Multicast Service Reflection using PIM-SM on IOS-XE : Multicast to Unicast

Contents

Introduction

The purpose of this article is to give you an understanding of the basic working of MSR (Multicast Service Replication) using IOS-XE platforms, through the form of a configuration lab guide.

Prerequisites

Requirements

Basic understanding of PIM-SM

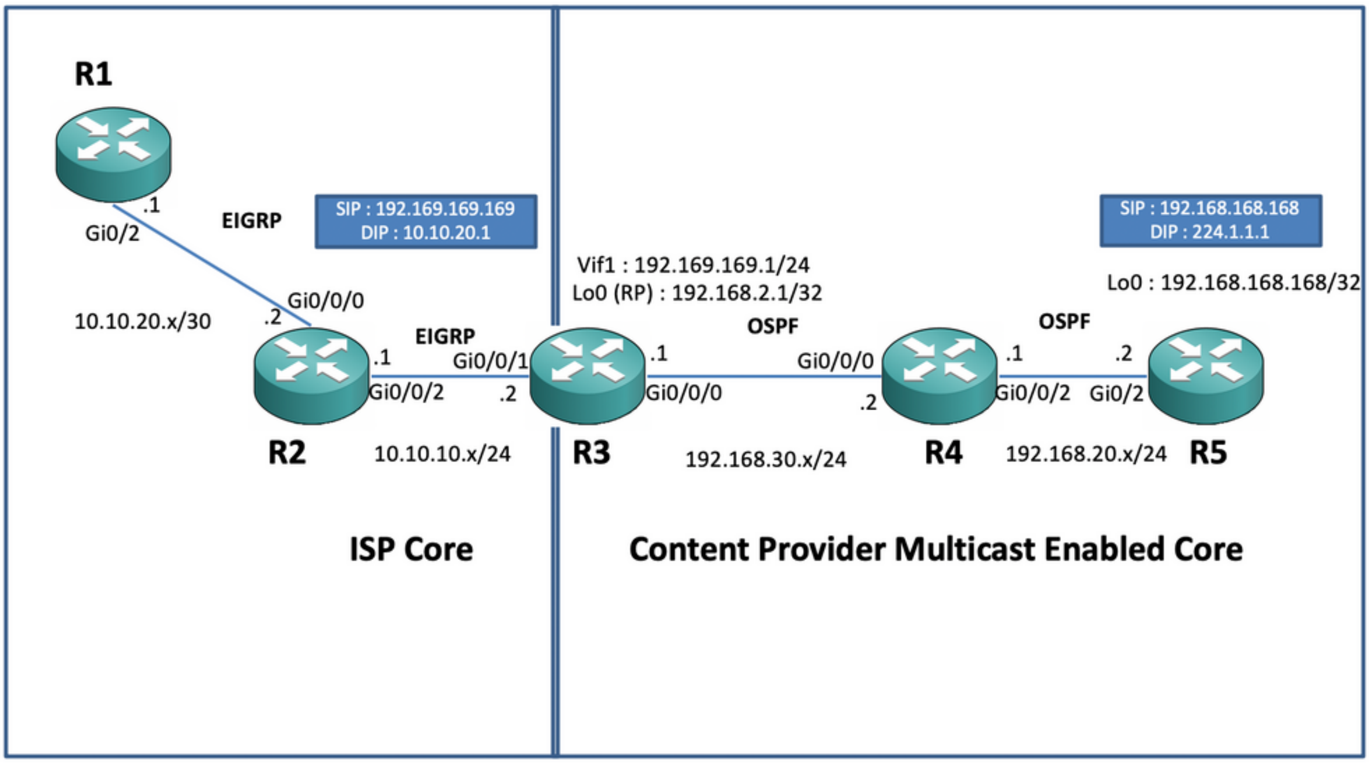
Components Used

ASR1000 (R2&R4), ISR4300 (R3), ISR2900 (R1&R5)

Configure

We would show below end-to-end configurations based on the below diagrammatic scenario for trnaslating the multicast.

Network Diagram



Configurations

In the above diagram, the node R1 acts as the receiver which is supposed to only get unicast multicast data feed from the multicast source.

The node R5 acts as the multicast source which generates multicast ICMP traffic sourced from its loopback 0 interface.

The node R2 is under the Content providers multicast core domain, and is running PIM-SM with underlay of OSPF.

The node R3 acts as the router which runs Multicast Service Replication Application and is in this case the mutlicast border router from which point the multicast data traffic is supposed to get translated into a unicast data packet towards the receiver. It uses OSPF and EIGRP with the Content Provider and ISP respectively and it houses the RP (Rendezvouz Point) on its loopback interface in the multicast core domain.

The node R4 is under the ISP Core control and is not multicast enabled and only understands how to reach the R3 node using underlay EIGRP routing.

