Copyright Information

Disclaimer

Table of Contents

MPLS VPN

Note

Load the *MPLS VPN* initial configurations prior to starting.

14.1 VRF Lite

- Create new VLAN 76 connecting R6 and SW1 using the subnet 155.X.76.0/24.
- Configure R6 and SW1 so that VLAN76 and VLAN67 subnets belong to different VRF tables named VPN_A and VPN_B
- Create two new Loopback interfaces in SW1 in the VRF tables VPN_A and VPN_B using the IP addresses 172.16.7.7/24 and 192.168.7.7/24.
- Both VRFs in SW1 should use R6 as the default gateway.
- Configure R6 so that you can ping between the abovementioned Loopback subnets.

14.2 MPLS LDP

- Configure R4, R5 and R6 for MPLS label exchange using IETF standard protocol.
- Authenticate LDP peering sessions using MD5 hash and the password value of CISCO.
- Use only a single command in R4 to enable LDP on all OSPF-enabled interfaces.
- Use the physical interfaces IP addresses for the LDP session between R4 and R6.

14.3 MPLS Label Filtering

- Configure R4, R5 and R6 so that the only labels advertised by LDP are the ones for Loopback0 interfaces of the mentioned routers.

14.4 MP-BGP VPNv4

- Create two new VRFs named VPN_A and VPN_B in R5 and assign VLAN58 and VLAN5 interfaces into these VRFs.
- Enable VPN route exchange between R5 and R6 using R6 as the BGP route-reflector.
- Make sure IPv4 peering sessions are not activated by default.
- By the end of this task you should be able to ping the connected interfaces inside the VPNs.

14.5 MP-BGP Prefix Filtering

- Create a new Loopback101 interface in R5's VRF VPN_A with the IP address of 172.16.5.5/24.
- Create a new Loopback102 interface in R6's VRF VPN_B with the IP address of 192.168.6.6/24.
- Configure the network to provide bi-directional connectivity between the two new subnets.
- Make sure R6's VPN_A does not see the prefix 172.16.5.0/24 and R5's VPN_B does not see the prefix 19216.6.0/24.

14.6 PE-CE Routing with RIP

- Use RIP as the PE-CE routing protocol for VPN_B sites.
- Remove all static routing used to provide VPN_B connectivity.
- Configure R4 so that VLAN43 connection belongs to VPN_B as well.
- Preserve RIP metric values learned from CE routers.

14.7 PE-CE Routing with OSPF

- Use OSPF as the PE-CE routing protocol for VPN_A sites configuring PE-CE routers in the same area 1.
- Use the same OSPF process IDs at R6 and R5 but ensure SW1 and SW2 can reach each other.
- Create a new Loopback interface in SW2 with the IP address 172.16.8.8/24 and make sure R6 sees only a /16 summary for this prefix.

14.8 OSPF Sham-Link

- Reconfigure SW1 to a pure CE router in VRF VPN_A, removing VRF-lite configuration.
- Preserve OSPF routing between VPN_A sites at R5 and R6.
- Provision a back-door link between SW1 and SW2 using any the interconnections as L3 interface.
- Configure VPN_A OSPF processes in R5 and R6 for the same domain-id.
- Ensure SW1 and SW2 prefer the path across MPLS core to the backdoor link.

14.9 PE-CE Routing with EIGRP

- Replace OSPF with EIGRP for PE-CE routing protocol used for VPN_A.
- Ensure the backdoor link is used for backup and the primary path between the two sites is across the MPLS VPN cloud.
- Make R4’s VLAN43 interface a part of VPN_A and advertise it into EIGRP.
- All EIGRP routers should be in the same AS.

14.10 EIGRP Site-of-Origin

- Configure R5 and R6 to prevent temporary routing loops that may occur due to mutual redistribution between EIGRP and MP-BGP.
- Ensure the path across the MPLS VPN core is used as the primary between SW1 and SW2.

14.11 PE-CE Routing with BGP

- Erase all EIGRP configurations for VPN_A in R5, R6, SW1 and SW2.
- Configure BGP AS 78 in SW1 and SW2 and peer the CE routers with R5 and R6.
- Advertise Loopback0 interface of SW1 and SW2 into BGP and ensure end-to-end reachability.

14.12 BGP SoO Attribute

- Configure “backdoor” BGP peering session between SW1 and SW2 using the inter-switch link configured previously.
- Ensure this configuration does not result in routing loops due to BGP loop-prevention mechanism being disabled with AS-Override feature.

14.13 Internet Access

- Enable RIP on R6’s connection to BB1 and configure R6 so that VPN_A customers may access these routes.
- You are allowed to use one static route to accomplish this task.
- Only the 150.X.0.0/16 subnets should be allowed to access the Internet.

14.14 AToM

- Provide P2P L2VPN connection between VLAN5 interface of R5 and the unused interface of R6.
- Use the method that provided minimum transport overhead.

14.15 L2TPV3

- Modify the solution for the previous task to use L2TPv3 instead of MPLS as the underlying transport.
- Ensure the packets are never fragmented in the core and automatic MTU detection is in progress.

14.16 MPLS VPN Performance Tuning

- Configure the PE and P routers to minimize the amount of time needed to propagate topology changes between CE-sites.

MPLS VPN Solutions

14.1 VRF Lite

- Create new VLAN 76 connecting R6 and SW1 using the subnet 155.X.76.0/24.
- Configure R6 and SW1 so that VLAN76 and VLAN67 subnets belong to different VRF tables named VPN_A and VPN_B
- Create two new Loopback interfaces in SW1 in the VRF tables VPN_A and VPN_B using the IP addresses 172.16.7.7/24 and 192.168.7.7/24.
- Both VRFs in SW1 should use R6 as the default gateway.
- Configure R6 so that you can ping between the abovementioned Loopback subnets.

Configuration

Note

VRFs or Virtual Routing and Forwarding tables are the fundamental building block for virtualizing a router: turning it into multiple virtual routers. Technically, VRF is a separate RIB (routing information base) and FIB (forwarding information base). These structures are also known as routing table and CEF table. Any interface in a router could be assigned to a VRF using the command **`ip vrf forwarding <VRF_NAME>.`** Notice that this command will erase all existing IP addresses configured on the interface in order to avoid potential address duplication in the new routing table. After this configuration, all packets received on the interface are routed and forwarded using the associated VRF table.

You may extend VRFs beyond a single router by properly mapping the VRFs to the links connecting two routers. This results in “parallel” VPNs being run across the multiple routers, also known as “VRF Lite” – the most basic way for building VPNs. This is the simplest way of creating non-overlapping VPNs in a network; however this solution has poor scalability, as you need to allocate a dedicated inter-router link for every VPN. Therefore, if you have two routers, and 100 VPNs, you will need to provision 100 connections between the two routers, one for every VPN. The connection could be either a separate interface, or some Layer 2 virtualization technique, such as Frame-Relay PVC of Ethernet VLAN.

By default, all interfaces (physical or sub-interfaces) are assigned to the VRF known as the **`global`**, which is the regular routing table used in non-VRF capable routers. In order to create a new VRF, issue the command **`ip vrf <VRF_NAME>`** which opens up the VRF configuration context mode. After this initial step, you need to define a route distinguisher (RD) for this particular VRF using the command **`rd X:Y`** where X and Y are 32-bit numbers. Route Distinguisher is a special 64-bit prefix prepended to every route in the respective VRF routing table. This is done for the purpose of distinguishing the prefixes internally inside the router and avoiding collisions in case two VRFs contain the same prefixes. The common format for RD is using the combination **`ASN:NN`** where ASN is the autonomous system number, and NN is the VRF number inside the router or more globally, the VPN number within the ASN. Alternatively, you may use the format **`IP-Address:NN`** where IP is the router's IP address and NN is the VRF name. The second format properly reflects the feature or RD being a local distinguisher, but using the format **`ASN:NN`** is more popular and common, as it allows easily associating a VRF with the particular VPN in the network.

It is possible to associate static routes or dynamic routing protocol processes with the VRFs. In this scenario, we deal with static routing only. The syntax for a VRF-bound static route is

```
ip route vrf <NAME> PREFIX MASK [interface] [next-hop]
```

Where **`[next-hop]`** is an IP address resolvable through the VRF. However, it is possible to use the static routes for "inter-VRF" communications. If you are using a static route with the **`[interface]`** specification, the interface could belong to any VRF. Note that with multi-access interfaces you will also need to specify the next-hop associated with the interface subnet. Cisco IOS will install a CEF entry in the "source" VRF using the information provided and will not attempt to resolve the next-hop recursively. Keep in mind that this trick only works with the non-recursive static routes that use directly connected interfaces.

R6:

```
!
! VRF for VPN_A
!
ip vrf VPN_A
 rd 100:1

!
! VRF for VPN_B
!
ip vrf VPN_B
 rd 100:2

!
! PE-CE Links
!
interface FastEthernet 0/0.67
 ip vrf forwarding VPN_A
 ip address 155.1.67.6 255.255.255.0
!
interface FastEthernet 0/0.76
 encapsulation dot1q 76
 ip vrf forwarding VPN_B
 ip address 155.1.76.6 255.255.255.0

!
! Inter-VRF static routes
!
ip route vrf VPN_A 192.168.7.0 255.255.255.0 FastEthernet0/0.76
155.1.76.7
!
ip route vrf VPN_B 172.16.7.0 255.255.255.0 FastEthernet0/0.67
155.1.67.7
```

SW1:

```
!
! Don’t forget the following command
!
ip routing
!
! VRF for VPN_A
!
ip vrf VPN_A
 rd 100:1

!
! VRF for VPN_B
!
ip vrf VPN_B
 rd 100:2

!
! VPN Interfaces
!
interface Vlan 67
 ip vrf forwarding VPN_A
 ip address 155.1.67.7 255.255.255.0
```

```
!
interface Vlan 76
 ip vrf forwarding VPN_B
 ip address 155.1.76.7 255.255.255.0
!
interface Loopback 101
 ip vrf forwarding VPN_A
 ip address 172.16.7.7 255.255.255.0
!
interface Loopback 102
 ip vrf forwarding VPN_B
 ip address 192.168.7.7 255.255.255.0
!
! Default routes for each VRF
!
ip route vrf VPN_A 0.0.0.0 0.0.0.0 155.1.67.6
ip route vrf VPN_B 0.0.0.0 0.0.0.0 155.1.76.6
```

SW1, SW2, SW3, SW4:

```
vlan 76
```

Verification

Note

Start by checking the VRF interfaces and basic connectivity. Notice that the test commands now use the “vrf” argument to select the routing table.

```
Rack1R6#show ip vrf
  Name                             Default RD          Interfaces
  VPN_A                            100:1               Fa0/0.67
  VPN_B                            100:2               Fa0/0.76

Rack1R6#ping vrf VPN_A 155.1.67.7

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 155.1.67.7, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/8 ms

Rack1R6#ping vrf VPN_B 155.1.76.7

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 155.1.76.7, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

Now, let’s look into inter-VRF connectivity in details. First, check the CEF table for VRF VPN_A in R6:

```
Rack1R6#show ip route vrf VPN_A 192.168.7.0
Routing entry for 192.168.7.0/24
  Known via "static", distance 1, metric 0
  Routing Descriptor Blocks:
  * 155.1.76.7, via FastEthernet0/0.76
      Route metric is 0, traffic share count is 1

Rack1R6#show ip cef vrf VPN_A 192.168.7.0 detail
192.168.7.0/24, epoch 0
  nexthop 155.1.76.7 FastEthernet0/0.76
```

It appears fully legit. Now ping 172.16.7.7 from within VRF VPN_B in SW1 and vice-versa:

```
Rack1SW1#ping vrf VPN_B 172.16.7.7 source loopback 102

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.7.7, timeout is 2 seconds:
Packet sent with a source address of 192.168.7.7
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/4/9 ms

Rack1SW1#ping vrf VPN_A 192.168.7.7 source loopback 101

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.7.7, timeout is 2 seconds:
Packet sent with a source address of 172.16.7.7
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/8 ms
```

14.2 MPLS LDP

- Configure R4, R5 and R6 for MPLS label exchange using IETF standard protocol.
- Authenticate LDP peering sessions using MD5 hash and the password value of CISCO.
- Use only a single command in R4 to enable LDP on all OSPF-enabled interfaces.
- Use the physical interfaces IP addresses for the LDP session between R4 and R6.

Configuration

Note

The problem with VRF Lite feature is the scalability issue. When your VPNs span multiple routers, you need a separate link for every VPN between every pair of routers involved. Even worse, you cannot auto-provision these links using any standard signaling protocol.

A technique that allows for overcoming this limitation emerged from the technology known as tag-switching. Originally developed as a way to accelerate IP packet switching it further emerged into advanced dynamic tunneling protocol. The name finally chosen for the technology was MPLS or Multi-Protocol Label Switching.

MPLS “tunnels” are known as LSP (Label Switching Path) are similar to Frame-Relay or ATM PVCs. As you know, Frame-Relay packets are switched based on the DLCI value found in the header. This DLCI value is purely local to every switch and thee DCLI value could be rewritten every time the packet is switched out. The DLCI value could be thought as a local label used to “switch” the incoming packet. The switching decision is made based on the local table that maps incoming DLCIs into outgoing DLCIs – effectively the Frame-Relay routing table. However, keep in mind that Frame-Relay is a Layer-2 protocol, and the next header following the Frame-Relay header normally belongs to the standard IP or any other Layer 3 protocol.

The similar principle employed is MPLS; a stack of “labels” is inserted between the original Layer 2 and the Layer 3 header. Per the RFC standard, every MPLS label is a 20-bit value, with a few special reserved label values. The actual tag is 32 bits however with a few special and reserved fields. MPLS labeling acts as a shim-layer, introducing extra set of virtual-paths deployed over the Layer 2 topology being used in the network. If a packet carries the stack of MPLS labels, every router performs switching based on the topmost label found in the stack. This is done using Label FIB or LFIB, which is maps incoming labels to their outgoing equivalents. Thus, MPLS-labeled packets are switched based on a Label Lookup/Switch instead of a lookup into the IP table for the destination prefix. The first router on the edge of MPLS cloud is know as LER or Label Edge Router and is responsible for inserting (pushing) the initial label. The routers inside the MPLS cloud are know as LSRs or Label Switching Routers – they swap labels found in the packets (perform pop and push operations) and switch packets further. The last LER along the LSP is responsible for popping the label and switching the packets further using traditional prefix lookup mechanics.

The core thing about MPLS is how the labels are assigned. A tunnel path is established for every IGP prefix found in the network. This allows replacing packet switching based on destination prefix lookup with switching based on label swapping. While it does not provide much performance improvements, it allows for other important features, such as MPLS VPNs, which we are going to discuss in a separate task. Now let’s see how MPLS routers exchange their label bindings.

In Frame-Relay, in order to provision a PVC you must manually program DLCI mappings along the path. MPLS is a dynamic technology, and uses special protocol called “Label Distribution Protocol” or LDP to exchange label values. By default, every router will generate a local label for every prefix found in its routing table and advertise it via LDP to its neighbors. This is the label the adjacent routers must use when switching for this prefix via the local router. You may think of LDP working similar to distance-vector protocols – it broadcasts all local prefixes with their respective labels. As soon as local router learns the labels used by its neighbors for the same prefixes it will program the LFIB with respective label values: incoming label (which has been locally generated) replaced with outgoing label (used by the neighbor routers).

LDP is a complicated protocol defined in RFC3036. We are not going to discuss all of its functionality in details. In short, LDP works as following:

1) In order to enable MPLS switching on a interface and start LDP on the same interface, you need to enter the interface-level command **`mpls ip`**. If you have too many interfaces to enable MPLS, you may use MPLS LDP auto configuration, available when you run OSPF as your IGP protocol. Under the OSPF process, enter the command **`mpls ldp autoconfig`** to activate LDP/MPLS switching on all interfaces running OSPF.

After MPLS has been enabled on an interface, LDP will attempt to discover valid neighbors on this interface. Initially, LDP sends discovery UDP packets to the multicast IP 224.0.0.5 port 646. This corresponds to “all routers” on the local segments.

2) Upon hearing from the other LDP routers, LDP learns their LDP Routing ID, which is by default the highest Loopback IP address. You may change the

Router-ID using the command **`mpls ldp router-id <interface> force`**. If for some reason the Loopback IP addresses are unreachable, TCP connection will not establish. If you want LDP to establish a TCP connection using the physical interface IP address, use the interface command **`mpls ldp discovery transport-address interface`** command at the interface level.

3) Using the Router-ID IP addresses as sources, two routers that heard from each other establish a TCP transport connection using the destination port of 646. This connection could be authenticated using MD5 hash TCP option. The hashing key is defined per-neighbor using the command **`mpls ldp neighbor <IP> password <password>.`** The IP address here is the neighbor's LDP Router-ID. In order to make the use of the passwords mandatory you need the global command **`mpls ldp password required`**.

4) After the transport connection has been established, the routers exchange prefix and label bindings, resulting in LFIB population. The exchange is performed over the TCP connection established previously.

After the LFIB databases have been populated, label switching may occur. Notice that it is important for all routers within MPLS domain to have the same prefixes in their routing tables – otherwise the label bindings will not match. This results in inability to use route summarization with MPLS and traditional IGPs.

And finally, a short historical note. The predecessor to LDP was Cisco proprietary protocol known as Tag Switching Protocol or TDP. It was the default label exchange protocol on Cisco routers until recent IOS releases. If for some reason you are required to use the legacy and proprietary protocol, you may do so using the command **`mpls label protocol ldp`** in the global configuration mode.

R4:
```
mpls ip
!
mpls ldp router-id Loopback0 force
!
interface FastEthernet 0/1
 mpls ldp discovery transport-address interface
!
router ospf 1
 mpls ldp autoconfig
!
mpls ldp password required
mpls ldp neighbor 150.1.5.5 password CISCO
mpls ldp neighbor 150.1.6.6 password CISCO
```

R6:
```
mpls ip
!
mpls ldp router-id Loopback0 force
!
interface FastEthernet 0/0.146
 mpls ldp discovery transport-address interface
 mpls ip
!
mpls ldp password required
mpls ldp neighbor 150.1.4.4 password CISCO
```

R5:
```
mpls ip
!
mpls ldp router-id Loopback0 force
!
interface Serial 0/1/0
 mpls ip
!
interface Serial 0/0/0
 mpls ip
!
mpls ldp password required
mpls ldp neighbor 150.1.4.4 password CISCO
```

Verification

Note

Check the MPLS LDP peering sessions first. Notice the IP addresses used for the TCP peering sessions. R4 and R6 use physical interface IP addresses to accomplish this.

