Enterprise Network Virtualization (Advanced)

LTRMPL-3102

Nicolae Matau (nmatau@cisco.com)

Tanja Hess (thess@cisco.com)

Introduction:

The objective of this lab is to introduce enterprise network engineers to the advanced concepts of network virtualization

The lab has the following major components.

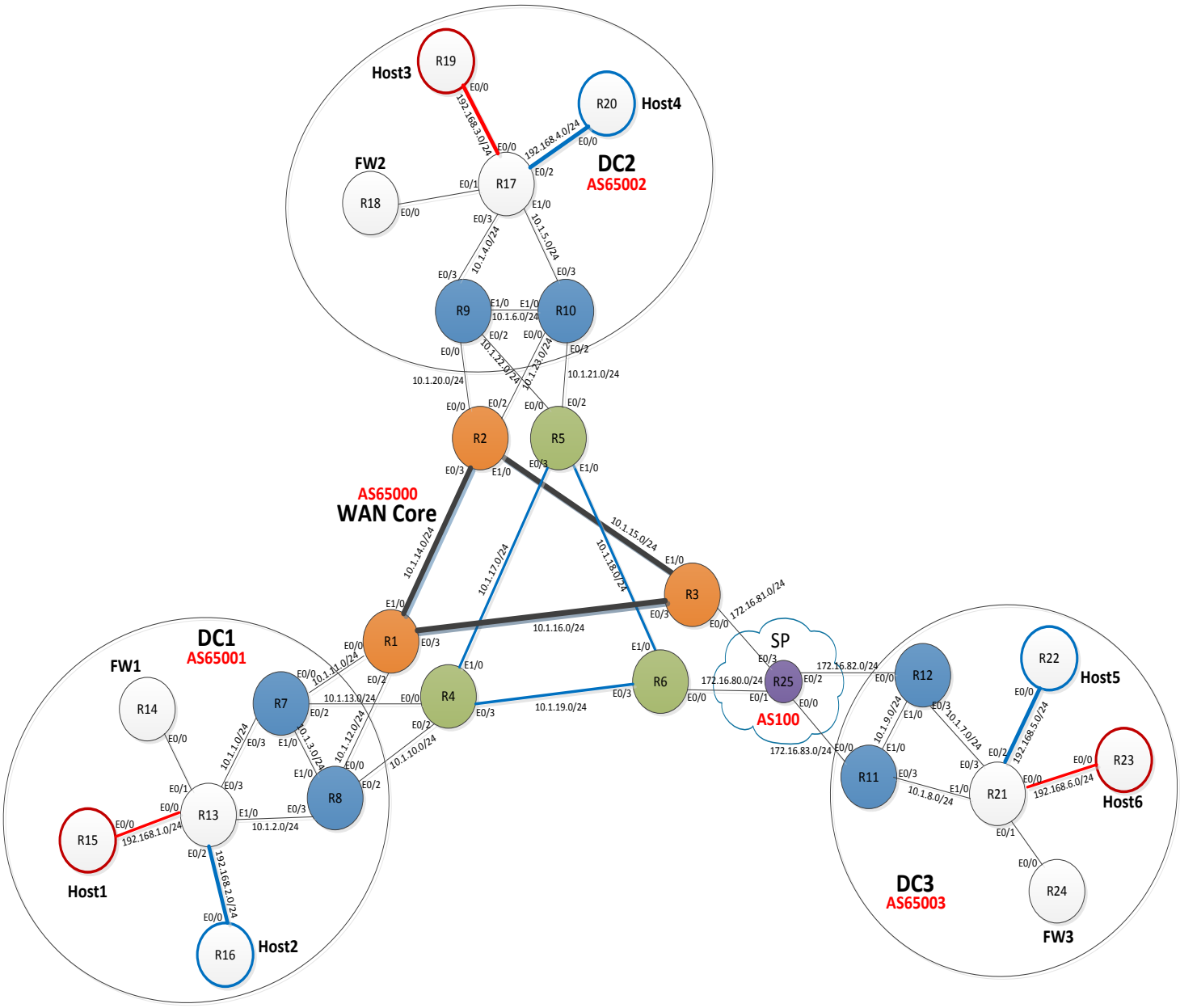
- Three site deployment
- A dual plane WAN Core with Inter-AS Option C
- Firewall at each site for inter-VRF traffic
- Ethernet over MPLS
- MPLS over GRE

This lab is not a design guide. The purpose of this lab is to teach certain elements of an overall design. Such as using VRFs, Firewalls, EoMPLS etc. These are the building blocks to a design. Many components are combined to give you exposure and hands on experience. There are other components that are also not mentioned such as QoS, management and best practices.

Prerequisites:

- Must know how to configure Cisco IOS
- Know MPLS basics
- Have attended at least Introduction to MPLS session
- Must understand basic IP routing, especially BGP

Lab Diagram:



Save your work often with a write mem

The Setup:

IP Address Assignment:

For a given subnet, the last octet of an IP address for any interface can be derived from the hostname. For example, R10 connected to 10.1.5.0/24 would be assigned the address of 10.1.5.**10**/24. The IP address on the loopback is also derived from the hostname with a prefix length of /32. For example, R10's loopback is 10.10.10.10/32.

Most links have a 255.255.255.0 mask for IPv4 except for the loopbacks which are a 255.255.255.255 for IPv4. All addresses have been pre-configured and all links are up.

In this lab, you don't need to configure or make any changes to the SP router (R25).

Lab1: Setup the IGP

The setup has an isolated OSPF domain for DC1, DC2 and DC3. As well as each WAN plane.

Enable OSPF on R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R17, and R21 using this configuration (identical on all these routers):

```
interface Loopback0
 ip ospf 100 area 0
!
interface Ethernet0/0
 ip ospf 100 area 0
!
interface Ethernet0/2
 ip ospf 100 area 0
!
interface Ethernet0/3
 ip ospf 100 area 0
!
interface Ethernet1/0
 ip ospf 100 area 0
!
router ospf 100
 passive-interface default
 no passive-interface Ethernet0/3
 no passive-interface Ethernet1/0
!
```

Verify that OSPF is coming up on the correct interfaces using the “show ip ospf neighbor” command. For example:

```
R2#sh ip ospf ne

Neighbor ID     Pri   State           Dead Time   Address         Interface
1.1.1.1           1   FULL/DR         00:00:36    10.1.14.1       Ethernet0/3
3.3.3.3           1   FULL/BDR        00:00:31    10.1.15.3       Ethernet1/0
```

And ensure that the OSPF routes are learned correctly using “show ip route ospf”:

```
R2#show ip route ospf

      1.0.0.0/32 is subnetted, 1 subnets
O        1.1.1.1 [110/11] via 10.1.14.1, 00:17:10, Ethernet0/3
      3.0.0.0/32 is subnetted, 1 subnets
O        3.3.3.3 [110/11] via 10.1.15.3, 00:12:25, Ethernet1/0
      10.0.0.0/8 is variably subnetted, 11 subnets, 2 masks
O        10.1.11.0/24 [110/20] via 10.1.14.1, 00:17:10, Ethernet0/3
O        10.1.12.0/24 [110/20] via 10.1.14.1, 00:17:10, Ethernet0/3
O        10.1.16.0/24 [110/20] via 10.1.15.3, 00:12:25, Ethernet1/0
                      [110/20] via 10.1.14.1, 00:17:10, Ethernet0/3
      172.16.0.0/24 is subnetted, 1 subnets
O        172.16.81.0 [110/20] via 10.1.15.3, 00:12:25, Ethernet1/0
```

Notice how the passive interfaces were used in this lab. That isolates OSPF to each DC from the WAN. That means each DC is an independent OSPF network.

Lab2: Implement LDP

Our goal is to enable LDP within each domain. And at the same time keep LDP contained to each domain similar to OSPF in the previous lab.

Enable LDP on R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R17, and R21 using the “mpls ip” command. Use the following configuration (same config on all the routers above):

```
mpls label protocol ldp
mpls ldp router-id Loopback0 force
!
interface Ethernet0/3
 mpls ip
!
interface Ethernet1/0
 mpls ip
!
```

Ensure that LDP came up properly using the command: “sh mpls ldp neighbor”:

```
R21#sh mpls ldp neighbor
    Peer LDP Ident: 11.11.11.11:0; Local LDP Ident 21.21.21.21:0
        TCP connection: 11.11.11.11.646 - 21.21.21.21.53043
        State: Oper; Msgs sent/rcvd: 13/13; Downstream
        Up time: 00:00:41
        LDP discovery sources:
          Ethernet1/0, Src IP addr: 10.1.8.11
        Addresses bound to peer LDP Ident:
          172.16.83.11    10.1.8.11       10.1.9.11       11.11.11.11
    Peer LDP Ident: 12.12.12.12:0; Local LDP Ident 21.21.21.21:0
        TCP connection: 12.12.12.12.646 - 21.21.21.21.37072
        State: Oper; Msgs sent/rcvd: 13/13; Downstream
        Up time: 00:00:41
        LDP discovery sources:
          Ethernet0/3, Src IP addr: 10.1.7.12
        Addresses bound to peer LDP Ident:
          172.16.82.12    10.1.7.12       10.1.9.12       12.12.12.12
```

Notice that we didn’t enable LDP between border routers (R7/R8, R9/R10, R11/R12) and the WAN routers. To achieve end to end label transport we will use BGP later.

Think of this design as a hierarchy. Each site runs independent OSPF and LDP.

Lab3: GRE Tunnels

The goal of this lab is to configure GRE tunnels between R3 and R12, and also between R6 and R11. Think of this as a way to connect DC3 back to the private WAN core over a service provider that can only provide an IP transport. In some real deployments this can be handy when bringing up a new site or when direct L1/L2 transport is not available.

Assign subnet 10.1.25.0/24 to the tunnel between R3 and R12. And assign 10.1.26.0/24 to the tunnel between R6 and R11.

