What is MPLS?

MPLS SP routers will use Labels to Forward Packets, But The main Benefit of MPLS is providing Layer Three connectivity to Customers instead of Layer 2 only such as Frame Relay. By providing Layer three connectivity SP edge Router is acting as part of Customer Internal Network.

As you can see above I have two companies (Cbtme & Traininghouse) and with MPLS SP I can connect each HQ of them with branches and same time I can connect one costumer site with another if they like to or if the two companies are going to merge, even if one of the sites ask for internet connection SP can provide it in same time. Above Topology represent one of the most common MPLS Application called MPLS Layer 3 VPN

P = Provider Router
PE = Provider Edge Router
CE = Customer Edge Router

To understand MPLS we need to ask ourselves many questions?
- How Labeling work inside MPLS SP cloud?
- How SP MPLS enabled Routers (P & PE) will communicate with each other?
- How customer router will communicate with PE?
- How single PE router can communicate with more than one customer while he had one routing table and how he will separate them and recognize them?
- How different Customers will communicate with PE if they are using same IPv4 addressing schema?

And many other questions will pop up while we try to answer above questions, so let’s start solving this puzzle.
MPLS need CEF to be running since it makes use of FIB & Adj Table

In RIB we see routes (network) and how to reach using next hop (same for RIB version created by CEF and we call it FIB -Forwarding Information Base)

In ARP Cache we see next hop address and what is his mac address (same for Adjacency Table created by CEF)

```
config t
ip cef < this command enable cef, but anyway by default CEF is enabled
```

```
int fas 0/0
ip route-cache cef < This command ensures CEF switching is done for packets that enter this interface.
```

Multiprotocol Label Switching (MPLS) is open standard defined in RFC 3031
Previously was Cisco proprietary called Tag Switching, that is why we will see this word many times when execute IOS MPLS related show commands `label = tag (in cisco world)`

Multiprotocol mean it support Ethernet, HDLC,PPP, frame relay, ATM in layer2 AND support ipv4, ipv6 in layer 3

In MPLS Traffic is switched between interfaces based on locally significant label values Label is just a number created on each router for each route (network) he can reach, then advertise these labels to his neighbors, so later forwarding frame will be based on Label Lookup.

**MPLS Domain (cloud) terminology:**
- LSR Label Switch Router = Provider = P
- E-LSR edge Label Switch Router = Provider Edge = PE
- LSP Label Switch Path = path from PE to another PE through many P's routers
- MPLS enabled interface is interface where packets will labeled & go out from it.

**MPLS label**
MPLS label format defined in RFC 3032
4 byte header added between layer 2 and layer headers (that’s why we call it two & half header).
**MPLS Labels is 32 bit:**
20 bit label number (locally significant to router)
3 bit EXP= class of service
S Bit = define last label in the label stack (called Bottom-Of-Stack bit)
8 bit TTL = time to live (indicate this last label before ip header)

**Label Stack** means we could have more than one label
So when more than one label assigned we call it label stack , normally we would have three labels if we use MPLS TE (traffic engineering) which beyond CCIE R&S and fill in CCIE SP track.

**LDP label** < used to carry LDP label Number  (LDP or any other label exchange Protocol)
**TE label**
**VPN label** < used with MP-BGP in MPLS VPN

Max labels in frame could be 3 , each one is 4 byte so total is 12 , each LSR must be able to handle bigger MTU  that is why we could need to write the following command under mpls enabled interface :

```
mpls mtu 1512
```

- The first label in the stack is called the **top label**, and the last label is called the **bottom label**.
- The Bottom-of-stack bit indicates whether the label is the last label in the stack. if set to 1 that indicate this the last label
- Receiving router uses the top label only

**How labels work ?**
According to topology in the next page
Packet  with label come from R3 to R2 int f0/0 with label X (where X is local label number created in R2 and advertised to other routers including R3)
Then send from R2 f0/1 with label Y (where Y is remote label number created by R4 and advertised to other routers including R2 where  R2 add this info on his LIB.
Operations happened to labels in MPLS routers:

**PUSH operation**: adds a new label to the IP packet or to the MPLS label stack of the packet. The push operation is commonly done by the ingress router except in some traffic engineering scenarios.

**SWAP operation**: the top most label is swapped by another one before switching the packet to the next downstream LSR. This is commonly done by intermediate LSRs in the provider network.

**POP operation**: removes the top most label from the label stack to prepare that packet for its final destination. This is commonly done by the egress router or by the router preceding the egress router as **Penultimate Hop Popping** or **PHP** in brief.

