:09, flags: PR
  Incoming interface: GigabitEthernet2, RPF nbr 192.168.5.2
  Outgoing interface list: Null

(*, 224.0.1.40), 00:44:50/00:02:53, RP 192.168.0.100, flags: SJCL
  Incoming interface: GigabitEthernet2, RPF nbr 192.168.5.2
  Outgoing interface list:
    GigabitEthernet1, Forward/Sparse, 00:43:56/00:02:53
```

If it is the case that both R1 and R2 have RP tree bit set to 1. you can then consider the router with the lowest AD; if equal, then take a look at the metric. If RP tree bit is true on both routers, the metric is compared toward RP IP address. If RP tree bit is 0, the metric is compared towards the source of the multicast stream.

If all these values are the same, the highest IP address sourcing assert message is the winner.

Assert mechanism on default mdt

If a source in the mVPN world is dual-homed to two Ingress Provider Edge (PE) routers, it could be possible for the two Ingress PE routers to both forward traffic for one (S,G) into the Multiprotocol Label Switching (MPLS) cloud. This is possible if, for example, there are two Egress PE routers and each Reverse Path Forwarding (RPF) to a different Ingress PE router. If both Ingress PE routers forward onto the Default MDT, then the assert mechanism will kick in and one Ingress PE wins the assert mechanism and the other loses so that one and only one Ingress PE continues to forward the Customer (C-) (S,G) onto the MDT. However, if for any reason the assert mechanism did not start on the Default MDT, then it is possible for both Ingress PE routers to begin to transmit the C-(S,G) multicast traffic onto one Data-MDT they initiate. Because the traffic is not on the Default MDT anymore, but on Data MDTs, both Ingress PE routers do not receive the C-(S,G) traffic from each other on the MDT/Tunnel interface. This can cause persistent duplicate traffic downstream. This document explains the solution to this problem.

Assert Mechanism on the Default MDT

The information in this section holds true for the Default MDT, regardless of the core tree protocol. The chosen core tree protocol is Protocol Independent Multicast (PIM). Cisco IOS is used for the examples, but everything that is mentioned applies equally for Cisco IOS-XR. All multicast Groups used are Source Specific Multicast (SSM) groups.

Look at Figure 1. Dual-Homed-Source-1. There are two Ingress PE routers (PE1 and PE2) and two Egress PE routers (PE3 and PE4). The Source is at CE1 with IP address 10.100.1.6. CE1 is dual-homed to PE1 and PE2.

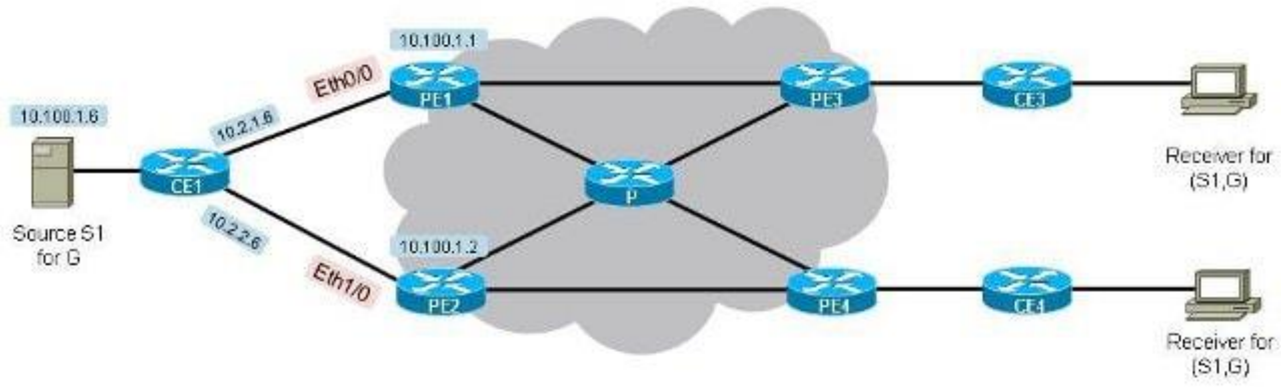


Figure 1. Dual-Homed-Source-1

The configuration on all PE routers (the Route Distinguisher (RD) can be different on the PE routers) is:

```
vrf definition one
 rd 1:1
 !
 address-family ipv4
  mdt default 232.10.10.10
  route-target export 1:1
  route-target import 1:1
 exit-address-family
!
```

In order to get both Ingress PE routers to start to forward the multicast stream (10.100.1.6,232.1.1.1) out onto the Default MDT, they must both receive a Join from an Egress PE. Look at the topology in Figure1. Dual-Homed-Source-1. You can see that by default, if all the costs of the edge links are the same and all of the costs of the core links are the same, then PE3 will RPF towards PE1 and PE4 will RPF towards PE2 for (10.100.1.6,232.1.1.1). They both RPF to their closest Ingress PE. This output confirms this:

```
PE3#show ip rpf vrf one 10.100.1.6
RPF information for ? (10.100.1.6)
  RPF interface: Tunnel0
  RPF neighbor: ? (10.100.1.1)
  RPF route/mask: 10.100.1.6/32
  RPF type: unicast (bgp 1)
  Doing distance-preferred lookups across tables
  BGP originator: 10.100.1.1
  RPF topology: ipv4 multicast base, originated from ipv4 unicast base
```

PE3 has RPF to PE1.

```
PE4#show ip rpf vrf one 10.100.1.6
RPF information for ? (10.100.1.6)
  RPF interface: Tunnel0
  RPF neighbor: ? (10.100.1.2)
  RPF route/mask: 10.100.1.6/32
  RPF type: unicast (bgp 1)
  Doing distance-preferred lookups across tables
  BGP originator: 10.100.1.2
  RPF topology: ipv4 multicast base, originated from ipv4 unicast base
```

PE4 has RPF to PE2. The reason that PE3 picks PE1 as the RPF neighbor is that the unicast route towards 10.100.1.6/32 in Virtual Routing/Forwarding (VRF) one is the best via PE1. PE3 actually receives the route 10.100.1.6/32 from both PE1 and PE2. All the criteria in the Border Gateway Protocol (BGP) Best Path Calculation Algorithm are the same, except for the cost towards the BGP next-hop address.

```
PE3#show bgp vpnv4 unicast vrf one  10.100.1.6/32
BGP routing table entry for 1:3:10.100.1.6/32, version 333
Paths: (2 available, best #1, table one)
  Advertised to update-groups:
     21
  Refresh Epoch 1
  Local, imported path from 1:1:10.100.1.6/32 (global)
    10.100.1.1 (metric 11) (via default) from 10.100.1.5 (10.100.1.5)
      Origin incomplete, metric 11, localpref 100, valid, internal,best
      Extended Community: RT:1:1 OSPF DOMAIN ID:0x0005:0x000000640200
        OSPF RT:0.0.0.0:2:0 OSPF ROUTER ID:10.2.4.1:0
      Originator: 10.100.1.1, Cluster list: 10.100.1.5
      Connector Attribute: count=1
       type 1 len 12 value 1:1:10.100.1.1
      mpls labels in/out nolabel/32
      rx pathid: 0, tx pathid: 0x0
  Refresh Epoch 1
  Local, imported path from 1:2:10.100.1.6/32 (global)
    10.100.1.2 (metric 21) (via default) from 10.100.1.5 (10.100.1.5)
      Origin incomplete, metric 11, localpref 100, valid, internal
      Extended Community: RT:1:1 OSPF DOMAIN ID:0x0005:0x000000640200
        OSPF RT:0.0.0.0:2:0 OSPF ROUTER ID:10.2.2.2:0
      Originator: 10.100.1.2, Cluster list: 10.100.1.5
      Connector Attribute: count=1
       type 1 len 12 value 1:2:10.100.1.2
      mpls labels in/out nolabel/29
      rx pathid: 0, tx pathid: 0
```

```
PE4#show bgp vpnv4 unicast vrf one  10.100.1.6/32
BGP routing table entry for 1:4:10.100.1.6/32, version 1050
Paths: (2 available, best #2, table one)
  Advertised to update-groups:
     2
  Refresh Epoch 1
  Local, imported path from 1:1:10.100.1.6/32 (global)
    10.100.1.1 (metric 21) (via default) from 10.100.1.5 (10.100.1.5)
      Origin incomplete, metric 11, localpref 100, valid, internal
      Extended Community: RT:1:1 OSPF DOMAIN ID:0x0005:0x000000640200
        OSPF RT:0.0.0.0:2:0 OSPF ROUTER ID:10.2.4.1:0
      Originator: 10.100.1.1, Cluster list: 10.100.1.5
      Connector Attribute: count=1
       type 1 len 12 value 1:1:10.100.1.1
      mpls labels in/out nolabel/32
      rx pathid: 0, tx pathid: 0
  Refresh Epoch 1
  Local, imported path from 1:2:10.100.1.6/32 (global)
    10.100.1.2 (metric 11) (via default) from 10.100.1.5 (10.100.1.5)
```

```
      Origin incomplete, metric 11, localpref 100, valid, internal, best
      Extended Community: RT:1:1 OSPF DOMAIN ID:0x0005:0x000000640200
        OSPF RT:0.0.0.0:2:0 OSPF ROUTER ID:10.2.2.2:0
      Originator: 10.100.1.2, Cluster list: 10.100.1.5
      Connector Attribute: count=1
       type 1 len 12 value 1:2:10.100.1.2
      mpls labels in/out nolabel/29
      rx pathid: 0, tx pathid: 0x0
```

The best path chosen by PE3 is the path advertised by PE1 because that has the lowest Interior Gateway Protocol (IGP) cost (11), versus the IGP cost (21) towards PE2. For PE4 it is the reverse. The topology reveals that from PE3 to PE1 there is only one hop, while from PE3 to PE2 there are two hops. Since all links have the same IGP cost, PE3 picks the path from PE1 as the best.

The Multicast Routing Information Base (MRIB) for (10.100.1.6,232.1.1.1) looks like this on PE1 and PE2 when there is no multicast traffic yet:

```
PE1#show ip mroute vrf one 232.1.1.1 10.100.1.6
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
…
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(10.100.1.6, 232.1.1.1), 00:00:12/00:03:17, flags: sT
  Incoming interface: Ethernet0/0, RPF nbr 10.2.1.6
  Outgoing interface list:
    Tunnel0, Forward/Sparse, 00:00:12/00:03:17

PE2#show ip mroute vrf one 232.1.1.1 10.100.1.6
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, E - Extranet,
…
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(10.100.1.6, 232.1.1.1), 00:00:47/00:02:55, flags: sT
  Incoming interface: Ethernet1/0, RPF nbr 10.2.2.6
  Outgoing interface list:
    Tunnel0, Forward/Sparse, 00:00:47/00:02:55
```

PE1 and PE2 both received a PIM Join for (10.100.1.6, 232.1.1.1). The Tunnel0 interface is in the Outgoing Interface List (OIL) for the multicast entry on both routers.

The multicast traffic starts to flow for (10.100.1.6, 232.1.1.1). "**Debug ip pim vrf one 232.1.1.1**" and "**debug ip mrouting vrf one 232.1.1.1**" show us that the arrival of multicast traffic onto Tunnel0 (in the OIL) of both Ingress PE routers, causes the assert mechanism to run.

PE1