R3 looks like this:

```
interface Tunnel0
 ip address 10.1.25.3 255.255.255.0
 ip ospf 100 area 0
 keepalive 3 3
 tunnel source 172.16.81.3
 tunnel destination 172.16.82.12
```

and R12 looks like this:

```
interface Tunnel0
 ip address 10.1.25.12 255.255.255.0
 ip ospf 100 area 0
 keepalive 3 3
 tunnel source 172.16.82.12
 tunnel destination 172.16.81.3
```

Do the same between R6 and R11 using subnet 10.1.26.0/24. Don't forget the "ip ospf 100 area 0" command under the tunnel.

The tunnels will show as being down since BGP to the SP (R25) is not configured yet. We will take care of that in the next lab.

Lab4: Implement BGP with Inter-AS Option C

This lab is broken into 4 sections.

1. eBGP to the SP (R25) to bring up the GRE tunnels that you setup in lab3.

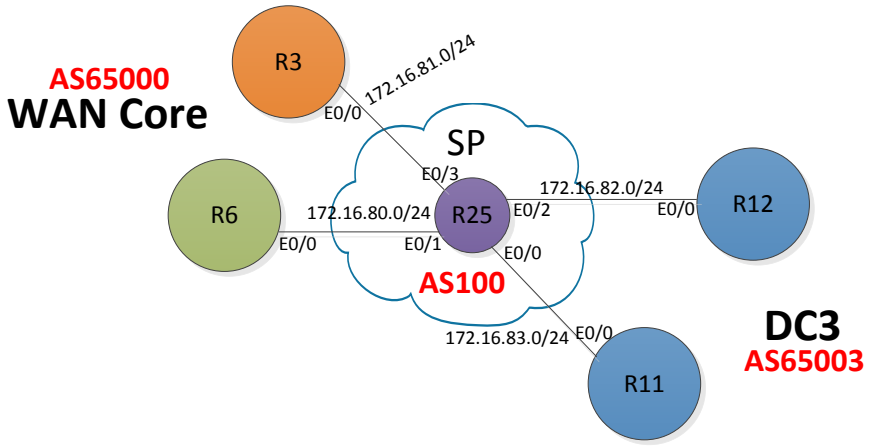
2. iBGP inside the WAN core layer (ie in between R1/R2/R3, and R4/R5/R6). Each WAN core layer is an independent OSPF, LDP and iBGP domain

3. eBGP between the DC border routers and the WAN core layer to exchange the PE loopback routes. That provides reachability from PE to PE across the private WAN core.

4. BGP between the PEs over the WAN core layer to exchange VPNv4 routes

BGP Section1:

The purpose of this section is to configure eBGP to the SP from R3, R6, R11 and R12.



To configure eBGP to the SP (R25) add the following BGP config to R6:

```
router bgp 65000
 neighbor 172.16.80.25 remote-as 100
 !
 address-family ipv4
  neighbor 172.16.80.25 activate
  neighbor 172.16.80.25 distribute-list 1 out
 exit-address-family
!
access-list 1 deny   any
```

Notice that we are using a distribute list to block all routes from being sent to the SP. We are only interested in receiving routes.

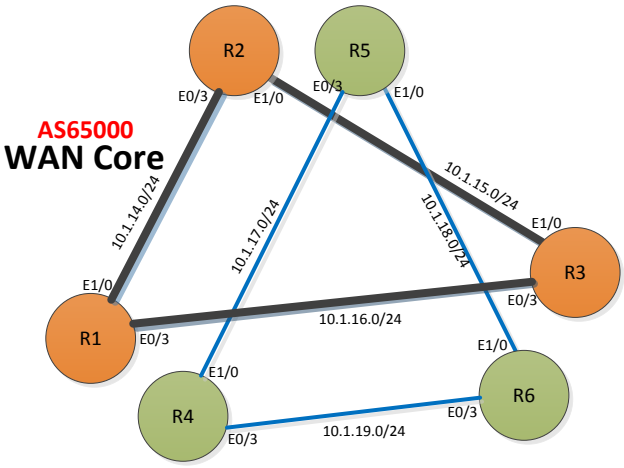
On R11 configure:

```
router bgp 65003
 neighbor 172.16.83.25 remote-as 100
 !
 address-family ipv4
  neighbor 172.16.83.25 activate
  neighbor 172.16.83.25 distribute-list 1 out
 exit-address-family
!
access-list 1 deny   any
```

Implement similar peering on R3 and R12. Verify that the BGP session came up and working using "show ip bgp summ". The GRE tunnels from previous lab should now come up.

BGP Section2:

The purpose of this lab is to configure iBGP between the routers inside the WAN core layer. Remember we have two independent WAN core layers running in parallel for HA.



On R1 configure:

```
router bgp 65000
 neighbor 2.2.2.2 remote-as 65000
 neighbor 2.2.2.2 update-source Loopback0
 neighbor 3.3.3.3 remote-as 65000
 neighbor 3.3.3.3 update-source Loopback0
 !
 address-family ipv4
  neighbor 2.2.2.2 activate
  neighbor 2.2.2.2 next-hop-self
```

```
 neighbor 2.2.2.2 send-label
 neighbor 3.3.3.3 activate
 neighbor 3.3.3.3 next-hop-self
 neighbor 3.3.3.3 send-label
exit-address-family
```

On R2 configure:

```
router bgp 65000
 neighbor 1.1.1.1 remote-as 65000
 neighbor 1.1.1.1 update-source Loopback0
 neighbor 3.3.3.3 remote-as 65000
 neighbor 3.3.3.3 update-source Loopback0
 !
 address-family ipv4
  neighbor 1.1.1.1 activate
  neighbor 1.1.1.1 next-hop-self
  neighbor 1.1.1.1 send-label
  neighbor 3.3.3.3 activate
  neighbor 3.3.3.3 next-hop-self
  neighbor 3.3.3.3 send-label
 exit-address-family
```

On R3 configure:

```
router bgp 65000
 neighbor 1.1.1.1 remote-as 65000
 neighbor 1.1.1.1 update-source Loopback0
 neighbor 2.2.2.2 remote-as 65000
 neighbor 2.2.2.2 update-source Loopback0
 !
 address-family ipv4
  neighbor 1.1.1.1 activate
  neighbor 1.1.1.1 next-hop-self
  neighbor 1.1.1.1 send-label
  neighbor 2.2.2.2 activate
  neighbor 2.2.2.2 next-hop-self
  neighbor 2.2.2.2 send-label
 exit-address-family
```

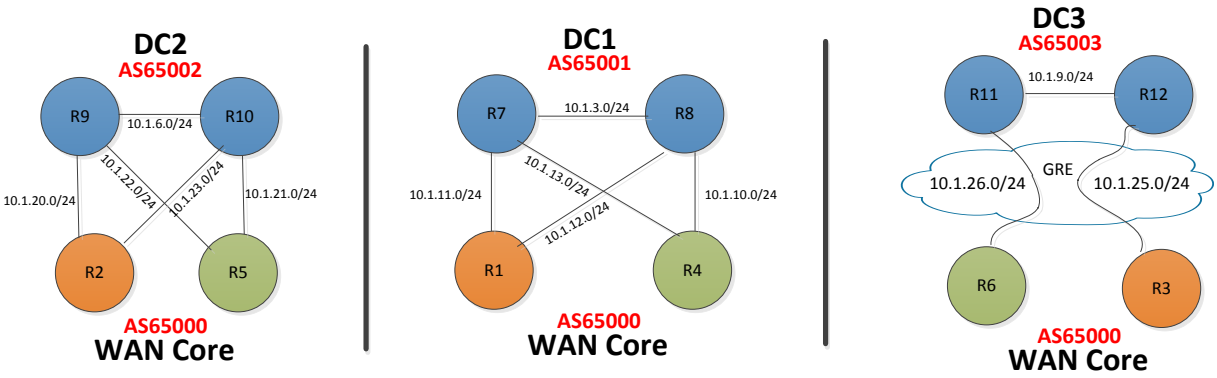
Notice that we using ‘send-label’ and ‘next-hop-self’. The purpose is to distribute a label with the BGP routes that we will receive from the DCs.

Configure R4, R5 and R6 in a similar way. Use the appropriate loopbacks and AS# for the peering sessions. The dual plane WAN core layers do not connect or peer with each other.

Verify that the BGP sessions came up ok using the ‘show ip bgp summ’ command.

BGP Section 3:

The purpose of this section is to configure eBGP between the border routers and the WAN core layer in order to exchange IPv4 routes. Remember that the WAN core layer doesn't need to learn any VPNv4 routes. The idea is to provide an alternate (HA) path on the links to WAN.



Starting with DC1 configure R7 and R8 like this (notice that R7 and R8 peer with each other directly too):

R7:

```
router bgp 65001
 neighbor 10.1.3.8 remote-as 65001
 neighbor 10.1.11.1 remote-as 65000
 neighbor 10.1.13.4 remote-as 65000
 !
 address-family ipv4
  network 13.13.13.13 mask 255.255.255.255
  neighbor 10.1.3.8 activate
  neighbor 10.1.11.1 activate
  neighbor 10.1.11.1 route-map OUTBOUND out
  neighbor 10.1.11.1 send-label
  neighbor 10.1.13.4 activate
  neighbor 10.1.13.4 route-map OUTBOUND out
  neighbor 10.1.13.4 send-label
 exit-address-family
!
ip as-path access-list 1 permit ^$
!
route-map OUTBOUND permit 10
 match as-path 1
 set mpls-label
```

On R8 configure the following:

```
router bgp 65001
 neighbor 10.1.3.7 remote-as 65001
 neighbor 10.1.10.4 remote-as 65000
 neighbor 10.1.12.1 remote-as 65000
 !
 address-family ipv4
  network 13.13.13.13 mask 255.255.255.255
  neighbor 10.1.3.7 activate
  neighbor 10.1.10.4 activate
  neighbor 10.1.10.4 route-map OUTBOUND out
  neighbor 10.1.10.4 send-label
  neighbor 10.1.12.1 activate
  neighbor 10.1.12.1 route-map OUTBOUND out
  neighbor 10.1.12.1 send-label
 exit-address-family
!
ip as-path access-list 1 permit ^$
!
route-map OUTBOUND permit 10
 match as-path 1
 set mpls-label
```

Notice the route-map. Essentially filters the routes being sent to the WAN core to only locally originated routes. Also notice the network statement that advertises the /32 loopback of the PE at that site. Using the network statement avoids redistributing OSPF back to BGP.