Below, you can find the relevant configurations present on the nodes present in the above topology diagram :

R1:

```
! no ip domain lookup ip cef no ipv6 cef ! interface GigabitEthernet0/2 ip address 10.10.20.1
255.255.255.0 duplex auto speed auto end ! router eigrp 100 network 10.10.20.0 0.0.0.255 !
```

R2:

```
! interface GigabitEthernet0/0/0 ip address 10.10.20.2 255.255.255.0 negotiation auto !
```

```
interface GigabitEthernet0/0/2 ip address 10.10.10.1 255.255.255.0 negotiation auto ! router
eigrp 100 network 10.10.10.0 0.0.0.255 network 10.10.20.0 0.0.0.255 !
```

R3:

```
! ip multicast-routing distributed ! interface Loopback0 ip address 192.168.2.1 255.255.255.255
ip pim sparse-mode ip ospf 1 area 0 ! interface GigabitEthernet0/0/0 ip address 192.168.30.1
255.255.255.0 ip pim sparse-mode ip ospf 1 area 0 negotiation auto ! interface
GigabitEthernet0/0/1 ip address 10.10.10.2 255.255.255.0 negotiation auto ! interface Vif1 ip
address 192.169.169.1 255.255.255.0 ip pim sparse-mode ip service reflect GigabitEthernet0/0/0
destination 224.1.1.0 to 10.10.20.0 mask-len 24 source 192.169.169.169 <<<< ip igmp static-group
224.1.1.1 ip ospf 1 area 0 ! router eigrp 100 network 10.10.10.0 0.0.0.255 ! router ospf 1 ! ip
pim rp-address 192.168.2.1 !
```

R4:

```
! ip multicast-routing distributed ! interface GigabitEthernet0/0/0 ip address 192.168.30.2
255.255.255.0 ip pim sparse-mode ip ospf 1 area 0 negotiation auto ! interface
GigabitEthernet0/0/2 ip address 192.168.20.1 255.255.255.0 ip pim sparse-mode ip ospf 1 area 0
negotiation auto ! router ospf 1 ! ip pim rp-address 192.168.2.1 !
```

R5:

```
! ip multicast-routing ip cef no ipv6 cef ! interface Loopback0 ip address 192.168.168.168
255.255.255.255 ip pim sparse-mode ip ospf 1 area 0 ! interface GigabitEthernet0/2 ip address
192.168.20.2 255.255.255.0 ip pim sparse-mode ip ospf 1 area 0 duplex auto speed auto ! router
ospf 1 ! ip pim rp-address 192.168.2.1 !
```

Verify

We can validate the configurations by performing a test ping for simulating multicast traffic from the R5 router with a source of its loopback 0 interface [192.168.168.168] destined to the multicast address 224.1.1.1. Then check the mroute entries on the node which is running the MSR application i.e. R3 :

```
R5(config)#do ping 224.1.1.1 sou lo 0 rep 10000000 Type escape sequence to abort. Sending
10000000, 100-byte ICMP Echos to 224.1.1.1, timeout is 2 seconds: Packet sent with a source
address of 192.168.168.168 ............................
```

```
R3#sh ip mroute 224.1.1.1 IP Multicast Routing Table Flags: D - Dense, S - Sparse, B - Bidir
Group, s - SSM Group, C - Connected, L - Local, P - Pruned, R - RP-bit set, F - Register flag, T
- SPT-bit set, J - Join SPT, M - MSDP created entry, 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, c - PFP-SA cache created entry Outgoing
interface flags: H - Hardware switched, A - Assert winner, p - PIM Join Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode (*, 224.1.1.1), 00:47:41/stopped, RP
192.168.2.1, flags: SJC Incoming interface: Null, RPF nbr 0.0.0.0 Outgoing interface list: Vif1,
Forward/Sparse, 00:46:36/00:01:23 <<<< (192.168.168.168, 224.1.1.1), 00:00:20/00:02:43, flags: T
Incoming interface: GigabitEthernet0/0/0, RPF nbr 192.168.30.2 Outgoing interface list: Vif1,
Forward/Sparse, 00:00:20/00:02:39 <<<<
```

```
R3#sh ip mroute 224.1.1.1 count Use "show ip mfib count" to get better response time for a large
number of mroutes. IP Multicast Statistics 3 routes using 2938 bytes of memory 2 groups, 0.50
average sources per group Forwarding Counts: Pkt Count/Pkts per second/Avg Pkt Size/Kilobits per
second Other counts: Total/RPF failed/Other drops(OIF-null, rate-limit etc) Group: 224.1.1.1,
Source count: 1, Packets forwarded: 1455, Packets received: 1458 <<<< RP-tree: Forwarding:
1/0/100/0, Other: 1/0/0 Source: 192.168.168.168/32, Forwarding: 1454/1/113/0, Other: 1457/3/0
R3#sh ip mroute 224.1.1.1 count Use "show ip mfib count" to get better response time for a large
number of mroutes. IP Multicast Statistics 3 routes using 2938 bytes of memory 2 groups, 0.50
average sources per group Forwarding Counts: Pkt Count/Pkts per second/Avg Pkt Size/Kilobits per
second Other counts: Total/RPF failed/Other drops(OIF-null, rate-limit etc) Group: 224.1.1.1,
Source count: 1, Packets forwarded: 1465, Packets received: 1468 <<<< RP-tree: Forwarding:
1/0/100/0, Other: 1/0/0 Source: 192.168.168.168/32, Forwarding: 1464/1/113/0, Other: 1467/3/0
```