```
PIM(1): Send v2 Assert on Tunnel0 for 232.1.1.1, source 10.100.1.6, metric
[110/11]
PIM(1): Assert metric to source 10.100.1.6 is [110/11]
MRT(1): not RPF interface, source address 10.100.1.6, group address 232.1.1.1
PIM(1): Received v2 Assert on Tunnel0 from 10.100.1.2
PIM(1): Assert metric to source 10.100.1.6 is [110/11]
PIM(1): We lose, our metric [110/11]
PIM(1): Prune Tunnel0/232.10.10.10 from (10.100.1.6/32, 232.1.1.1)
MRT(1): Delete Tunnel0/232.10.10.10 from the olist of (10.100.1.6, 232.1.1.1)
MRT(1): Reset the PIM interest flag for (10.100.1.6, 232.1.1.1)
MRT(1): set min mtu for (10.100.1.6, 232.1.1.1) 1500->18010 - deleted
PIM(1): Received v2 Join/Prune on Tunnel0 from 10.100.1.3, not to us
PIM(1): Join-list: (10.100.1.6/32, 232.1.1.1), S-bit set
```

PE2

```
PIM(1): Received v2 Assert on Tunnel0 from 10.100.1.1
PIM(1): Assert metric to source 10.100.1.6 is [110/11]
PIM(1): We win, our metric [110/11]
PIM(1): (10.100.1.6/32, 232.1.1.1) oif Tunnel0 in Forward state
PIM(1): Send v2 Assert on Tunnel0 for 232.1.1.1, source 10.100.1.6, metric
[110/11]
PIM(1): Assert metric to source 10.100.1.6 is [110/11]
PIM(1): Received v2 Join/Prune on Tunnel0 from 10.100.1.3, to us
PIM(1): Join-list: (10.100.1.6/32, 232.1.1.1), S-bit set
PIM(1): Update Tunnel0/10.100.1.3 to (10.100.1.6, 232.1.1.1), Forward state, by
PIM SG Join
```

If the metric and distance is the same of both routers towards the Source 10.100.1.6, then there is a tie-breaker in order to determine the assert winner. The tie-breaker is the highest IP address of the PIM neighbor on the Tunnel0 (Default MDT). In this case, this is PE2:

```
PE1#show ip pim vrf one neighbor
PIM Neighbor Table
Mode: B - Bidir Capable, DR - Designated Router, N - Default DR Priority,
      P - Proxy Capable, S - State Refresh Capable, G - GenID Capable,
      L - DR Load-balancing Capable
Neighbor              Interface                   Uptime/Expires     Ver    DR
Address
Prio/Mode
10.100.1.4           Tunnel0                     06:27:57/00:01:29  v2     1 / DR S P G
10.100.1.3           Tunnel0                     06:28:56/00:01:24  v2     1 / S P G
10.100.1.2           Tunnel0                     06:29:00/00:01:41  v2     1 / S P G


PE1#show ip pim vrf one interface

Address              Interface                  Ver/    Nbr     Query  DR            DR
                                                   Mode    Count   Intvl  Prior
10.2.1.1            Ethernet0/0                v2/S    0        30       1
10.2.1.1
10.2.4.1            Ethernet1/0                v2/S    0        30       1
10.2.4.1
10.100.1.1          Lspvif1                    v2/S    0        30       1
```

```
10.100.1.1
10.100.1.1       Tunnel0                      v2/S    3      30     1
10.100.1.4
```

PE1 removed Tunnel0 from the OIL of the multicast entry because of the asserts. Since the OIL became empty, the multicast entry is pruned.

```
PE1#show ip mroute vrf one 232.1.1.1 10.100.1.6
IP Multicast Routing Table
…
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(10.100.1.6, 232.1.1.1), 00:17:24/00:00:01, flags: sPT       ← pruned
  Incoming interface: Ethernet0/0, RPF nbr 10.2.1.6
  Outgoing interface list: Null
```

PE2 has the A-flag set on the interface Tunnel0, because it is the assert winner.

```
PE2#show ip mroute vrf one 232.1.1.1 10.100.1.6
IP Multicast Routing Table
…
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(10.100.1.6, 232.1.1.1), 00:17:20/00:02:54, flags: sT
  Incoming interface: Ethernet1/0, RPF nbr 10.2.2.6
  Outgoing interface list:
    Tunnel0, Forward/Sparse, 00:17:20/00:02:54, A ← assert ‘win’
```

PE2 periodically sends an assert on Tunnel0 (Default MDT), just before the assert timer expires. As such PE2 remains the assert winner.

```
PE2#
PIM(1): Send v2 Assert on Tunnel0 for 232.1.1.1, source 10.100.1.6, metric
[110/11]
PIM(1): Assert metric to source 10.100.1.6 is [110/11]
```

The assert mechanism also works with a Tunnel interface in the OIL. The asserts are exchanged over the Default MDT when the Ingress PE routers receive C-(S,G) multicast traffic on the associated Tunnel interface which is in the OIL.

Assert Mechanism with Data MDTs

Most of the time when Data MDTs are configured, the assert mechanism will still run on the Default MDT as the C-(S,G) traffic is only switched over from the Default MDT to the Data MDTs after three seconds. Then the same occurs as previously described. Note that **there is only one tunnel interface per multicast-enabled VRF**: the Default MDT and all Data MDTs use one tunnel interface only. This tunnel interface is used in the OIL on the Ingress PE routers or as an RPF interface on the Egress PE routers.

In some cases it is possible that the assert mechanism is not triggered before the Data MDTs are signaled. Then it is possible that the C-(S,G) multicast traffic starts to be forwarded on a Data MDT on both Ingress PE routers PE1 and PE2. In such cases, **this could lead to permanent duplicate C-(S,G) multicast traffic across** the MPLS core network. In order to avoid this, this solution was implemented: when an Ingress PE router sees another Ingress PE router announce a Data MDT for which the PE router is also an Ingress PE router, it joins that Data MDT. In principle, only Egress PE routers (that have a downstream receiver) would join the Data MDT. Because Ingress PE routers join the Data MDT announced by other Ingress PE routers, it leads to the Ingress PE router receiving multicast traffic from the Tunnel interface which is present in the OIL, and hence this triggers the assert mechanism and leads to one of the Ingress PE routers to stop forwarding the C-(S,G) multicast traffic onto its Data MDT (with the Tunnel interface), while the other Ingress PE (the assert winner) can continue to forward the C-(S,G) multicast traffic onto its Data MDT.

For the next example, assume that the Ingress PE routers PE1 and PE2 never saw the C-(S,G) multicast traffic from each other on the Default MDT. The traffic is on the Default MDT for only three seconds and it is not difficult to understand that this can occur if there is, for example, temporary traffic loss on the core network.

The configuration for Data MDT is added to all PE routers. The configuration on all PE routers (the RD can be different on the PE routers) is:

```
vrf definition one
 rd 1:1
 !
 address-family ipv4
  mdt default 232.10.10.10
  mdt data 232.11.11.0 0.0.0.0
  route-target export 1:1
  route-target import 1:1
 exit-address-family
!
```

As soon as PE1 and PE2 see the traffic from the Source, they create a C-(S,G) entry. Both Ingress PE routers forward the C-(S,G) multicast traffic onto the Default MDT. Egress PE routers PE3 and PE4 receive the multicast traffic and forward it. Due to a temporary issue, PE2 does not see the traffic from PE1 and vice-versa on the Default MDT. They both send a Data MDT Join Type Length Value (TLV) out on the Default MDT.

If there is no C-(S,G) traffic, you see this multicast state on the Ingress PE routers:

```
PE1#show ip mroute vrf one 232.1.1.1 10.100.1.6
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,
…
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(10.100.1.6, 232.1.1.1), 00:00:45/00:02:44, flags: sT
  Incoming interface: Ethernet0/0, RPF nbr 10.2.1.6
  Outgoing interface list:
    Tunnel0, Forward/Sparse, 00:00:45/00:02:42

PE2#show ip mroute vrf one 232.1.1.1 10.100.1.6
IP Multicast Routing Table
…
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode
```

```
(10.100.1.6, 232.1.1.1), 00:02:18/00:03:28, flags: sT
  Incoming interface: Ethernet1/0, RPF nbr 10.2.2.6
  Outgoing interface list:
    Tunnel0, Forward/Sparse, 00:02:18/00:03:28
```

The y-flag is not yet set. Both Ingress PE routers have the Tunnel0 interface in the OIL. This is due to the fact that PE3 has RPF towards PE1 and PE4 has RPF towards PE2 for C-(S,G).

When multicast traffic for C-(S,G) starts to flow, both PE1 and PE2 forward the traffic. The threshold for Data MDT is crossed on both Ingress PE routers and both send out a Data MDT Join TLV and after three seconds start forwarding onto their Data MDT. Notice that PE1 joins the Data MDT sourced by PE2 and PE2 joins the Data MDT sourced by PE1.

```
PE1#show ip mroute vrf one 232.1.1.1 10.100.1.6
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, E - Extranet,
       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,
       G - Received BGP C-Mroute, g - Sent BGP C-Mroute,
       N - Received BGP Shared-Tree Prune, n - BGP C-Mroute suppressed,
       Q - Received BGP S-A Route, q - Sent BGP S-A Route,
       V - RD & Vector, v - Vector, p - PIM Joins on route,
       x - VxLAN group
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(10.100.1.6, 232.1.1.1), 00:01:26/00:03:02, flags: sTy
  Incoming interface: Ethernet0/0, RPF nbr 10.2.1.6
  Outgoing interface list:
    Tunnel0, Forward/Sparse, 00:01:26/00:03:02

PE2#show ip mroute vrf one 232.1.1.1 10.100.1.6
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
…
       Y - Joined MDT-data group, y - Sending to MDT-data group,
…
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(10.100.1.6, 232.1.1.1), 00:00:41/00:02:48, flags: sTy
  Incoming interface: Ethernet1/0, RPF nbr 10.2.2.6
  Outgoing interface list:
    Tunnel0, Forward/Sparse, 00:00:41/00:02:48
```

Both PE1 and PE2 receive traffic for C-(S,G) on the Tunnel0 interface (but now from the Data MDT, not the Default MDT) and the assert mechanism kicks in. Only PE2 continues to forward the C-(S,G) traffic on its Data MDT:

```
PE1#
PIM(1): Send v2 Assert on Tunnel0 for 232.1.1.1, source 10.100.1.6, metric
[110/11]
PIM(1): Assert metric to source 10.100.1.6 is [110/11]
```

```
MRT(1): not RPF interface, source address 10.100.1.6, group address 232.1.1.1
PIM(1): Received v2 Assert on Tunnel0 from 10.100.1.2
PIM(1): Assert metric to source 10.100.1.6 is [110/11]
PIM(1): We lose, our metric [110/11]
PIM(1): Prune Tunnel0/232.11.11.0 from (10.100.1.6/32, 232.1.1.1)
MRT(1): Delete Tunnel0/232.11.11.0 from the olist of (10.100.1.6, 232.1.1.1)
MRT(1): Reset the PIM interest flag for (10.100.1.6, 232.1.1.1)
PIM(1): MDT Tunnel0 removed from (10.100.1.6,232.1.1.1)
MRT(1): Reset the y-flag for (10.100.1.6,232.1.1.1)
PIM(1): MDT next_hop change from: 232.11.11.0 to 232.10.10.10 for (10.100.1.6,
232.1.1.1) Tunnel0
MRT(1): set min mtu for (10.100.1.6, 232.1.1.1) 1500->18010 - deleted
PIM(1): MDT threshold dropped for (10.100.1.6,232.1.1.1)
PIM(1): Receive MDT Packet (9889) from 10.100.1.2 (Tunnel0), length (ip: 44,
udp: 24), ttl: 1
PIM(1): TLV type: 1 length: 16 MDT Packet length: 16
PE2#
PIM(1): Received v2 Assert on Tunnel0 from 10.100.1.1
PIM(1): Assert metric to source 10.100.1.6 is [110/11]
PIM(1): We win, our metric [110/11]
PIM(1): (10.100.1.6/32, 232.1.1.1) oif Tunnel0 in Forward state
PIM(1): Send v2 Assert on Tunnel0 for 232.1.1.1, source 10.100.1.6, metric
[110/11]
PIM(1): Assert metric to source 10.100.1.6 is [110/11]
PE2#
PIM(1): Received v2 Join/Prune on Tunnel0 from 10.100.1.3, to us
PIM(1): Join-list: (10.100.1.6/32, 232.1.1.1), S-bit set
PIM(1): Update Tunnel0/10.100.1.3 to (10.100.1.6, 232.1.1.1), Forward state, by
PIM SG Join
MRT(1): Update Tunnel0/232.10.10.10 in the olist of (10.100.1.6, 232.1.1.1),
Forward state - MAC built
MRT(1): Set the y-flag for (10.100.1.6,232.1.1.1)
PIM(1): MDT next_hop change from: 232.10.10.10 to 232.11.11.0 for (10.100.1.6,
232.1.1.1) Tunnel0
```

PE1 no longer has the tunnel interface in the OIL.