Now configure R1 and R4 to peer with DC1 to match what you did above. On R1 configure:

```
router bgp 65000
 neighbor 10.1.11.7 remote-as 65001
 neighbor 10.1.12.8 remote-as 65001
 !
 address-family ipv4
  neighbor 10.1.11.7 activate
  neighbor 10.1.11.7 send-label
  neighbor 10.1.12.8 activate
  neighbor 10.1.12.8 send-label
 exit-address-family
```

On R4 configure:

```
router bgp 65000
 neighbor 10.1.10.8 remote-as 65001
 neighbor 10.1.13.7 remote-as 65001
 !
 address-family ipv4
  neighbor 10.1.10.8 activate
  neighbor 10.1.10.8 send-label
  neighbor 10.1.13.7 activate
  neighbor 10.1.13.7 send-label
 exit-address-family
```

Verify that the BGP sessions came up using the “show ip bgp summ” command.

Implement a similar config for R9, R10, R2 and R5 (DC2 connections to the WAN core layer).

For DC3 connectivity, configure R11 like this:

```
router bgp 65003
 neighbor 10.1.26.6 remote-as 65000
 neighbor 12.12.12.12 remote-as 65003
 neighbor 12.12.12.12 update-source Loopback0
 !
 address-family ipv4
  network 21.21.21.21 mask 255.255.255.255
  neighbor 10.1.26.6 activate
  neighbor 10.1.26.6 route-map OUTBOUND out
  neighbor 10.1.26.6 send-label
  neighbor 12.12.12.12 activate
 exit-address-family
!
ip as-path access-list 1 permit ^$
!
route-map OUTBOUND permit 10
 match as-path 1
 set mpls-label
```

and the other end of the GRE tunnel R6 like this:

```
router bgp 65000
 neighbor 10.1.26.11 remote-as 65003
 !
 address-family ipv4
  neighbor 10.1.26.11 activate
  neighbor 10.1.26.11 send-label
 exit-address-family
```

Implement a similar config for R12 and R3 using the appropriate addresses for their respective tunnel. Notice that only R12 needs the route-map

Ensure the BGP sessions over the GRE tunnels came up ok using the “show ip bgp summ” command.

Add a redistribute statement under OSPF on R7, R8, R9, R10, R11 and R12:

```
router ospf 100
 redistribute bgp <AS# for that site> subnets
```

that allows the PE at each DC to use the local border routers to reach the other DCs.

Now check to see if you can ping from R17 to R13:

```
R17#ping 13.13.13.13 source 17.17.17.17
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 13.13.13.13, timeout is 2 seconds:
Packet sent with a source address of 17.17.17.17
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/2 ms
```

If ping is not working between R17 and R13 check the previous steps. Make sure the routes are learned properly and redistribution into OSPF is configured.

To make sure the label transport is working properly execute this traceroute:

```
R13#traceroute 17.17.17.17 source 13.13.13.13
Type escape sequence to abort.
Tracing the route to 17.17.17.17
VRF info: (vrf in name/id, vrf out name/id)
  1 10.1.1.7 [MPLS: Label 27 Exp 0] 3 msec
    10.1.2.8 [MPLS: Label 27 Exp 0] 2 msec
    10.1.1.7 [MPLS: Label 27 Exp 0] 1 msec
  2 10.1.12.1 [MPLS: Label 38 Exp 0] 1 msec
    10.1.11.1 [MPLS: Label 38 Exp 0] 1 msec
    10.1.12.1 [MPLS: Label 38 Exp 0] 1 msec
  3 10.1.14.2 [MPLS: Label 19 Exp 0] 1 msec 1 msec 1 msec
  4 10.1.20.9 [MPLS: Label 26 Exp 0] 1 msec 1 msec 2 msec
  5 10.1.4.17 1 msec *  2 msec
```

If you get a label assigned at every hop that means the end to end label path is working.

To check at each hop if the label is assigned correctly use this command:

```
R13#sh mpls for
R13#sh mpls forwarding-table 17.17.17.17
Local      Outgoing   Prefix          Bytes Label   Outgoing    Next Hop
Label      Label      or Tunnel Id    Switched      interface
25         27         17.17.17.17/32  0             Et0/3       10.1.1.7
           27         17.17.17.17/32  0             Et1/0       10.1.2.8
```

Then go to the next hop and check again:

```
R7#sh mpls forwarding-table 17.17.17.17
Local      Outgoing   Prefix          Bytes Label   Outgoing    Next Hop
Label      Label      or Tunnel Id    Switched      interface
27         38         17.17.17.17/32  360           Et0/0       10.1.11.1
```

And the next hop:

```
R1#sh mpls forwarding-table 17.17.17.17
Local      Outgoing   Prefix          Bytes Label   Outgoing    Next Hop
Label      Label      or Tunnel Id    Switched      interface
38         19         17.17.17.17/32  31983         Et1/0       10.1.14.2
```

And so on until you reach the last hop in DC2. You can also do the reverse for 13.13.13.13. If the label is assigned properly then you get bidirectional end to end connectivity.

Lab5: Configure VRFs and iBGP Peering Between PEs

In this network we have 3 PEs: R13, R17 and R21. Configure the following VRFs on the all PEs:

```
vrf definition blue
 rd 200:2
 route-target export 200:2
 route-target import 200:2
 !
 address-family ipv4
 exit-address-family
 !
 address-family ipv6
 exit-address-family
!
vrf definition red
 rd 200:1
 route-target export 200:1
 route-target import 200:1
 !
 address-family ipv4
 exit-address-family
 !
 address-family ipv6
 exit-address-family
!
```

Assign VRF red to ethernet 0/0 on all PEs using the command “vrf forwarding red”.

For example R13:

```
R13#config t
Enter configuration commands, one per line.  End with CNTL/Z.
R13(config)#int e0/0
R13(config-if)#vrf forwarding red
% Interface Ethernet0/0 IPv4 disabled and address(es) removed due to enabling VRF red
R13(config-if)#ip address 192.168.1.13 255.255.255.0
R13(config-if)#
```

Notice how the IP address on the interface was removed when you applied the vrf forwarding command.

R17 would look like this:

```
R17#config t
Enter configuration commands, one per line.  End with CNTL/Z.
R17(config)#interface Ethernet0/0
R17(config-if)#vrf forwarding red
% Interface Ethernet0/0 IPv4 disabled and address(es) removed due to enabling VRF red
R17(config-if)#
R17(config-if)# ip address 192.168.3.17 255.255.255.0
```

Do the same for VRF blue. Don't forget to re-add the IP address to the interface.

Implement similar config on R21 for VRFs red and blue.

In order to exchange the VRF routes we need an eBGP mesh between the PEs under the VPNv4 address family. There is no need to exchange IPv4 routes between PEs.

On R13 enable BGP, peer with PE2 and PE3 (R17 and R21) over the WAN core layer using the loopbacks. And add redistribute connected under the VRF address family. Here's how the config would look like on R13:

```
router bgp 65001
 neighbor 17.17.17.17 remote-as 65002
 neighbor 17.17.17.17 ebgp-multihop 10
 neighbor 17.17.17.17 update-source Loopback0
 neighbor 21.21.21.21 remote-as 65003
 neighbor 21.21.21.21 ebgp-multihop 10
 neighbor 21.21.21.21 update-source Loopback0
 !
 address-family ipv4
  no neighbor 17.17.17.17 activate
  no neighbor 21.21.21.21 activate
 exit-address-family
 !
address-family vpnv4
  neighbor 17.17.17.17 activate
  neighbor 17.17.17.17 send-community extended
  neighbor 17.17.17.17 route-map OUTBOUND out
  neighbor 21.21.21.21 activate
  neighbor 21.21.21.21 send-community extended
  neighbor 21.21.21.21 route-map OUTBOUND out
 exit-address-family
 !
 address-family ipv4 vrf blue
  redistribute connected
 exit-address-family
 !
 address-family ipv4 vrf red
  redistribute connected
 exit-address-family
!
ip as-path access-list 1 permit ^$
!
route-map OUTBOUND permit 10
 match as-path 1
```

R17 looks like this:

```
router bgp 65002
 neighbor 13.13.13.13 remote-as 65001
 neighbor 13.13.13.13 ebgp-multihop 10
 neighbor 13.13.13.13 update-source Loopback0
 neighbor 21.21.21.21 remote-as 65003
 neighbor 21.21.21.21 ebgp-multihop 10
 neighbor 21.21.21.21 update-source Loopback0
 !
 address-family ipv4
  no neighbor 13.13.13.13 activate
  no neighbor 21.21.21.21 activate
 exit-address-family
 !
address-family vpnv4
  neighbor 13.13.13.13 activate
  neighbor 13.13.13.13 send-community extended
  neighbor 13.13.13.13 route-map OUTBOUND out
  neighbor 21.21.21.21 activate
  neighbor 21.21.21.21 send-community extended
  neighbor 21.21.21.21 route-map OUTBOUND out
 exit-address-family
 !
 address-family ipv4 vrf blue
  redistribute connected
 exit-address-family
 !
 address-family ipv4 vrf red
  redistribute connected
 exit-address-family
!
ip as-path access-list 1 permit ^$
!
route-map OUTBOUND permit 10
 match as-path 1
```

Configure R21 to participate in the eBGP mesh. Don't forget the redistribute connected under the address family for VRFs red and blue. And also the route-map (OUTBOUND).

