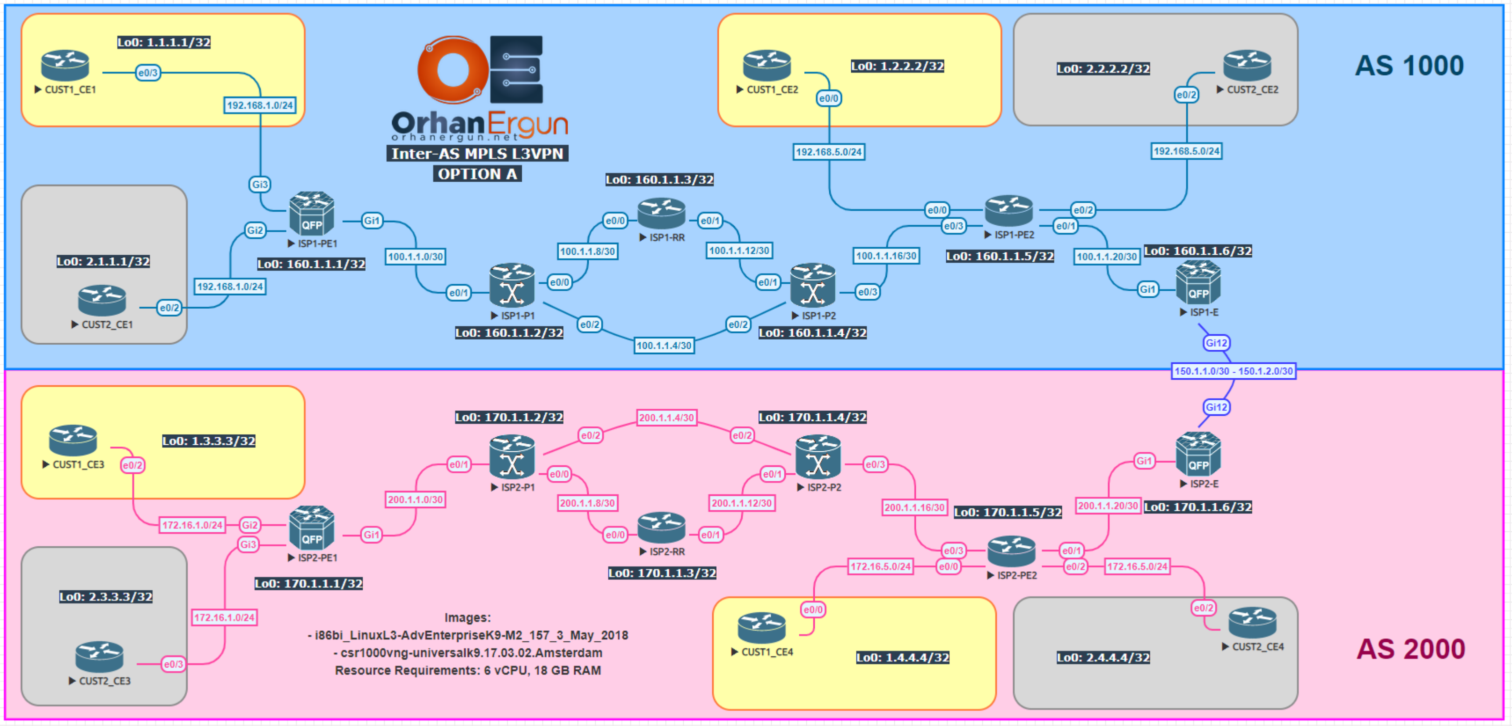
Inter-AS MPLS L3VPN Option A

Topology:



Images:
- i86bi_LinuxL3-AdvEnterpriseK9-M2_157_3_May_2018
- csr1000vng-universalk9.17.03.02.Amsterdam
Resource Requirements: 6 vCPU, 18 GB RAM

BGP AF/SAF: VPNv4 Unicast

PE-CE Routing Protocols:

- OSPF (Cust1)
- EIGRP Named-Mode (Cust2)

Task 01:

- Provide MPLS L3VPN service to the customers (Cust1 PE-CE: OSPF, Cust2 PE-CE: EIGRP)
- ISP1-RR and ISP2-RR should be configured as the Route-Reflectors

Solution:

You are already familiar with the L3VPN configuration, so we are not going to go through detailed explanation of the configuration steps:

ISP1-PE1:

```
vrf definition CUST1
 rd 1000:1
 !
 address-family ipv4
  route-target export 160.1.1.1:1
  route-target import 160.1.1.5:1
  route-target import 160.1.1.6:1
 exit-address-family
!
!
interface GigabitEthernet3
 vrf forwarding CUST1
 ip address 192.168.1.1 255.255.255.0
 negotiation auto
 no mop enabled
 no mop sysid
!
router ospf 1 vrf CUST1
 redistribute bgp 1000
 network 0.0.0.0 255.255.255.255 area 0
!
!
router bgp 1000
 !
 address-family ipv4 vrf CUST1
  redistribute ospf 1 match internal external 1 external 2
 exit-address-family
!
vrf definition CUST2
 rd 1000:2
 !
 address-family ipv4
  route-target export 160.1.1.1:2
  route-target import 160.1.1.5:2
  route-target import 160.1.1.6:2
 exit-address-family
!
!
interface GigabitEthernet2
 vrf forwarding CUST2
 ip address 192.168.1.1 255.255.255.0
 negotiation auto
 no mop enabled
 no mop sysid
!
router eigrp CUST2
!
address-family ipv4 unicast vrf CUST2 autonomous-system 2
  !
  topology base
   redistribute bgp 1000
  exit-af-topology
  network 0.0.0.0
 exit-address-family
!
router bgp 1000
 !
 address-family ipv4 vrf CUST2
  redistribute eigrp 2
 exit-address-family
!
router bgp 1000
 bgp log-neighbor-changes
 no bgp default ipv4-unicast
 neighbor 160.1.1.3 remote-as 1000
 neighbor 160.1.1.3 update-source Loopback0
 !
 address-family ipv4
 exit-address-family
 !
 address-family vpnv4
  neighbor 160.1.1.3 activate
  neighbor 160.1.1.3 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf CUST1
  redistribute ospf 1 match internal external 1 external 2
 exit-address-family
 !
 address-family ipv4 vrf CUST2
  redistribute eigrp 2
 exit-address-family
```

ISP1-RR:

```
router bgp 1000
 bgp log-neighbor-changes
 no bgp default ipv4-unicast
 neighbor 160.1.1.1 remote-as 1000
 neighbor 160.1.1.1 update-source Loopback0
 neighbor 160.1.1.5 remote-as 1000
 neighbor 160.1.1.5 update-source Loopback0
 neighbor 160.1.1.6 remote-as 1000
 neighbor 160.1.1.6 update-source Loopback0
 !
 address-family vpnv4
  neighbor 160.1.1.1 activate
  neighbor 160.1.1.1 send-community extended
  neighbor 160.1.1.1 route-reflector-client
  neighbor 160.1.1.5 activate
  neighbor 160.1.1.5 send-community extended
  neighbor 160.1.1.5 route-reflector-client
  neighbor 160.1.1.6 activate
  neighbor 160.1.1.6 send-community extended
  neighbor 160.1.1.6 route-reflector-client
 exit-address-family
```

ISP1-PE2:

```
router bgp 1000
 bgp log-neighbor-changes
 no bgp default ipv4-unicast
 neighbor 160.1.1.3 remote-as 1000
 neighbor 160.1.1.3 update-source Loopback0
 !
 address-family ipv4
 exit-address-family
 !
 address-family vpnv4
  neighbor 160.1.1.3 activate
  neighbor 160.1.1.3 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf CUST1
  redistribute ospf 1 match internal external 1 external 2
 exit-address-family
 !
 address-family ipv4 vrf CUST2
  redistribute eigrp 2
 exit-address-family
!
vrf definition CUST1
 rd 1000:1
 !
 address-family ipv4
  route-target export 160.1.1.5:1
  route-target import 160.1.1.1:1
  route-target import 160.1.1.6:1
 exit-address-family
!
!
interface Ethernet0/0
 vrf forwarding CUST1
 ip address 192.168.5.1 255.255.255.0
 duplex auto
!
router ospf 1 vrf CUST1
 redistribute bgp 1000 subnets
 network 0.0.0.0 255.255.255.255 area 0
!
!
router bgp 1000
 !
 address-family ipv4 vrf CUST1
  redistribute ospf 1 match internal external 1 external 2
 exit-address-family
!
vrf definition CUST2
 rd 1000:2
 !
 address-family ipv4
  route-target export 160.1.1.5:2
  route-target import 160.1.1.1:2
  route-target import 160.1.1.6:2
 exit-address-family
!
!
interface Ethernet0/2
 vrf forwarding CUST2
 ip address 192.168.5.1 255.255.255.0
 duplex auto
!
router eigrp CUST2
!
address-family ipv4 unicast vrf CUST2 autonomous-system 2
  !
  topology base
   redistribute bgp 1000
  exit-af-topology
  network 0.0.0.0
 exit-address-family
!
router bgp 1000
 !
 address-family ipv4 vrf CUST2
  redistribute eigrp 2
 exit-address-family
!
```

Service Provider 1 basic configuration is done.