```
PE1#show ip mroute vrf one 232.1.1.1 10.100.1.6
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
…
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(10.100.1.6, 232.1.1.1), 00:10:23/00:00:04, flags: sPT
  Incoming interface: Ethernet0/0, RPF nbr 10.2.1.6
  Outgoing interface list: Null
```

PE2 has the A-flag set on the Tunnel0 interface:

```
PE2#show ip mroute vrf one 232.1.1.1 10.100.1.6
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, E - Extranet,
       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,
       G - Received BGP C-Mroute, g - Sent BGP C-Mroute,
       N - Received BGP Shared-Tree Prune, n - BGP C-Mroute suppressed,
       Q - Received BGP S-A Route, q - Sent BGP S-A Route,
       V - RD & Vector, v - Vector, p - PIM Joins on route,
       x - VxLAN group
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(10.100.1.6, 232.1.1.1), 00:10:00/00:02:48, flags: sTy
  Incoming interface: Ethernet1/0, RPF nbr 10.2.2.6
  Outgoing interface list:
    Tunnel0, Forward/Sparse, 00:08:40/00:02:48, A
```

The assert mechanism also works when Data MDTs are used. The asserts are exchanged over the Default MDT when the Ingress PE routers receive C-(S,G) multicast traffic on the associated Tunnel interface which is in the OIL.

Partitioned MDT

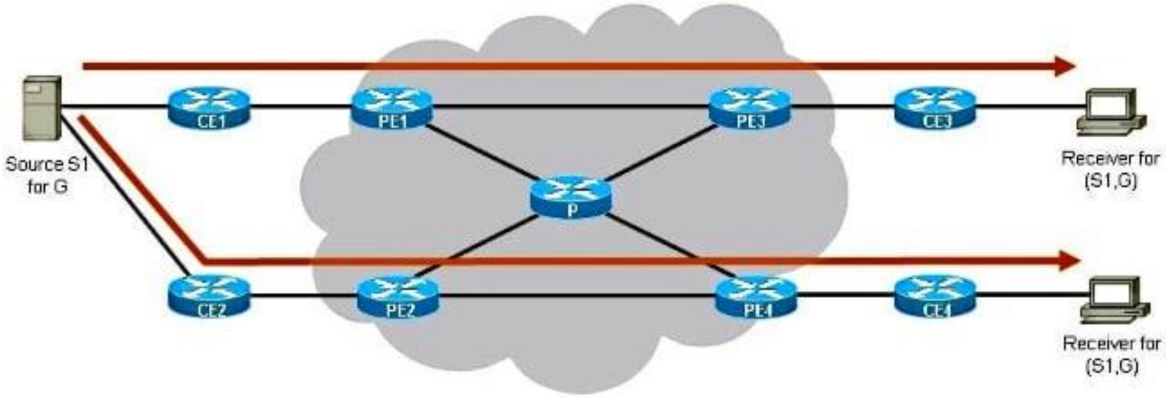
This problem could potentially happen on multi-homed and redundantly connected sources, flowing through an mpls network. We need to add some other information before moving on with this.

RPF implies that the incoming interface is **checked towards the source**. Although the interface is checked to determine that it is the correct one towards the source, it is not checked to determine that it is the correct RPF neighbor on that interface. On a multi-access interface, there could be more than one neighbor to which you could RPF. **The result could be that the router receives twice the same multicast stream on that interface and forwards both**.

In networks where **Protocol Independent Multicast (PIM)** runs on the multi-access interface, this is not an issue, because the duplicate multicast stream causes the **assert mechanism to run** and one multicast stream will no longer be received. In some cases, PIM does not run on the Multicast Distribution Tree (MDT), which is a multi-access interface. In those cases, Border Gateway Protocol (**BGP**) is the **overlay signaling protocol**.

In the profiles with Partitioned MDT, even **if PIM runs as the overlay protocol**, it can be impossible to have asserts. The reason for this is that one Ingress Provider Edge (PE) does not join the Partitioned MDT from another Ingress PE in the scenarios where there are two or more Ingress PE routers. Each Ingress PE router can forward the multicast stream onto its Partitioned MDT without the other Ingress PE router seeing the multicast traffic. The fact that two different Egress PE routers each join an MDT towards a different Ingress PE router for the same multicast stream is a valid scenario: it is called **Anycast Source**. This allows different receivers to join the same multicast stream but over a different path in the Multiprotocol Label Switching (MPLS) core. See Figure 1 for an example of Anycast Source.

There are two Ingress PE routers: PE1 and PE2. There are two Egress PE routers: PE3 and PE4. Each Egress PE router has a different Ingress PE router as its RPF neighbor. PE3 has PE1 as its RPF neighbor. PE4 has PE2 as its RPF neighbor. The Egress PE routers pick their closest Ingress PE router as their RPF neighbor. The Stream (S1,G) will go from S1 to Receiver 1 over the top path and from S1 to Receiver 2 over the bottom path. There is no intersection of the two streams over the two paths (**each path in the MPLS core is a different Partitioned MDT**).

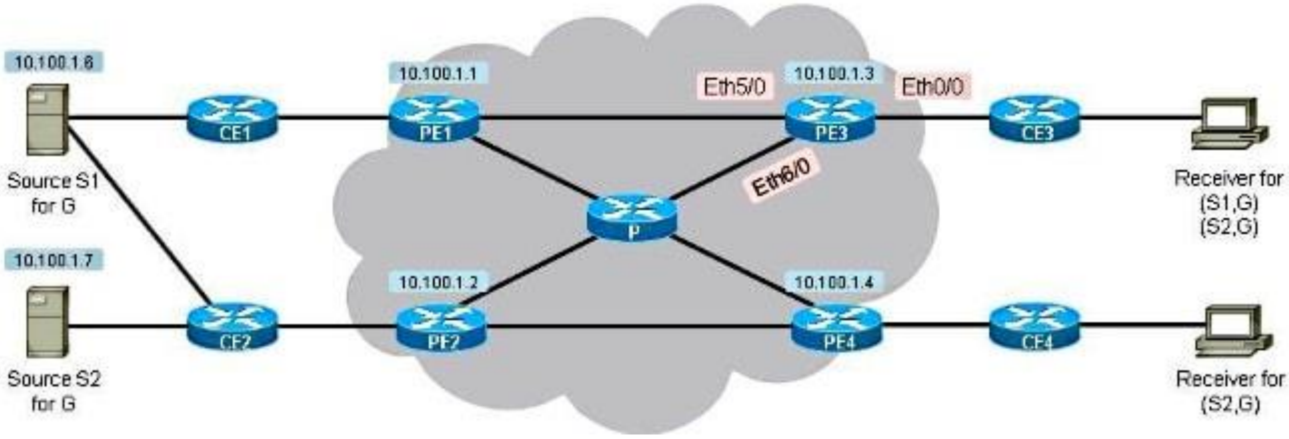


If the MDT was a Default MDT - such as in the Default MDT profiles - then this would not work because **the two multicast streams would be on the same Default MDT and the assert mechanism would run** (PE2 **OR** PE1 would stop forwarding S1 traffic in the core, since they see each other in the logical multicast 'mpls core' interface). If the MDT is a Data MDT in the Default MDT profiles, then all Ingress PE routers join the Data MDT from the other Ingress PE routers and as such see the multicast traffic from each other and the assert mechanism runs again. If the overlay protocol is BGP, then there is Upstream Multicast Hop (UMH) selection and only one Ingress PE router is selected as the forwarder, but this is per MDT. **Anycast Source** is one of the big advantages of running Partitioned MDT.

Duplicated traffic problem

The regular RPF check confirms that the packets arrive at the router from the correct RPF interface. There is no check to confirm that the packets are received from the correct RPF neighbor on that interface.

See Figure 2. It shows an issue where duplicate traffic is persistently forwarded in a scenario with Partitioned MDT. It shows that the regular RPF check in the case of Partitioned MDT is not sufficient in order to avoid duplicate traffic.



There are two receivers. The first receiver is set up to receive traffic for (S1,G) and (S2,G). The second receiver is set up to receive traffic for (S2,G) only. There is Partitioned MDT, and BGP is the overlay signaling protocol. Note that Source S1 is reachable via both PE1 and PE2. The core-tree protocol is Multipoint Label Distribution Protocol (mLDP).

Each PE router advertises a Type 1 BGP IPv4 mVPN route (i.e. auto discovery, this is the list of PEs where vrf one is configured), which indicates that it is a candidate to be the root of a Partitioned MDT.

```
PE3#show bgp ipv4 mvpn vrf one
…
     Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 1:3 (default for vrf one)
*>i  [1][1:3][10.100.1.1]/12
                      10.100.1.1               0    100      0 ?
*>i  [1][1:3][10.100.1.2]/12
                       10.100.1.2               0    100      0 ?
*>  [1][1:3][10.100.1.3]/12
                       0.0.0.0                           32768 ?
*>i  [1][1:3][10.100.1.4]/12
                       10.100.1.4               0    100      0 ?
```

PE3 finds PE1 as the RPF neighbor for S1 after a lookup for the unicast route for S1. PE3 selects PE1 as the RPF neighbor for (S1,G) and joins the Partitioned MDT with PE1 as root.

```
PE3#show bgp vpnv4 unicast vrf one 10.100.1.6/32      ← source address
BGP routing table entry for 1:3:10.100.1.6/32, version 16
Paths: (2 available, best #2, table one)
Advertised to update-groups:
     5
Refresh Epoch 2
65001, imported path from 1:2:10.100.1.6/32 (global)
   10.100.1.2 (metric 21) (via default) from 10.100.1.5 (10.100.1.5)
     Origin incomplete, metric 0, localpref 100, valid, internal
     Extended Community: RT:1:1 MVPN AS:1:0.0.0.0 MVPN VRF:10.100.1.2:1
     Originator: 10.100.1.2, Cluster list: 10.100.1.5
     mpls labels in/out nolabel/20
     rx pathid: 0, tx pathid: 0
Refresh Epoch 2
65001, imported path from 1:1:10.100.1.6/32 (global)
   10.100.1.1 (metric 11) (via default) from 10.100.1.5 (10.100.1.5)
     Origin incomplete, metric 0, localpref 100, valid, internal, best
     Extended Community: RT:1:1 MVPN AS:1:0.0.0.0 MVPN VRF:10.100.1.1:1
     Originator: 10.100.1.1, Cluster list: 10.100.1.5
     mpls labels in/out nolabel/29
     rx pathid: 0, tx pathid: 0x0

PE3#show ip rpf vrf one 10.100.1.6
RPF information for ? (10.100.1.6)
  RPF interface: Lspvif0              ← mdt in the CORE
  RPF neighbor: ? (10.100.1.1)
  RPF route/mask: 10.100.1.6/32
  RPF type: unicast (bgp 1)
  Doing distance-preferred lookups across tables
  RPF topology: ipv4 multicast base, originated from ipv4 unicast base
```

PE3 selects PE2 as the RPF neighbor for (S2,G) and joins the Partitioned MDT with PE2 as root.