It is important to make sure the PEs are not transit routers, hence the route map under the VPNv4 address family. Check PE3 (R21) to make sure it is only advertising the connected routes for VRFs red and blue:

```
R21#sh ip bgp vpnv4 all neighbors 17.17.17.17 advertised-routes
BGP table version is 21, local router ID is 21.21.21.21

     Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 200:1 (default for vrf red)
 *>  192.168.6.0      0.0.0.0                  0         32768 ?
Route Distinguisher: 200:2 (default for vrf blue)
 *>  192.168.5.0      0.0.0.0                  0         32768 ?
```

Test connectivity from Host1 (R15):

```
R15#ping 192.168.3.19
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.3.19, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 1/34/133 ms
R15#
R15#
R15#ping 192.168.6.23
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.6.23, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 1/1/1 ms
```

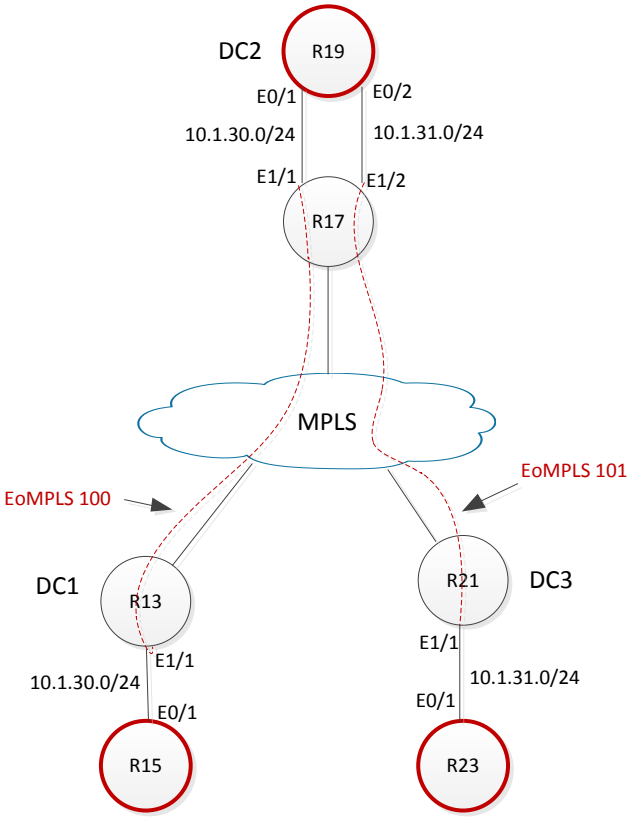
Test for VRF blue from Host2 (R16):

```
R16#ping 192.168.4.20
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.4.20, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 1/4/8 ms
R16#
R16#ping 192.168.5.22
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.5.22, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 1/1/2 ms
R16#
```

Check with traceroute as well. If everything is working ok that means you got the control plane (BGP, OSPF) and the dataplane (the labeled path) working properly.

Lab6: Ethernet over MPLS (EoMPLS)

The goal of this lab is to configure two EoMPLS circuits across the infrastructure we setup so far. Think of port mode EoMPLS as virtual wires.



The IP addresses on R15, R19, and R23 used to connect over EoMPLS are already set up.

On DC1's PE (R13) configure a port mode EoMPLS circuit to connect ethernet 1/1 to R17's ethernet 1/1 interface. On R13 configure this:

```
interface Ethernet1/1
 no shut
 xconnect 17.17.17.17 100 encapsulation mpls
```

and on R17 configure this:

```
interface Ethernet1/1
 no shut
 xconnect 13.13.13.13 100 encapsulation mpls
```

Configure the EoMPLS circuit between R17's ethernet 1/2 and R21's ethernet 1/1 with VC ID of 101 (don't forget the 'no shut').

Verify that R15 can ping R19 over the EoMPLS PW:

```
R15#ping 10.1.30.19
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.30.19, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/2 ms
```

On R17 check to make sure the EoMPLS circuits are up:

R17#sh mpls l2transport vc

Local intf	Local circuit	Dest address	VC ID	Status
Et1/1	Ethernet	13.13.13.13	100	UP
Et1/2	Ethernet	21.21.21.21	101	UP

And check on the stats of vc 100:

```
R17#sh mpls l2transport vc 100 detail
Local interface: Et1/1 up, line protocol up, Ethernet up
  Destination address: 13.13.13.13, VC ID: 100, VC status: up
    Output interface: Et1/0, imposed label stack {27 37}
    Preferred path: not configured
    Default path: active
    Next hop: 10.1.5.10
  Create time: 00:07:27, last status change time: 00:05:56
  Signaling protocol: LDP, peer 13.13.13.13:0 up
    Targeted Hello: 17.17.17.17(LDP Id) -> 13.13.13.13
    Status TLV support (local/remote)   : enabled/supported
      Label/status state machine        : established, LruRru
      Last local dataplane   status rcvd: no fault
      Last local SSS circuit status rcvd: no fault
      Last local SSS circuit status sent: no fault
      Last local  LDP TLV    status sent: no fault
      Last remote LDP TLV    status rcvd: no fault
    MPLS VC labels: local 36, remote 37
    Group ID: local 0, remote 0
    MTU: local 1500, remote 1500
    Remote interface description:
  Sequencing: receive disabled, send disabled
  VC statistics:
    packet totals: receive 56, send 53
    byte totals:   receive 6031, send 6886
    packet drops:  receive 0, seq error 0, send 0
```

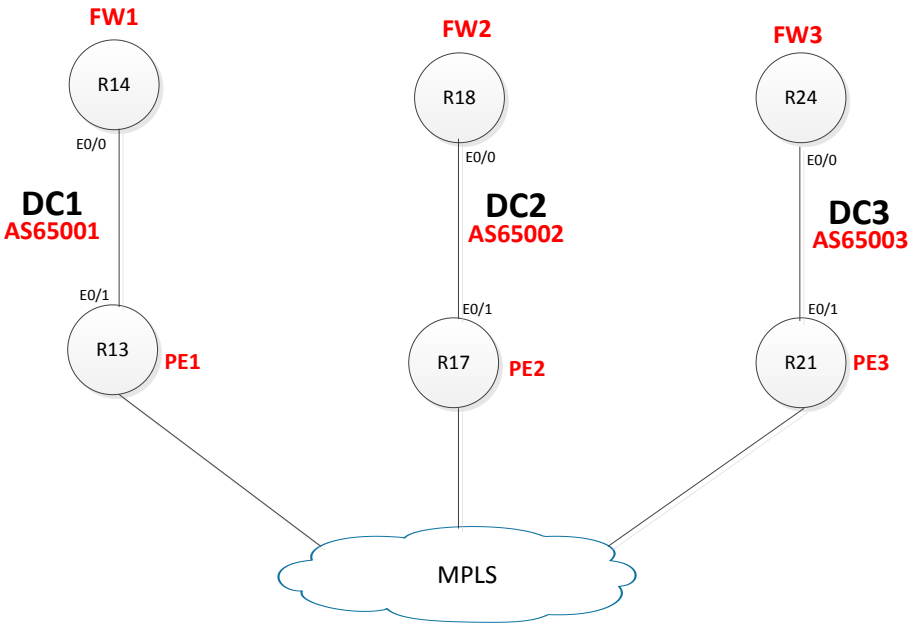
In a real life environment the transport will need to support the extra MTU. For example if you try to ping between R19 and R23 using an MTU of 1500 you will get this:

```
R19#ping 10.1.30.15 size 1500
Type escape sequence to abort.
Sending 5, 1500-byte ICMP Echos to 10.1.30.15, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

Lab7: Implement the Firewalls

There are 3 firewalls in this network. One firewall in each DC. The idea is to use the appropriate firewall for inter-VRF traffic.

Here's a representation of how the firewalls connect to the PEs for reference:



Start with DC1. On PE1 (R13) add an extra VRF called the "gray VRF". The Gray VRF provides the path in between firewalls when inter-VRF traffic is needed across DCs.

Use this config for R13, R17 and R21:

```
vrf definition gray
 rd 200:3
 route-target export 200:3
 route-target import 200:3
 !
 address-family ipv4
 exit-address-family
 !
 address-family ipv6
 exit-address-family
!
```

The link between each PE and the directly attached firewalls provides a transport for each VRF towards the firewall. For that we use subinterfaces. Configure the 3 PEs with the following (on links facing the local firewall):

R13	R17	R21
interface Ethernet0/1.1 encapsulation dot1Q 1 native ! interface Ethernet0/1.2 vrf forwarding red encapsulation dot1Q 2 ip address 10.1.102.13 255.255.255.0 ! interface Ethernet0/1.3 vrf forwarding blue encapsulation dot1Q 3 ip address 10.1.103.13 255.255.255.0 ! interface Ethernet0/1.4 vrf forwarding gray encapsulation dot1Q 4 ip address 10.1.104.13 255.255.255.0	interface Ethernet0/1.1 encapsulation dot1Q 1 native ! interface Ethernet0/1.2 vrf forwarding red encapsulation dot1Q 2 ip address 10.1.112.17 255.255.255.0 ! interface Ethernet0/1.3 vrf forwarding blue encapsulation dot1Q 3 ip address 10.1.113.17 255.255.255.0 ! interface Ethernet0/1.4 vrf forwarding gray encapsulation dot1Q 4 ip address 10.1.114.17 255.255.255.0	interface Ethernet0/1.1 encapsulation dot1Q 1 native ! interface Ethernet0/1.2 encapsulation dot1Q 2 vrf forwarding red ip address 10.1.122.21 255.255.255.0 ! interface Ethernet0/1.3 encapsulation dot1Q 3 vrf forwarding blue ip address 10.1.123.21 255.255.255.0 ! interface Ethernet0/1.4 encapsulation dot1Q 4 vrf forwarding gray ip address 10.1.124.21 255.255.255.0

Configure the 3 firewalls with the following to match:

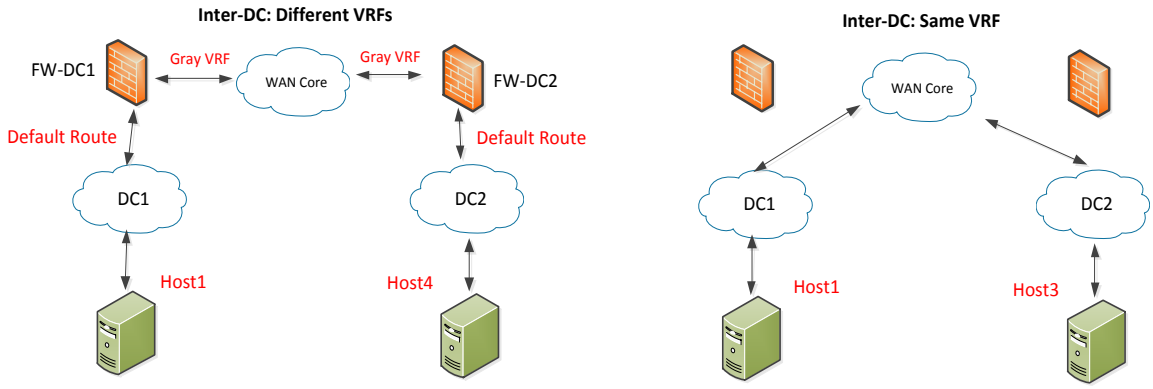
FW (R14)	FW2 (R18)	FW (R24)
interface Ethernet0/0.1 encapsulation dot1Q 1 native ! interface Ethernet0/0.2 description vrf red encapsulation dot1Q 2 ip address 10.1.102.14 255.255.255.0 ! interface Ethernet0/0.3 description vrf blue encapsulation dot1Q 3 ip address 10.1.103.14 255.255.255.0 ! interface Ethernet0/0.4 description vrf gray encapsulation dot1Q 4 ip address 10.1.104.14 255.255.255.0	interface Ethernet0/0.1 encapsulation dot1Q 1 native ! interface Ethernet0/0.2 description vrf red encapsulation dot1Q 2 ip address 10.1.112.18 255.255.255.0 ! interface Ethernet0/0.3 description vrf blue encapsulation dot1Q 3 ip address 10.1.113.18 255.255.255.0 ! interface Ethernet0/0.4 description vrf gray encapsulation dot1Q 4 ip address 10.1.114.18 255.255.255.0	interface Ethernet0/0.1 encapsulation dot1Q 1 native ! interface Ethernet0/0.2 description vrf red encapsulation dot1Q 2 ip address 10.1.122.24 255.255.255.0 ! interface Ethernet0/0.3 description vrf blue encapsulation dot1Q 3 ip address 10.1.123.24 255.255.255.0 ! interface Ethernet0/0.4 description vrf gray encapsulation dot1Q 4 ip address 10.1.124.24 255.255.255.0

We will use the same firewall policy on **all the firewalls**. Configure FW1 (R14), FW2 (R18) and FW3 (R24) with the following:

```
class-map type inspect match-any services
 match access-group 100
!
policy-map type inspect fw_rules
 class type inspect services
  pass
 class class-default
  drop
!
zone security zone1
zone security zone2
zone security zone3
zone security zone4
zone-pair security pair1 source zone1 destination zone3
 service-policy type inspect fw_rules
zone-pair security pair2 source zone2 destination zone3
 service-policy type inspect fw_rules
zone-pair security pair3 source zone1 destination zone2
 service-policy type inspect fw_rules
zone-pair security pair4 source zone3 destination zone1
 service-policy type inspect fw_rules
zone-pair security pair5 source zone3 destination zone2
 service-policy type inspect fw_rules
!
interface Ethernet0/0.2
 zone-member security zone1
!
interface Ethernet0/0.3
 zone-member security zone2
!
interface Ethernet0/0.4
 zone-member security zone3
!
access-list 100 permit icmp any any
access-list 100 permit udp any any range 33434 33534
```

The firewall policy above is very basic. It is designed to allow ICMP and traceroute through to verify connectivity and the path we are taking through the network.

The objective of the design is to force inter-VRF traffic to use the local firewall at each DC. That means if a host in DC1 wants to access a host in DC2, the packets will traverse FW1 on the way out, then FW2 before reaching host2. The return traffic comes back the same way. Traffic symmetry is required for the firewalls to work properly. For intra-VRF traffic the firewall is bypassed.



To get this working the firewalls send down a default route into VRFs red and blue. That makes the firewall attract traffic that needs to leave the VRF. We also need to make sure the PEs do **not** propagate the default route to other DCs (that means each DC has one unique default route for each VRF that is local **at** that DC).

Configure the three PEs (R13, R17, and R21) like this:

PE1 (R13)	PE2 (R17)
router bgp 65001 ! address-family vpnv4 neighbor 17.17.17.17 route-map PE_to_PE out neighbor 21.21.21.21 route-map PE_to_PE out exit-address-family ! ip as-path access-list 1 permit ^$ ip as-path access-list 2 permit ^$ ip as-path access-list 2 permit ^65101 65001$! ! route-map PE_to_PE permit 10 match as-path 2 ! route-map OUTBOUND permit 10 match as-path 1	router bgp 65002 ! address-family vpnv4 neighbor 13.13.13.13 route-map PE_to_PE out neighbor 21.21.21.21 route-map PE_to_PE out exit-address-family ! ip as-path access-list 1 permit ^$ ip as-path access-list 2 permit ^$ ip as-path access-list 2 permit ^65102 65002$! route-map PE_to_PE permit 10 match as-path 2 ! route-map OUTBOUND permit 10 match as-path 1

Implement similar route-map on PE3 (R21) on the peering sessions with R13 and R17.

The PE_to_PE route-map will propagate the locally originated routes AND the routes that passed through the firewall (ie the gray VRF). However the default route generated by the firewall will not get advertised since it won't match the as-path ACL.

Now we simply need to create the appropriate peering sessions between each PE and the local firewall.

Configure PE1 & PE2 (R13 and R17) like this:

PE1 (R13)	PE2 (R17)
router bgp 65001 ! address-family ipv4 vrf blue redistribute connected neighbor 10.1.103.14 remote-as 65101 neighbor 10.1.103.14 activate neighbor 10.1.103.14 route-map OUTBOUND out exit-address-family ! address-family ipv4 vrf gray redistribute connected neighbor 10.1.104.14 remote-as 65101 neighbor 10.1.104.14 activate neighbor 10.1.104.14 allowas-in 1 exit-address-family ! address-family ipv4 vrf red redistribute connected neighbor 10.1.102.14 remote-as 65101 neighbor 10.1.102.14 activate neighbor 10.1.102.14 route-map OUTBOUND out exit-address-family	router bgp 65002 ! address-family ipv4 vrf blue redistribute connected neighbor 10.1.113.18 remote-as 65102 neighbor 10.1.113.18 activate neighbor 10.1.113.18 route-map OUTBOUND out exit-address-family ! address-family ipv4 vrf gray redistribute connected neighbor 10.1.114.18 remote-as 65102 neighbor 10.1.114.18 activate neighbor 10.1.114.18 allowas-in 1 exit-address-family ! address-family ipv4 vrf red redistribute connected neighbor 10.1.112.18 remote-as 65102 neighbor 10.1.112.18 activate neighbor 10.1.112.18 route-map OUTBOUND out exit-address-family

And FW1 and FW2 (R14 and R18) like this:

FW1 (R14)	FW2 (R18)
router bgp 65101 neighbor 10.1.102.13 remote-as 65001 neighbor 10.1.102.13 default-originate neighbor 10.1.102.13 route-map OUTBOUND out neighbor 10.1.103.13 remote-as 65001 neighbor 10.1.103.13 default-originate neighbor 10.1.103.13 route-map OUTBOUND out neighbor 10.1.104.13 remote-as 65001 ! ip as-path access-list 1 permit ^$! route-map OUTBOUND permit 10 match as-path 1	router bgp 65102 neighbor 10.1.112.17 remote-as 65002 neighbor 10.1.112.17 default-originate neighbor 10.1.112.17 route-map OUTBOUND out neighbor 10.1.113.17 remote-as 65002 neighbor 10.1.113.17 default-originate neighbor 10.1.113.17 route-map OUTBOUND out neighbor 10.1.114.17 remote-as 65002 ! ip as-path access-list 1 permit ^$! route-map OUTBOUND permit 10 match as-path 1

Implement similar peering sessions between PE3 (R21) and FW3 (R24) using the appropriate addresses and route-maps.

On R13 check to see that VRFs red and blue learned a default route:

```
R13#sh ip route vrf red

Gateway of last resort is 10.1.102.14 to network 0.0.0.0

B*    0.0.0.0/0 [20/0] via 10.1.102.14, 05:09:44
      10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
C        10.1.102.0/24 is directly connected, Ethernet0/1.2
L        10.1.102.13/32 is directly connected, Ethernet0/1.2
B        10.1.112.0/24 [20/0] via 17.17.17.17, 05:31:45
B        10.1.122.0/24 [20/0] via 21.21.21.21, 04:44:42
      192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.1.0/24 is directly connected, Ethernet0/0
L        192.168.1.13/32 is directly connected, Ethernet0/0
B     192.168.3.0/24 [20/0] via 17.17.17.17, 06:09:44
B     192.168.6.0/24 [20/0] via 21.21.21.21, 04:44:42

R13#sh ip route vrf blue

Gateway of last resort is 10.1.103.14 to network 0.0.0.0

B*    0.0.0.0/0 [20/0] via 10.1.103.14, 05:09:55
      10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
C        10.1.103.0/24 is directly connected, Ethernet0/1.3
L        10.1.103.13/32 is directly connected, Ethernet0/1.3
B        10.1.113.0/24 [20/0] via 17.17.17.17, 06:09:55
B        10.1.123.0/24 [20/0] via 21.21.21.21, 04:44:53
      192.168.2.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.2.0/24 is directly connected, Ethernet0/2
L        192.168.2.13/32 is directly connected, Ethernet0/2
B     192.168.4.0/24 [20/0] via 17.17.17.17, 06:09:55
B     192.168.5.0/24 [20/0] via 21.21.21.21, 04:44:53
```

Check the other PEs and you should see the default route pointing only to the local PE. Now check R13 to see which routes are being advertised:

```
R13#sh ip bgp vpnv4 all neighbors 17.17.17.17 advertised-routes

     Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 200:1 (default for vrf red)
 *>  10.1.102.0/24    0.0.0.0                  0         32768 ?
 *>  192.168.1.0      0.0.0.0                  0         32768 ?
Route Distinguisher: 200:2 (default for vrf blue)
 *>  10.1.103.0/24    0.0.0.0                  0         32768 ?
 *>  192.168.2.0      0.0.0.0                  0         32768 ?
Route Distinguisher: 200:3 (default for vrf gray)
 *>  10.1.102.0/24    10.1.104.14                            0 65101 65001 ?
 *>  10.1.103.0/24    10.1.104.14                            0 65101 65001 ?
 *>  10.1.104.0/24    0.0.0.0                  0         32768 ?
 *>  192.168.1.0      10.1.104.14                            0 65101 65001 ?
 *>  192.168.2.0      10.1.104.14                            0 65101 65001 ?