```
Rack1R4#show mpls ldp neighbor
    Peer LDP Ident: 150.1.5.5:0; Local LDP Ident 150.1.4.4:0
        TCP connection: 150.1.5.5.42010 - 150.1.4.4.646
        State: Oper; Msgs sent/rcvd: 27/27; Downstream
        Up time: 00:11:22
        LDP discovery sources:
          Serial0/0/0.1, Src IP addr: 155.1.0.5
        Addresses bound to peer LDP Ident:
          155.1.58.5      155.1.5.5       155.1.0.5       155.1.45.5
          150.1.5.5
    Peer LDP Ident: 150.1.6.6:0; Local LDP Ident 150.1.4.4:0
        TCP connection: 155.1.146.6.49606 - 155.1.146.4.646
        State: Oper; Msgs sent/rcvd: 12/12; Downstream
        Up time: 00:00:03
        LDP discovery sources:
          FastEthernet0/1, Src IP addr: 155.1.146.6
        Addresses bound to peer LDP Ident:
          155.1.146.6     150.1.6.6
```

Confirm authentication for the neighbors:

```
Rack1R4#show mpls ldp neighbor password
    Peer LDP Ident: 150.1.5.5:0; Local LDP Ident 150.1.4.4:0
        TCP connection: 150.1.5.5.42010 - 150.1.4.4.646
        Password: required, neighbor, in use
        State: Oper; Msgs sent/rcvd: 28/28
    Peer LDP Ident: 150.1.6.6:0; Local LDP Ident 150.1.4.4:0
        TCP connection: 155.1.146.6.49606 - 155.1.146.4.646
        Password: required, neighbor, in use
        State: Oper; Msgs sent/rcvd: 12/12
```

Note

Check MPLS forwarding tables of all three routers. Notice the labels assigned to the Loopback0 interfaces. For example, R6 advertises label 17 for 150.X.5.5/32 and receives label 16 for this prefix from R4. Many prefixes have the "Pop Label" action as a result of implicit-null binding.

```
Rack1R6#show mpls forwarding-table
Local  Outgoing      Prefix            Bytes Label   Outgoing    Next Hop
Label  Label or VC   or Tunnel Id      Switched      interface
16     16            150.1.5.5/32      0             Fa0/0.146   155.1.146.4
17     Pop Label     150.1.4.4/32      0             Fa0/0.146   155.1.146.4
18     Pop Label     204.12.1.0/24     0             Fa0/0.146   155.1.146.4
19     18            155.1.58.0/24     0             Fa0/0.146   155.1.146.4
20     Pop Label     155.1.45.0/24     0             Fa0/0.146   155.1.146.4
21     17            155.1.5.0/24      0             Fa0/0.146   155.1.146.4
22     Pop Label     155.1.0.0/24      0             Fa0/0.146   155.1.146.4

Rack1R4#show mpls forwarding-table
Local  Outgoing      Prefix            Bytes Label   Outgoing    Next Hop
Label  Label or VC   or Tunnel Id      Switched      interface
16     Pop Label     150.1.5.5/32      0             Se0/0/0.1   point2point
       No Label      150.1.5.5/32      0             Se0/1/0     point2point
17     Pop Label     155.1.5.0/24      0             Se0/0/0.1   point2point
       No Label      155.1.5.0/24      0             Se0/1/0     point2point
18     Pop Label     155.1.58.0/24     0             Se0/0/0.1   point2point
       No Label      155.1.58.0/24     0             Se0/1/0     point2point
19     Pop Label     150.1.6.6/32      0             Fa0/1       155.1.146.6
Rack1R4#

Rack1R5#show mpls forwarding-table
Local  Outgoing      Prefix            Bytes Label   Outgoing    Next Hop
Label  Label or VC   or Tunnel Id      Switched      interface
16     Pop Label     150.1.4.4/32      0             Se0/0/0     155.1.0.4
       No Label      150.1.4.4/32      0             Se0/1/0     point2point
17     Pop Label     155.1.146.0/24    0             Se0/0/0     155.1.0.4
       No Label      155.1.146.0/24    0             Se0/1/0     point2point
18     Pop Label     204.12.1.0/24     0             Se0/0/0     155.1.0.4
       No Label      204.12.1.0/24     0             Se0/1/0     point2point
19     19            150.1.6.6/32      0             Se0/0/0     155.1.0.4
       No Label      150.1.6.6/32      0             Se0/1/0     point2point
```

Note

Do a traceroute from R6 to R5 to and notice the MPLS label popping in the output.

```
Rack1R6#traceroute 150.1.5.5

Type escape sequence to abort.
Tracing the route to 150.1.5.5

  1 155.1.146.4 [MPLS: Label 16 Exp 0] 48 msec 4 msec 48 msec
  2 155.1.45.5 0 msec *  0 msec
```

14.3 MPLS Label Filtering

- Configure R4, R5 and R6 so that the only labels advertised by LDP are the ones for Loopback0 interfaces of the mentioned routers.

Configuration

Note

By default, LDP will generate and advertise labels for every prefix found in the local routing table. If you want to change this behavior and generate labels only for specific prefixes, you may use an access-list to select the prefixes eligible for label generation.

```
R4, R5, R6:
access-list 10 permit 150.1.0.0 0.0.255.255
!
no mpls ldp advertise-labels
mpls ldp advertise-labels for 10
```

Verification

Note

Check MPLS forwarding tables in R4, R5 and R6 and notice that only the Loopback0 prefixes now have labels assigned.

```
Rack1R6#show mpls forwarding-table
Local  Outgoing      Prefix            Bytes Label   Outgoing   Next Hop
Label  Label or VC   or Tunnel Id      Switched      interface
16     16            150.1.5.5/32      0             Fa0/0.146  155.1.146.4
17     Pop Label     150.1.4.4/32      0             Fa0/0.146  155.1.146.4
18     No Label      204.12.1.0/24     0             Fa0/0.146  155.1.146.4
19     No Label      155.1.58.0/24     0             Fa0/0.146  155.1.146.4
20     No Label      155.1.45.0/24     0             Fa0/0.146  155.1.146.4
21     No Label      155.1.5.0/24      0             Fa0/0.146  155.1.146.4
22     No Label      155.1.0.0/24      0             Fa0/0.146  155.1.146.4

Rack1R4#show mpls forwarding-table
Local  Outgoing      Prefix            Bytes Label   Outgoing   Next Hop
Label  Label or VC   or Tunnel Id      Switched      interface
16     Pop Label     150.1.5.5/32      0             Se0/0/0.1  point2point
       No Label      150.1.5.5/32      96            Se0/1/0    point2point
17     No Label      155.1.5.0/24      0             Se0/0/0.1  point2point
       No Label      155.1.5.0/24      0             Se0/1/0    point2point
18     No Label      155.1.58.0/24     0             Se0/0/0.1  point2point
       No Label      155.1.58.0/24     0             Se0/1/0    point2point
19     Pop Label     150.1.6.6/32      0             Fa0/1      155.1.146.6

Rack1R5#show mpls forwarding-table
Local  Outgoing      Prefix            Bytes Label   Outgoing   Next Hop
Label  Label or VC   or Tunnel Id      Switched      interface
16     Pop Label     150.1.4.4/32      0             Se0/0/0    155.1.0.4
       No Label      150.1.4.4/32      0             Se0/1/0    point2point
17     No Label      155.1.146.0/24    0             Se0/0/0    155.1.0.4
       No Label      155.1.146.0/24    0             Se0/1/0    point2point
18     No Label      204.12.1.0/24     0             Se0/0/0    155.1.0.4
       No Label      204.12.1.0/24     0             Se0/1/0    point2point
19     19            150.1.6.6/32      0             Se0/0/0    155.1.0.4
       No Label      150.1.6.6/32      0             Se0/1/0    point2point
```

14.4 MP-BGP VPNv4

- Create two new VRFs named VPN_A and VPN_B in R5 and assign VLAN58 and VLAN5 interfaces into these VRFs.
- Enable VPN route exchange between R5 and R6 using R6 as the BGP route-reflector.
- Make sure IPv4 peering sessions are not activated by default.
- By the end of this task you should be able to ping the connected interfaces inside the VPNs.

Configuration

Note

MPLS technology is perfect candidate for dynamic tunneling solution to resolve the scalability issue associated with VRF Lite. The idea of MPLS VPNs is establishing a full-mesh of dynamic MPLS LSRs between the PE (Provided Edge) routers and using those for tunneling VPN packets across the network core. In order to select the proper VRF instance on the endpoint PE router, an additional label is needed that selects the proper FIB entry associated with the target VRF. This requires two labels in MPLS stack – one label (the topmost) is the transport label, which is being swapped the entire path between the two PEs and the other label (innermost) is the VPN label, which is used to select the proper outgoing VRF CEF entry.

When the tunneling solution has been found, a way to distribute VPN routes between the sites was needed. You cannot normally establish IGP protocol adjacencies across MPLS LSRs as those are unidirectional. And even if a bi-directional tunneling solution like mGRE would be in use, establishing hundreds of adjacencies for OSPF across a network core will not work well from scaling perspective. Based on all that, a choice for BGP as universal prefix redistribution protocol has been made.

In order to support the new features, BGP functionality has been enhanced to handle the “VRF” specific routes. A new special MP-BGP (multiprotocol BGP) address family named “VPNv4” (VPN IPv4) has been added to BGP along with new NLRI format. Every VPNv4 prefix has the RD associated with it and the corresponding MPLS label, in addition to the normal BGP attributes. This allows for transporting different VPN routes together and performing best-path selection independently for every different RD. VPNv4 address-family capability is activated per-neighbor using the respective address-family configuration. By default, when you create a new BGP neighbor using the command **`neighbor <IP> remote-as N`** the default IPv4 unciast address-family is activated for this neighbor. If for some reason you don’t want this behavior and only need the

VPNv4 prefixes to be sent, you may disable the default behavior via the command **`no bgp default ipv4-unicast`**.

There is one special limitation for iBGP peering sessions that you want to enable for VPNv4 prefix exchange. First of all, they must be sourced off a Loobpack0 interface and secondly, this interface must have a /32 mask. This is needed because the BGP peering IP address is used as the NEXT_HOP for the locally originated VPNv4 prefixes. When the remote BGP router receives those prefixes, it performs recursive routing lookup for the NEXT_HOP value and finds a label in LFIB. This label is used as the transport label in the receiving router. Effectively, the NEXT_HOP is used to build the “tunnel” or the transport LSP between the two PEs. The VPN label is generated by BGP process on the advertising router and directly corresponds to the local VRF route. As for the /32 restriction, it is needed to guarantee the fact the transport LSP terminates on the particular PE router, and not some shared network segment.

In order to inject a particular VRF’s routes into BGP, you need to activate the respective address-family under the BGP process and enable route redistribution (e.g. static or connected). All the respective routes belonging to the particular VRF will be injected into BGP table with their RDs and have their VPN labels generated. The import process is a bit more complicated and based on the concept of “Route Targets”.

Route Target is an instance of BGP extended community attributes. These BGP attributes are transitive, and encoded as 64-bit values (as opposed to normal 32-bit communities). They are used for “enhanced” tagging of VPNv4 prefixes. The need for route-target arises from the fact that one cannot just use Route Distinguishers for prefix importing/exporting, as the routes with the same RD may eventually belong to multiple VRFs, when you share their routes.

Here is how route-target based import works. By default, all prefixes redistributed from a VRF into BGP process are tagged with the extended community **`X:Y`** specified under VRF configuration via the command **`route-target export X:Y`**. You may specify as many export commands as you want to tag prefixes with multiple attributes. On the receiving side, the VRF will import the BGP VPNv4 prefixes with the route-targets matching the local command **`route-target import X:Y`**. That is, the import process is based entirely on the route-targets, and not the RDs. If the imported routers used to have RD different from the one used by the local VRF, they are “naturalized” by having the RD changed to the local value. Theoretically, you may assign a route-target to every VPN site, and specify fine-tuned import policies, to select the remote site routes accepted locally. Finally, notice that the use of command **`route-target both X:Y`** means import and export statements at the same time.

R4:
```
router bgp 100
!
! Disable automatic IPv4 neighbor activation
!
no bgp default ipv4-unicast
 neighbor 150.1.5.5 remote-as 100
 neighbor 150.1.5.5 update-source Loopback0
 neighbor 150.1.6.6 remote-as 100
 neighbor 150.1.6.6 update-source Loopback0
address-family vpnv4 unicast
 neighbor 150.1.5.5 activate
 neighbor 150.1.6.6 activate
!
! Enable sending of ext. communities with VPNv4 routes
!
 neighbor 150.1.5.5 send-community extended
 neighbor 150.1.6.6 send-community extended
 neighbor 150.1.5.5 route-reflector-client
 neighbor 150.1.6.6 route-reflector-client
```

R5:
```
ip vrf VPN_A
 rd 100:1
 route-target both 100:1
!
ip vrf VPN_B
 rd 100:2
 route-target both 100:2
!
interface FastEthernet 0/0
 ip vrf forwarding VPN_A
 ip address 155.1.58.5 255.255.255.0
!
interface FastEthernet 0/1
 ip vrf forwarding VPN_B
 ip address 155.1.5.5 255.255.255.0
```

R6:
```
ip vrf VPN_A
 rd 100:1
 route-target both 100:1
!
ip vrf VPN_B
 rd 100:2
 route-target both 100:2
```

```
R5 & R6:
router bgp 100
 no bgp default ipv4-unicast
 neighbor 150.1.4.4 remote-as 100
 neighbor 150.1.4.4 update-source Loopback0
 address-family vpnv4 unicast
 neighbor 150.1.4.4 activate
 neighbor 150.1.4.4 send-community extended
!
! Activate the address families for VRFs and redistribute routes
!
 address-family ipv4 vrf VPN_A
  redistribute connected
  redistribute static
 address-family ipv4 vrf VPN_B
  redistribute connected
  redistribute static
```

Verification

Note

Check the PE route and look for BGP-learned prefixes in every VRF's routing table:

```
Rack1R5#show ip route vrf VPN_A

Routing Table: VPN_A
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS
level-2
       ia - IS-IS inter area, * - candidate default, U - per-user
static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     155.1.0.0/24 is subnetted, 2 subnets
C       155.1.58.0 is directly connected, FastEthernet0/0
B       155.1.67.0 [200/0] via 150.1.6.6, 00:03:08
```

```
Rack1R5#show ip route vrf VPN_B

Routing Table: VPN_B
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS
level-2
       ia - IS-IS inter area, * - candidate default, U - per-user
static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     155.1.0.0/24 is subnetted, 2 subnets
C       155.1.5.0 is directly connected, FastEthernet0/1
B       155.1.76.0 [200/0] via 150.1.6.6, 00:04:03
```

Now check the route-reflector BGP table for VPNv4 prefixes. Notice how prefixes are grouped based on the RD.