```
PE3#show bgp vpnv4 unicast vrf one 10.100.1.7/32
BGP routing table entry for 1:3:10.100.1.7/32, version 18
Paths: (1 available, best #1, table one)
Advertised to update-groups:
     6
Refresh Epoch 2
65002, imported path from 1:2:10.100.1.7/32 (global)
     10.100.1.2 (metric 21) (via default) from 10.100.1.5 (10.100.1.5)
     Origin incomplete, metric 0, localpref 100, valid, internal, best
     Extended Community: RT:1:1 MVPN AS:1:0.0.0.0 MVPN VRF:10.100.1.2:1
     Originator: 10.100.1.2, Cluster list: 10.100.1.5
     mpls labels in/out nolabel/29
     rx pathid: 0, tx pathid: 0x0

PE3#show ip rpf vrf one 10.100.1.7
RPF information for ? (10.100.1.7)
  RPF interface: Lspvif0
  RPF neighbor: ? (10.100.1.2)
  RPF route/mask: 10.100.1.7/32
  RPF type: unicast (bgp 1)
  Doing distance-preferred lookups across tables
  RPF topology: ipv4 multicast base, originated from ipv4 unicast base
```

PE4 selects PE2 as the RPF neighbor for (S1,G) and joins the Partitioned MDT with PE1 as root.

```
PE4#show bgp vpnv4 unicast vrf one 10.100.1.6/32
BGP routing table entry for 1:4:10.100.1.6/32, version 138
Paths: (2 available, best #1, table one)
Advertised to update-groups:
     2
Refresh Epoch 2
65001, imported path from 1:2:10.100.1.6/32 (global)
     10.100.1.2 (metric 11) (via default) from 10.100.1.5 (10.100.1.5)
     Origin incomplete, metric 0, localpref 100, valid, internal, best
     Extended Community: RT:1:1 MVPN AS:1:0.0.0.0 MVPN VRF:10.100.1.2:1
     Originator: 10.100.1.2, Cluster list: 10.100.1.5
     mpls labels in/out nolabel/20
     rx pathid: 0, tx pathid: 0x0
Refresh Epoch 2
65001, imported path from 1:1:10.100.1.6/32 (global)
   10.100.1.1 (metric 21) (via default) from 10.100.1.5 (10.100.1.5)
     Origin incomplete, metric 0, localpref 100, valid, internal
     Extended Community: RT:1:1 MVPN AS:1:0.0.0.0 MVPN VRF:10.100.1.1:1
     Originator: 10.100.1.1, Cluster list: 10.100.1.5
     mpls labels in/out nolabel/29
     rx pathid: 0, tx pathid: 0

PE4#show ip rpf vrf one 10.100.1.6
RPF information for ? (10.100.1.6)
  RPF interface: Lspvif0
  RPF neighbor: ? (10.100.1.2)
  RPF route/mask: 10.100.1.6/32
  RPF type: unicast (bgp 1)
  Doing distance-preferred lookups across tables
  RPF topology: ipv4 multicast base, originated from ipv4 unicast base
```

Notice that the RPF interface is **Lspvif0** for both S1 (10.100.1.6) and S2 (10.100.1.7).

PE3 joins the Partitioned MDT from PE2 for (S2,G), and PE4 joins the Partitioned MDT from PE2 for (S1,G). PE1 joins the Partitioned MDT from PE1 for (S1,G). You can see this by the type 7 BGP IPv4 mVPN routes received on PE1 and PE2.

```
PE1#show bgp ipv4 mvpn vrf one
BGP table version is 302, local router ID is 10.100.1.1
…
Network          Next Hop           Metric LocPrf Weight Path
Route Distinguisher: 1:1 (default for vrf one)
*>i [7][1:1][1][10.100.1.6/32][232.1.1.1/32]/22
                       10.100.1.3              0    100      0 ?
```

```
PE2#show bgp ipv4 mvpn vrf one
…
   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 1:2 (default for vrf one)
*>i [7][1:2][1][10.100.1.6/32][232.1.1.1/32]/22
                       10.100.1.4               0    100      0 ?
*>i [7][1:2][1][10.100.1.7/32][232.1.1.1/32]/22
                       10.100.1.3               0    100      0 ?
```

The multicast entries on PE3 and PE4:

```
PE3#show ip mroute vrf one 232.1.1.1
…
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(10.100.1.7, 232.1.1.1), 21:18:24/00:02:46, flags: sTg
 Incoming interface: Lspvif0, RPF nbr 10.100.1.2
 Outgoing interface list:
   Ethernet0/0, Forward/Sparse, 00:11:48/00:02:46

(10.100.1.6, 232.1.1.1), 21:18:27/00:03:17, flags: sTg
 Incoming interface: Lspvif0, RPF nbr 10.100.1.1
 Outgoing interface list:
   Ethernet0/0, Forward/Sparse, 00:11:48/00:03:17


PE4#show ip mroute vrf one 232.1.1.1
IP Multicast Routing Table
…
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(10.100.1.6, 232.1.1.1), 20:50:13/00:02:37, flags: sTg
 Incoming interface: Lspvif0, RPF nbr 10.100.1.2
 Outgoing interface list:
   Ethernet0/0, Forward/Sparse, 20:50:13/00:02:37
```

This shows that PE3 joins the Point-to-Multipoint (P2MP) tree rooted at PE1 and also the tree rooted at PE2:

```
PE3#show mpls mldp database
* Indicates MLDP recursive forwarding is enabled

LSM ID : A   Type: P2MP   Uptime : 00:18:40
 FEC Root                : 10.100.1.1
 Opaque decoded          : [gid 65536 (0x00010000)]
 Opaque length          : 4 bytes
 Opaque value            : 01 0004 00010000
 Upstream client(s) :
   10.100.1.1:0    [Active]
     Expires          : Never          Path Set ID : A
     Out Label (U) : None          Interface    : Ethernet5/0*
     Local Label (D): 29            Next Hop     : 10.1.5.1
 Replication client(s):
   MDT (VRF one)
     Uptime            : 00:18:40      Path Set ID : None
     Interface        : Lspvif0
```

```
LSM ID : B   Type: P2MP   Uptime : 00:18:40
 FEC Root              : 10.100.1.2
 Opaque decoded        : [gid 65536 (0x00010000)]
 Opaque length        : 4 bytes
 Opaque value          : 01 0004 00010000
 Upstream client(s) :
   10.100.1.5:0  [Active]
     Expires         : Never          Path Set ID : B
     Out Label (U) : None          Interface    : Ethernet6/0*
     Local Label (D): 30           Next Hop     : 10.1.3.5
 Replication client(s):
   MDT (VRF one)
     Uptime         : 00:18:40     Path Set ID : None
     Interface      : Lspvif0
```

This shows that PE4 joins the P2MP tree rooted at PE2:

```
PE4#show mpls mldp database
* Indicates MLDP recursive forwarding is enabled

LSM ID : 3   Type: P2MP   Uptime : 21:17:06
 FEC Root              : 10.100.1.2
 Opaque decoded        : [gid 65536 (0x00010000)]

 Opaque value          : 01 0004 00010000
 Upstream client(s) :
   10.100.1.2:0   [Active]
     Expires         : Never          Path Set ID : 3
     Out Label (U) : None          Interface    : Ethernet5/0*
     Local Label (D): 29           Next Hop     : 10.1.6.2
 Replication client(s):
   MDT (VRF one)
     Uptime           : 21:17:06     Path Set ID : None
     Interface      : Lspvif0
```

S1 and S2 stream for the Group 232.1.1.1 with 10 pps. You can see the streams at PE3 and PE4. However, at PE3, you can see the rate for (S1, G) as 20 pps.

```
PE3#show ip mroute vrf one 232.1.1.1 count
Use "show ip mfib count" to get better response time for a large number of
mroutes.
…
Group: 232.1.1.1, Source count: 2, Packets forwarded: 1399687, Packets received:
2071455
  Source: 10.100.1.7/32, Forwarding: 691517/10/28/2, Other: 691517/0/0
  Source: 10.100.1.6/32, Forwarding: 708170/20/28/4, Other: 1379938/671768/0

PE4#show ip mroute vrf one 232.1.1.1 count
Use "show ip mfib count" to get better response time for a large number of
mroutes.
…
Group: 232.1.1.1, Source count: 1, Packets forwarded: 688820, Packets received:
688820
  Source: 10.100.1.6/32, Forwarding: 688820/10/28/2, Other: 688820/0/0

PE3#show interfaces ethernet0/0 | include rate
 Queueing strategy: fifo
  30 second input rate 0 bits/sec, 0 packets/sec
  30 second output rate 9000 bits/sec, 30 packets/sec
```

There is a duplicate stream. This duplication is the result of the presence of stream (S1, G) on the Partitioned MDT from PE1 and on the Partitioned MDT from PE2. This second Partitioned MDT, from PE2, was joined by PE3 in order to get the stream (S2, G). But, because PE4 joined the Partitioned MDT from PE2 in order to get (S1, G), (S1, G) is also present on the Partitioned MDT from PE2. Hence, PE3 receives the stream (S1, G) from both Partitioned MDTs it joined.

PE3 cannot discriminate between the packets for (S1, G) it receives from PE1 and PE2. Both streams are received on the correct RPF interface: Lspvif0.

```
PE3#show ip multicast vrf one mpls vif

Interface    Next-hop            Application      Ref-Count    Table / VRF
name    Flags
 Lspvif0      0.0.0.0             MDT                 N/A        1    (vrf one) 0x1
```

The packets could arrive on different incoming physical interfaces on PE3 or on the same interface. In any case, the packets from the different streams for (S1, G) do arrive with a different MPLS label at PE3:

```
PE3#show mpls forwarding-table vrf one
Local      Outgoing    Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id      Switched      interface
29    [T] No Label    [gid 65536 (0x00010000)][V]    \
                                       768684      aggregate/one
30    [T] No Label    [gid 65536 (0x00010000)][V]    \
                                       1535940      aggregate/one

[T]      Forwarding through a LSP tunnel.
       View additional labelling info with the 'detail' option
```

Solution

The solution is to have a stricter RPF. **With strict RPF, the router checks from which neighbor the packets are received on the RPF interface**. Without strict RPF, the only check is to determine if the incoming interface is the RPF interface, but not if the packets are received from the correct RPF neighbor on that interface.

Notes for Cisco IOS

Here are some important notes about RPF with Cisco IOS.

- When you change to/from strict RPF mode, either configure it before you configure the Partitioned MDT or clear BGP. If you only configure the strict RPF command, it will not create another Lspvif interface immediately.
- Strict RPF is not enabled by default in Cisco IOS.
- It is not supported to have the **strict-rpf** command with Default MDT profiles.

Configuration

You can configure strict RPF on PE3 for the Virtual Routing and Forwarding (VRF).