Penultimate hop popping is an operation performed by a certain LSR in the MPLS network before sending the packet to the Label Edge Router (LER). The process is done by removing the top most label of the MPLS packet to reduce the overhead of the double lookup on the LER.
**How MPLS routers will POP, PUSH or SWAP Labels?**

Label Transport Protocols used to exchange labels between P’s and PE’s:

- LDP Standard
- TDP Cisco (FIB called TIB, LFIB called TFIB) TCP 711
- RSVP used for mpls TE

We only concern about LDP in CCIE R&S

**LDP** RFC 3036

Neighbor automatically discover and send hello messages using UDP port# 646 to 224.0.0.2.

Then (neighbor adjacency) LDP establish TCP session with LDP peer (Two LSR) TCP port# 646 to remote Ldp router-id

We can make E-LSR receive packets without labels, if we make last router to him remove labels instead this process called PHP.

PHP accomplished through implicit NULL label advertisement for connected prefix

PHP is penultimate hop popping which means remove the label one hop before its destination.

Label advertisement:
- Advertise FEC for connected IGP interfaces
- Advertise FEC for IGP learned routes

**Remember, LDP send hello message using UDP then open TCP session with Neighbor to send/receive Labels.**
**What is a Forwarding Equivalence Class (FEC)?**

FEC is a group of IP packets which are forwarded in the same manner, over the same path, and with the same forwarding treatment. An FEC might correspond to a destination IP subnet, but it also might correspond to any traffic class that the Edge-LSR considers significant. For example, all traffic with a certain value of IP precedence might constitute a FEC.

**What is the range of Labels Numbers we can use?**

In Cisco IOS we can use numbers from 16 to 100,000, BUT from 0 to 15 is reserved.

**Reserved Labels**

Labels 0 through 15 are reserved labels.

An LSR cannot use them in the normal case for forwarding packets.

An LSR assigns a specific function to each of these labels.

Label 0 is the explicit NULL label, whereas label 3 is the implicit NULL label. Label 1 is the router alert label, whereas label 14 is the OAM alert label.

The other reserved labels between 0 and 15 have not been assigned yet.

**Implicit NULL Label (value 3)**

The egress LSR—running Cisco IOS—assigns the implicit NULL label to its connected and summarized prefixes. The benefit of this is that if the egress LSR were to assign a label for these FECs, it would receive the packets with one label on top of it. It would then have to do two lookups. First, it would have to look up the label in the LFIB, just to figure out that the label needs to be removed; then it would have to perform an IP lookup. These are two lookups, and the first is unnecessary.

The use of implicit NULL at the end of an LSP is called penultimate hop popping (PHP). The LFIB entry for the LSP on the PHP router shows a "Pop Label" as the outgoing label PHP is the default mode in Cisco IOS. In the case of IPv4-over-MPLS, Cisco IOS only advertises the implicit NULL label for directly connected routes and summarized routes.

A value of 3 represents the "Implicit NULL Label". This is a label that an LSR can assign and distribute. However, it never actually appears in the encapsulation. It indicates that the LSR pops the top label from the stack and forwards the rest of the packet (labeled or unlabeled) through the outgoing interface (as per the entry in Lfib). Although this value might never appear in the encapsulation, it needs to be specified in the Label Distribution Protocol, so a value is reserved.

A value of 2 represents the "IPv6 Explicit NULL Label". It indicates that the label stack must be popped, and the packet forwarding must be based on the IPv6 header.

**Explicit NULL Label (value 0)**

When a label is removed, the EXP bits are also removed. Because the EXP bits are exclusively used for quality of service (QoS), the QoS part of the packet is lost when the top label is removed. In some cases, you might want to keep this QoS information and have it delivered to the egress LSR. Implicit NULL cannot be used in that case.

A value of 0 represents the "IPv4 Explicit NULL Label". This label indicates that the label stack must be popped, and the packet forwarding must be based on the IPv4 header. This helps to keep Exp bits safe until the egress router. It is used in MPLS based QoS.
If you want to Force the process to stop relying on the PHP behavior. This can be accomplished by telling R to use the explicit-null configuration.