```
Rack1R4#show bgp vpnv4 unicast all
BGP table version is 5, local router ID is 150.1.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
*>i155.1.58.0/24    150.1.5.5                0    100      0 ?
*>i155.1.67.0/24    150.1.6.6                0    100      0 ?
Route Distinguisher: 100:2
*>i155.1.5.0/24     150.1.5.5                0    100      0 ?
*>i155.1.76.0/24    150.1.6.6                0    100      0 ?
```

Note

Check the route-target values associated with the various prefixes in R4’s VPNv4 BGP table:

```
Rack1R4#show bgp vpnv4 unicast all 155.1.76.0 255.255.255.0
BGP routing table entry for 100:2:155.1.76.0/24, version 5
Paths: (1 available, best #1, no table)
  Advertised to update-groups:
        2
  Local, (Received from a RR-client)
    150.1.6.6 (metric 2) from 150.1.6.6 (150.1.6.6)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      Extended Community: RT:100:2
      mpls labels in/out nolabel/19

Rack1R4#show bgp vpnv4 unicast all 155.1.67.0 255.255.255.0
BGP routing table entry for 100:1:155.1.67.0/24, version 3
Paths: (1 available, best #1, no table)
  Advertised to update-groups:
        2
  Local, (Received from a RR-client)
    150.1.6.6 (metric 2) from 150.1.6.6 (150.1.6.6)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      Extended Community: RT:100:1
      mpls labels in/out nolabel/21
```

Note

Now check the end-to-end connectivity across each VPN. Notice the label stack used to tunnel the VPN packets.

```
Rack1R5#ping vrf VPN_A 155.1.67.6

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 155.1.67.6, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 60/60/64 ms

Rack1R5#traceroute vrf VPN_A 155.1.67.6

Type escape sequence to abort.
Tracing the route to 155.1.67.6

  1 155.1.0.4 [MPLS: Labels 19/21 Exp 0] 60 msec 64 msec 60 msec
  2 155.1.67.6 32 msec *  28 msec

Rack1R5#ping vrf VPN_B 155.1.76.6

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 155.1.76.6, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 60/60/64 ms
```

```
Rack1R5#**traceroute vrf VPN_B 155.1.76.7**

Type escape sequence to abort.
Tracing the route to 155.1.76.7

  1 155.1.0.4 [MPLS: Labels 19/19 Exp 0] 60 msec 60 msec 60 msec
  2 155.1.76.6 28 msec 28 msec 32 msec
  3 155.1.76.7 32 msec *  28 msec

Rack1R5#**traceroute vrf VPN_B 155.1.76.6**

Type escape sequence to abort.
Tracing the route to 155.1.76.6

  1 155.1.0.4 [MPLS: Labels 19/19 Exp 0] 60 msec 60 msec 60 msec
  2 155.1.76.6 32 msec *  32 msec
```

Note

Check the MPLS labels (VPN labels) assigned to the VPN prefixes at the PE routers. There are two labels in stack, one VPN label and the other is transport label. You may find the VPN label in the BGP table and the transport label could be found by looking up the VPNv4 BGP next-hop in the MPLS forwarding table.

```
Rack1R5#show ip bgp vpnv4 vrf VPN_A 155.1.67.0
BGP routing table entry for 100:1:155.1.67.0/24, version 61
Paths: (1 available, best #1, table VPN_A)
  Not advertised to any peer
  Local
    150.1.6.6 (metric 66) from 150.1.4.4 (150.1.4.4)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      Extended Community: RT:100:1
      Originator: 150.1.6.6, Cluster list: 150.1.4.4
      mpls labels in/out nolabel/21

Rack1R5#show mpls forwarding-table 150.1.6.6
Local  Outgoing      Prefix            Bytes Label   Outgoing   Next
Hop
Label  Label or VC   or Tunnel Id      Switched      interface
19     19            150.1.6.6/32      0             Se0/0/0
155.1.0.4
       No Label      150.1.6.6/32      0             Se0/1/0
point2point
```

14.5 MP-BGP Prefix Filtering

- Create a new Loopback101 interface in R5’s VRF VPN_A with the IP address of 172.16.5.5/24.
- Create a new Loopback102 interface in R6’s VRF VPN_B with the IP address of 192.168.6.6/24.
- Configure the network to provide bi-directional connectivity between the two new subnets.
- Make sure R6’s VPN_A does not see the prefix 172.16.5.0/24 and R5’s VPN_B does not see the prefix 192.168.6.0/24.

Configuration

Note

As we found in the previous task, route-target tagging applies to all routes injected from VRF into BGP process. Sometimes it is required to have more granular control over route tagging. In order to accomplish this, Cisco IOS has the feature know as export/import maps. They are configured under VRF configuration context using the command **`{import|export} map <ROUTE_MAP_NAME>.`** The special export route-map associated with the VRF could match the prefixes based on the prefix-lists, access-lists or extended-communities. All routes not permitted in export route-map are not exported into the BGP process. The export route-map may also set extended-community selectively, using the command **`set extcommunity rt`**. This allows for selective tagging of VPN routes.

The import map is used less often than the export map, but still has some good use. First of all, it allows controlling all routes imported into VRF from BGP based on prefix-lists, access-lists, or extended/standard communities (or any other applicable BGP attribute). This might be helpful when, for example, you want to import all other sites routes with exception to the default route. Notice that by default all prefixes not permitted with the import-map are implicitly denied and not imported.

```
R5:
interface Loopback101
 ip vrf forwarding VPN_A
 ip address 172.16.5.5 255.255.255.0
!
ip prefix-list LO101 permit 172.16.5.0/24
!
route-map VPN_A_EXPORT permit 10
 match ip address prefix-list LO101
 set extcommunity rt 100:55
!
route-map VPN_A_EXPORT permit 20
```

```
 set extcommunity rt 100:1
!
ip vrf VPN_A
 export map VPN_A_EXPORT
 route-target import 100:66
```

R6:

```
interface Loopback102
 ip vrf forwarding VPN_B
 ip address 192.168.6.6 255.255.255.0
!
ip prefix-list LO102 permit 192.168.6.0/24
!
route-map VPN_B_EXPORT permit 10
 match ip address prefix-list LO102
 set extcommunity rt 100:66
!
route-map VPN_B_EXPORT permit 20
 set extcommunity rt 100:2
!
ip vrf VPN_B
 export map VPN_B_EXPORT
 route-target import 100:55
```

Verification

Note

Check the prefix 172.16.5.0/24 in R6's VPN_A and VPN_B routing tables.

```
Rack1R6#show ip route vrf VPN_B 172.16.5.0
Routing entry for 172.16.5.0/24
  Known via "bgp 100", distance 200, metric 0, type internal
  Last update from 150.1.5.5 00:00:59 ago
  Routing Descriptor Blocks:
  * 150.1.5.5 (Default-IP-Routing-Table), from 150.1.4.4, 00:00:59 ago
      Route metric is 0, traffic share count is 1
      AS Hops 0

Rack1R6#show ip route vrf VPN_A 172.16.5.0
% Network not in table
```

Note

Perform symmetric check in R5's VRFs for the prefix 192.168.6.0/24

```
Rack1R5#show ip route vrf VPN_A 192.168.6.0
Routing entry for 192.168.6.0/24
  Known via "bgp 100", distance 200, metric 0, type internal
  Last update from 150.1.6.6 00:02:27 ago
  Routing Descriptor Blocks:
  * 150.1.6.6 (Default-IP-Routing-Table), from 150.1.4.4, 00:02:27 ago
      Route metric is 0, traffic share count is 1
      AS Hops 0

Rack1R5#show ip route vrf VPN_B 192.168.6.0
% Network not in table
```

Check the route-target values associated with the new prefixes.

```
Rack1R4#show ip bgp vpnv4 rd 100:1 172.16.5.0
BGP routing table entry for 100:1:172.16.5.0/24, version 7
Paths: (1 available, best #1, no table)
  Advertised to update-groups:
        2
  Local, (Received from a RR-client)
    150.1.5.5 (metric 65) from 150.1.5.5 (150.1.5.5)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      Extended Community: RT:100:55
      mpls labels in/out nolabel/22

Rack1R4#show ip bgp vpnv4 rd 100:2 192.168.6.0
BGP routing table entry for 100:2:192.168.6.0/24, version 9
Paths: (1 available, best #1, no table)
  Advertised to update-groups:
        2
  Local, (Received from a RR-client)
    150.1.6.6 (metric 2) from 150.1.6.6 (150.1.6.6)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      Extended Community: RT:100:66
      mpls labels in/out nolabel/25
```

14.6 PE-CE Routing with RIP

- Use RIP as the PE-CE routing protocol for VPN_B sites.
- Remove all static routing used to provide VPN_B connectivity.
- Configure R4 so that VLAN43 connection belongs to VPN_B as well.
- Preserve RIP metric values learned from CE routers.

Configuration

Note

Even though MP-BGP is the “carrier” routing protocol for MPLS VPNs, a wide variety of PE-CE routing protocols could be used. The PE-CE routing protocols run independently at every site, using MP-BGP to exchange their routing information. The route exchange is performed via prefix redistribution between MP-BGP process and the respective PE-CE routing protocol. A number of features have been added to MP-BGP in order to preserve some valuable routing protocol information that otherwise might be lost during redistribution. These features vary from one PE-CE protocol to another.

We start with the most simple of all PE-CE protocols, RIPv2. With RIPv2 you may enable the respective address family under the process configuration mode using the command **`address-family ipv4 vrf <VRF_NAME>.`** You may then specify the network commands as usual. Notice that the timer settings, version and auto-summary settings are global for all VRFs with this approach. When you need to redistribute the MP-BGP routes into RIP, use the command **`redistribute bgp <N> metric {X|transparent}`** under the respective address family. Here N is the BGP process number (AS#) and X is the metric assigned to the RIP routes. If you are using the keyword **`transparent`**, the RIP metrics will be recovered from BGP MED attribute, which in turn is copied from RIP metrics learned at the remote site. This allows for transparent preservation of RIPv2 metric values across the VPN and better path selection in case of backdoor links.

Naturally, if you want to inject the respective VRF’s RIP routes into BGP, you simply redistribute them under the corresponding address-family. The MED value for the new VPNv4 prefixes will be initialized from the RIP metrics of redistributed routes.

R4:
```
ip vrf VPN_B
 rd 100:2
 no route-target both 100:1
 route-target both 100:2
!
interface FastEthernet0/0
 ip vrf forwarding VPN_B
 ip address 204.12.1.4 255.255.255.0
!
router rip
 version 2
 no auto-summary
 address-family ipv4 vrf VPN_B
!
! The keyword “transparent” carries RIP hop count in
! BGP metric attribute
!
 redistribute bgp 100 metric transparent
 network 204.12.1.0
!
router bgp 100
 address-family ipv4 vrf VPN_B
  redistribute rip
```

R6:
```
router rip
 version 2
 no auto-summary
 address-family ipv4 vrf VPN_B
 redistribute bgp 100 metric transparent
 network 155.1.0.0
!
no ip route vrf VPN_B 172.16.7.0 255.255.255.0 FastEthernet0/0.67
155.1.67.7
```

SW1:
```
no ip route vrf VPN_B 0.0.0.0 0.0.0.0 155.1.76.6
!
router rip
 version 2
 no auto-summary
 address-family ipv4 vrf VPN_B
 network 155.1.0.0
 network 172.16.0.0
```

Verification

Note

Check VPN_A routes in SW1 and make sure you see the RIP prefixes advertised by BB3:

```
Rack1SW1#show ip route vrf VPN_B

Routing Table: VPN_B
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS
level-2
       ia - IS-IS inter area, * - candidate default, U - per-user
static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

R    204.12.1.0/24 [120/1] via 155.1.76.6, 00:00:04, Vlan76
     155.1.0.0/24 is subnetted, 2 subnets
R       155.1.5.0 [120/1] via 155.1.76.6, 00:00:04, Vlan76
C       155.1.76.0 is directly connected, Vlan76
     172.16.0.0/24 is subnetted, 1 subnets
R       172.16.5.0 [120/1] via 155.1.76.6, 00:00:04, Vlan76
C    192.168.7.0/24 is directly connected, Loopback102
     31.0.0.0/16 is subnetted, 4 subnets
R       31.3.0.0 [120/2] via 155.1.76.6, 00:00:04, Vlan76
R       31.2.0.0 [120/2] via 155.1.76.6, 00:00:04, Vlan76
R       31.1.0.0 [120/2] via 155.1.76.6, 00:00:04, Vlan76
R       31.0.0.0 [120/2] via 155.1.76.6, 00:00:05, Vlan76
     30.0.0.0/16 is subnetted, 4 subnets
R       30.2.0.0 [120/2] via 155.1.76.6, 00:00:05, Vlan76
R       30.3.0.0 [120/2] via 155.1.76.6, 00:00:05, Vlan76
R       30.0.0.0 [120/2] via 155.1.76.6, 00:00:05, Vlan76
R       30.1.0.0 [120/2] via 155.1.76.6, 00:00:05, Vlan76
```

Note

Check the routing table for VPN_B in R4 and look for the same prefixes:

```
Rack1R4#show ip route vrf VPN_B

Routing Table: VPN_A
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS
level-2
       ia - IS-IS inter area, * - candidate default, U - per-user
static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

C    204.12.1.0/24 is directly connected, FastEthernet0/0
     155.1.0.0/24 is subnetted, 2 subnets
B       155.1.58.0 [200/0] via 150.1.5.5, 00:32:29
B       155.1.67.0 [200/0] via 150.1.6.6, 00:32:29
     31.0.0.0/16 is subnetted, 4 subnets
R       31.3.0.0 [120/1] via 204.12.1.254, 00:00:25, FastEthernet0/0
R       31.2.0.0 [120/1] via 204.12.1.254, 00:00:25, FastEthernet0/0
R       31.1.0.0 [120/1] via 204.12.1.254, 00:00:25, FastEthernet0/0
R       31.0.0.0 [120/1] via 204.12.1.254, 00:00:25, FastEthernet0/0
     30.0.0.0/16 is subnetted, 4 subnets
R       30.2.0.0 [120/1] via 204.12.1.254, 00:00:25, FastEthernet0/0
R       30.3.0.0 [120/1] via 204.12.1.254, 00:00:00, FastEthernet0/0
R       30.0.0.0 [120/1] via 204.12.1.254, 00:00:00, FastEthernet0/0
R       30.1.0.0 [120/1] via 204.12.1.254, 00:00:00, FastEthernet0/0
```

Note

Look at R6’s BGP table to see the RIP prefixes being carried in BGP updates. Notice the “metric” field value:

```
Rack1R6#show ip bgp vpnv4 vrf VPN_B
BGP table version is 210, local router ID is 150.1.6.6
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:2 (default for vrf VPN_B)
*>i30.0.0.0/16      150.1.4.4                1    100      0 ?
*>i30.1.0.0/16      150.1.4.4                1    100      0 ?
*>i30.2.0.0/16      150.1.4.4                1    100      0 ?
*>i30.3.0.0/16      150.1.4.4                1    100      0 ?
*>i31.0.0.0/16      150.1.4.4                1    100      0 ?
*>i31.1.0.0/16      150.1.4.4                1    100      0 ?
*>i31.2.0.0/16      150.1.4.4                1    100      0 ?
*>i31.3.0.0/16      150.1.4.4                1    100      0 ?
*>i155.1.5.0/24     150.1.5.5                0    100      0 ?
*> 155.1.76.0/24    0.0.0.0                  0         32768 ?
*>i172.16.5.0/24    150.1.5.5                0    100      0 ?
*  172.16.7.0/24    155.1.67.7               0         32768 ?
*> 192.168.6.0      0.0.0.0                  0         32768 ?
*>i204.12.1.0       150.1.4.4                0    100      0 ?
```

Note

Test end-to-end connectivity across VPN_B:

```
Rack1SW1#traceroute vrf VPN_B 30.0.0.1

Type escape sequence to abort.
Tracing the route to 30.0.0.1

  1 155.1.76.6 0 msec 0 msec 8 msec
  2 204.12.1.4 0 msec 0 msec 0 msec
  3 204.12.1.254 9 msec *  0 msec
```

14.7 PE-CE Routing with OSPF

- Use OSPF as the PE-CE routing protocol for VPN_A sites configuring PE-CE routers in the same area 1.
- Use the same OSPF process IDs at R6 and R5 but ensure SW1 and SW2 can reach each other.
- Create a new Loopback interface in SW2 with the IP address 172.16.8.8/24 and make sure R6 sees only a /16 summary for this prefix.

Configuration

Note

OSPF is a link-state routing protocol, which relies on building a network topology graph prior to calculating the optimal routes. However, as we remember MPLS VPNs rely on using MP-BGP to transport the routing information across the MPLS backbone. This means no OSPF adjacencies could be established across the core. However, as we know, OSPF uses distance-vector like updates sent across Area 0 to redistribute routing information between different areas. This allows for interpreting the MP-BGP cloud as one “super area 0” that is used to link all OSPF areas at different sites. This special “virtual area” is called OSPF super-backbone and it is emulated by passing OSPF VRF routing information in MP-BGP updates. The use of the super-backbone allows for avoiding using area 0 at all, as the super-backbone performs the same function. You may have non-zero OSPF areas at different VPN sites connecting via MP-BGP mesh without any need for area 0 at any site. However, it’s perfectly legit to have area 0 at different sites along with non-zero areas attached to the super-backbone as well. The main design principle that should be followed is that all areas are connected to the super-backbone in loop-less start-like manner.

We briefly discussed BGP extended communities before. With OSPF, special extended communities are used to propagate addition OSPF prefix information. All OSPF routes redistributed into MP-BGP are treated pretty much like Type 3 summary LSAs, as they enter the super-backbone from other areas. When injected into BGP, OSPF prefixes have two extended-community attributes attached to them. One of the attributes is known as domain-id, which equal to the ospf process numbers in the local router OR explicitly configured using the command **`domain-id`** under the OSPF process. The purpose of this attribute is to identify OSPF processes belonging to different VPNs. It is assumed that you configured all OSPF processes within the same VPN using the same domain-id (e.g. the same process number). If for some reason you exchange routes between two different VPNs using different domain-ids, the OSPF process will interpret all such prefixes as if there are Type-5 External LSAs, effectively external routes.

The other extended community attribute is known as OSPF route-type which has three significant fields: source area, route-type and option. They are usually depicted as tripe X:Y:Z. Here Y=2 for intra-area learned prefix, Y=3 for inter-area routes, Y=5 for external prefixes and Y=7 for NSSA routes. The last value, Z, signalizes the metric type for Y=5 or 7 - it's either 1 for E1 and 2 for E2. Notice that all routes redistributed from BGP into OSPF appear like *inter-area* routes even if they belong to the same area number at different sites. This effect is due to the fact that LSAs cross the super-backbone and essentially are inter-area routes.

The last BGP attribute used to carry OSPF information is MED or metric, which copies the original route's metric from the routing table. The route-type attribute is needed to allow for proper OSPF best-path calculation when routes are inserted into OSPF database based on redistribution from BGP. Notice that the routes travelling the MP-BGP cloud do not get any increment in their metric unless you manually change the MED attribute for incoming BGP prefixes. This might be needed sometimes for proper OSPF best-path selection.

All the above mechanics is implemented automatically once you configure route redistribution between a VRF process and BGP. Notice that you create a separate OSPF process for every VRF using the command **`router ospf X vrf NAME`**, unlike with RIP, where the whole process is shared among VRFs.

The fact that there is an additional backbone area introduces the possibility of routing loops if for some reason the VPN sites are not connected in a star-like manner. To reduce this risk, OSPF implements some basic loop prevention rules. First of all, all summary LSAs generated from the routes redistributed from BGP have special "Down" bit set in LSA headers. If a router receives a summary-LSA with the down bit set on an interface that belongs to VRF, it simply drops this LSA. This is needed to prevent the case of routing loops for multihomed sites, when a summary LSA is flooded across the CE site and delivered back to another PE. However, this feature may have undesirable effect when you have a CE router configured with multiple VRFs. In this case, you may want to enter the OSPF process command **`capability vrf-lite`** on the CE router. This will disable the default loop-prevention capability. However, some IOS versions do not support this feature (e.g. older IOSes or some Catalyst IOS revisions). If you have such router configured for multi-VRF and experience route blackholing, configure PE routers with different domain-IDs – this will force all redistributed routes to become external and bypass the down-bit check. This is the trick used in the solution for this particular scenario, as the Catalyst switches do not support the OSPF "VRF-lite" feature.

```
R5:
router ospf 100 vrf VPN_A
 domain-id 0.0.0.5
 log-adjacency-changes
 redistribute bgp 100 subnets
 network 0.0.0.0 255.255.255.255 area 1
!
router bgp 100
address-family ipv4 vrf VPN_A
 redistribute ospf 100 vrf VPN_A

R6:
router ospf 100 vrf VPN_A
 domain-id 0.0.0.6
 log-adjacency-changes
 redistribute bgp 100 subnets
 network 0.0.0.0 255.255.255.255 area 1
 !
 ! We use the summary address because the routes are generated
 ! using type 5 external LSAs.
 !
 summary-address 172.16.0.0 255.255.0.0
!
router bgp 100
address-family ipv4 vrf VPN_A
 redistribute ospf 100 vrf VPN_A

SW1:
no ip route vrf VPN_B 0.0.0.0 0.0.0.0 155.1.76.6
!
router ospf 1 vrf VPN_A
 network 0.0.0.0 0.0.0.0 area 1

SW2:
ip routing
!
router ospf 1
 network 0.0.0.0 0.0.0.0 area 1
!
interface Loopback100
 ip address 172.16.8.8 255.255.255.0
```

Verification

Note

Check the routing tables of SW2 and SW1 VRF_A. Notice that SW1 sees the /16 summary for 172.16.8.8/24 prefix (actually it was /32 due to the OSPF loopback network type).

```
Rack1SW2#show ip route ospf
     155.1.0.0/24 is subnetted, 3 subnets
O E2    155.1.67.0 [110/1] via 155.1.58.5, 00:17:22, Vlan58
     172.16.0.0/16 is variably subnetted, 4 subnets, 3 masks
O E2    172.16.7.7/32 [110/2] via 155.1.58.5, 00:17:05, Vlan58
O       172.16.5.5/32 [110/2] via 155.1.58.5, 00:17:34, Vlan58
O E2    172.16.0.0/16 [110/2] via 155.1.58.5, 00:05:12, Vlan58
O E2 192.168.6.0/24 [110/1] via 155.1.58.5, 00:17:34, Vlan58

Rack1SW1#show ip route vrf VPN_A

Routing Table: VPN_A
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS
level-2
       ia - IS-IS inter area, * - candidate default, U - per-user
static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     155.1.0.0/24 is subnetted, 3 subnets
O E2    155.1.8.0 [110/2] via 155.1.67.6, 00:17:53, Vlan67
O E2    155.1.58.0 [110/1] via 155.1.67.6, 00:17:53, Vlan67
C       155.1.67.0 is directly connected, Vlan67
     172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks
C       172.16.7.0/24 is directly connected, Loopback101
O E2    172.16.0.0/16 [110/2] via 155.1.67.6, 00:06:06, Vlan67
     150.1.0.0/32 is subnetted, 1 subnets
O E2    150.1.8.8 [110/2] via 155.1.67.6, 00:17:53, Vlan67
```

Note

Check that OSPF processes in R5 and R6 are connected to OSPF super-backbone. Notice that both routers are ASBRs since they redistribute BGP routes.

```
Rack1R5#show ip ospf 100
 Routing Process "ospf 100" with ID 172.16.5.5
   Domain ID type 0x0005, value 0.0.0.5
 Start time: 1d07h, Time elapsed: 16:48:19.640
 Supports only single TOS(TOS0) routes
 Supports opaque LSA
 Supports Link-local Signaling (LLS)
 Supports area transit capability
 Connected to MPLS VPN Superbackbone, VRF VPN_A
 It is an area border and autonomous system boundary router
 Redistributing External Routes from,
    bgp 100, includes subnets in redistribution
 Router is not originating router-LSAs with maximum metric
 Initial SPF schedule delay 5000 msecs
 Minimum hold time between two consecutive SPFs 10000 msecs
 Maximum wait time between two consecutive SPFs 10000 msecs
 Incremental-SPF disabled
 Minimum LSA interval 5 secs
 Minimum LSA arrival 1000 msecs
 LSA group pacing timer 240 secs
 Interface flood pacing timer 33 msecs
 Retransmission pacing timer 66 msecs
 Number of external LSA 4. Checksum Sum 0x02404D
 Number of opaque AS LSA 0. Checksum Sum 0x000000
 Number of DCbitless external and opaque AS LSA 0
 Number of DoNotAge external and opaque AS LSA 0
 Number of areas in this router is 1. 1 normal 0 stub 0 nssa
 Number of areas transit capable is 0
 External flood list length 0
 IETF NSF helper support enabled
 Cisco NSF helper support enabled
    Area 1
        Number of interfaces in this area is 2 (1 loopback)
        Area has no authentication
        SPF algorithm last executed 16:48:16.004 ago
        SPF algorithm executed 2 times
        Area ranges are
        Number of LSA 3. Checksum Sum 0x01063B
        Number of opaque link LSA 0. Checksum Sum 0x000000
        Number of DCbitless LSA 0
        Number of indication LSA 0
        Number of DoNotAge LSA 0
        Flood list length 0

Rack1R6#show ip ospf 100
 Routing Process "ospf 100" with ID 155.1.67.6
   Domain ID type 0x0005, value 0.0.0.6
 Start time: 1w1d, Time elapsed: 17:18:03.756
 Supports only single TOS(TOS0) routes
 Supports opaque LSA
 Supports Link-local Signaling (LLS)
 Supports area transit capability
 Connected to MPLS VPN Superbackbone, VRF VPN_A
 It is an area border and autonomous system boundary router
 Redistributing External Routes from,
    bgp 100, includes subnets in redistribution
 Router is not originating router-LSAs with maximum metric
 Initial SPF schedule delay 5000 msecs
 Minimum hold time between two consecutive SPFs 10000 msecs
 Maximum wait time between two consecutive SPFs 10000 msecs
 Incremental-SPF disabled
 Minimum LSA interval 5 secs
 Minimum LSA arrival 1000 msecs
 LSA group pacing timer 240 secs
 Interface flood pacing timer 33 msecs
 Retransmission pacing timer 66 msecs
 Number of external LSA 4. Checksum Sum 0x021F17
 Number of opaque AS LSA 0. Checksum Sum 0x000000
 Number of DCbitless external and opaque AS LSA 0
 Number of DoNotAge external and opaque AS LSA 0
 Number of areas in this router is 1. 1 normal 0 stub 0 nssa
 Number of areas transit capable is 0
 External flood list length 0
 IETF NSF helper support enabled
 Cisco NSF helper support enabled
    Area 1
        Number of interfaces in this area is 1
        Area has no authentication
        SPF algorithm last executed 17:17:59.820 ago
        SPF algorithm executed 2 times
        Area ranges are
        Number of LSA 3. Checksum Sum 0x0118E5
        Number of opaque link LSA 0. Checksum Sum 0x000000
        Number of DCbitless LSA 0
        Number of indication LSA 0
        Number of DoNotAge LSA 0
        Flood list length 0
```

Note

Now check the extended community attributes associated with OSPF prefixes carried from R5 to R6. Take any prefix, for example 172.16.8.8. Pay attention to the Domain ID encoded in 0x000000**05**0200 as 0x05 and the route-type attribute value of 0.0.0.1:2:0 which stands for intra-area route received from area 1.