We will configure the ISP2 the same way:

ISP2-PE1:

```
router bgp 2000
 bgp log-neighbor-changes
 no bgp default ipv4-unicast
 neighbor 170.1.1.3 remote-as 2000
 neighbor 170.1.1.3 update-source Loopback0
 !
 address-family ipv4
 exit-address-family
 !
 address-family vpnv4
  neighbor 170.1.1.3 activate
  neighbor 170.1.1.3 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf CUST1
  redistribute ospf 1 match internal external 1 external 2
 exit-address-family
 !
 address-family ipv4 vrf CUST2
  redistribute eigrp 2
 exit-address-family
vrf definition CUST1
 rd 2000:1
 !
 address-family ipv4
  route-target export 170.1.1.1:1
  route-target import 170.1.1.5:1
  route-target import 170.1.1.6:1
 exit-address-family
!
!
interface GigabitEthernet2
 vrf forwarding CUST1
 ip address 172.16.1.1 255.255.255.0
 negotiation auto
 no mop enabled
 no mop sysid
!
router ospf 1 vrf CUST1
 redistribute bgp 2000
 network 0.0.0.0 255.255.255.255 area 0
!
!
router bgp 2000
 !
 address-family ipv4 vrf CUST1
  redistribute ospf 1 match internal external 1 external 2
 exit-address-family
!
vrf definition CUST2
 rd 2000:2
 !
 address-family ipv4
  route-target export 170.1.1.1:2
  route-target import 170.1.1.5:2
  route-target import 170.1.1.6:2
 exit-address-family
!
!
interface GigabitEthernet3
 vrf forwarding CUST2
 ip address 172.16.1.1 255.255.255.0
 negotiation auto
 no mop enabled
 no mop sysid
!
router eigrp CUST2
!
```

```
address-family ipv4 unicast vrf CUST2 autonomous-system 2
  !
  topology base
   redistribute bgp 2000
  exit-af-topology
  network 0.0.0.0
 exit-address-family
!
router bgp 2000
 !
 address-family ipv4 vrf CUST2
  redistribute eigrp 2
 exit-address-family
!
```

ISP2-RR:

```
router bgp 2000
 bgp log-neighbor-changes
 no bgp default ipv4-unicast
 neighbor 170.1.1.1 remote-as 2000
 neighbor 170.1.1.1 update-source Loopback0
 neighbor 170.1.1.5 remote-as 2000
 neighbor 170.1.1.5 update-source Loopback0
 neighbor 170.1.1.6 remote-as 2000
 neighbor 170.1.1.6 update-source Loopback0
 !
 address-family vpnv4
  neighbor 170.1.1.1 activate
  neighbor 170.1.1.1 send-community extended
  neighbor 170.1.1.1 route-reflector-client
  neighbor 170.1.1.5 activate
  neighbor 170.1.1.5 send-community extended
  neighbor 170.1.1.5 route-reflector-client
  neighbor 170.1.1.6 activate
  neighbor 170.1.1.6 send-community extended
  neighbor 170.1.1.6 route-reflector-client
 exit-address-family

ISP2-PE2:
router bgp 2000
 bgp log-neighbor-changes
 no bgp default ipv4-unicast
 neighbor 170.1.1.3 remote-as 2000
 neighbor 170.1.1.3 update-source Loopback0
 !
 address-family ipv4
 exit-address-family
 !
 address-family vpnv4
  neighbor 170.1.1.3 activate
  neighbor 170.1.1.3 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf CUST1
  redistribute ospf 1 match internal external 1 external 2
 exit-address-family
 !
 address-family ipv4 vrf CUST2
  redistribute eigrp 2
 exit-address-family
vrf definition CUST1
 rd 2000:1
 !
 address-family ipv4
  route-target export 170.1.1.5:1
  route-target import 170.1.1.1:1
  route-target import 170.1.1.6:1
 exit-address-family
```

```
!
!
interface Ethernet0/0
 vrf forwarding CUST1
 ip address 172.16.5.1 255.255.255.0
 duplex auto
!
router ospf 1 vrf CUST1
 redistribute bgp 2000 subnets
 network 0.0.0.0 255.255.255.255 area 0
!
!
router bgp 2000
 !
 address-family ipv4 vrf CUST1
  redistribute ospf 1 match internal external 1 external 2
 exit-address-family
!
vrf definition CUST2
 rd 2000:2
 !
 address-family ipv4
  route-target export 170.1.1.5:2
  route-target import 170.1.1.1:2
  route-target import 170.1.1.6:2
 exit-address-family
!
!
interface Ethernet0/2
 vrf forwarding CUST2
 ip address 172.16.5.1 255.255.255.0
 duplex auto
!
router eigrp CUST2
```

```
!
address-family ipv4 unicast vrf CUST2 autonomous-system 2
  !
  topology base
   redistribute bgp 2000
  exit-af-topology
  network 0.0.0.0
 exit-address-family
!
router bgp 2000
 !
 address-family ipv4 vrf CUST2
  redistribute eigrp 2
 exit-address-family
!
```