Total number of prefixes 9
```

On R15 try a traceroute to see the path the packet is taking. For example intra-VRF:

```
R15#traceroute 192.168.3.19
Type escape sequence to abort.
Tracing the route to 192.168.3.19
VRF info: (vrf in name/id, vrf out name/id)
  1 192.168.1.13 1 msec 1 msec 1 msec
  2 10.1.2.8 [MPLS: Labels 27/35 Exp 0] 2 msec 1 msec 1 msec
  3 10.1.12.1 [MPLS: Labels 29/35 Exp 0] 0 msec 1 msec 1 msec
  4 10.1.14.2 [MPLS: Labels 28/35 Exp 0] 0 msec 0 msec 1 msec
  5 10.1.20.9 [MPLS: Labels 22/35 Exp 0] 1 msec 0 msec 1 msec
  6 192.168.3.17 0 msec 1 msec 0 msec
  7 192.168.3.19 1 msec *  2 msec
```

And inter-VRF:

```
R15#traceroute 192.168.4.20
Type escape sequence to abort.
Tracing the route to 192.168.4.20
VRF info: (vrf in name/id, vrf out name/id)
  1 192.168.1.13 2 msec 1 msec 2 msec
  2 10.1.102.14 1 msec 1 msec 0 msec
  3 10.1.104.13 1 msec 1 msec 1 msec
  4 10.1.2.8 [MPLS: Labels 27/47 Exp 0] 2 msec 1 msec 1 msec
  5 10.1.12.1 [MPLS: Labels 29/47 Exp 0] 2 msec 1 msec 1 msec
  6 10.1.14.2 [MPLS: Labels 28/47 Exp 0] 2 msec 1 msec 1 msec
  7 10.1.20.9 [MPLS: Labels 22/47 Exp 0] 1 msec 1 msec 1 msec
  8 10.1.114.17 [MPLS: Label 47 Exp 0] 1 msec 1 msec 1 msec
  9 10.1.114.18 1 msec 6 msec 7 msec
 10 10.1.113.17 1 msec 1 msec 0 msec
 11 192.168.4.20 1 msec *  3 msec
```

Bonus Lab8: VRF IPv6 connectivity (6VPE)

The goal of this lab is to provide IPv6 unicast connectivity between VRF red DC locations, using the previously configured MPLS infrastructure.

This lab has 2 sections:

1) Activate IPv6 routing on the PEs for VRF red
2) Configure VPNv6 between the PEs

Section 1: IPv6 routing VRF red

For simplicity, the following IPv6 subnets will be used on the DC VRF red segment.

DC1 - 2001:192:168:1::/64
DC2 – 2001:192:168:3::/64
DC3 – 2001:192:168:6::/64

Activate IPv6 address family and IPv6 routing for VRF red, on all PEs (R13, R17, R21):

```
!
vrf definition red
!
 address-family ipv6
 exit-address-family
!
!
ipv6 unicast-routing
ipv6 cef
!
```

Configure appropriate IPv6 addresses on all PEs (R13, R17, R21) VRF red local interface:

For example on R13 configure this:

```
R13(config)#
interface Ethernet0/0
 vrf forwarding red
 ipv6 address 2001:192:168:1::13/64
!
```

And for R17 configure this:

```
R17(config)#
interface Ethernet0/0
 vrf forwarding red
```

```
ipv6 address 2001:192:168:3::17/64
!
```

Also, configure the appropriate interface for R21.

For simulating IPv6 hosts into VRF red, configure also IPv6 addresses for Host1 (R15), Host3 (R19) and Host6(R23) as follows:

On Host1(R15), configure the IPv6 address:

```
R15(config)#
interface Ethernet0/0
ipv6 address 2001:192:168:1::15/64
!
```

Continue with Host3(R19)

```
R19(config)#
interface Ethernet0/0
ipv6 address 2001:192:168:3::19/64
!
```

Also, configure Host6(R23) with the appropriate IPv6 address.

On each previously configured Hosts, test the IPv6 connectivity their correspondent PE and that and IPv6 default route is present in their IPv6 routing table.

For example in Host1(R15), the results should look like:

```
R15#ping 2001:192:168:1::13
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:192:168:1::13, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
R15#
```

Observe the IPv6 default routing entry:

```
R15#show ipv6 route
IPv6 Routing Table - default - 4 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, HA - Home Agent, MR - Mobile Router, R - RIP
       H - NHRP, I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
       IS - ISIS summary, D - EIGRP, EX - EIGRP external, NM - NEMO
       ND - ND Default, NDp - ND Prefix, DCE - Destination, NDr - Redirect
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, l - LISP
ND  ::/0 [2/0]
     via FE80::A8BB:CCFF:FE00:D00, Ethernet0/0
C   2001:192:168:1::/64 [0/0]
     via Ethernet0/0, directly connected
L   2001:192:168:1::15/128 [0/0]
     via Ethernet0/0, receive
L   FF00::/8 [0/0]
```

```
    via Null0, receive
!
```

Test the same commands on Host3(R19) and Host6(R23).

Section 2: VPNv6 between the PEs

For providing IPv6 connectivity inside VRF red, VPNv6 address family will be configured into MP-BGP on the PEs. This is known as “6VPE” configuration.

It is worth mentioning that, for the scope of this lab, no IPv6 configuration is required on the other routers, beside the PEs. The core (R1 to R6) or distribution routers (R7 to R12), do not require any change.

The following configuration will be applied to R13:

```
R13(config)#
router bgp 65001
!
 address-family vpnv6
  neighbor 17.17.17.17 activate
  neighbor 17.17.17.17 send-community extended
  neighbor 17.17.17.17 route-map PE_to_PE out
  neighbor 21.21.21.21 activate
  neighbor 21.21.21.21 send-community extended
  neighbor 21.21.21.21 route-map PE_to_PE out
 exit-address-family
 !
address-family ipv6 vrf red
  redistribute connected
 exit-address-family
!
```

And the following to R17:

```
R17(config)#
router bgp 65002
!
 address-family vpnv6
  neighbor 13.13.13.13 activate
  neighbor 13.13.13.13 send-community extended
  neighbor 13.13.13.13 route-map PE_to_PE out
  neighbor 21.21.21.21 activate
  neighbor 21.21.21.21 send-community extended
  neighbor 21.21.21.21 route-map PE_to_PE out
 exit-address-family
 !
!
 address-family ipv6 vrf red
  redistribute connected
 exit-address-family
!
```

Please apply a correspondent configuration to R21.

For checking if the PEs advertise VPNv6 prefixes to each other, please check the MP-BGP vpnv6 table on all PEs. Also verify the VPNv6 BGP prefixes entries, and the IPv6 routing and CEF tables.

For example on R13, the other PEs are present into the vpvn6 address family and it receives one prefix form each neighbour:

```
R13#sh bgp vpnv6 unicast all summary
BGP router identifier 13.13.13.13, local AS number 65001
BGP table version is 9, main routing table version 9
4 network entries using 768 bytes of memory
4 path entries using 368 bytes of memory
3/3 BGP path/bestpath attribute entries using 432 bytes of memory
6 BGP AS-PATH entries using 144 bytes of memory
3 BGP extended community entries using 72 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 1784 total bytes of memory
BGP activity 38/0 prefixes, 38/0 paths, scan interval 60 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
17.17.17.17     4        65002      27      27        9    0    0 00:11:40        1
21.21.21.21     4        65003      27      33        9    0    0 00:11:14        1
R13#
```

R13, vpnv6 BGP table, with local and remote entries:

```
R13#sh bgp vpnv6 unicast all
BGP table version is 9, local router ID is 13.13.13.13
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 200:1 (default for vrf red)
 *>  2001:192:168:1::/64
                    ::                       0         32768 ?
 *>  2001:192:168:3::/64
                    ::FFFF:17.17.17.17
                                             0             0 65002 ?
 *>  2001:192:168:6::/64
                    ::FFFF:21.21.21.21
                                             0             0 65003 ?
```

On R13, check how a remote VPNv6 prefix should look like, and notice the VPN label:

```
R13#sh bgp vpnv6 unicast all 2001:192:168:6::/64
BGP routing table entry for [200:1]2001:192:168:6::/64, version 9
Paths: (1 available, best #1, table red)
  Not advertised to any peer
  Refresh Epoch 1
```

```
65003
  ::FFFF:21.21.21.21 (metric 1) from 21.21.21.21 (21.21.21.21)
    Origin incomplete, metric 0, localpref 100, valid, external, best
    Extended Community: RT:200:1
    Connector Attribute: count=1
     type 1 len 12 value 200:1:21.21.21.21
    mpls labels in/out nolabel/22
```

Please also observe the remote prefixes next hop, which is an IPv4 mapped address.

Check the IPv6 routing table. For example in R13:

```
R13#show ipv6 route vrf red bgp
IPv6 Routing Table - red - 7 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, HA - Home Agent, MR - Mobile Router, R - RIP
       H - NHRP, I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
       IS - ISIS summary, D - EIGRP, EX - EIGRP external, NM - NEMO
       ND - ND Default, NDp - ND Prefix, DCE - Destination, NDr - Redirect
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, l - LISP
B   2001:192:168:3::/64 [20/0]
     via 17.17.17.17%default, indirectly connected
B   2001:192:168:6::/64 [20/0]
     via 21.21.21.21%default, indirectly connected
R13#
```

Verify the CEF table and check the MPLS stack (2 labels) used for transiting IPv6 packets. For example on R3:

```
R13#show ipv6 cef vrf red 2001:192:168:6::/64 det
2001:192:168:6::/64, epoch 0, flags rib defined all labels
  recursive via 21.21.21.21 label 22
    nexthop 10.1.1.7 Ethernet0/3 label 28
    nexthop 10.1.2.8 Ethernet1/0 label 28
R13#
```

Please check the same commands, for all IPv6 prefixes on all PEs, R13, R17 and R21.

Now check the IPv6 full mesh connectivity between the Host1(R15), Host3(R19) and Host6(R23).

For example on R15, the ipv6 ping works for all other hosts and the traceroute is showing the transit path, including the MPLS stack:

```
R15#ping 2001:192:168:3::19
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:192:168:3::19, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/2 ms

R15#ping 2001:192:168:6::23
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 2001:192:168:6::23, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/4/19 ms

R15#traceroute 2001:192:168:3::19
Type escape sequence to abort.
Tracing the route to 2001:192:168:3::19

  1 2001:192:168:1::13 0 msec 0 msec 0 msec
  2 ::FFFF:10.1.1.7 [MPLS: Labels 27/27 Exp 0] 7 msec 1 msec 0 msec
  3 ::FFFF:10.1.11.1 [MPLS: Labels 27/27 Exp 0] 1 msec 1 msec 0 msec
  4 ::FFFF:10.1.14.2 [MPLS: Labels 27/27 Exp 0] 1 msec 0 msec 1 msec
  5 ::FFFF:10.1.20.9 [MPLS: Labels 22/27 Exp 0] 1 msec 0 msec 1 msec
  6 2001:192:168:3::17 0 msec 0 msec 0 msec
  7 2001:192:168:3::19 1 msec 1 msec 0 msec

R15#traceroute 2001:192:168:6::23
Type escape sequence to abort.
Tracing the route to 2001:192:168:6::23

  1 2001:192:168:1::13 2 msec 4 msec 5 msec
  2 ::FFFF:10.1.2.8 [MPLS: Labels 28/22 Exp 0] 1 msec 1 msec 1 msec
  3 ::FFFF:10.1.12.1 [MPLS: Labels 35/22 Exp 0] 1 msec 1 msec 1 msec
  4 ::FFFF:10.1.16.3 [MPLS: Labels 33/22 Exp 0] 1 msec 1 msec 1 msec
  5 ::FFFF:10.1.25.12 [MPLS: Labels 19/22 Exp 0] 1 msec 0 msec 1 msec
  6 2001:192:168:6::21 0 msec 1 msec 0 msec
  7 2001:192:168:6::23 1 msec 0 msec 1 msec
R15#
```

Check the same commands on Host3(R19) and Host6(R23) and test IPv6 connectivity to the other hosts.

For exercise, the same order of commands can be applied for activating IPv6 connectivity inside VRF blue.

Bonus Lab9: VRF multicast IPv4 (MVPN)

The goal of this lab is to provide IPv4 multicast connectivity between VRF red DC locations, using the previously configured infrastructure. This is known as MVPN(multicast VPN). For the scope of this lab, the MVPN used model will be Draft Rosen.

The lab has 3 sections:

1) Configure the global IPv4 multicast routing into the distribution/core network
2) Activate IPv4 multicast routing on the PEs, global and VRF red
3) Configure MVPN between the PEs

Section 1: IPv4 multicast routing Distribution and Core network

The Draft Rosen MPN model requires that global IPv4 multicast routing to be activated inside the transit network between the PEs, in our lab case the core and distribution network. For these, only SSM (source specific multicast) will be used, which is best practice.

Activate multicast routing on all transit routers, R1 to R12, with RPF proxy vector and PIM-SSM default range (232.0.0.0/8). Apply the following command on all these routers:

```
ip multicast-routing
ip multicast rpf proxy vector
!
ip pim ssm default
!
```

Activate PIM protocol on all core and distribution transit interface.

On R1,R2,R4,R5 respective R7 to R10, apply the following commands:

```
!
interface Ethernet0/0
 ip pim sparse-mode
!
interface Ethernet0/2
 ip pim sparse-mode
!
interface Ethernet0/3
 ip pim sparse-mode
!
interface Ethernet1/0
 ip pim sparse-mode
!
```

On R3,R6 respective R11 and R12 apply the commands:

```
interface Ethernet0/3
```

```
 ip pim sparse-mode
!
interface Ethernet1/0
 ip pim sparse-mode
!
interface Tunnel0
 ip pim sparse-mode
!
```

Please check on all transit routers that PIM neighbourship was formed on all transit interfaces. For example on R1:

```
R1#sh ip pim ne
PIM Neighbor Table
Mode: B - Bidir Capable, DR - Designated Router, N - Default DR Priority,
      P - Proxy Capable, S - State Refresh Capable, G - GenID Capable
Neighbor          Interface          Uptime/Expires    Ver   DR
Address                                                Prio/Mode
10.1.11.7         Ethernet0/0        01:31:12/00:01:35 v2    1 / DR S P G
10.1.12.8         Ethernet0/2        01:31:13/00:01:36 v2    1 / DR S P G
10.1.16.3         Ethernet0/3        01:31:12/00:01:32 v2    1 / DR S P G
10.1.14.2         Ethernet1/0        01:31:13/00:01:34 v2    1 / DR S P G
R1#
```

Section 2: IPv4 multicast routing on the PEs, global and VRF red

On the Draft Rosen MPN model, global multicast routing must be activate on the PEs, including the source loopback; also multicast routing will be activated for the VRF red. The “customer” VRF red, will use ASM (any source multicast), which will require configuring a multicast RP (rendezvous point) inside the VRF.

For configuring global multicast routing, the following commands will be used on the PEs (R13, R17 and R21):

```
ip multicast-routing
!
ip pim ssm default
!
```

For enabling the PIM protocol to the global interfaces and loopack, the following commands will be used on the PEs (R13, R17 and R21):

```
interface Ethernet0/3
 ip pim sparse-mode
!
interface Ethernet1/0
 ip pim sparse-mode
!
Interface Loopback0
ip pim sparse-mode
!
```

The same commands will be applied for configuring multicast routing inside VRF red, on all PEs (R13, R17 and R21), including RPF proxy vector with RD:

```
!
ip multicast-routing vrf red
ip multicast vrf red rpf proxy rd vector
!
```

For enabling multicast routing for the hosts, configure PIM on the local interfaces inside VRF red, on all PEs (R13, R17 and R21):

```
!
interface Ethernet0/0
 vrf forwarding red
 ip pim sparse-mode
!
```

For the purpose of this lab, R13 will be the RP inside VRF red. The RP ip address will be 192.168.13.13 for simplicity, and will be configured on loopback 13 on R13.

```
R13(config)#
!
interface Loopback13
 vrf forwarding red
 ip address 192.168.13.13 255.255.255.0
 ip pim sparse-mode
!
```

This RP address will be configured statically as RP address on all PEs(R13, R17 and R21):

```
!
ip pim vrf red rp-address 192.168.13.13
!
```

Please be aware that, for this lab, the DC1 will have the RP locally, and the DC2 and DC3 remotely into DC1.

Please verify on all PEs (R13, R17 and R21) that they have PIM neighbourship with the all directly connected distribution routers. For example on R13:

```
R13#sh ip pim ne
PIM Neighbor Table
Mode: B - Bidir Capable, DR - Designated Router, N - Default DR Priority,
      P - Proxy Capable, S - State Refresh Capable, G - GenID Capable
Neighbor          Interface                Uptime/Expires    Ver   DR
Address                                                      Prio/Mode
10.1.1.7          Ethernet0/3              01:56:21/00:01:31 v2    1 / S P G
10.1.2.8          Ethernet1/0              01:56:22/00:01:32 v2    1 / S P G
R13#
```

Also check that all PEs, have the right RP configured and this is reachable; for example on R17:

```
R17#sh ip pim vrf red rp mapping
PIM Group-to-RP Mappings

Group(s): 224.0.0.0/4, Static
    RP: 192.168.13.13 (?)

R17#ping vrf red 192.168.13.13
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.13.13, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 5/5/5 ms
R17#
```

Section 3: Configure MPVN between the PEs

The draft Rosen MPN model requires that MP-BGP mdt address family to be activated.

For activating the mdt address family, use the following commands on the PEs (R13, R17 and R21). For example on R13:

```
R13(config)#
router bgp 65001
!
 address-family ipv4 mdt
  neighbor 17.17.17.17 activate
  neighbor 17.17.17.17 send-community extended
  neighbor 17.17.17.17 route-map PE_to_PE out
  neighbor 21.21.21.21 activate
  neighbor 21.21.21.21 send-community extended
  neighbor 21.21.21.21 route-map PE_to_PE out
 exit-address-family
 !
```

Activate the MVPN configuration for VRF red using the following commands on all PEs (R13, R17 and R21).