Also, you can take captures to verify that the packets are indeed getting translated to the intended unicast destination address on the R2 node by using EPC (Embedded Packet Capture) feature on the IOS-XE router :

```
R2#mon cap TAC int gi 0/0/2 both match any R2#mon cap TAC buff siz 50 circular R2#mon cap TAC
start Started capture point : TAC R2# *Aug 12 06:50:40.195: %BUFCAP-6-ENABLE: Capture Point TAC
enabled. R2#sh mon cap TAC buff br | i ICMP 6 114 10.684022 192.169.169.169 -> 10.10.20.1 0 BE
ICMP <<<< 7 114 10.684022 192.169.169.169 -> 10.10.20.1 0 BE ICMP <<<< 8 114 12.683015
192.169.169.169 -> 10.10.20.1 0 BE ICMP <<<< 9 114 12.683015 192.169.169.169 -> 10.10.20.1 0 BE
ICMP <<<<
```

Here, the important point to note is that regularly when you perform multicast ICMP pings in "lab environments", you would usually expect that you receive back the ICMP echo reply packets back from the receiver side towards the source, assuming that there is complete reachability between the two(source and receiver). However, in this scenario its important to note that even if we try to advertise the NATted source address for the multicast ICMP packets i.e. 192.169.169.169 all the way till the receiver i.e. R1 through EIGRP, still the unicast ICMP echo replies wont cross the R3 router, since the reverse NAT is not configured on the MSR application node. We can test this, by trying to perform the EIGRP route advertisement of the Vif 1 interface on R3 into EIGRP (ISP Core routing) :

```
ISR4351(config)#router eigrp 100 ISR4351(config-router)#network 192.169.169.0 0.0.0.255 <<<<
```

Now, we can check in the captures taken on the R2 node on the ICMP echo replies being sent towards R3 :

```
R2#sh mon cap TAC buff br | i ICMP <snip> 249 114 317.847948 192.169.169.169 -> 10.10.20.1 0 BE
ICMP 250 114 317.847948 192.169.169.169 -> 10.10.20.1 0 BE ICMP 251 114 317.847948 10.10.20.1 ->
192.169.169.169 0 BE ICMP <<<< 252 114 317.847948 10.10.20.1 -> 192.169.169.169 0 BE ICMP <<<<
253 114 319.847948 192.169.169.169 -> 10.10.20.1 0 BE ICMP 254 114 319.847948 192.169.169.169 ->
10.10.20.1 0 BE ICMP 255 114 319.848955 10.10.20.1 -> 192.169.169.169 0 BE ICMP <<<< 256 114
319.848955 10.10.20.1 -> 192.169.169.169 0 BE ICMP <<<< 259 114 321.848955 192.169.169.169 ->
10.10.20.1 0 BE ICMP 260 114 321.848955 192.169.169.169 -> 10.10.20.1 0 BE ICMP 261 114
321.848955 10.10.20.1 -> 192.169.169.169 0 BE ICMP <<<< 262 114 321.848955 10.10.20.1 ->
192.169.169.169 0 BE ICMP <<<<
```

But the pings would still fail as seen on the source R5 :

```
R5(config)#do ping 224.1.1.1 sou lo 0 rep 10000000 Type escape sequence to abort. Sending
10000000, 100-byte ICMP Echos to 224.1.1.1, timeout is 2 seconds: Packet sent with a source
address of 192.168.168.168
......................................................................
...................................................................... <snip>
```

Now to get the replies to reach all the way till the source, we can configure NAT port forwarding on the MSR application node R3 to translate the destined traffic towards 192.169.169.169 to 192.168.168.168, by configuring extendable NAT :

```
R3(config)#int gi 0/0/1 R3(config-if)#ip nat out R3(config-if)#int gi 0/0/0 R3(config-if)#ip nat
ins R3(config-if)#exit R3(config)#ip nat inside source static 192.168.168.168 192.169.169.169
extendable <<<<
```

Now on checking the source R5 node, we can see the response come back :

```
R5(config)#do ping 224.1.1.1 sou lo 0 rep 10000000 Type escape sequence to abort. Sending
10000000, 100-byte ICMP Echos to 224.1.1.1, timeout is 2 seconds: Packet sent with a source
address of 192.168.168.168
...................................................................... <snip> Reply to request
716 from 10.10.20.1, 1 ms Reply to request 716 from 10.10.20.1, 1 ms Reply to request 717 from
10.10.20.1, 1 ms Reply to request 717 from 10.10.20.1, 1 ms Reply to request 718 from
10.10.20.1, 1 ms Reply to request 718 from 10.10.20.1, 1 ms
```

The above was just performed to explain the packet flow and to understand how to establish the reverse unicast path/flow for the data traffic and the downstream multicast traffic. Since in the regular production scenario's, you would usually not come accross cases/instances where the multicast applications running on the server/source side require a reverse acknowledgement packets from the receivers in a unicast form.

By the above tests and validations, it should have given a brief overview on how to run multicast service replication application on one of the multicast border nodes and how to deploy the same if the same shown above were to be expanded to a large scale deployement.