```
vrf definition one
 rd 1:3
 !
 address-family ipv4
  mdt auto-discovery mldp
  mdt strict-rpf interface                 ← can't be configured with default mdt
  mdt partitioned mldp p2mp                ← partitioned mdt
  mdt overlay use-bgp
  route-target export 1:1
  route-target import 1:1
 exit-address-family
!
```

The RPF information has changed:

```
PE3#show ip rpf vrf one 10.100.1.6
RPF information for ? (10.100.1.6)
  RPF interface: Lspvif0
  Strict-RPF interface: Lspvif1
  RPF neighbor: ? (10.100.1.1)
  RPF route/mask: 10.100.1.6/32
  RPF type: unicast (bgp 1)
  Doing distance-preferred lookups across tables
  RPF topology: ipv4 multicast base, originated from ipv4 unicast base

PE3#show ip rpf vrf one 10.100.1.7
RPF information for ? (10.100.1.7)
  RPF interface: Lspvif0
  Strict-RPF interface: Lspvif2
  RPF neighbor: ? (10.100.1.2)
  RPF route/mask: 10.100.1.7/32
  RPF type: unicast (bgp 1)
  Doing distance-preferred lookups across tables
  RPF topology: ipv4 multicast base, originated from ipv4 unicast base
```

PE3 created a **Lspvif** interface per Ingress PE. The **Lspvif** interface is created per Ingress PE, per Address Family (AF), and per VRF. The RPF for 10.100.1.6 now points to interface Lspvif1 and the RPF for 10.100.1.7 now points to interface Lspvif2. Lspvif0 is the default mdt.

```
PE3#show ip multicast vrf one mpls vif
Interface    Next-hop            Application      Ref-Count    Table / VRF
name    Flags
Lspvif0      0.0.0.0               MDT                N/A         1    (vrf one) 0x1
Lspvif1      10.100.1.1            MDT                N/A         1    (vrf one) 0x1
Lspvif2      10.100.1.2            MDT                N/A         1    (vrf one) 0x1
```

Now, the RPF check for packets (S1, G) from PE1 are checked against RPF interface Lspvif1. These packets come in with MPLS label 29. The RPF check for packets (S2, G) from PE2 are checked against RPF interface Lspvif2. These packets come in with MPLS label 30. The streams arrive on PE3 through different incoming interfaces, but this could also be the same interface. However, due to the fact that mLDP never uses Penultimate-Hop-Popping (PHP), there is always a regular MPLS label on top of the multicast packets. The (S1, G) packets that arrive from PE1 and from PE2 are on two different Partitioned MDTs and therefore have a different MPLS label. Hence, PE3 can discriminate between the (S1, G) stream that comes from PE1 and the (S1, G) stream that comes from PE2. In this way, the packets can be kept apart by PE3 and an RPF can be performed against different Ingress PE routers. The mLDP database on PE3 now shows the different Lspvif interfaces per Ingress PE.

```
PE3#show mpls mldp database
* Indicates MLDP recursive forwarding is enabled

LSM ID : C    Type: P2MP   Uptime : 00:05:58
FEC Root           : 10.100.1.1
Opaque decoded     : [gid 65536 (0x00010000)]
Opaque length     : 4 bytes
Opaque value       : 01 0004 00010000
Upstream client(s) :
   10.100.1.1:0    [Active]
     Expires        : Never         Path Set ID  : C
     Out Label (U)  : None         Interface    : Ethernet5/0*
     Local Label (D): 29            Next Hop     : 10.1.5.1
```

```
Replication client(s):
   MDT (VRF one)
     Uptime         : 00:05:58     Path Set ID : None
     Interface      : Lspvif1

LSM ID : D   Type: P2MP   Uptime : 00:05:58
FEC Root            : 10.100.1.2
Opaque decoded      : [gid 65536 (0x00010000)]
Opaque length      : 4 bytes
Opaque value        : 01 0004 00010000
Upstream client(s) :
   10.100.1.5:0    [Active]
     Expires        : Never         Path Set ID : D
     Out Label (U)  : None         Interface    : Ethernet6/0*
     Local Label (D): 30            Next Hop      : 10.1.3.5
Replication client(s):
   MDT (VRF one)
     Uptime         : 00:05:58     Path Set ID : None
     Interface      : Lspvif2
```

Strict RPF or RPF per Ingress PE works due to the fact that the multicast streams come in to the Ingress PE with a different MPLS label per ingress PE:

```
PE3#show mpls forwarding-table vrf one
Local      Outgoing    Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id      Switched      interface
29    [T] No Label   [gid 65536 (0x00010000)][V]    \
                                         162708        aggregate/one
30    [T] No Label   [gid 65536 (0x00010000)][V]    \
                                         162750        aggregate/one

[T]     Forwarding through a LSP tunnel.
       View additional labelling info with the 'detail' option
```

The proof that strict RPF works is that there is no longer a duplicate stream (S1,G) forwarded on PE3. The duplicate stream still arrives on PE3, but it is dropped due to the RPF failure. The RPF failure counter is at 676255 and increases constantly at a rate of 10 pps.

```
PE3#show ip mroute vrf one 232.1.1.1 count
Use "show ip mfib count" to get better response time for a large number of
mroutes.
…
Group: 232.1.1.1, Source count: 2, Packets forwarded: 1443260, Packets received:
2119515
Source: 10.100.1.7/32, Forwarding: 707523/10/28/2, Other: 707523/0/0
Source: 10.100.1.6/32, Forwarding: 735737/10/28/2, Other: 1411992/676255/0
```

The output rate at PE3 is now 20 pps, which is 10 pps for each stream (S1,G) and (S2,G):

```
PE3#show interfaces ethernet0/0 | include rate
Queueing strategy: fifo
30 second input rate 0 bits/sec, 0 packets/sec
30 second output rate 6000 bits/sec, 20 packets/sec
```

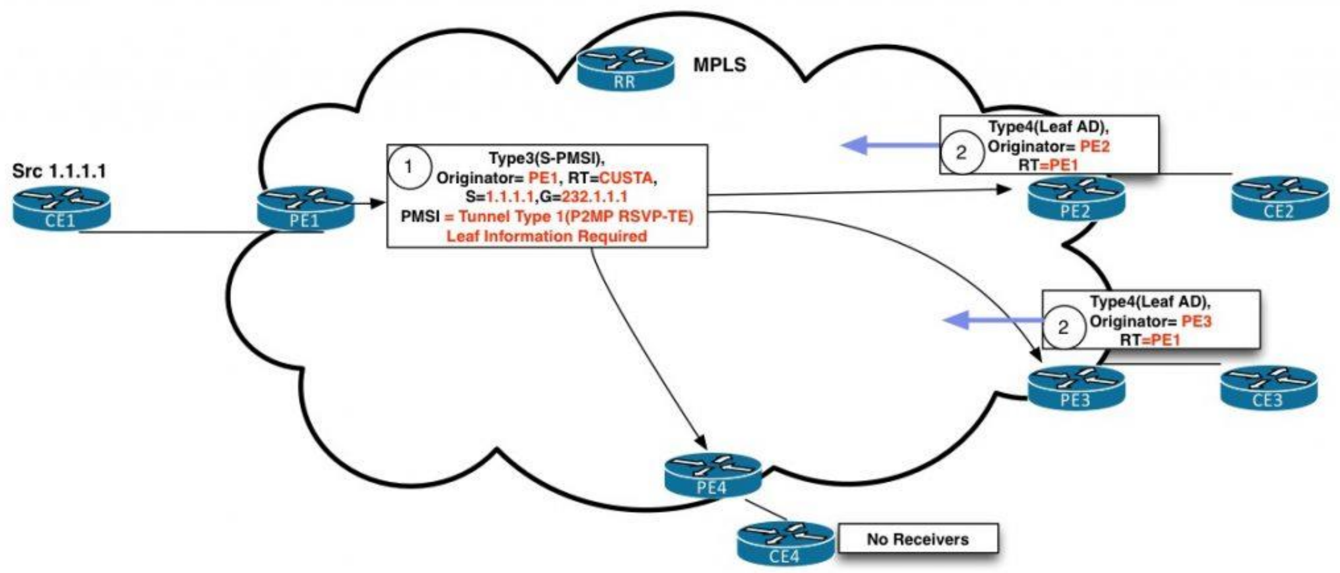
Conclusion

Strict RPF Check must be used for the mVPN deployment models that use Partitioned MDT. Things might appear to work, even if you do not configure the strict RPF check for the mVPN deployment models with Partitioned MDT: the multicast streams are delivered to the receivers. However, there is the possibility that there is duplicate multicast traffic when sources are connected to multiple Ingress PE routers. This leads to a waste of bandwidth in the network and can adversely affect the multicast application on the receivers.

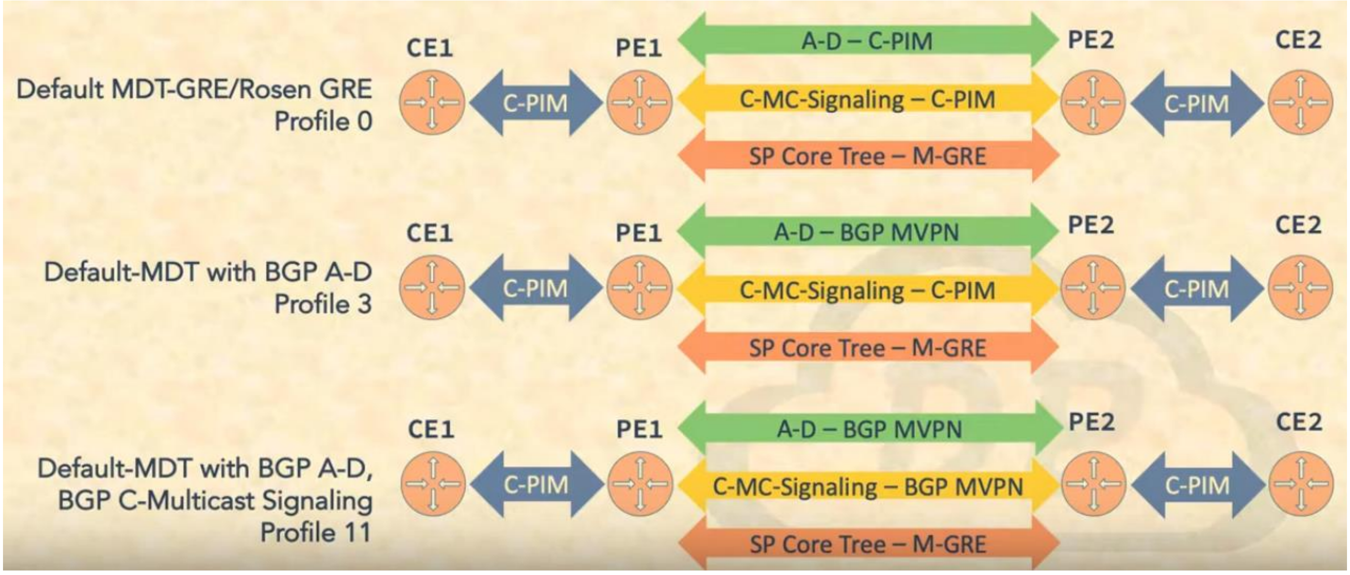
Hence, it is a must to configure strict RPF Check for the mVPN deployment models that uses Partitioned MDT.

Profile 10: VRF Static - P2MP TE - BGP-AD

It must be noted that unlike mldp P2MP trees, which are built ‘downstream on demand’ (i.e. from the receivers toward the root PE), P2MP TE trees are built from the root toward the receivers. So if the Provider wants to provide S-PMSI (Specific-Provider Multicast Service Interface) service using P2MP RSVP-TE tunnel there is a problem. Take a look back at Fig.4 and assume that the P-Tunnel is “P2MP RSVP-TE”. After PE1 advertises a Type3 (S-PMSI) route to every PE informing them that he is the Root PE, it is unaware of the interested Receiver PE’s. So it is unable to initiate P2MP RSVP-TE to interested receivers. This problem can be solved with the help of BGP Type 4 Leaf AD route. In Fig.6, PE1 advertises BGP route Type 3 with the “Leaf Info Required” bit set. When the Receiver PEs get the route, the interested PEs (PE2 and PE3) respond with a BGP Route Type 4 Leaf AD. Once PE1 discovered the leaf(s), it signals a P2MP RSVP LSP.



Profile 11: Default MDT GRE, BGP-AD, BGP C-Mcast Signaling

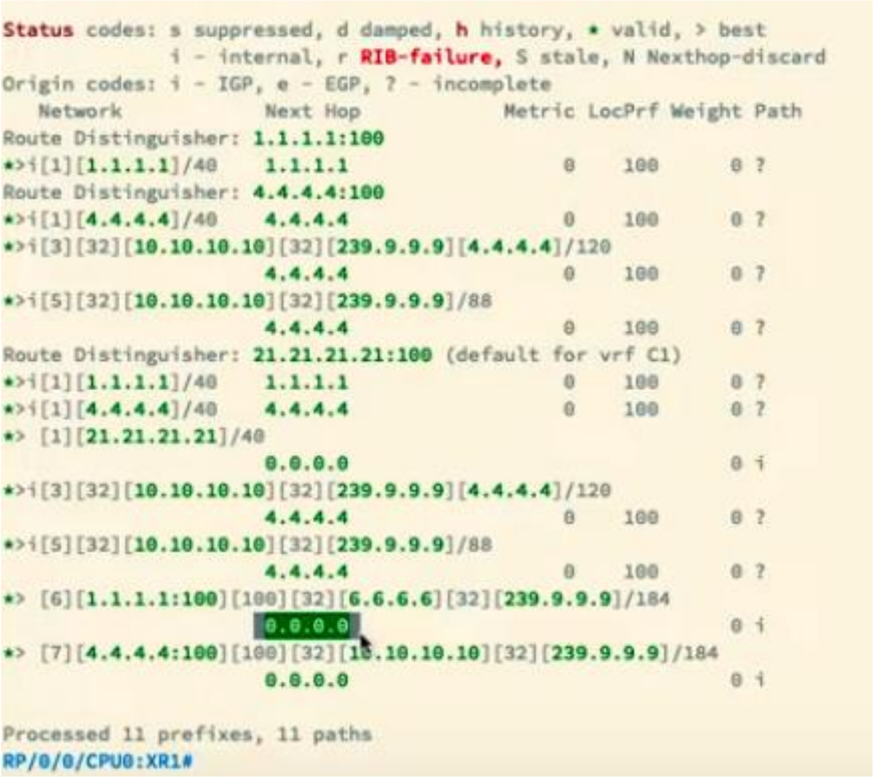

Default MDT-GRE/Rosen GRE Profile 0
CE1 PE1 PE2 CE2
C-PIM
A-D – C-PIM
C-MC-Signaling – C-PIM
SP Core Tree – M-GRE
C-PIM

Default-MDT with BGP A-D Profile 3
CE1 PE1 PE2 CE2
C-PIM
A-D – BGP MVPN
C-MC-Signaling – C-PIM
SP Core Tree – M-GRE
C-PIM

Default-MDT with BGP A-D, BGP C-Multicast Signaling Profile 11
CE1 PE1 PE2 CE2
C-PIM
A-D – BGP MVPN
C-MC-Signaling – BGP MVPN
SP Core Tree – M-GRE
C-PIM