```
Rack1R6#show bgp vpnv4 unicast vrf VPN_A 172.16.8.8
BGP routing table entry for 100:1:172.16.8.8/32, version 236
Paths: (1 available, best #1, table VPN_A)
  Not advertised to any peer
  Local
    150.1.5.5 (metric 66) from 150.1.4.4 (150.1.4.4)
      Origin incomplete, metric 2, localpref 100, valid, internal, best
      Extended Community: RT:100:1 OSPF DOMAIN ID:0x0005:0x000000050200
        OSPF RT:0.0.0.1:2:0 OSPF ROUTER ID:172.16.5.5:0
      Originator: 150.1.5.5, Cluster list: 150.1.4.4
      mpls labels in/out nolabel/25
```

Note

Now you may want to check the OSPF database in SW1 to see what type of LSA it has for the subnet 172.16.8.0/24

```
Rack1SW1#show ip ospf database external 172.16.8.8

            OSPF Router with ID (150.1.7.7) (Process ID 1)

Rack1SW1#show ip ospf database external 172.16.0.0

            OSPF Router with ID (150.1.7.7) (Process ID 1)

                Type-5 AS External Link States

  Routing Bit Set on this LSA
  LS age: 779
  Options: (No TOS-capability, DC)
  LS Type: AS External Link
  Link State ID: 172.16.0.0 (External Network Number )
  Advertising Router: 155.1.67.6
  LS Seq Number: 80000001
  Checksum: 0x32DD
  Length: 36
  Network Mask: /16
        Metric Type: 2 (Larger than any link state path)
        TOS: 0
        Metric: 2
        Forward Address: 0.0.0.0
        External Route Tag: 0
```

Note

Now verify end-to-end connectivity across the VPN:

```
Rack1SW1#traceroute vrf VPN_A 172.16.8.8

Type escape sequence to abort.
Tracing the route to 172.16.8.8

  1 155.1.67.6 0 msec 0 msec 0 msec
  2 155.1.146.4 101 msec 50 msec 101 msec
  3 155.1.58.5 0 msec 8 msec 0 msec
  4 155.1.58.8 9 msec *  0 mses
```

14.8 OSPF Sham-Link

- Reconfigure SW1 to a pure CE router in VRF VPN_A, removing VRF-lite configuration.
- Preserve OSPF routing between VPN_A sites at R5 and R6.
- Provision a back-door link between SW1 and SW2 using any the interconnections as L3 interface.
- Configure VPN_A OSPF processes in R5 and R6 for the same domain-id.
- Ensure SW1 and SW2 prefer the path across MPLS core to the backdoor link.

Configuration

Note

As we learned in the previous scenario, OSPF prefixes are transported via MPLS VPN core using MP-BGP and interpreted either as type-3 summary LSAs or type-5 external LSAs on the receiving end. This may pose some difficulties in case when there is a backdoor link connecting two VPN sites directly. Many times the backdoor link is supposed to be used as a backup path (i.e. it’s a slow leased line) and the MPLS VPN cloud should be used as the primary way between the two sites. However, is the link is in the same area as the PE/CE routers, the PE routers will prefer the path across the back-door link, as OSPF treats all paths across it as intra-area, and the prefixes received via MP-BGP are interpreted as inter-area.

This is a fundamental issue, which requires the MPLS core to be a part of the same OSPF area to resolve the conflict. The solution is called OSPF sham-link, which is a special tunnel similar to virtual-link connecting two PE routers and configured in the same area as the PE routers. This link is used to establish an OSPF adjacency and exchange LSAs. The LSAs are then loaded in the OSPF database and the sham-link is used for intra-area path computations. However, when the routes are being installed in the respective VRF RIB, the forwarding information is based on looking up the MP-BGP learned routes based on exact prefix-match. The corresponding VPN and transport labels are then used for actual packet forwarding across the MPLS core. Thus the information loaded across the sham-link is used only for SPF calculations and best-path selection – the actual forwarding is being done based on the information learned via MP-BGP.

Configuring sham-links is a bit different from configuring virtual-links. First and most important, sham-links are sourced off actual interfaces configured in the respective VRF. Commonly these are Loopback interfaces, used as endpoints for the sham-link tunnel. Notice that the IP addresses for these interfaces should be advertised into the VRF routing table by means other than OSPF, most commonly via BGP. After you have the endpoints reachable across the MPLS VPN core, you may configure the sham-link using the command **`area 1 sham-link <SRC> <DST> cost X`**. Here <SRC> and <DST> are the IP addresses for the sham-link source and destination. The cost is the OSPF metric value associated with traversing the MPLS core. If the endpoints are reachable, OSPF will establish an adjacency on the link treating it a point-to-point connection and re-calculating the shortest paths.

In the solution below, we change the OSPF network command to cover only the interfaces that should establish OSPF adjacencies. The sham-links endpoints should not be advertised into OSPF. Also note that the OSPF summary command we configured in the previous tasks no longer has any effect. SW1 and SW2 see each other's routes as intra-area even though they are sent across the MPLS core network.

R5:
```
router ospf 100 vrf VPN_A
 no domain-id 0.0.0.5
 area 1 sham-link 150.1.55.55 150.1.66.66 cost 1
 !
 ! We need to make sure we're not advertising the sham-link
 ! endpoints into OSPF
 !
 no network 0.0.0.0 0.0.0.0 area 1
 network 155.1.58.5 0.0.0.0 area 1
!
interface Loopback 200
 ip vrf forwarding VPN_A
 ip address 150.1.55.55 255.255.255.255
!
router bgp 100
address-family ipv4 vrf VPN_A
 network 150.1.55.55 mask 255.255.255.255
```

R6:
```
router ospf 100 vrf VPN_A
 no domain-id 0.0.0.5
 area 1 sham-link 150.1.66.66 150.1.55.55 cost 1
 !
 ! We need to make sure we're not advertising the sham-link
 ! endpoints into OSPF
 !
 no network 0.0.0.0 0.0.0.0 area 1
 network 155.1.67.6 0.0.0.0 area 1
!
interface Loopback 200
 ip vrf forwarding VPN_A
 ip address 150.1.66.66 255.255.255.255
!
```

```
router bgp 100
address-family ipv4 vrf VPN_A
 network 150.1.66.66 mask 255.255.255.255
```

SW1:

```
!
! Adjust the cost to make sure the backdoor link is used for backup
!
interface FastEthernet 0/15
 no switchport
 ip address 155.1.78.7 255.255.255.0
 ip ospf cost 9999
!
interface Vlan 67
 ip address 155.1.67.7 255.255.255.0
!
interface Loopback 101
 ip address 172.16.7.7 255.255.255.0
!
router ospf 1
 network 0.0.0.0 0.0.0.0 area 1
```

SW2:

```
interface FastEthernet 0/15
 no switchport
 ip address 155.1.78.8 255.255.255.0
 ip ospf cost 9999
```

Verification

Note

Check the sham-link status:

```
Rack1R5#show ip ospf sham-links
Sham Link OSPF_SL0 to address 150.1.66.66 is up
Area 1 source address 150.1.55.55
  Run as demand circuit
  DoNotAge LSA allowed. Cost of using 1 State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40,
    Hello due in 00:00:06
    Adjacency State FULL (Hello suppressed)
    Index 2/2, retransmission queue length 0, number of retransmission
0
    First 0x0(0)/0x0(0) Next 0x0(0)/0x0(0)
    Last retransmission scan length is 0, maximum is 0
    Last retransmission scan time is 0 msec, maximum is 0 msec
```

Note

Confirm that the endpoint addresses were learned via BGP:

```
Rack1R5# sh ip route vrf VPN_A 150.1.66.66
Routing entry for 150.1.66.66/32
  Known via "bgp 100", distance 200, metric 0, type internal
  Redistributing via ospf 100
  Advertised by ospf 100 subnets
  Last update from 150.1.6.6 00:05:56 ago
  Routing Descriptor Blocks:
  * 150.1.6.6 (Default-IP-Routing-Table), from 150.1.4.4, 00:05:56 ago
      Route metric is 0, traffic share count is 1
      AS Hops 0

Rack1R6#show ip route vrf VPN_A 150.1.55.55
Routing entry for 150.1.55.55/32
  Known via "bgp 100", distance 200, metric 0, type internal
  Redistributing via ospf 100
  Advertised by ospf 100 subnets
  Last update from 150.1.5.5 00:04:06 ago
  Routing Descriptor Blocks:
  * 150.1.5.5 (Default-IP-Routing-Table), from 150.1.4.4, 00:04:06 ago
      Route metric is 0, traffic share count is 1
      AS Hops 0
```

Note

Check that both SW1 and SW2 prefer reaching each other across the MPLS core, and all the prefixes are seen as OSPF intra-area.

```
Rack1SW1#show ip route ospf
     155.1.0.0/24 is subnetted, 7 subnets
O       155.1.8.0 [110/4] via 155.1.67.6, 00:01:47, Vlan67
O       155.1.58.0 [110/3] via 155.1.67.6, 00:01:47, Vlan67
     172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks
O       172.16.8.8/32 [110/4] via 155.1.67.6, 00:01:47, Vlan67
O E2 192.168.6.0/24 [110/1] via 155.1.67.6, 00:01:47, Vlan67
     150.1.0.0/16 is variably subnetted, 4 subnets, 2 masks
O E2    150.1.66.66/32 [110/1] via 155.1.67.6, 00:01:47, Vlan67
O E2    150.1.55.55/32 [110/1] via 155.1.67.6, 00:01:47, Vlan67
O       150.1.8.8/32 [110/4] via 155.1.67.6, 00:01:47, Vlan67
```

```
Rack1SW2#show ip route ospf
     155.1.0.0/24 is subnetted, 7 subnets
O       155.1.7.0 [110/4] via 155.1.58.5, 00:02:05, Vlan58
O       155.1.37.0 [110/4] via 155.1.58.5, 00:02:05, Vlan58
O       155.1.79.0 [110/4] via 155.1.58.5, 00:02:05, Vlan58
O       155.1.67.0 [110/3] via 155.1.58.5, 00:02:05, Vlan58
     172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks
O       172.16.7.7/32 [110/4] via 155.1.58.5, 00:02:05, Vlan58
O E2 192.168.6.0/24 [110/1] via 155.1.58.5, 00:02:05, Vlan58
     150.1.0.0/16 is variably subnetted, 4 subnets, 2 masks
O E2    150.1.66.66/32 [110/1] via 155.1.58.5, 00:02:06, Vlan58
O E2    150.1.55.55/32 [110/1] via 155.1.58.5, 00:02:06, Vlan58
O       150.1.7.7/32 [110/4] via 155.1.58.5, 00:02:06, Vlan58
```

Note

Check the end-to-end connectivity between SW1 and SW2:

```
Rack1SW1#traceroute 150.1.8.8

Type escape sequence to abort.
Tracing the route to 150.1.8.8

  1 155.1.67.6 0 msec 8 msec 0 msec
  2 155.1.146.4 50 msec 101 msec 50 msec
  3 155.1.58.5 17 msec 17 msec 17 msec
  4 155.1.58.8 25 msec *  17 msec
```

Note

Check the data-plane information for prefix 150.1.8.8 in R5 to confirm it's being switched using the MPLS VPN information:

```
Rack1R6#show bgp vpnv4 unicast vrf VPN_A 150.1.8.8
BGP routing table entry for 100:1:150.1.8.8/32, version 418
Paths: (1 available, best #1, table VPN_A, RIB-failure(17))
  Not advertised to any peer
  Local
    150.1.5.5 (metric 66) from 150.1.4.4 (150.1.4.4)
      Origin incomplete, metric 2, localpref 100, valid, internal, best
      Extended Community: RT:100:1 OSPF DOMAIN ID:0x0005:0x000000640200
        OSPF RT:0.0.0.1:2:0 OSPF ROUTER ID:172.16.5.5:0
      Originator: 150.1.5.5, Cluster list: 150.1.4.4
      mpls labels in/out nolabel/35

Rack1R6#show mpls forwarding-table 150.1.5.5
Local  Outgoing      Prefix            Bytes Label   Outgoing   Next
Hop
Label  Label or VC   or Tunnel Id      Switched      interface
16     16            150.1.5.5/32      0             Fa0/0.146
155.1.146.4
```

```
Rack1R6#show ip cef vrf VPN_A 150.1.8.8
150.1.8.8/32
  nexthop 155.1.146.4 FastEthernet0/0.146 label 16 35
```

Note

Now confirm that the backdoor link works if the primary link fails. Shutdown SW1's connection to R6 and confirm that the prefixes are available via the backup path.

SW1:

```
interface Vlan 67
 shutdown

Rack1SW1#show ip route ospf
     155.1.0.0/24 is subnetted, 7 subnets
O       155.1.8.0 [110/10000] via 155.1.78.8, 00:00:09,
FastEthernet0/15
O       155.1.58.0 [110/10000] via 155.1.78.8, 00:00:09,
FastEthernet0/15
O       155.1.67.0 [110/10002] via 155.1.78.8, 00:00:09,
FastEthernet0/15
     172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks
O       172.16.8.8/32 [110/10000] via 155.1.78.8, 00:00:09,
FastEthernet0/15
O E2 192.168.6.0/24 [110/1] via 155.1.78.8, 00:00:09, FastEthernet0/15
     150.1.0.0/16 is variably subnetted, 4 subnets, 2 masks
O E2    150.1.66.66/32 [110/1] via 155.1.78.8, 00:00:09,
FastEthernet0/15
O E2    150.1.55.55/32 [110/1] via 155.1.78.8, 00:00:09,
FastEthernet0/15
O       150.1.8.8/32 [110/10000] via 155.1.78.8, 00:00:09,
FastEthernet0/15
```

14.9 PE-CE Routing with EIGRP

- Replace OSPF with EIGRP for PE-CE routing protocol used for VPN_A.
- Ensure the backdoor link is used for backup and the primary path between the two sites is across the MPLS VPN cloud.
- Make R4's VLAN43 interface a part of VPN_A and advertise it into EIGRP.
- All EIGRP routers should be in the same AS.

Configuration

Note

EIGRP is advanced distance vector protocol that uses composite metric for best path selection. One issue with transporting EIGRP routes over MP-BGP is preserving the original metric values, the route type, the source AS# and the remote Router ID. These all are encoded using special BGP extended-community attributes that allow the remote site to properly decode the incoming routing update information. Why don't just carry the metric in MED attribute? The problem is that the remote site may happen to have different K values and have different resulting metric. Not something you would really see in real life, but still the implementation account for this. Of course, the local EIGRP process will treat all prefixes originated in different remote AS# as external, per the normal EIGRP rules. Thus the AS# and the Router-ID information could be crucial to resolve potential routing loops in scenarios with backdoor links between the VPN sites.

Another special attribute used with EIGRP prefixes redistributed into MP-BGP is know as "cost attribute". While attribute was designed to have pretty wide use, the main idea is to change the BGP best-path selection process. The problem with MPLS VPN route redistribution is that the same route may enter the PE's BGP table using redistribution – i.e. learned from a CE router and via a BGP update – i.e. learned from a remote site. Per BGP best-path selection process, locally redistributed prefixes have BGP weight of 32768 which make them always win the best-path selection process. Therefore, BGP will always choose the locally received update even if the remote site has better EIGRP metric to reach the destination.

To resolve this issue, EIGRP prefixes redistributed into MP-BGP will have the Cost attribute value set to their composite metric. BGP process will honor the Cost attribute value before ANY other best-path selection options if the attribute is present. The prefix with the lowest cost will immediately win the best-path selection and will be redistributed into local EIGRP process. This process happens automatically and does not require any additional configurations. Notice that the Cost attribute is NOT needed with OSPF because the prefixes received via MP-BGP are treated as inter-area and always less preferred compared to the same prefixes learned as intra-area from a CE device. As for the case when RIP is used for PE-CE routing, Cisco IOS does not implement Cost attribute at all, and thus the locally redistributed prefixes will always get preferred over MP-BGP learned, preventing effective RIP deployment in scenarios with backdoor links. But this is not considered a big issue as no one really uses RIP for large-scale deployments and backup routing nowadays.

When configuring EIGRP for MPLS VPNs, you share the same process among multiple VRFs. You need to configure an address family per VRF under the process and for every address-family configure an AS number using the command **`autonomous-system N`**. This command is mandatory to enable EIGRP for the particular VRF. After that, all you need is enter the normal network statements just like with the normal EIGRP configuration.

R4:
```
ip vrf VPN_A
 rd 100:1
 route-target both 100:1
!
router eigrp 100
 no auto-summary
 address-family ipv4 vrf VPN_A
  autonomous-system 100
  network 204.12.1.0 0.0.0.255
  redistribute bgp 100 metric 1 1 1 1 1
!
router bgp 100
 address-family ipv4 vrf VPN_A
  redistribute eigrp 100
!
interface FastEthernet 0/0
 ip vrf forwarding VPN_A
 ip address 204.12.1.4 255.255.255.0
```

R5:
```
no router ospf 100
!
router eigrp 100
 no auto-summary
 address-family ipv4 vrf VPN_A
  autonomous-system 100
  network 155.1.58.5 0.0.0.0
  redistribute bgp 100 metric 1 1 1 1 1
!
router bgp 100
 address-family ipv4 vrf VPN_A
  redistribute eigrp 100
```

R6:
```
no router ospf 100
!
router eigrp 100
 no auto-summary
 address-family ipv4 vrf VPN_A
  autonomous-system 100
  network 155.1.67.6 0.0.0.0
  redistribute bgp 100 metric 1 1 1 1 1
!
router bgp 100
 address-family ipv4 vrf VPN_A
  redistribute eigrp 100
```

SW1:
```
no router ospf 1
!
router eigrp 100
 no auto-summary
 network 0.0.0.0 0.0.0.0
!
! Make sure this link is backup
!
interface FastEthernet 0/15
 delay 1000
```

SW2:
```
no router ospf 1
!
router eigrp 100
 no auto-summary
 network 0.0.0.0 0.0.0.0
!
! Make sure this link is backup
!
interface FastEthernet 0/15
 delay 1000
```

Verification

Note

Check the EIGRP routes at SW1 and SW2. Notice that the primary path is chosen via the PE-routers.