R1(config)# mpls ldp explicit-null

R1# show mpls forwarding-table
Local Outgoing Prefix Bytes Label Outgoing Next Hop
Label Label or VC or Tunnel Id Switched interface
16 explicit-n 1.1.1.1/32 0 Fa0/0 172.16.15.5
17 explicit-n 2.2.2.2/32 0 Fa0/0 172.16.15.5
18 explicit-n 3.3.3.3/32 0 Fa0/0 172.16.15.5
19 explicit-n 4.4.4.4/32 0 Fa0/0 172.16.15.5
20 explicit-n 5.5.5.5/32 0 Fa0/0 172.16.15.5
21 explicit-n 172.16.45.0/24 0 Fa0/0 172.16.15.5
22 52 192.1.4.4/32 0 Fa0/0 172.16.15.5
23 explicit-n 192.1.5.5/32 0 Fa0/0 172.16.15.5

Observe that the Pop Label entry we saw previously has now been replaced with the explicit-n label. This means that the PHP behavior has now been turned off.

Router Alert Label (value 1)
This label can be present anywhere in the label stack except at the bottom. When the Router Alert label is the top label, it alerts the LSR that the packet needs a closer look. Therefore, the packet is not forwarded in hardware, but it is looked at by a software process.

A value of 1 represents the "Router Alert Label". When a received packet contains this label value at the top of the label stack, it is delivered to a local software module for processing. The actual packet forwarding is determined by the label beneath it in the stack. However, if the packet is forwarded further, the Router Alert Label should be pushed back onto the label stack before forwarding. The use of this label is analogous to the use of the "Router Alert Option" in IP packets (for example, ping with record route option)

OAM Alert Label (value 14)
Operation and Maintenance (OAM) Alert label , Cisco IOS does support the use of label 14. It does perform MPLS OAM, but not by using label 14.

Unreserved Labels
Because the label value has 20 bits, the labels from 16 through 1,048,575 (220 – 1) are used for normal packet forwarding. In Cisco IOS, the default range is 16 through 100,000.

you can let router choose labels numbers or define range of labels numbers can be use on each router

R1(config)# mpls label range 16 1048575

R1# show mpls label range
Downstream Generic label region: Min/Max label: 16/1048575

Special-Purpose MPLS Label Values :
http://www.iana.org/assignments/mpls-label-values/mpls-label-values.xhtml
Types of Tables used in MPLS world

RIB – Routing Information Base  
```
sh ip route / sh ip route vrf AS100
```

ARP Cache  
```
sh arp / sh ip arp
```

FIB – Forwarding Information Base  
```
sh ip cef / sh ip cef vrf AS100
```

Adjacency Table  
```
sh adjacency
```

LFIB – Label Forwarding Instance Base  
```
sh mpls forwarding-table
```

This is the table that the router uses to forward labeled packets going through the network. Much like the RIB uses the FIB to forward traffic, so the LIB uses the LFIB to forward traffic.

LIB – Label Information Base  
```
sh mpls ldp bindings
```

This is the place where the router will keep all known MPLS labels.

MPLS LDP Basic Configuration

```
config t
mpls label protocol ldp
mpls ldp router-id loopback0
int fas 0/0
mpls ip
```

We can specify label protocol under interface as well, but both peers directly connected interface must use same protocol.

```
int f0/0
mpls label protocol ldp
```

The story of LDP Labels

By looking at the top label of the received labeled packet and the corresponding entry in the LFIB, the LSR knows how to forward the packet. The LSR determines what label operation needs to be performed—swap, push, or pop—and what the next hop is to which the packet needs to be forwarded.

**POP** = removing label from packet

**PUSH** = adding label to packet

**SWAP** = replace local label on packet with another (remote) label

Remember IP packet forward using RIB (FIB in CEF)

Labeled Packet forward using LFIB

```
R1# sh mpls forwarding-table
Local Outgoing Prefix Bytes tag Outgoing Next Hop
Tag tag or VC or Tunnel Id switched interface
16 Untagged 10.1.1.0/24 0 Et0/0/0 10.200.200.2
17 16 10.200.202.0/24 0 Et0/0/0 10.200.200.2
```
The local label (or tag) is the label that this LSR assigns and distributes to the other LSRs. As such, this LSR expects labeled packets to come to it with these labels as the top ones in the label stack. If this LSR were to receive a labeled packet with the top label 22, it would swap the label with label 17 and then forward it on the Ethernet0/0/0 interface. This is an example of the **label-to-label forwarding case**.

If the outgoing label (tag) is Untagged. This is an example of the **label-to-IP forwarding case**.