```
vrf one
```

```
 address-family ipv4 unicast
  import route-target
   1:1
  !
  export route-target
   1:1
  !
!
router pim
 address-family ipv4
  interface Loopback0
   enable
!
interface GigabitEthernet0/0/0/3 <<< PIM is enabled for global context interface
!
!
route-policy rpf-for-one
 set core-tree pim-default       ← old Rosen gre encapsulation
end-policy
!
vrf one
 address-family ipv4
  rpf topology route-policy rpf-for-one
  mdt c-multicast-routing bgp
 !
 interface GigabitEthernet0/1/0/0
  enable
!
multicast-routing
 address-family ipv4
  interface Loopback0
   enable
 !
 interface GigabitEthernet0/0/0/3 <<< Multicast is enabled for global context intf
  enable
!
mdt source Loopback0
!
vrf one
 address-family ipv4
  mdt source Loopback0
   mdt data 232.100.100.0/24
   mdt default ipv4 232.100.1.1
  rate-per-route
  interface all enable
  bgp auto-discovery pim         ← not sure about this command … what happens ?
  !
  accounting per-prefix
!
```



```
Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 1.1.1.1:100
*>i[1][1.1.1.1]/40    1.1.1.1                  0    100      0 ?
Route Distinguisher: 4.4.4.4:100
*>i[1][4.4.4.4]/40    4.4.4.4                  0    100      0 ?
*>i[3][32][10.10.10.10][32][239.9.9.9][4.4.4.4]/120
                      4.4.4.4                  0    100      0 ?
*>i[5][32][10.10.10.10][32][239.9.9.9]/88
                      4.4.4.4                  0    100      0 ?
Route Distinguisher: 21.21.21.21:100 (default for vrf C1)
*>i[1][1.1.1.1]/40    1.1.1.1                  0    100      0 ?
*>i[1][4.4.4.4]/40    4.4.4.4                  0    100      0 ?
*> [1][21.21.21.21]/40
                      0.0.0.0                                0 i
*>i[3][32][10.10.10.10][32][239.9.9.9][4.4.4.4]/120
                      4.4.4.4                  0    100      0 ?
*>i[5][32][10.10.10.10][32][239.9.9.9]/88
                      4.4.4.4                  0    100      0 ?
*> [6][1.1.1.1:100][100][32][6.6.6.6][32][239.9.9.9]/184
                      0.0.0.0                                0 i
*> [7][4.4.4.4:100][100][32][10.10.10.10][32][239.9.9.9]/184
                      0.0.0.0                                0 i

Processed 11 prefixes, 11 paths
RP/0/0/CPU0:XR1#
```

The above output is for “show bgp ipv4 mvpn”, follow hereafter routes explanation.

`[3][32][10.10.10.10][32][239.9.9.9][4.4.4.4]/120`

[3] route-type 3, case 1: Selective-Provider Multicast Service Interface to inform the receiving router (XR1) in this example to join the specific source (S,G) over the data mdt. Since the same message could be used for IpV6, probably 32 are the two addresses’ length in bits.

[32][10.10.10.10]	source multicast address to be joined
[32][239.9.9.9]	destination multicast group to be joined
[4.4.4.4]	the router announcing the network, toward which the SSM must be sent

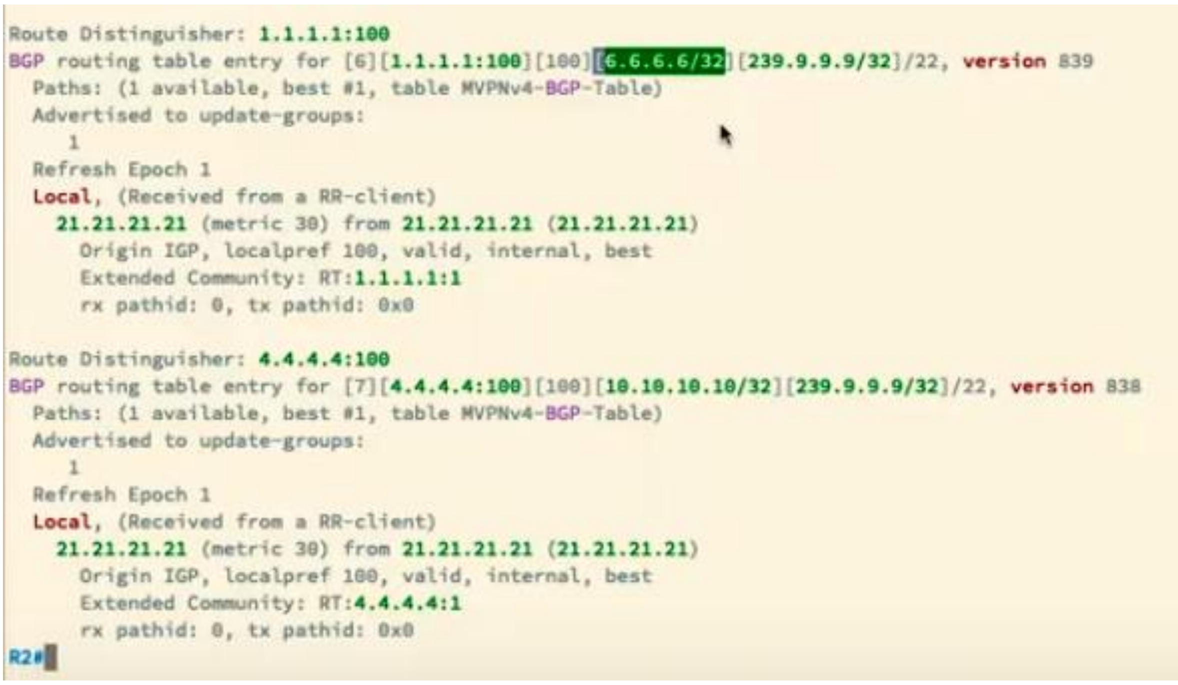
`[3][32][10.10.10.10][32][239.9.9.9]/88`

`[6][1.1.1.1:100][100][32][6.6.6.6][32][239.9.9.9]/184`

[6]	PIM join/prune message for a shared tree
[1.1.1.1:100]	route distinguisher or RD
[100]	source Autonomous System (useful in case of inter-as multicast)
[32][6.6.6.6]	RP (Rendevouz Point) ip address
[32][239.9.9.9]	multicast group for which the RP is the above one

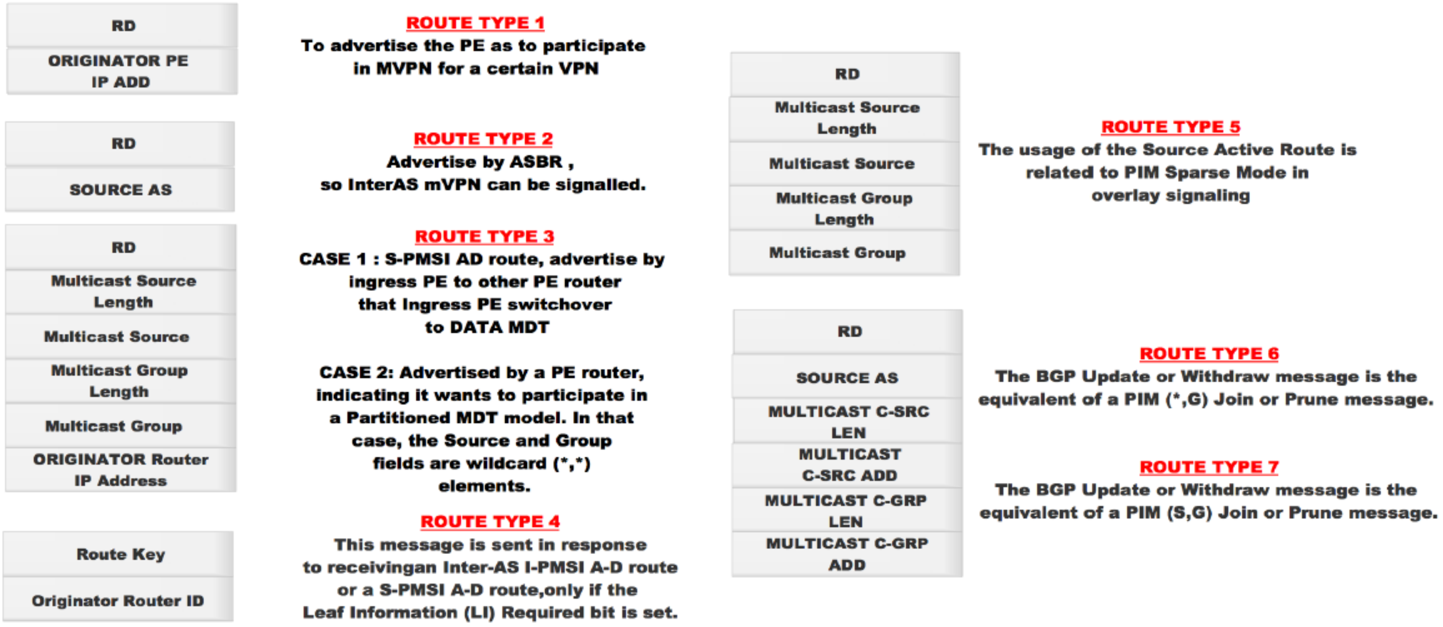
`[7][4.4.4.4:100][100][32][10.10.10.10][32][239.9.9.9]/184`

[7]	
[4.4.4.4:100]	route distinguisher or RD
[100]	source Autonomous System (useful in case of inter-as multicast)
[32][10.10.10.10]	source ip address
[32][239.9.9.9]	multicast group for the source