```
!
vrf definition red
!
 address-family ipv4
  bgp next-hop Loopback0
  mdt default 232.0.0.1
  mdt data 231.0.1.0 0.0.0.255 threshold 1
  mdt data threshold 1
 exit-address-family
 !
```

For checking that the MVPN model functions correctly, check the full mesh PIM neighbourship which must be formed between the PEs over the MVPN overlay tunnel. For example on R13:

```
R13#sh ip pim vrf red 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
Neighbor        Interface             Uptime/Expires    Ver   DR
Address                                                 Prio/Mode
21.21.21.21     Tunnel2               02:10:04/00:01:27 v2    1 / DR S P G
17.17.17.17     Tunnel2               02:10:34/00:01:37 v2    1 / S P G
R13#
```

For checking the MP-BGP mdt address family neighbourship and prefixes used the following commands, for example on R13:

```
R13#sh bgp ipv4 mdt all sum
BGP router identifier 13.13.13.13, local AS number 65001
BGP table version is 5, main routing table version 5
3 network entries using 492 bytes of memory
3 path entries using 204 bytes of memory
3/3 BGP path/bestpath attribute entries using 408 bytes of memory
6 BGP AS-PATH entries using 144 bytes of memory
3 BGP extended community entries using 72 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 1320 total bytes of memory
BGP activity 38/0 prefixes, 38/0 paths, scan interval 60 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
17.17.17.17     4        65002     161     163        5    0    0 02:14:17
1
21.21.21.21     4        65003     162     168        5    0    0 02:13:51
1
R13#
```

Here we can see R13 has 2 neighbours, from which receives one prefix from each.

For checking the BGP mdt table:

```
R13#sh bgp ipv4 mdt all
BGP table version is 5, local router ID is 13.13.13.13
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 200:1 (default for vrf red)
 *>  13.13.13.13/32  0.0.0.0                       0 ?
 *>  17.17.17.17/32  17.17.17.17              0          0 65002 ?
 *>  21.21.21.21/32  21.21.21.21              0          0 65003 ?
```

Here we can see the local prefix, and the remote prefixes.

With this command, we can check the status of one remote prefix, with the RD and the used advertised SSM group:

```
R13#sh bgp ipv4 mdt all 17.17.17.17/32
BGP routing table entry for 200:1:17.17.17.17/32     version 4
Paths: (1 available, best #1, table IPv4-MDT-BGP-Table)
  Not advertised to any peer
```

```
  Refresh Epoch 1
  65002
    17.17.17.17 from 17.17.17.17 (17.17.17.17)
      Origin incomplete, metric 0, localpref 100, valid, external, best,
      MDT group address: 232.0.0.1
```

We can check the status of the MVPN overlay tunnel and PIM information received from BGP:

```
R13#sh ip pim mdt
  * implies mdt is the default MDT
  MDT Group/Num   Interface   Source              VRF
* 232.0.0.1       Tunnel2     Loopback0           red
R13#
R13#sh ip pim mdt bgp
MDT (Route Distinguisher + IPv4)           Router ID        Next Hop
  MDT group 232.0.0.1
    200:1:17.17.17.17                      17.17.17.17      17.17.17.17
    200:1:21.21.21.21                      21.21.21.21      21.21.21.21
R13#
```

On one PE, we can check that they receive multicast traffic from all other PEs on the default group 232.0.0.1, and itself sends multicast traffic into this group. For example on R13:

```
R13#sh 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, 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,
       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

(21.21.21.21, 232.0.0.1), 02:26:33/stopped, flags: sTIZ
  Incoming interface: Ethernet1/0, RPF nbr 10.1.2.8
  Outgoing interface list:
    MVRF red, Forward/Sparse, 02:26:33/00:00:26

(17.17.17.17, 232.0.0.1), 02:26:59/stopped, flags: sTIZ
  Incoming interface: Ethernet1/0, RPF nbr 10.1.2.8
  Outgoing interface list:
    MVRF red, Forward/Sparse, 02:26:59/00:00:00

(13.13.13.13, 232.0.0.1), 02:26:59/00:03:00, flags: sT
  Incoming interface: Loopback0, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet1/0, Forward/Sparse, 02:26:59/00:03:00

(*, 224.0.1.40), 02:28:25/00:02:36, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
```

```
    Loopback0, Forward/Sparse, 02:28:24/00:02:36

R13#
```

On some of the distribution and/or core routers, the multicast routing should appear as forwarding for the default group 232.0.0.1. for example on R1 and R4:

```
R1#sh 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, 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,
       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

(21.21.21.21, 232.0.0.1), 02:31:24/00:02:44, flags: sTv
  Incoming interface: Ethernet0/3, RPF nbr 10.1.16.3, vector 3.3.3.3
  Outgoing interface list:
    Ethernet0/2, Forward/Sparse, 02:31:24/00:02:44

(17.17.17.17, 232.0.0.1), 02:31:50/00:03:10, flags: sTv
  Incoming interface: Ethernet1/0, RPF nbr 10.1.14.2, vector 2.2.2.2
  Outgoing interface list:
    Ethernet0/2, Forward/Sparse, 02:31:50/00:03:10

(*, 224.0.1.40), 02:33:17/00:02:47, RP 0.0.0.0, flags: DPL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list: Null

R1#

R4#sh 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, 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,
       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

(13.13.13.13, 232.0.0.1), 02:31:45/00:03:21, flags: sTv
  Incoming interface: Ethernet0/2, RPF nbr 10.1.10.8, vector 10.1.10.8
  Outgoing interface list:
```

```
    Ethernet0/3, Forward/Sparse, 02:31:19/00:02:44
    Ethernet1/0, Forward/Sparse, 02:31:45/00:03:21

(*, 224.0.1.40), 02:33:12/00:02:53, RP 0.0.0.0, flags: DPL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list: Null

R4#
```

For testing the full mesh multicast connectivity between the Host1(R15), Host3(R19) and Host6(R23), the following multicast sources and receiving groups will be tested:

Source and multicast group	Host1(R15)	Host3(R19)	Host6(R23)
Host1: 239.0.0.15		receiver	receiver
Host3: 239.0.0.19	receiver		receiver
Host6: 239.0.0.23	receiver	receiver	

For configuring multicast receiver, the following commands will be used on the Hosts; for example on Host3(R19):

```
R19(config)#
!
interface Ethernet0/0
 ip igmp join-group 239.0.0.23
 ip igmp join-group 239.0.0.15
!
```

For testing the multicast connectivity, ping to a multicast destination will be used. The multicast works correctly if on one Host, we receive reply from all the other Hosts.

In case of Host1(R15), we receive replies from Host2(R19) and Host3(R23):

```
R15#ping 239.0.0.15 repeat 100
Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 239.0.0.15, timeout is 2 seconds:

Reply to request 0 from 192.168.3.19, 3 ms
Reply to request 0 from 192.168.6.23, 3 ms
Reply to request 1 from 192.168.3.19, 3 ms
Reply to request 1 from 192.168.6.23, 3 ms
Reply to request 2 from 192.168.6.23, 2 ms
Reply to request 2 from 192.168.3.19, 3 ms
Reply to request 3 from 192.168.3.19, 3 ms
Reply to request 3 from 192.168.6.23, 3 ms
….
```

In case of Host2(R19), we receive replies from Host1(R15) and Host3(R23):

```
R19#ping 239.0.0.19 repeat 100
Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 239.0.0.19, timeout is 2 seconds:

Reply to request 0 from 192.168.6.23, 31 ms
Reply to request 1 from 192.168.6.23, 4 ms
Reply to request 1 from 192.168.1.15, 22 ms
Reply to request 1 from 192.168.1.15, 4 ms
Reply to request 2 from 192.168.6.23, 2 ms
Reply to request 2 from 192.168.1.15, 2 ms
Reply to request 3 from 192.168.6.23, 2 ms
Reply to request 3 from 192.168.1.15, 2 ms
Reply to request 4 from 192.168.1.15, 2 ms
Reply to request 4 from 192.168.6.23, 3 ms
….
```

In case of Host3(R23), we receive replies from Host1(R15) and Host2(R19):

```
R23#ping 239.0.0.23 repeat 100
Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 239.0.0.23, timeout is 2 seconds:

Reply to request 0 from 192.168.1.15, 22 ms
Reply to request 0 from 192.168.3.19, 48 ms
Reply to request 1 from 192.168.1.15, 4 ms
Reply to request 1 from 192.168.1.15, 5 ms
Reply to request 1 from 192.168.3.19, 5 ms
Reply to request 2 from 192.168.3.19, 4 ms
Reply to request 2 from 192.168.1.15, 4 ms
Reply to request 3 from 192.168.3.19, 3 ms
Reply to request 3 from 192.168.1.15, 3 ms
Reply to request 4 from 192.168.1.15, 2 ms
Reply to request 4 from 192.168.3.19, 2 ms
….
```

So we can conclude that multicast is working between the Hosts inside VRF red.

Also we can check the multicast routing table on the PEs, for example on R13:

```
R13#sh ip mroute  vrf red
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,
       V - RD & Vector, v - Vector
Outgoing interface flags: H - Hardware switched, A - Assert winner
```

```
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.0.0.19), 02:56:47/stopped, RP 192.168.13.13, flags: SJC
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet0/0, Forward/Sparse, 02:56:47/00:02:18
    Tunnel2, Forward/Sparse, 02:54:46/00:02:59

(192.168.3.19, 239.0.0.19), 00:00:23/00:02:40, flags: T
  Incoming interface: Tunnel2, RPF nbr 17.17.17.17
  Outgoing interface list:
    Ethernet0/0, Forward/Sparse, 00:00:23/00:02:36

(*, 239.0.0.23), 02:56:47/stopped, RP 192.168.13.13, flags: SJC
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet0/0, Forward/Sparse, 02:56:47/00:02:17
    Tunnel2, Forward/Sparse, 02:55:16/00:02:33

(192.168.6.23, 239.0.0.23), 00:00:19/00:02:42, flags: T
  Incoming interface: Tunnel2, RPF nbr 21.21.21.21
  Outgoing interface list:
    Ethernet0/0, Forward/Sparse, 00:00:19/00:02:40

(*, 239.0.0.15), 02:55:16/00:02:51, RP 192.168.13.13, flags: S
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Tunnel2, Forward/Sparse, 02:55:16/00:02:51

(192.168.1.15, 239.0.0.15), 00:00:53/00:02:06, flags: T
  Incoming interface: Ethernet0/0, RPF nbr 0.0.0.0
  Outgoing interface list:
    Tunnel2, Forward/Sparse, 00:00:53/00:02:51

(*, 224.0.1.40), 02:56:49/00:02:56, RP 192.168.13.13, flags: SJCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet0/0, Forward/Sparse, 02:56:48/00:02:16
    Tunnel2, Forward/Sparse, 02:55:15/00:02:56

R13#
```

For exercise, the same order of commands can be applied for activating IPv4 multicast connectivity inside VRF blue.