All routers running OSPF and all networks advertised and reachable
Now let’s assume R5 will advertise to R4 that he use label 500 for route 5.5.5.5

R4 will keep this info on LFIB and send to R3,R5 saying I am using label number 400 for 5.5.5.5

R3 will do the same will send R4 & R2 saying I am using label 300 for 5.5.5.5

R2 will do the same will send R3 & R1 saying I am using label 200 for 5.5.5.5

R1 will receive this info from R2 and send to R2 saying i will use 100 for 5.5.5.5

Notice R1 had three interfaces
Two interfaces are IP packet enabled interfaces
One interface MPLS enabled which is facing R2
If R6 ping R7 7.7.7.7 no MPLS operation will be used here
But if R7 ping 5.5.5.5 R5 in this case R1 will label this packet with 100 and send out from his mpls labeled interface

R1#show mpls forwarding-table 10.200.254.4 detail

Local Outgoing Prefix Bytes tag Outgoing Next Hop
tag tag or VC or Tunnel Id switched interface
23 16 10.200.254.4/32 0 Tu1 point2point

MAC/Encaps=14/22, MRU=1496, Tag Stack (20 16), via Et0/0/0
00604700881D00024A4008008847 0001400000010000
No output feature configured

If the detail keyword is specified, you can see all the labels that change in the label stack. From left to right between {}, you see the first label, which is the swapped label (20), and then the pushed label (16) onto the swapped label. Without the detail keyword, you see only the pushed label (16).

The CEF adjacency table determines the outgoing data link encapsulation. The adjacency table provides the necessary Layer 2 information to forward the packet to the next-hop LSR.
R1#show adjacency detail

It is possible for something to go wrong in the MPLS network and the LSR to start receiving labeled packets with a top label that the LSR does not find in its LFIB. The LSR can theoretically try two things: strip off the labels and try to forward the packet, or drop the packet. The Cisco LSR drops the packet.
**MPLS LDP Lab**

![MPLS LDP Lab Diagram]

**R1**
- int loop 0
- ip add 1.1.1.1 255.255.255.255
- int f0/0
- ip add 10.1.12.1 255.255.255.0
- int f0/1
- ip add 10.1.16.1 255.255.255.0
- int f1/0
- ip add 10.1.17.1 255.255.255.0

router ospf 100
router-id 0.0.0.1
network 0.0.0.0 0.0.0.0 are 0

**R2**
- int loop 0
- ip add 2.2.2.2 255.255.255.255
- int f0/0
- ip add 10.1.12.2 255.255.255.0
- int f0/1
- ip add 10.1.23.2 255.255.255.0
router ospf 100
router-id 0.0.0.2
network 0.0.0.0 0.0.0.0 are 0

**R3**
- int loop 0
- ip add 3.3.3.3 255.255.255.255
- int f0/0
- ip add 10.1.34.3 255.255.255.0
int f0/1
ip add 10.1.23.3 255.255.255.0
router ospf 100
router-id 0.0.0.3
network 0.0.0.0 0.0.0.0 are 0

R4
int loop 0
ip add 4.4.4.4 255.255.255.255
int f0/0
ip add 10.1.45.4 255.255.255.0
int f0/1
ip add 10.1.34.4 255.255.255.0
router ospf 100
router-id 0.0.0.4
network 0.0.0.0 0.0.0.0 are 0

R5
int loop 0
ip add 5.5.5.5 255.255.255.255
int f0/0
ip add 10.1.45.5 255.255.255.0
router ospf 100
router-id 0.0.0.5
network 0.0.0.0 0.0.0.0 are 0

R6
int loop 0
ip add 6.6.6.6 255.255.255.255
int f0/0
ip add 10.1.16.6 255.255.255.0
router ospf 100
router-id 0.0.0.6
network 0.0.0.0 0.0.0.0 are 0

R7
int loop 0
ip add 7.7.7.7 255.255.255.255
int f0/0
ip add 10.1.17.7 255.255.255.0
router ospf 100
router-id 0.0.0.7
network 0.0.0.0 0.0.0.0 are 0

CEF made FIB from RIB use show ip cef to check this table
CEF made adj table from adj l2 mac table use sh adjacency to check this table
R1
mpls label protocol ldp
mpls ldp router-id loop0 f
mpls label range 100 120
int f0/0
mpls ip

R2
mpls label protocol ldp
mpls ldp router-id loop0 f
mpls label range 200 220
int f0/0
mpls ip
int f0/1
mpls ip

sh mpls ldp discovery
sh mpls ldp interfaces
sh mpls ldp neighbors

R3
mpls label protocol ldp
mpls ldp router-id loop0 f
mpls label range 300 320
int f0/0
mpls ip
int f0/1
mpls ip