```
Route Distinguisher: 1.1.1.1:100
BGP routing table entry for [6][1.1.1.1:100][100][6.6.6.6/32][239.9.9.9/32]/22, version 839
  Paths: (1 available, best #1, table MVPNv4-BGP-Table)
  Advertised to update-groups:
     1
  Refresh Epoch 1
  Local, (Received from a RR-client)
    21.21.21.21 (metric 30) from 21.21.21.21 (21.21.21.21)
      Origin IGP, localpref 100, valid, internal, best
      Extended Community: RT:1.1.1.1:1
      rx pathid: 0, tx pathid: 0x0

Route Distinguisher: 4.4.4.4:100
BGP routing table entry for [7][4.4.4.4:100][100][10.10.10.10/32][239.9.9.9/32]/22, version 838
  Paths: (1 available, best #1, table MVPNv4-BGP-Table)
  Advertised to update-groups:
     1
  Refresh Epoch 1
  Local, (Received from a RR-client)
    21.21.21.21 (metric 30) from 21.21.21.21 (21.21.21.21)
      Origin IGP, localpref 100, valid, internal, best
      Extended Community: RT:4.4.4.4:1
      rx pathid: 0, tx pathid: 0x0
R2#
```

Beware to the extended community RT for the last two routes … this is a filtering mechanism to avoid remote PEs importing unnecessary routes. Tricky stuff …



RD
ORIGINATOR PE IP ADD

ROUTE TYPE 1
To advertise the PE as to participate in MVPN for a certain VPN

RD
SOURCE AS

ROUTE TYPE 2
Advertise by ASBR , so InterAS mVPN can be signalled.

RD
Multicast Source Length
Multicast Source
Multicast Group Length
Multicast Group
ORIGINATOR Router IP Address

ROUTE TYPE 3
CASE 1 : S-PMSI AD route, advertise by ingress PE to other PE router that Ingress PE switchover to DATA MDT

CASE 2: Advertised by a PE router, indicating it wants to participate in a Partitioned MDT model. In that case, the Source and Group fields are wildcard (*,*) elements.

Route Key
Originator Router ID

ROUTE TYPE 4
This message is sent in response to receivingan Inter-AS I-PMSI A-D route or a S-PMSI A-D route,only if the Leaf Information (LI) Required bit is set.

RD
Multicast Source Length
Multicast Source
Multicast Group Length
Multicast Group

ROUTE TYPE 5
The usage of the Source Active Route is related to PIM Sparse Mode in overlay signaling

RD
SOURCE AS
MULTICAST C-SRC LEN
MULTICAST C-SRC ADD
MULTICAST C-GRP LEN
MULTICAST C-GRP ADD

ROUTE TYPE 6
The BGP Update or Withdraw message is the equivalent of a PIM (*,G) Join or Prune message.

ROUTE TYPE 7
The BGP Update or Withdraw message is the equivalent of a PIM (S,G) Join or Prune message.

Profile 12: Default MDT, MLDP - P2MP, BGP-AD, BGP C-Mcast Signaling

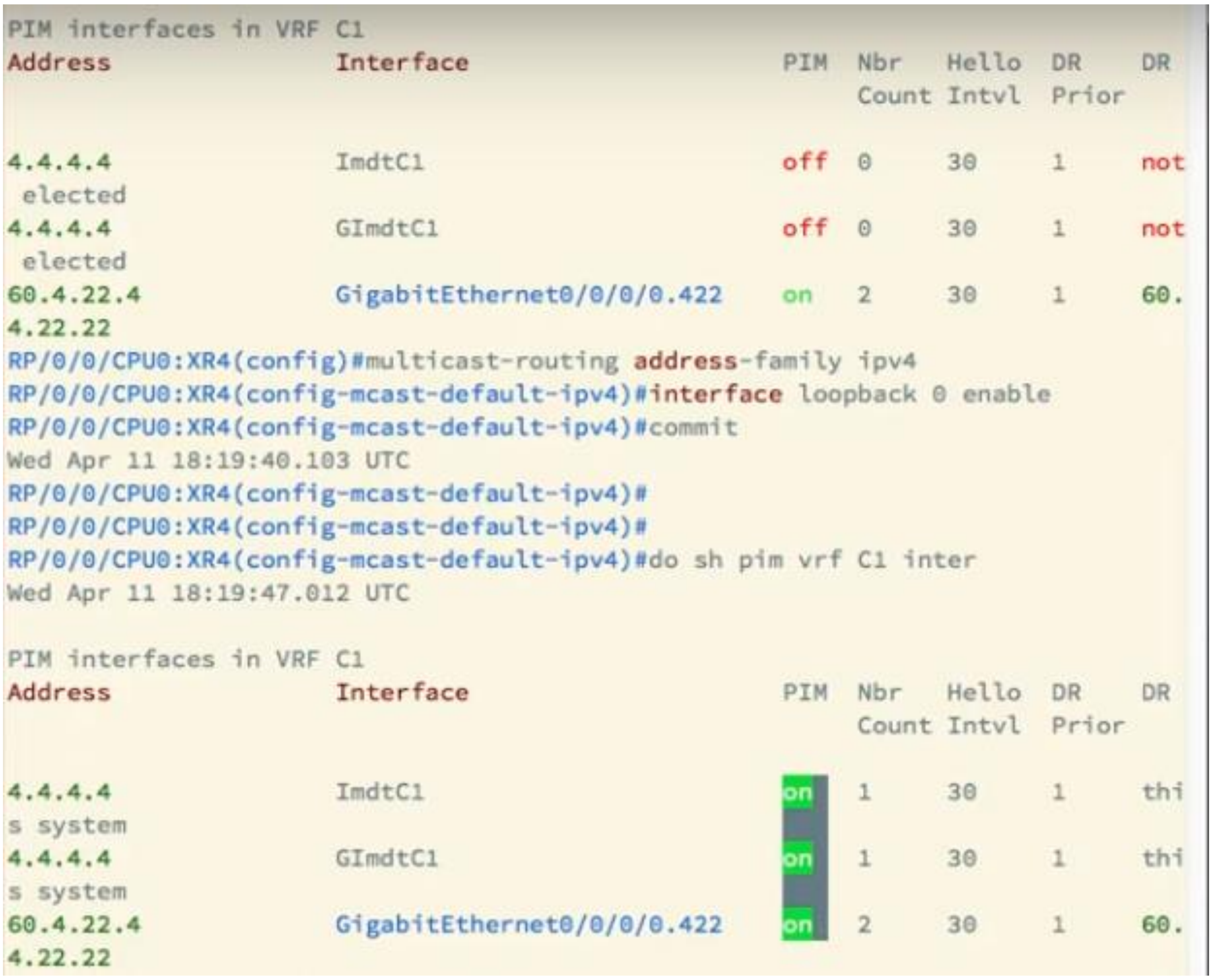
This profile is very similar to **profile 5**, except that customer’s multicast signaling is translated and propagated through bgp. Mldp is used inside the core instead of a native pim tree based on GRE encapsulation. Thus there will be bgp auto-discovery to dynamically learn all the PEs belonging to the same multicast vrf, and each of them will instantiate a p2mp lsp to send messages to all the other PEs. The N trees together form an ‘ethernet LAN’ sharing the same broadcast domain. Optionally mdt data trees can be allowed to optimize bandwidth consumption in the core.

```
vrf one
 address-family ipv4 unicast
  import route-target
```

```
   1:1
  !
  export route-target
   1:1
 !
!
route-policy rpf-for-one
 set core-tree mldp-default
end-policy
!
router pim
 vrf one
  address-family ipv4
   rpf topology route-policy rpf-for-one
   mdt c-multicast-routing bgp
  !
  interface GigabitEthernet0/1/0/0     ← phy to CE or loopback
   enable
!
multicast-routing
 vrf one
  address-family ipv4
   mdt source Loopback0
    mdt default mldp p2mp
    mdt data 100
   rate-per-route
   interface all enable
   bgp auto-discovery mldp
   !
   accounting per-prefix
  !
 !
!
mpls ldp
 mldp
  logging notifications
  address-family ipv4
!
```

Troubleshooting commands

```
show pim <vrf name> topology

! if the following command shows ‘Off’ pim on the overlay interfaces, you MUST enable
! loopback0 for multicast on the GLOBAL TABLE
show pim <vrf name> interfaces

show bgp ipv4 mvpn
```



```
PIM interfaces in VRF C1
Address              Interface                PIM  Nbr   Hello  DR    DR
                                                   Count Intvl  Prior

4.4.4.4              ImdtC1                   off  0     30     1     not
 elected
4.4.4.4              GImdtC1                  off  0     30     1     not
 elected
60.4.22.4            GigabitEthernet0/0/0/0.422  on   2     30     1     60.
4.22.22
RP/0/0/CPU0:XR4(config)#multicast-routing address-family ipv4
RP/0/0/CPU0:XR4(config-mcast-default-ipv4)#interface loopback 0 enable
RP/0/0/CPU0:XR4(config-mcast-default-ipv4)#commit
Wed Apr 11 18:19:40.103 UTC
RP/0/0/CPU0:XR4(config-mcast-default-ipv4)#
RP/0/0/CPU0:XR4(config-mcast-default-ipv4)#
RP/0/0/CPU0:XR4(config-mcast-default-ipv4)#do sh pim vrf C1 inter
Wed Apr 11 18:19:47.012 UTC

PIM interfaces in VRF C1
Address              Interface                PIM  Nbr   Hello  DR    DR
                                                   Count Intvl  Prior

4.4.4.4              ImdtC1                   on   1     30     1     thi
s system
4.4.4.4              GImdtC1                  on   1     30     1     thi
s system
60.4.22.4            GigabitEthernet0/0/0/0.422  on   2     30     1     60.
4.22.22
```