```
Rack1SW1#show ip route eigrp
D    204.12.1.0/24 [90/28416] via 155.1.67.6, 00:37:23, Vlan67
     155.1.0.0/24 is subnetted, 7 subnets
D       155.1.8.0 [90/28672] via 155.1.67.6, 00:39:53, Vlan67
D       155.1.58.0 [90/28416] via 155.1.67.6, 00:39:53, Vlan67
     172.16.0.0/24 is subnetted, 2 subnets
D       172.16.8.0 [90/156416] via 155.1.67.6, 00:39:53, Vlan67
D EX 192.168.6.0/24
           [170/2560256512] via 155.1.78.8, 00:39:53, FastEthernet0/15
     150.1.0.0/16 is variably subnetted, 4 subnets, 2 masks
D EX    150.1.66.66/32 [170/2560000512] via 155.1.67.6, 00:39:54,
Vlan67
D EX    150.1.55.55/32 [170/2560000512] via 155.1.67.6, 00:39:54,
Vlan67
D       150.1.8.0/24 [90/156416] via 155.1.67.6, 00:39:54, Vlan67
Rack1SW1#
```

Note

Check the BGP prefixes for redistributed EIGRP routes in R5 and R6 and notice the extended communities used to carry the additional EIGRP information. Also, notice the Cost attribute that carries the original route’s metric.

```
Rack1R5#show bgp vpnv4 unicast vrf VPN_A 204.12.1.0
BGP routing table entry for 100:1:204.12.1.0/24, version 505
Paths: (1 available, best #1, table VPN_A)
Flag: 0x820
  Not advertised to any peer
  Local
    150.1.4.4 (metric 65) from 150.1.4.4 (150.1.4.4)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      Extended Community: RT:100:1 Cost:pre-bestpath:128:28160
0x8800:32768:0
        0x8801:100:2560 0x8802:65280:25600 0x8803:65281:1500
      mpls labels in/out nolabel/25
```

Note

Take another prefix, which is reachable both via the backdoor link and MPLS VPN cloud. Notice that BGP has only the path across the MPLS VPN could – this is because the CE router has accepted this path as well and suppressed advertising the backup path to the PE router. This is all thanks to BGP cost attribute.

```
Rack1R5#show bgp vpnv4 unicast vrf VPN_A 150.1.8.0
BGP routing table entry for 100:1:150.1.8.0/24, version 446
Paths: (1 available, best #1, table VPN_A)
  Advertised to update-groups:
        1
  Local
    155.1.58.8 from 0.0.0.0 (150.1.5.5)
      Origin incomplete, metric 156160, localpref 100, weight 32768,
valid, sourced, best
      Extended Community: RT:100:1 Cost:pre-bestpath:128:156160
        0x8800:32768:0 0x8801:100:130560 0x8802:65281:25600
0x8803:65281:1500
      mpls labels in/out 25/nolabel
```

Note

Now verify end-to-end connectivity and confirm that the backup path (backdoor link) works as well.

```
Rack1SW1#traceroute 150.1.8.8

Type escape sequence to abort.
Tracing the route to 150.1.8.8

  1 155.1.67.6 0 msec 0 msec 0 msec
  2 155.1.146.4 101 msec 50 msec 101 msec
  3 155.1.58.5 17 msec 25 msec 17 msec
  4 155.1.58.8 25 msec *  17 msec

Rack1SW1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1SW1(config)#interface vlan 67
Rack1SW1(config-if)#shutdown
Rack1SW1(config-if)#^Z

Rack1SW1#sh ip route 150.1.8.0
Routing entry for 150.1.8.0/24
  Known via "eigrp 100", distance 90, metric 409600, type internal
  Redistributing via eigrp 100
  Last update from 155.1.78.8 on FastEthernet0/15, 00:00:05 ago
  Routing Descriptor Blocks:
  * 155.1.78.8, from 155.1.78.8, 00:00:05 ago, via FastEthernet0/15
      Route metric is 409600, traffic share count is 1
      Total delay is 15000 microseconds, minimum bandwidth is 100000
Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 1
```

14.10 EIGRP Site-of-Origin

- Configure R5 and R6 to prevent temporary routing loops that may occur due to mutual redistribution between EIGRP and MP-BGP.
- Ensure the path across the MPLS VPN core is used as the primary between SW1 and SW2.

Configuration

✐ **Note**

With multihomed scenarios similar to the one we are using, BGP and EIGRP perform mutual redistribution at PE routers. This may potentially result in transient routing loops, when a prefix withdrawn in MP-BGP at one PE router is not timely propagated into EIGRP and EIGRP feed the invalid information back to BGP at another PE router, keeping the false information circulating between the PE routers until it's eliminated by counting to infinity. The core of the problem is the mutual redistribution that allows the information learned from BGP at one site re-enter BGP at another site. This problem is commonly resolved by tagging the prefixes from one protocol while redistributing them into another, and then blocking the prefixes from re-entering the domain of origin based on the tag.

However, implementing the solution using EIGRP route tags and BGP communities could be cumbersome if done using the regular methods. Therefore, Cisco IOS implements a special feature known as EIGRP Site-of-Origin which uses an extended community appended to BGP and EIGRP routing updates (EIGRP's TLV format allows adding this information easily). The feature is configured at interface level using the command **`ip vrf sitemap <ROUTE_MAP>`** where ROUTE_MAP is a regular route-map that applies the command **`set extcommunity soo ASN:XX`** where ASN:XX is the "VPN identifier" common for all PEs of the same multi-homed site.

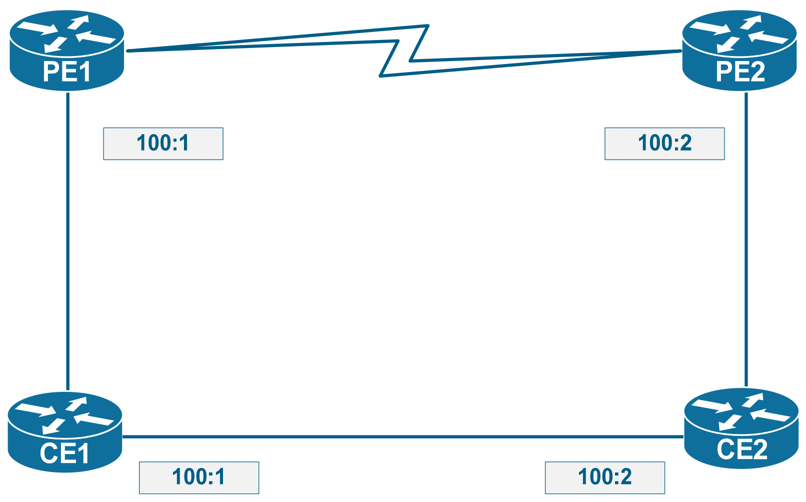
All BGP prefixes redistributed into EIGRP and sent over the interface with SoO set will have this extended community appended, but only if the community is not already present. If the update to be sent already has the same extended community value set, it is discarded as being redistributed back to the same site. The next thing that this feature does is applies the same extended community to all EIGRP routes received on the interface and redistributed into BGP. All these actions are performed by IOS automatically – you only need to apply the route-map to the relevant interfaces.

There are two common ways of applying VRF sitemap feature.

1) The SoO is applied on the PE interfaces facing the CE routers. Every PE

router uses the same SoO value. This prevents PE routers from learning MP-BGP originated routes from CE routers. However, the side-effect is that MPLS core could not be used as a backup path between the segments of the multi-homed sites in case if the backdoor link fails. This is because MP-BGP updates for in-site EIGRP prefixes carry the same SoO that is applied to the PE interfaces facing the CE routers and thus cannot be redistributed back to the same site.

2) If you need to preserve the path across the MPLS core network, you should use different SoO values at every PE router of multi-homed site. However, this means the MP-BGP information injected into EIGRP at one PE router will reach the other PEs without being blocked. To prevent this effect, additional SoO interface should be configured on the CE routers with the backdoor link. Look at the figure below and notice the SoO settings on the respective PE/CE interfaces:



In this case, if PE1 redistributes a prefix into MP-BGP it will be tagged with 100:1. The prefix will reach PE1 and pass down to CE2. The prefix will finally be stopped at CE1 by SoO filtering and will never loop back to PE1 again. The same applies to prefixes redistributed into MP-BGP by PE2.

R5:
```
route-map EIGRP_SOO
 set extcommunity soo 100:15
!
interface FastEthernet 0/0
 ip vrf sitemap EIGRP_SOO
```

R6:
```
route-map EIGRP_SOO
 set extcommunity soo 100:16
!
interface FastEthernet 0/0.67
 ip vrf sitemap EIGRP_SOO
```

SW1:
```
route-map EIGRP_SOO
 set extcommunity soo 100:16
!
interface FastEthernet 0/15
 ip vrf sitemap EIGRP_SOO
```

SW2:
```
route-map EIGRP_SOO
 set extcommunity soo 100:15
!
interface FastEthernet 0/15
 ip vrf sitemap EIGRP_SOO
```

Verification

Note

Check the BGP prefixes for EIGRP routes and notice the SoO community values:

```
Rack1R6#show bgp vpnv4 unicast vrf VPN_A 150.1.8.0
BGP routing table entry for 100:1:150.1.8.0/24, version 680
Paths: (1 available, best #1, table VPN_A)
  Not advertised to any peer
  Local
    150.1.5.5 (metric 66) from 150.1.4.4 (150.1.4.4)
      Origin incomplete, metric 156160, localpref 100, valid, internal,
best
      Extended Community: SoO:100:15 RT:100:1 Cost:pre-
bestpath:128:156160
        0x8800:32768:0 0x8801:100:130560 0x8802:65281:25600
0x8803:65281:1500
      Originator: 150.1.5.5, Cluster list: 150.1.4.4
      mpls labels in/out nolabel/25

Rack1R6#show bgp vpnv4 unicast vrf VPN_A 150.1.7.0
BGP routing table entry for 100:1:150.1.7.0/24, version 700
Paths: (1 available, best #1, table VPN_A)
  Advertised to update-groups:
        1
  Local
    155.1.67.7 from 0.0.0.0 (150.1.6.6)
      Origin incomplete, metric 156160, localpref 100, weight 32768,
valid, sourced, best
      Extended Community: SoO:100:16 RT:100:1 Cost:pre-
bestpath:128:156160
        0x8800:32768:0 0x8801:100:130560 0x8802:65281:25600
0x8803:65281:1500
      mpls labels in/out 18/nolabel
```

Note

Check the routing tables in the CE routers and confirm that primary paths are via the MPLS core:

```
Rack1SW1#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS
level-2
       ia - IS-IS inter area, * - candidate default, U - per-user
static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

D    204.12.1.0/24 [90/28416] via 155.1.67.6, 00:33:53, Vlan67
     155.1.0.0/24 is subnetted, 7 subnets
D       155.1.8.0 [90/28672] via 155.1.67.6, 00:33:53, Vlan67
C       155.1.7.0 is directly connected, Vlan7
D       155.1.58.0 [90/28416] via 155.1.67.6, 00:33:53, Vlan67
C       155.1.37.0 is directly connected, FastEthernet0/3
C       155.1.78.0 is directly connected, FastEthernet0/15
C       155.1.79.0 is directly connected, Vlan79
C       155.1.67.0 is directly connected, Vlan67
     172.16.0.0/24 is subnetted, 2 subnets
D       172.16.8.0 [90/156416] via 155.1.67.6, 00:33:54, Vlan67
C       172.16.7.0 is directly connected, Loopback101
D EX 192.168.6.0/24
           [170/2560256512] via 155.1.78.8, 00:33:54, FastEthernet0/15
     150.1.0.0/16 is variably subnetted, 4 subnets, 2 masks
C       150.1.7.0/24 is directly connected, Loopback0
D EX    150.1.66.66/32 [170/2560000512] via 155.1.67.6, 00:33:55,
Vlan67
D EX    150.1.55.55/32 [170/2560000512] via 155.1.67.6, 00:33:55,
Vlan67
D       150.1.8.0/24 [90/156416] via 155.1.67.6, 00:33:55, Vlan67

Rack1SW2#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS
level-2
       ia - IS-IS inter area, * - candidate default, U - per-user
static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set
```

```
D    204.12.1.0/24 [90/28416] via 155.1.58.5, 00:35:33, Vlan58
     155.1.0.0/24 is subnetted, 7 subnets
C       155.1.8.0 is directly connected, Vlan8
D       155.1.7.0 [90/28672] via 155.1.58.5, 00:35:33, Vlan58
C       155.1.58.0 is directly connected, Vlan58
D       155.1.37.0 [90/30976] via 155.1.58.5, 00:35:33, Vlan58
C       155.1.78.0 is directly connected, FastEthernet0/15
D       155.1.79.0 [90/28672] via 155.1.58.5, 00:35:33, Vlan58
D       155.1.67.0 [90/28416] via 155.1.58.5, 00:35:33, Vlan58
     172.16.0.0/24 is subnetted, 2 subnets
C       172.16.8.0 is directly connected, Loopback100
D       172.16.7.0 [90/156416] via 155.1.58.5, 00:35:34, Vlan58
D EX 192.168.6.0/24 [170/2560000512] via 155.1.58.5, 01:30:38, Vlan58
     150.1.0.0/16 is variably subnetted, 4 subnets, 2 masks
D       150.1.7.0/24 [90/156416] via 155.1.58.5, 00:35:34, Vlan58
D EX    150.1.66.66/32 [170/2560000512] via 155.1.58.5, 00:35:34,
Vlan58
D EX    150.1.55.55/32 [170/2560000512] via 155.1.58.5, 00:35:34,
Vlan58
C       150.1.8.0/24 is directly connected, Loopback0
```

Note

Shutdown the PE-CE link at SW1 and confirm that the backup link is still usable:

```
Rack1SW1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1SW1(config)#interface vlan 67
Rack1SW1(config-if)#shutdown

Rack1SW1#show ip route eigrp
D    204.12.1.0/24 [90/284416] via 155.1.78.8, 00:00:03,
FastEthernet0/15
     155.1.0.0/24 is subnetted, 6 subnets
D       155.1.8.0 [90/281856] via 155.1.78.8, 00:00:03,
FastEthernet0/15
D       155.1.58.0 [90/281856] via 155.1.78.8, 00:00:03,
FastEthernet0/15
     172.16.0.0/24 is subnetted, 2 subnets
D       172.16.8.0 [90/409600] via 155.1.78.8, 00:00:03,
FastEthernet0/15
D EX 192.168.6.0/24
           [170/2560256512] via 155.1.78.8, 00:36:26, FastEthernet0/15
     150.1.0.0/16 is variably subnetted, 4 subnets, 2 masks
D EX    150.1.66.66/32
           [170/2560256512] via 155.1.78.8, 00:00:03, FastEthernet0/15
D EX    150.1.55.55/32
           [170/2560256512] via 155.1.78.8, 00:00:03, FastEthernet0/15
D       150.1.8.0/24 [90/409600] via 155.1.78.8, 00:00:04,
FastEthernet0/15
```

14.11 PE-CE Routing with BGP

- Erase all EIGRP configurations for VPN_A in R5, R6, SW1 and SW2.
- Configure BGP AS 78 in SW1 and SW2 and peer the CE routers with R5 and R6.
- Advertise Loopback0 interface of SW1 and SW2 into BGP and ensure end-to-end reachability.

Configuration

Note

BGP seems to be a natural fit for PE-CE routing due to its perfect integration with core MP-BGP routing. Advantages – perfect scalability and precise route control. Disadvantage – slow convergence for intra-site routing. However, as a balanced option you may use IGP for intra-site routing and BGP for PE-CE peering to get the best of both worlds.

One common problem that arises from the use of BGP is that multiple sites may re-use the same AS number. Based on BGP loop-prevention mechanism, this will filter BGP updates sent between the sites. There are two solutions to this problem: the first one is configuring the **`allowas-in`** option inbound for CE peering session. The other option is configuring the **`as-override`** option on PE routers for eBGP peering sessions with CE routers. What this option does, is compares the remote-AS number with the AS number stored in the end of AS_PATH attribute. If they match, the AS number in AS_PATH is replaced with the local PE router’s AS number. The AS numbers used in BGP peering sessions remain the same, only AS_PATH attribute is changed when updates are relayed. Notice that the length of AS_PATH attribute remains the same, which allows for proper best-path selection.

Configuring BGP for PE-CE routing requires only activating the respective VRF’s address family under the global BGP process and configuring BGP peering sessions under this VRF. There is no need to configure any redistribution in this case, as routes are propagated into VPNv4 table automatically. Just as usual, prefix import and export is controlled by route-target extended communities. If you want to, you may additionally redistribute other IGP protocols into BGP, or redistribute connected interfaces.

R5:
```
no router eigrp 100
!
router bgp 100
 address-family ipv4 vrf VPN_A
 neighbor 155.1.58.8 remote-as 78
 neighbor 155.1.58.8 as-override
```

R6:
```
no router eigrp 100
!
router bgp 100
 address-family ipv4 vrf VPN_A
 neighbor 155.1.67.7 remote-as 78
 neighbor 155.1.67.7 as-override
```

SW1:
```
no router eigrp 100
!
 router bgp 78
 neighbor 155.1.67.6 remote-as 100
 network 150.1.7.0 mask 255.255.255.0
```

SW2:
```
no router eigrp 100
!
router bgp 78
 neighbor 155.1.58.5 remote-as 100
 network 150.1.8.0 mask 255.255.255.0
```

Verification

Note

Check the BGP routes learned at SW1. Look into the BGP table and confirm that the “source” AS 58 no longer appears in the AS_PATH and is replaced by the “core” AS 100. After this, verify end-to-end connectivity across the MPLS core.