ISP2 basic configuration is also done.

So the customer sites inside each ISP have end to end reachability.

But what about customer sites reachability between two different sites which are connected to the different ISPs?

They don’t have any connectivity so far (for example CUST1_CE1 cannot ping CUST1_CE3).

The solution is using Inter-AS MPLS VPN different options.

There are 4 option: A, B, C and D.

The difference between those options are mostly how Edge routers interact with each other at the edge of the network (also in option C we will discuss about RR interaction on different Service Providers).

Task 02:

- Make sure that all customer sites have IP reachability (Use Option A)

Solution:

Option A is not scalable, because of the need for defining separate VRF for each customer on the edge routers (ASBRs).

In this topology we have two ASBRs (ISP1-E and ISP2-E). these ASBRs are connected to each other using GigabitEthernet12.

We should configure two sub-interfaces on this interface for each of the customers (imagin if we had 100 Customers!).

```
ISP1-E:
vrf definition CUST1
 rd 1000:1
 !
 address-family ipv4
  route-target export 160.1.1.6:1
  route-target import 160.1.1.1:1
  route-target import 160.1.1.5:1
 exit-address-family
!
!
interface GigabitEthernet12
 no shutdown
!
interface GigabitEthernet12.1
 encapsulation dot1Q 1 native
 vrf forwarding CUST1
 ip address 150.1.1.1 255.255.255.252
!
router bgp 1000
 !
 address-family ipv4 vrf CUST1
  neighbor 150.1.1.2 remote-as 2000
  neighbor 150.1.1.2 activate
  neighbor 150.1.1.2 send-community extended
 exit-address-family
!
vrf definition CUST2
 rd 1000:2
 !
 address-family ipv4
  route-target export 160.1.1.6:2
  route-target import 160.1.1.1:2
  route-target import 160.1.1.5:2
 exit-address-family
!
!
interface GigabitEthernet12
 no ip address
 negotiation auto
 no mop enabled
 no mop sysid
!
interface GigabitEthernet12.2
 encapsulation dot1Q 2
 vrf forwarding CUST2
 ip address 150.1.2.1 255.255.255.252
!
router bgp 1000
 !
 address-family ipv4 vrf CUST2
  neighbor 150.1.2.2 remote-as 2000
  neighbor 150.1.2.2 activate
  neighbor 150.1.2.2 send-community extended
 exit-address-family
!
```

Now, each customer data will be carried in an isolated way thanks to the dot1q tags.

The edge device needs to form BGP session with the Route-Reflector inside it's own ISP (VPNv4 Unicast AF/SAF needs to be enabled):

```
ISP1-E:
router bgp 1000
 bgp log-neighbor-changes
 no bgp default ipv4-unicast
 neighbor 160.1.1.3 remote-as 1000
 neighbor 160.1.1.3 update-source Loopback0
 !
 address-family ipv4
 exit-address-family
 !
 address-family vpnv4
  neighbor 160.1.1.3 activate
  neighbor 160.1.1.3 send-community extended
 exit-address-family
 !
```

If you take a closer look at the first set of the configuration, you will realize that for each of the customers, The Edge devices are forming eBGP neighborship, you can also use any other dynamic routing protocol between ASBRs, but BGP is easier, no need for redistribution and there is only a single protocol (BGP) to troubleshoot and take care of!