```
show mpls ldp database
```

Troubleshooting Multicast Routing

Abstract

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

Common Reasons for Multicast Problems

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

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

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

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

PIM SM Troubleshooting Routine

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

- **Step 1:** Ensure RP information propagation through the topology. First, confirm that the BSR/MA hears all the candidate RP announcements. If the MA/BSR collects all the information, make sure there is an RP for group “G” using the command **show ip pim rp mapping**. After this, proceed to every router in the domain, starting with the ones closest to BSR/MA and check that they have the RP mapping information. Refer to **Troubleshooting Auto-RP** and **Troubleshooting PIM BSR** for detailed techniques on fixing the Auto-RP/BSR problems.
- **Step 2:** Ensure all receivers “R” have joined the group “G”. Use the command **show ip igmp groups** to validate this. If you don’t have actual receivers, simulate them on the routers using the command **ip igmp join**. Next, from every leaf router that has a receiver attached, issue the **mtrace** command back to the RP address. Use the command **mtrace [RP-IP-Address]** to accomplish this. For more information about the **mtrace** command, refer to the section **Understanding the *mtrace* command**. If you can successfully **mtrace** back to the RP, this means the leaf router is able to join the (*,G) tree for this RP. If the **mtrace** breaks at some point, this most likely means an RPF failure occurs at this point. Read the **Resolving RPF failures in control plane** for information on fixing this problem. Fix all RPF problems so that receiver can join the RP-tree.
- **Step 3:** Ensure that DR can reach the RP using unicast packet exchange. Ping the RP address off the DR address for the source segment. Remember you can change the registration source address on the DR using the command **ip pim register-source [interface]**
- **Step 4:** This step should be performed on the leaf routers connecting the receivers “R” or emulating the receivers themselves. Use the **mtrace** command and trace the route back to the “S” addresses. This procedure ensures that every leaf node is able to switch to the shortest-tree upon receiving

the first packet down the RP-tree. This operation is not required if you are not using the SPTs, i.e. if the command **ip pim spt-threshold infinity** is used at the respective leaf router.

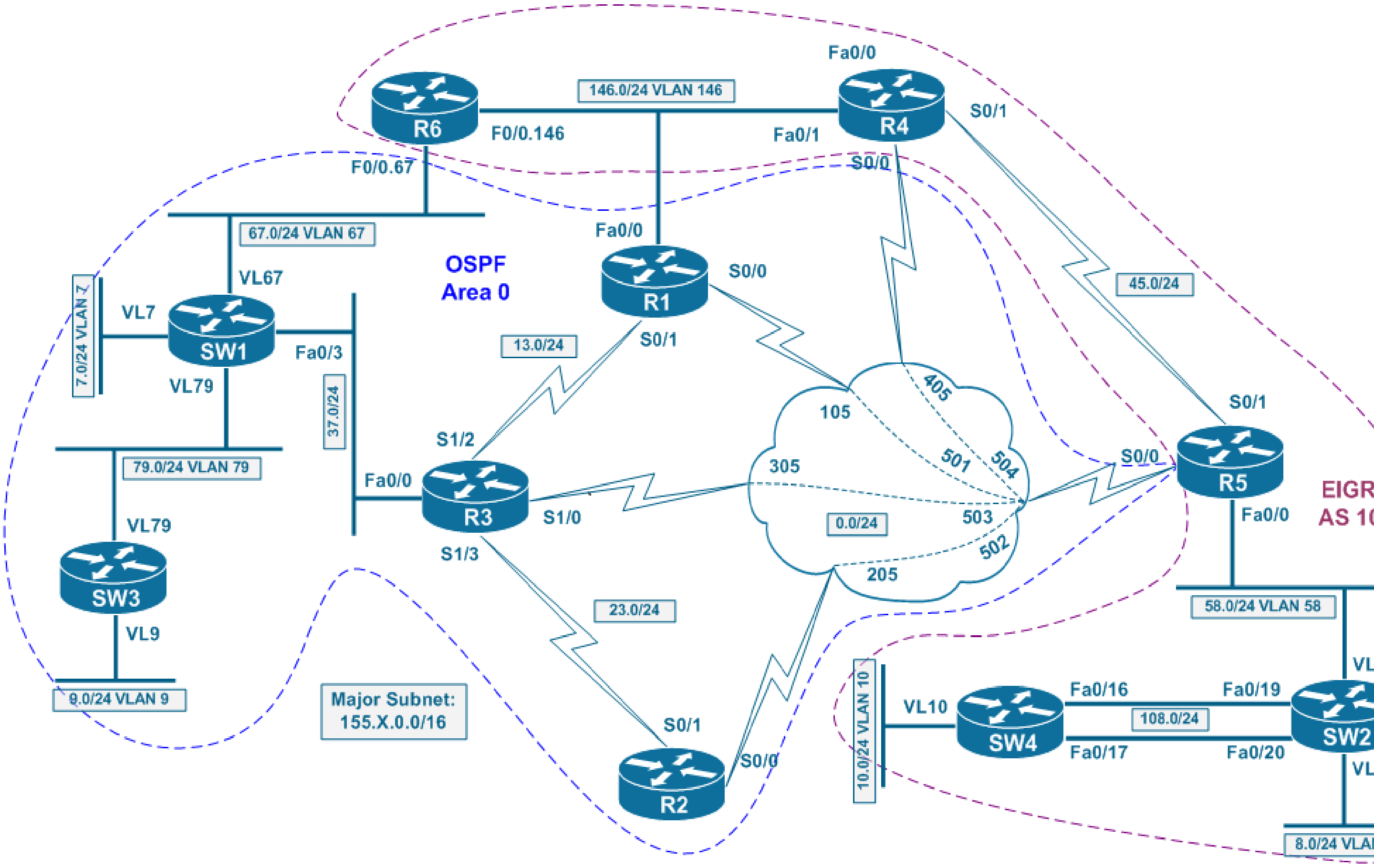
- **Step 5:** Simulate ICMP echo traffic from the source to the multicast group "G" either from the actual sources "S" or from the routers attached directly to the sources and ensure you are receiving response from every receiver "R". In some cases, you are not required to receive the replies back, but only make traffic flow down to the receivers. It is more common, however, to ensure bi-directional traffic exchange between the senders and receivers. If this step fails, follow the procedures outlined in the **Troubleshooting multicast data-plane failures** section.

Understanding the *mtrace* command

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

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

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



```
Rack29R5#mtrace 150.29.10.10

Type escape sequence to abort.

mtrace from 150.29.10.10 to 155.29.58.5 via RPF

From source (?) to destination (?)

Querying full reverse path...

 0  155.29.58.5

-1  155.29.58.5 PIM  [150.29.10.0/24]

-2  155.29.58.8 PIM  [150.29.10.0/24]

-3  155.29.108.10 PIM  [150.29.10.0/24]

-4  150.29.10.10
```

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

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

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

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

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

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

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

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

Resolving RPF failures in control plane

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

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

Troubleshooting Auto-RP

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

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

```
Rack29R5#show ip pim autorp
AutoRP Information:
  AutoRP is enabled.
PIM AutoRP Statistics: Sent/Received

  RP Announce: 0/0, RP Discovery: 0/0
```

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

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

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

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

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

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

As a heads-up, since Auto-RP uses PIM DM (typically) for multicast flooding it often has issues when running on NBMA segments. Common problems include inability of PIM DM to pass traffic across the hub in the hub-and-spoke topology due to the split-horizon forwarding problem and improper PIM Assert winner selection on the NBMA segment. If the MA is located at any of the spokes, the Auto-RP discovery messages will not make it through to the other spokes – the solution is either moving the MA to the hub, or using tunnel interfaces from hub to “other” spokes. As an alternative to using the tunnels, in production network you may want to split the multipoint interface to the collection of logical point-to-point subinterfaces. Frame-Relay implementation in

Cisco IOS even supports the use of the same address on different subinterfaces to support such “splitting” procedure. As for PIM Assert winning, make sure the hub router, or the router having direct layer 2 connections with every other node on the NBMA segment always has the best PIM priority on the segment. This will prevent a non-fully connected node from being elected as Assert winner by mistake and improper segment flooding.

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

Troubleshooting PIM BSR

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

- **Step 1:** Ensure that every RP in the domain learns of the BSR. Use the command **show ip pim bsr-router** to validate this. If some RPs appear to be missing this information, perform **mroute** off the candidate RP back to the BSR IP address and see if there are any problems on the path. Fix any problems detected by **mtrace** using the RPF fixup tools described above. If the problem persists, use the command **debug ip pim bsr** on all routers on the reverse path to the BSR and see if there are any RPF failure messages similar to the one below:
-
  ```
  PIM-BSR(0): bootstrap (150.29.3.3) on non-RPF path Serial0/0/0 or from
  non-RPF neighbor 155.29.45.4 discarded
  ```

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

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

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

Troubleshooting Multicast Data Plane

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

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

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

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

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

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

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

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

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

Case 2: Shortest Tree Failure

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

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

Decoding the *show ip mroute* command output

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

```
Rack29SW3#show ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry,
       X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group
       V - RD & Vector, v - Vector
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode
```

```
(*, 224.0.1.40), 00:03:53/00:02:11, RP 0.0.0.0, flags: DCL

  Incoming interface: Null, RPF nbr 0.0.0.0

  Outgoing interface list:

    Vlan79, Forward/Sparse, 00:03:52/00:02:11
```

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

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

```
Rack29R6#show ip mroute 224.0.1.39

IP Multicast Routing Table
...
(*, 224.0.1.39), 00:11:44/stopped, RP 0.0.0.0, flags: D

  Incoming interface: Null, RPF nbr 0.0.0.0

  Outgoing interface list:

    FastEthernet0/0.146, Forward/Sparse-Dense, 00:11:44/00:00:00

(150.29.6.6, 224.0.1.39), 00:11:44/00:02:56, flags: T

  Incoming interface: Loopback0, RPF nbr 0.0.0.0

  Outgoing interface list:

    FastEthernet0/0.146, Forward/Sparse-Dense, 00:11:44/00:00:00
```

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

```
Rack29R4#show ip mroute 224.0.1.39

IP Multicast Routing Table
…
(*, 224.0.1.39), 21:24:46/stopped, RP 0.0.0.0, flags: D

  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/1/0, Forward/Sparse, 21:24:46/00:00:00
    FastEthernet0/1, Forward/Sparse-Dense, 21:24:46/00:00:00
(150.29.6.6, 224.0.1.39), 21:21:42/00:02:55, flags: T
  Incoming interface: FastEthernet0/1, RPF nbr 155.29.146.6
  Outgoing interface list:
    Serial0/1/0, Forward/Sparse, 21:21:42/00:00:00
```

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

```
Rack29R4#show ip mroute 224.0.1.40
IP Multicast Routing Table
...
(*, 224.0.1.40), 21:35:41/stopped, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/1/0, Forward/Sparse, 21:35:41/00:00:00
    FastEthernet0/1, Forward/Sparse, 21:35:41/00:00:00
(150.29.5.5, 224.0.1.40), 00:02:26/00:00:33, flags: L
  Incoming interface: Null, RPF nbr 155.29.0.5
  Outgoing interface list:
    FastEthernet0/1, Forward/Sparse, 00:02:26/00:00:00
    Serial0/1/0, Forward/Sparse, 00:02:26/00:00:00
```

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

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

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

```
Rack29R4#show ip mroute 224.0.1.40
IP Multicast Routing Table
...
(150.29.5.5, 224.0.1.40), 00:01:44/00:01:15, flags: L
  Incoming interface: Serial0/1/0, RPF nbr 155.29.45.5, mroute
  Outgoing interface list:
    FastEthernet0/1, Forward/Sparse, 00:01:44/00:00:00
```

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

Summary

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