```
Rack1SW1#show ip route bgp
B    204.12.1.0/24 [20/0] via 155.1.67.6, 00:00:40
     155.1.0.0/24 is subnetted, 6 subnets
B       155.1.58.0 [20/0] via 155.1.67.6, 00:00:40
     150.1.0.0/16 is variably subnetted, 4 subnets, 2 masks
B       150.1.66.66/32 [20/0] via 155.1.67.6, 00:00:40
B       150.1.55.55/32 [20/0] via 155.1.67.6, 00:00:40
B       150.1.8.0/24 [20/0] via 155.1.67.6, 00:00:40

Rack1SW1#show ip bgp
BGP table version is 9, local router ID is 172.16.7.7
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 150.1.7.0/24     0.0.0.0                  0         32768 i
*> 150.1.8.0/24     155.1.67.6                             0 100 100 i
*> 150.1.55.55/32   155.1.67.6                             0 100 i
*> 150.1.66.66/32   155.1.67.6               0             0 100 i
*> 155.1.58.0/24    155.1.67.6                             0 100 ?
r> 155.1.67.0/24    155.1.67.6               0             0 100 ?
*> 204.12.1.0       155.1.67.6                             0 100 ?

Rack1SW1#traceroute
Protocol [ip]:
Target IP address: 150.1.8.8
Source address: 150.1.7.7
Numeric display [n]:
Timeout in seconds [3]:
Probe count [3]:
Minimum Time to Live [1]:
Maximum Time to Live [30]:
Port Number [33434]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Type escape sequence to abort.
Tracing the route to 150.1.8.8

  1 155.1.67.6 [AS 100] 0 msec 9 msec 0 msec
  2 155.1.146.4 50 msec 101 msec 50 msec
  3 155.1.58.5 [AS 100] 42 msec 51 msec 50 msec
  4 155.1.58.8 [AS 100] 25 msec *  17 msec
```

14.12 BGP SoO Attribute

- Configure “backdoor” BGP peering session between SW1 and SW2 using the inter-switch link configured previously.
- Ensure this configuration does not result in routing loops due to BGP loop-prevention mechanism being disabled with AS-Override feature.

Configuration

Note

The use of AS-Override feature allows circumventing the BGP loop prevention mechanism. However, the drawback is that this may result in routing-loops for multi-homed sites with backdoor links. If a site injects a prefix into MP-BGP the prefix may re-enter the same site via another PE as AS-Override will replace the source AS.

In order to overcome this issue for multi-homed sites, BGP implements SoO attribute similar to the one used for EIGRP. However, this time you configure the SoO value per-neighbor peering session in PE routers. There are two basic methods of doing this:

1) Setting the SoO attribute for inomcing/outgoing prefixes on the peering session using the command **`neighbor <IP> soo <VALUE>`**. This command is available since IOS 12.4(11)T and is more elegant way of configuring the feature.

2) Setting the SoO attribute using a route-map command **`set extcommunity soo <VALUE>`** and applying this route map inbound to the neighbor session via the command **`neighbor <IP> route-map <NAME> in`**.

BGP SoO works similarly to the EIGRP SoO: if an incoming or outgoing update has the SoO value matching the locally configured one, the update is dropped. Otherwise, the update is tagged with the SoO extended community. Based on this similarity, you may quickly find out that using the same SoO at all PEs will prevent using the MPLS VPN core as a backup path between the partitions in case if the backdoor link fails. The solution is, again, using different SoO values at PE routers and applying SoO at the CE routers connected to the backdoor link. However, in this scenario we simply configure the SoO on the PE routers.

R5:
```
router bgp 100
 address-family ipv4 vrf VPN_A
 neighbor 155.1.58.8 soo 100:1
```

R6:
```
router bgp 100
 address-family ipv4 vrf VPN_A
 neighbor 155.1.67.7 soo 100:1
```

SW1:
```
router bgp 78
 neighbor 155.1.78.8 remote-as 78
```

SW2:
```
router bgp 78
 neighbor 155.1.78.7 remote-as 78
```

Verification

Note

Look into BGP tables of PE routers and confirm that BGP prefixes learned from PEs are now tagged with SoO attribute. Start with R5 and the prefix 150.1.8.0/24. Confirm that this prefix is not advertised to the CE router (SW1):

```
Rack1R6#show ip bgp vpnv4 vrf VPN_A 150.1.8.0
BGP routing table entry for 100:1:150.1.8.0/24, version 25
Paths: (1 available, best #1, table VPN_A)
Flag: 0x820
  Advertised to update-groups:
        1
  78
    155.1.67.7 from 155.1.67.7 (172.16.7.7)
      Origin IGP, localpref 100, valid, external, best
      Extended Community: SoO:100:1 RT:100:1
      mpls labels in/out 35/nolabel

Rack1R6#sh ip bgp vpnv4 vrf VPN_A neighbors 155.1.67.7 advertised-
routes
BGP table version is 23, local router ID is 150.1.6.6
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN_A)
*>i150.1.55.55/32   150.1.5.5                0    100      0 i
*> 150.1.66.66/32   0.0.0.0                  0         32768 i
*>i155.1.58.0/24    150.1.5.5                0    100      0 ?
*> 155.1.67.0/24    0.0.0.0                  0         32768 ?
*>i204.12.1.0       150.1.4.4                0    100      0 ?

Total number of prefixes 5
```

Note

R5 has two prefixes in the BGP table for 150.1.7.0: one learned from R6 via R4 (the RR) and another learned from SW2. Both are tagged with the SoO of 100:1. R5 does not advertise the prefix 150.1.7.0/24 to SW2 per the rules of SoO filtering.

```
Rack1R5#show ip bgp vpnv4 vrf VPN_A 150.1.7.0
BGP routing table entry for 100:1:150.1.7.0/24, version 2
Paths: (2 available, best #2, table VPN_A)
Flag: 0x820
```

```
  Advertised to update-groups:
        1
  78
    150.1.6.6 (metric 66) from 150.1.4.4 (150.1.4.4)
      Origin IGP, metric 0, localpref 100, valid, internal
      Extended Community: SoO:100:1 RT:100:1
      Originator: 150.1.6.6, Cluster list: 150.1.4.4
      mpls labels in/out 42/48
  78
    155.1.58.8 from 155.1.58.8 (172.16.8.8)
      Origin IGP, localpref 100, valid, external, best
      Extended Community: SoO:100:1 RT:100:1
      mpls labels in/out 42/nolabel

Rack1R5#show ip bgp vpnv4 vrf VPN_A neighbors 155.1.58.8 advertised-
routes
BGP table version is 25, local router ID is 150.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN_A)
*> 150.1.55.55/32   0.0.0.0                  0         32768 i
*>i150.1.66.66/32   150.1.6.6                0    100      0 i
*> 155.1.58.0/24    0.0.0.0                  0         32768 ?
*>i155.1.67.0/24    150.1.6.6                0    100      0 ?
*> 172.16.5.0/24    0.0.0.0                  0         32768 ?
*>i192.168.6.0      150.1.6.6                0    100      0 ?
*>i204.12.1.0       150.1.4.4                0    100      0 ?

Total number of prefixes 7
```

14.13 Internet Access

- Enable RIP on R6's connection to BB1 and configure R6 so that VPN_A customers may access these routes.
- You are allowed to use one static route to accomplish this task.
- Only the 150.X.0.0/16 subnets should be allowed to access the Internet.

Configuration

Note

Internet access is a common service, which often needs to be combined with MPLS VPN services provided by an SP. In many cases, the solution is simply providing an addition PE-CE link that belongs to the global routing table in PE, or the special VRF containing Internet routes. In some situations, however, it is desirable to use a single point of attachment for both type of service. In this case, two problems need to be addressed:

1) Injecting the Internet routes into the respective VRF. In many situations, it is enough to inject just the default route. If the Internet access is provided via the global routing table, special type of static routes should be used. If the Internet access is provided via a special VPN, classic route export/import could be used.

2) The Internet routing table should learn about the prefixes found in the VPN that needs VPN access. In many situations, VPNs use private IP addressing, and thus a kind of NAT mechanism is required to hide the source IP addresses and translate them into routable equivalents.

In this scenario, we are going to deal with the most common case, when Internet access is provided via the global routing table. We assume the VPNs using private IP addressing, which requires NAT to be configured when access the Internet. In our case, "Internet" is emulated by the prefixes learned from BB1. Here is the list of the steps required to configure NAT-based Internet access via the global routing table:

1) Create a special default VRF route that resolves via the global routing table. The syntax is **`ip route vrf <NAME> 0.0.0.0 0.0.0.0 <NEXT_HOP> global`**. The last keyword means that even though the route is bound to a VRF, the next hop should be resolved using the global routing table, and FIB should be programmed accordingly. You may need to redistribute this route into MP-BGP to propagate it to all VPN sites.

2) Enable NAT on the Internet and VPN links. You need to set up all VPN-facing links as NAT inside and the Internet facing link as NAT outside. After this, create

a global NAT address pool if needed. This address pool should be reachable via the global routing table.

3) The last step is the core of NAT-based VPN Internet access. You need to configure a source NAT translation rule that matches the VPN source IP addresses and specifies either the global pool or the global interface for address translation. This rule should end with the keyword **`vrf <VRF_NAME>`** which selects the source IP addresses only from the particular VRF. The ability to apply the NAT rules to IP addresses found in particular VRF is a new feature of NAT translation engine.

After these three steps have been configured, the VPN clients will use the injected default route to access all non-matched prefixes. The packets will be routed to the Internet-access router which has the NAT rule set up. After this, the source IP addresses will be changed to the addresses routable via the global table, and respective NAT entries will be created. Even though the packets originally belong to the VRF, they will be switches using the global routing table and the NAT entries will serve as “entry points” back to the VRF for the returning packets.

Notice that our solution uses BGP, and thus we cannot simply redistribute the static default route. We use the BGP address-family command **`default-information originate`** to accomplish this task. This command requires the default route to be advertised into local BGP table by using either redistribution or the **`network`** command. Only after this the default route will be propagated to all peers.

R6:

```
router rip
 version 2
 no auto-summary
 network 54.0.0.0
!
ip route vrf VPN_A 0.0.0.0 0.0.0.0 54.1.1.254 global
!
router bgp 100
!
! Let all sites know about the default route.
!
 address-family ipv4 vrf VPN_A
 default-information originate
 redistribute static
!
interface Serial 0/0/0
 ip nat outside
!
interface FastEthernet 0/0.146
 ip nat inside
!
interface FastEthernet 0/0.67
 ip nat inside
!
ip access-list standard VPN_PREFIXES
 permit 150.1.0.0 0.0.255.255
!
ip nat inside source list VPN_PREFIXES interface Serial 0/0/0 vrf VPN_A
overload
```

Verification

Note

Check the routing table for any of the CEs for the default route:

```
Rack1SW1#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS
level-2
       ia - IS-IS inter area, * - candidate default, U - per-user
static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is 155.1.67.6 to network 0.0.0.0

B    204.12.1.0/24 [20/0] via 155.1.67.6, 00:00:15
     155.1.0.0/24 is subnetted, 6 subnets
C       155.1.7.0 is directly connected, Vlan7
B       155.1.58.0 [20/0] via 155.1.67.6, 00:00:15
C       155.1.37.0 is directly connected, FastEthernet0/3
C       155.1.78.0 is directly connected, FastEthernet0/15
C       155.1.79.0 is directly connected, Vlan79
C       155.1.67.0 is directly connected, Vlan67
     172.16.0.0/24 is subnetted, 2 subnets
B       172.16.5.0 [200/0] via 155.1.58.5, 00:00:27
C       172.16.7.0 is directly connected, Loopback101
B    192.168.6.0/24 [200/0] via 155.1.58.5, 00:00:05
B    192.168.7.0/24 [20/0] via 155.1.67.6, 00:00:28
     150.1.0.0/16 is variably subnetted, 4 subnets, 2 masks
C       150.1.7.0/24 is directly connected, Loopback0
B       150.1.66.66/32 [20/0] via 155.1.67.6, 00:00:29
B       150.1.55.55/32 [20/0] via 155.1.67.6, 00:00:16
B       150.1.8.0/24 [200/0] via 155.1.78.8, 01:41:57
B*   0.0.0.0/0 [20/0] via 155.1.67.6, 00:00:29
```

Note

Test connectivity to any of the “Internet” routes, learned by R6 via RIP:

```
Rack1R6#show ip route rip
R    212.18.1.0/24 [120/1] via 54.1.1.254, 00:00:11, Serial0/0/0
R    212.18.0.0/24 [120/1] via 54.1.1.254, 00:00:11, Serial0/0/0
R    212.18.3.0/24 [120/1] via 54.1.1.254, 00:00:11, Serial0/0/0
R    212.18.2.0/24 [120/1] via 54.1.1.254, 00:00:11, Serial0/0/0

Rack1SW1#ping 212.18.1.1 source loopback0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 212.18.1.1, timeout is 2 seconds:
Packet sent with a source address of 150.1.7.7
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 25/30/34 ms
Rack1SW1#

Rack1R6#show ip nat translations
Pro Inside global      Inside local       Outside local      Outside
global
icmp 54.1.1.6:3        150.1.7.7:3        212.18.1.1:3
212.18.1.1:3

Rack1SW2#ping 212.18.1.1 source loopback 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 212.18.1.1, timeout is 2 seconds:
Packet sent with a source address of 150.1.8.8
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 25/32/34 ms
Rack1SW2#