We configure the second ISP edge router the same way:

```
ISP2-E:
router bgp 2000
 bgp log-neighbor-changes
 no bgp default ipv4-unicast
 neighbor 170.1.1.3 remote-as 2000
 neighbor 170.1.1.3 update-source Loopback0
 !
 address-family ipv4
 exit-address-family
 !
 address-family vpnv4
  neighbor 170.1.1.3 activate
  neighbor 170.1.1.3 send-community extended
 exit-address-family
!
vrf definition CUST1
 rd 2000:1
 !
 address-family ipv4
  route-target export 170.1.1.6:1
  route-target import 170.1.1.1:1
  route-target import 170.1.1.5:1
 exit-address-family
!
!
interface GigabitEthernet12
 no shutdown
!
interface GigabitEthernet12.1
 encapsulation dot1Q 1 native
 vrf forwarding CUST1
 ip address 150.1.1.2 255.255.255.252
!
router bgp 2000
 !
 address-family ipv4 vrf CUST1
  neighbor 150.1.1.1 remote-as 1000
  neighbor 150.1.1.1 activate
  neighbor 150.1.1.1 send-community extended
 exit-address-family
!
vrf definition CUST2
 rd 2000:2
 !
 address-family ipv4
  route-target export 170.1.1.6:2
  route-target import 170.1.1.1:2
  route-target import 170.1.1.5:2
 exit-address-family
!
!
interface GigabitEthernet12
 no ip address
 negotiation auto
 no mop enabled
 no mop sysid
!
interface GigabitEthernet12.2
 encapsulation dot1Q 2
 vrf forwarding CUST2
 ip address 150.1.2.2 255.255.255.252
!
router bgp 2000
 !
 address-family ipv4 vrf CUST2
  neighbor 150.1.2.1 remote-as 1000
  neighbor 150.1.2.1 activate
  neighbor 150.1.2.1 send-community extended
 exit-address-family
!
```

That was all we needed to do!

NOTE: Each of the Service Providers can have their own Route-Distinguisher and Route-Target values, there is no need to configure the same value on both of them.

The traffic path (when CUST1_CE1 wants to send some traffic to CUST1_CE3):

- CUST1_CE1 sends the traffic to ISP1-PE1
- ISP1-PE1 received the VPN label advertised for 1.3.3.3/32 from ISP1-E (in fact RR got the route from ISP1-E and advertised it to ISP1-PE1)
- ISP1-E sends the packet to ISP2-E
- ISP2-E sends it towards ISP2-PE1 (because it also received VPN label for that prefix from ISP2-PE1

Verification:

```
CUST1_CE1#trace 1.3.3.3 source lo 0

Type escape sequence to abort.

Tracing the route to 1.3.3.3

VRF info: (vrf in name/id, vrf out name/id)

  1 192.168.1.1 1 msec 0 msec 0 msec

  2 100.1.1.2 [MPLS: Labels 2005/6010 Exp 0] 6 msec 2 msec 2 msec

  3 100.1.1.6 [MPLS: Labels 4006/6010 Exp 0] 1 msec 2 msec 1 msec

  4 100.1.1.18 [MPLS: Labels 5011/6010 Exp 0] 2 msec 1 msec 1 msec

  5 150.1.1.1 [MPLS: Label 6010 Exp 0] 2 msec 1 msec 2 msec

  6 150.1.1.2 1 msec 2 msec 1 msec

  7 200.1.1.21 [MPLS: Labels 5012/1010 Exp 0] 3 msec 3 msec 3 msec

  8 200.1.1.17 [MPLS: Labels 4007/1010 Exp 0] 2 msec 3 msec 3 msec

  9 200.1.1.5 [MPLS: Labels 2007/1010 Exp 0] 2 msec 3 msec 3 msec

 10 172.16.1.1 [MPLS: Label 1010 Exp 0] 3 msec 3 msec 3 msec

 11 172.16.1.100 4 msec 3 msec 3 msec
```

As you realize each ISP have their own VPN or Service labels, ISP1 VPN label for that prefix is 6010 and ISP2 VPN label is 1010.

When ISP1-E forwards the packet to ISP2-E, there is no label, so MPLS does not need to be enabled on Gig12 sub-interfaces.

```
ISP1-E#show mpls forwarding-table vrf CUST1

Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop

Label      Label      or Tunnel Id     Switched      interface

6010       No Label   1.3.3.3/32[V]    2430          Gi12.1     150.1.1.2

6011       No Label   1.4.4.4/32[V]    0             Gi12.1     150.1.1.2

6012       No Label   172.16.1.0/24[V] 0             Gi12.1     150.1.1.2

6013       No Label   172.16.5.0/24[V] 0             Gi12.1     150.1.1.2

ISP1-E#show ip cef vrf CUST1 1.3.3.3 detail

1.3.3.3/32, epoch 2, flags [rib only nolabel, rib defined all labels]

  dflt local label info: other/6010 [0x2]

  recursive via 150.1.1.2

    attached to GigabitEthernet12.1
```