Rack1R6#show ip nat translations
Pro Inside global      Inside local       Outside local      Outside
global
icmp 54.1.1.6:0        150.1.8.8:0        212.18.1.1:0
212.18.1.1:0
```

14.14 AToM

- Provide P2P L2VPN connection between VLAN5 interface of R5 and the unused interface of R6.
- Use the method that provided minimum transport overhead.

Configuration

Note

Point-to-Point L2 VPNs provide transparent L2 connectivity to the attached circuits at two separate sites connected via a packet network. The general idea of L2 VPNs is providing transparent connectivity without the need for provisioning actual L2 connection across the SP core. There are two common solutions for P2P L2VPNs: Any Transport over MPLS or AToM and Layer 2 Tunneling Protocol Version 3 or L2TPv3. Both technologies work as simple P2P tunnel know as “pseudowire”, capturing L2 frames at one side and transporting them across the packet network to the other side of the connection. Some of the information of the original frame might be dropped at ingress and recovered at egress, depending on the technology.

In this task we will be concerned with AToM, which provides less encapsulation overhead than L2TPv3 as it uses MPLS encapsulation for tunneling. Also, we will be dealing with Ethernet as the only underlying technology for L2VPN connectivity. First of all, AToM transport is configured manually, using the interface-level command **`xconnect`**. This command takes additional parameters, including pseudowire class, destination IP address and the VC (virtual circuit) ID. The pseudowire class is defined globally using the command **`pseudowire-class <NAME>`** and specifies additional L2VPN parameters. As soon as AToM learns the destination IP address and the VC ID, it attempts to establish a targeted LDP session with the remote end. The source IP address used for this session is based on the configured MPLS LDP router identifier, i.e. per the normal LDP rules. If the remote end is configured properly, the session will become active and the parties will attempt to exchange the labels identifying their attached circuits (ACs). For the negotiations to succeed, the following need to match:

1) VC identifiers at every end must match, so that the connection is uniquely addressed. It’s up to you to provision an optimal numbering scheme
2) VC type: either port-mode or VLAN-mode for Ethernet L2VPNs. Port-Mode connection is configured by applying the **`xconnect`** command to the physical interface level. It will replicate all Ethernet frames received, including all tagged frames. In VLAN-based mode you configure the **`xconnect`** statement under a subinterface with the proper encapsulation and VLAN number specified. This mode will only intercept and forward the respective VLAN frames across the

pseudowire.
3) MTU settings for the interfaces being connected. Notice that MTU mismatch often happens in situations when you connect Gigabit and Fast Ethernet ports together.
4) Authentication setting must match. Since AToM uses LDP for label exchange, you may find that your LDP password policy may require you to authenticate the new session as well.

If all parameters match, the label exchange will succeed, and pseudowire will become active. The packets intercepted at one end of the connected, are prepended with two MPLS labels: the topmost label is the transport label for the other end’s /32 IP address and the second label identifies the remote AC. AToM additionally prepends special “Control Word” that carries extra information, which is not relevant in the context of our discussion. The resulting MPLS packet is label switched to the remote end, where the AC label is used to select the proper outgoing circuit and switch the packet.

Notice that if you are using AToM in VLAN-based mode, you might or might not have to use the same VLAN id at both ends. This depends on the platform hardware/software limitation to do VLAN rewrite; Most ISR IOS-based platforms can do that, though.

```
R5:
default interface FastEthernet 0/1
interface FastEthernet 0/1
 xconnect 150.1.6.6 100 encapsulation mpls
!
! We need the password per our LDP password policy
!
mpls ldp neighbor 150.1.6.6 password CISCO
```

```
R6:
interface FastEthernet 0/1
 no shutdown
 xconnect 150.1.5.5 100 encapsulation mpls
!
mpls ldp neighbor 150.1.5.5 password CISCO
```

Verification

Note

Issue the following command to check the status of LDP session and L2 transport connection. Notice the labels used for AC identification and the MTU setting. Additionally, the command displays the label stack used to transport the actual L2 frames.

```
Rack1R6#show mpls l2transport vc detail
Local interface: Fa0/1 up, line protocol up, Ethernet up
  Destination address: 150.1.5.5, VC ID: 100, VC status: up
    Output interface: Fa0/0.146, imposed label stack {16 40}
    Preferred path: not configured
    Default path: active
    Next hop: 155.1.146.4
  Create time: 00:00:22, last status change time: 00:00:15
  Signaling protocol: LDP, peer 150.1.5.5:0 up
    MPLS VC labels: local 27, remote 40
    Group ID: local 0, remote 0
    MTU: local 1500, remote 1500
    Remote interface description:
  Sequencing: receive disabled, send disabled
  VC statistics:
    packet totals: receive 896, send 28
    byte totals:   receive 53918, send 2520
    packet drops:  receive 0, seq error 0, send 0
```

Note

Now configure SW3 and SW4 as the CE devices for L2 VPN. Assign them any IP addresses on the same subnet and confirm that you can ping across:

```
SW4:
interface FastEthernet 0/6
 no switchport
 ip address 56.56.56.10 255.255.255.0

SW3:
interface FastEthernet 0/5
 no switchport
 ip address 56.56.56.9 255.255.255.0

Rack1SW3#ping 56.56.56.10

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 56.56.56.10, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 33/35/42 ms
```

14.15 L2TPV3

- Modify the solution for the previous task to use L2TPv3 instead of MPLS as the underlying transport.
- Ensure the packets are never fragmented in the core and automatic MTU detection is in progress.

Configuration

Note

L2TPv3 is an alternative to MPLS for dynamic tunneling. It is based on the L2TPv2 protocol, which was used for PPP tunneling in dial VPDNs. L2TPv3 does not require MPLS layer deployed in the network, as it uses normal IP packets (either IP protocol 115 or UDP packets) to tunnel the payload. This allows for using L2TPv3 in the networks that lack MPLS support, or in scenario that require inter-provider operations, where one of the operators does not implement MPLS.

The deployment simplicity comes at price, of course. First, L2TPv3 has significantly larger overhead than MPLS-based tunneling. This might raise concerns related to the maximum MTU settings of the underlying transport network. Secondly, L2TPv3 does not allow for effective traffic engineering like MPLS does, as MPLS implements locally significant tagging whereas L2TPv3 uses globally significant “label” – encapsulating IP packet.

Configuring L2TPv3 pseudowire is pretty much similar to setting up an AToM pseudowire, it just has more options. First of all, it is mandatory to specify a pseudowire class with the xconnect command that uses L2TPv3 encapsulation. This is due to the fact that you need to specify the source IP address used for L2TPv3 session via the pw-command **`ip local interface <IFNAME>`**. The other important options for L2TPv3 include forcefully setting DFbit using pw-class command **`ip dfbit set`**, which avoids in-core fragmentation and performance degradation. By default, DF-bit is not set for tunneled packets. If you want to copy the TOS byte from encapsulate packets (if they are IP) use the command **`ip tos reflect`** or the command **`ip tos <VALUE>`** to manually set this TOS byte. One other feature is the packet sequencing, which allows dropping out of order packets but adds additional overhead. It is enabled using the command **`sequencing`**, and usually needed only with the applications that are sensitive to packet reordering, such as SANs.

Normally, when using L2TPv3 you may want to increase the core network MTU by the value that is big enough to accommodate the client network MTU plus L2TPv3 overhead. In the situations when you cannot do that, you may have to implement some solutions at client side to avoid PMTU problems, such as using the IOS command **`ip tcp adjust-mss`**. One L2TPv3 command that may particularly help you with customer PMTU discovery is **`ip local pmtu`** which allows the encapsulating PE to forward back to the customer ICMP unreachable messages thus informing the IP-capable end-devices of fragmentation issues. This command should always be used in conjunction with **`ip dfbit set`** to ensure all packets are subject to drop if they cannot be fragmented.

Similar to AToM, L2TPv3 support port-mode and VLAN-mode VC types, but unlike the MPLS-based technology, it does not require the MTU value to match in order to bring up the pseudowire.

R5:

```
pseudowire-class L2TPV3
 encapsulation l2tpv3
 ip local interface Loopback0
 ip pmtu
 ip dfbit set
 ip tos reflect
!
default interface FastEthernet 0/1
interface FastEthernet 0/1
 xconnect 150.1.6.6 100 encapsulation l2tpv3 pw-class L2TPV3
```

R6:

```
pseudowire-class L2TPV3
 encapsulation l2tpv3
 ip local interface Loopback0
 ip pmtu
 ip dfbit set
 ip tos reflect
!
default interface FastEthernet 0/1
interface FastEthernet 0/1
 xconnect 150.1.5.5 100 encapsulation l2tpv3 pw-class L2TPV3
```

Verification

Note

Use the show command below on any of the PE routers. Pay special attention to the Session UP and Circuit UP messages which signal the healthy P2P connection. Typical problems with connection establishment include unreachable endpoint IP addresses, shutdown interface, mismatching parameters or mismatching VC-types.

Notice the FW header information which is the dump of the encapsulating L2TPv3 header. You may use this information to adjust the core network MTU sizes properly.

```
Rack1R5#show l2tp session all

L2TP Session Information Total tunnels 1 sessions 1

Session id 19547 is up, tunnel id 38503
  Remote session id is 55660, remote tunnel id 34507
  Locally initiated session
Call serial number is 1694600001
Remote tunnel name is Rack1R6
  Internet address is 150.1.6.6
Local tunnel name is Rack1R5
  Internet address is 150.1.5.5
IP protocol 115
  Session is L2TP signaled
  Session state is established, time since change 00:00:35
    4 Packets sent, 6 received
    562 Bytes sent, 923 received
  Last clearing of counters never
  Counters, ignoring last clear:
    4 Packets sent, 6 received
    562 Bytes sent, 923 received
    Receive packets dropped:
      out-of-order:             0
      total:                    0
    Send packets dropped:
      exceeded session MTU:     0
      total:                    0
  DF bit on, ToS reflect enabled, ToS value 0, TTL value 255
  Sending UDP checksums are disabled
  Receiving UDP checksums are verified
  Session PMTU enabled, path MTU is not known
  No session cookie information available
  FS cached header information:
    encap size = 24 bytes
    45000014 00004000 FF734469 96010505
    96010606 0000D96C
  Sequencing is off
  Conditional debugging is disabled
  SSM switch id is 4098, SSM segment id is 8196
  Unique ID is 5
Session Layer 2 circuit, type is Ethernet, name is FastEthernet0/1
  Session vcid is 100
  Circuit state is UP
    Local circuit state is UP
    Remote circuit state is UP
```

Note

Now configure SW3 and SW4 as the CE devices for L2 VPN. Assign them any IP addresses on the same subnet and confirm that you can ping across:

```
SW4:
interface FastEthernet 0/6
 no switchport
 ip address 56.56.56.10 255.255.255.0

SW3:
interface FastEthernet 0/5
 no switchport
 ip address 56.56.56.9 255.255.255.0

Rack1SW3#ping 56.56.56.10

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 56.56.56.10, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 33/35/42 ms
```

Note

Verify that Path MTU discovery is working for the pseudowire. Make SW3 send large packets with DF-bit set. Notice that PE router is sending the ICMP unreachables to the source IP address found in encapsulated packets, allowing SW3 to properly implement PMTU discovery.

```
Rack1SW3#debug ip icmp
ICMP packet debugging is on

Rack1SW3#ping 56.56.56.10 size 1540 df-bit

Type escape sequence to abort.
Sending 5, 1540-byte ICMP Echos to 56.56.56.10, timeout is 2 seconds:
Packet sent with the DF bit set

ICMP: dst (56.56.56.10): frag. needed and DF set.
ICMP: time exceeded rcvd from 56.56.56.10
ICMP: dst (56.56.56.10): frag. needed and DF set.
ICMP: dst (56.56.56.10): frag. needed and DF set.
```

```
ICMP: dst (56.56.56.10): frag. needed and DF set.
ICMP: dst (56.56.56.10): frag. needed and DF set.
Success rate is 0 percent (0/5)
```

Note

Notice that the above procedure also permitted R5 to correctly discover Path MTU. This could be seen in the output of the same command we used before:

```
Rack1R5#show l2tp session all

L2TP Session Information Total tunnels 1 sessions 1

Session id 19547 is up, tunnel id 38503
  Remote session id is 55660, remote tunnel id 34507
  Locally initiated session
Call serial number is 1694600001
Remote tunnel name is Rack1R6
  Internet address is 150.1.6.6
Local tunnel name is Rack1R5
  Internet address is 150.1.5.5
IP protocol 115
  Session is L2TP signaled
  Session state is established, time since change 00:09:16
    114 Packets sent, 130 received
    31488 Bytes sent, 34186 received
  Last clearing of counters never
  Counters, ignoring last clear:
    114 Packets sent, 130 received
    31488 Bytes sent, 34186 received
    Receive packets dropped:
      out-of-order:             0
      total:                    0
    Send packets dropped:
      exceeded session MTU:     0
      total:                    0
  DF bit on, ToS reflect enabled, ToS value 0, TTL value 255
  Sending UDP checksums are disabled
  Receiving UDP checksums are verified
  Session PMTU enabled, path MTU is 1496 bytes
  No session cookie information available
<snip>
```

14.16 MPLS VPN Performance Tuning

- Configure the PE and P routers to minimize the amount of time needed to propagate topology changes between CE-sites.

Configuration

Note

MPLS VPNs utilize MP-BGP as the transport for IGP routing updates between VPN sites. BGP was designed as highly-scalable protocol, which however limits the ability to quickly propagate changes in network topology. One of the reasons for that is the goal to maintain network stability, which, of course, slows convergence. Thus, it is not uncommon to see network changes at one site to take a minute or even more to propagate to other sites, even with “fast” protocols such as OSPF and EIGRP. The following main factors affect the amount of time it takes to propagate from one PE to another across the MP-BGP core:

1) The time it takes for the IGP update to be redistributed into BGP. In the past this was limited by **`bgp scan-interval`** general scanning interval, but recent IOS versions made this feature even driven. This makes IGP to BGP route redistribution almost instant.

2) The time it takes the local BGP speaker and other BGP speakers to propagate updates to their peers. This time is controlled by the timer known as peer advertisement-interval. This interval specifies the periodic “batching” events: BGP waits for the timer to expire and accumulates the updates instead of sending every one immediately. The purpose is preventing constant load on the peer’s CPU for best-path re-calculations. The default interval is 5 seconds, and could be set as low as 0 seconds using the command **`neighbor <IP> advertisement-interval`**.

3) The time it takes the PE router’s BGP process to import the MP-BGP VPNv4 prefixes into the local VRF table. This timer is controlled by the command **`bgp scan-time import <5-60>`** which should be entered under VPNv4 address-family configuration. The default value is 15 second, and setting it to 5 seconds may significantly decrease the update propagation time.

The above discussed commands could be tuned to their minimal values, resulting in change propagation time between 5-10 seconds. This is probably the best time you can get with MPLS VPNs – everything else depends on IGP convergence. Also notice that by tuning this parameters you do not improve the time it takes to detect link or router failures in the MPLS core.

R4:

```
router bgp 100
 address-family vpnv4 unicast
 neighbor 150.1.5.5 advertisement-interval 0
 neighbor 150.1.6.6 advertisement-interval 0
 bgp scan import 5
```

R5 & R6:

```
router bgp 100
 address-family vpnv4 unicast
 neighbor 150.1.4.4 advertisement-interval 0
 bgp scan import 5
```

Verification

Note

Check the optimized BGP timers in one of the routers. The other routers configuration is verified similarly.

```
Rack1R5#show bgp vpnv4 unicast all neighbors 150.1.4.4  | inc adv
    Route refresh: advertised and received(new)
    New ASN Capability: advertised and received
    Address family VPNv4 Unicast: advertised and received
  Default minimum time between advertisement runs is 0 seconds
```

Note

To validate the import timer, run the following debugging command and notice the timestamps increments:

```
Rack1R5#debug ip bgp events
BGP events debugging is on

Rack1R5#
23:27:41: BGP(4): Import timer expired. Walking from 55 to 55
Rack1R5#
23:27:46: BGP(4): Import timer expired. Walking from 55 to 55
Rack1R5#
23:27:51: BGP(4): Import timer expired. Walking from 55 to 55
```