R4
mpls label protocol ldp
mpls ldp router-id loop0 f
mpls label range 400 420
int f0/0
mpls ip
int f0/1
mpls ip

R5
mpls label protocol ldp
mpls ldp router-id loop0 f
mpls label range 500 520
int f0/0
mpls ip

MPLS create LFIB from FIB use sh mpls forwarding-table to check this table
R1#sh mpls forwarding-table 5.5.5.5
Local Outgoing Prefix Bytes tag Outgoing Next Hop
tag tag or VC or Tunnel Id switched interface
107 208 5.5.5.5/32 0 Fa0/0 10.1.12.2

R2#sh mpls forwarding-table 5.5.5.5
Local Outgoing Prefix Bytes tag Outgoing Next Hop
tag tag or VC or Tunnel Id switched interface
208 308 5.5.5.5/32 0 Fa0/1 10.1.23.3

Mean my local label for 5.5.5.5 is 308
And will send packets to 10.1.34.4 using F0/0 with label 408

R3#sh mpls forwarding-table 5.5.5.5
Local Outgoing Prefix Bytes tag Outgoing Next Hop
tag tag or VC or Tunnel Id switched interface
308 408 5.5.5.5/32 0 Fa0/0 10.1.34.4

R4#sh mpls forwarding-table 5.5.5.5
Local Outgoing Prefix Bytes tag Outgoing Next Hop
tag tag or VC or Tunnel Id switched interface
408 Pop tag 5.5.5.5/32 0 Fa0/0 10.1.45.5

Mean my local label for 5.5.5.5 is 408
and will send packets to 10.1.45.5 using f0/0 with NO label ( POP )

R5#sh mpls forwarding-table 5.5.5.5
Local Outgoing Prefix Bytes tag Outgoing Next Hop
tag tag or VC or Tunnel Id switched interface

R4#sh mpls ldp bindings 5.5.5.5 32
tib entry: 5.5.5.5/32, rev 24
local binding: tag: 408
remote binding: tsr: 5.5.5.5:0, tag: imp-null
remote binding: tsr: 3.3.3.3:0, tag: 308

R4#sh mpls ldp discovery
Local LDP Identifier:
4.4.4.4:0 < Local LDP router id
Discovery Sources:
Interfaces:
FastEthernet0/0 (ldp): xmit/recv
LDP Id: 5.5.5.5:0 < LDP neighbor router id
FastEthernet0/1 (ldp): xmit/recv
LDP Id: 3.3.3.3:0 < LDP neighbor router id

R6#ping 5.5.5.5
R6#traceroute 5.5.5.5
Type escape sequence to abort.
Tracing the route to 5.5.5.5
1 10.1.16.1 40 msec 48 msec 52 msec
2 10.1.12.2 [MPLS: Label 208 Exp 0] 152 msec 156 msec 156 msec
3 10.1.23.3 [MPLS: Label 308 Exp 0] 140 msec 172 msec 156 msec
4 10.1.34.4 [MPLS: Label 408 Exp 0] 148 msec 184 msec 140 msec
5 10.1.45.5 144 msec 144 msec 184 msec

Mean to reach 5.5.5.5 we went to 10.1.16.1 then we go through 3 MPLS routers then finally we reach 10.1.45.5 where 5.5.5.5 exists.
We will talk later how to prevent seeing MPLS hops by Customers.
show mpls ldp bindings  
show mpls ldp discovery  
show mpls ldp forwarding  

To display the contents of the Label Information Base (LIB)  
To display the status of the LDP discovery process  
To display the LDP forwarding state installed in MPLS forwarding  

**Change LDP parameters**  
R1# show mpls ldp parameters  
Protocol version: 1  

*Downstream label generic region: min label: 100; max label: 120*  
Session hold time: 180 sec; keep alive interval: 60 sec  
*Discovery hello: holdtime: 15 sec; interval: 5 sec*  
*Discovery targeted hello: holdtime: 90 sec; interval: 10 sec*  
Downstream on Demand max hop count: 255  
Downstream on Demand Path Vector Limit: 255  
LDP for targeted sessions  
LDP initial/maximum backoff: 15/120 sec  
LDP loop detection: off  

Default LSR hello is 5 sec , hold down (dead) is 15 sec but we can change it let’s say to 15 , hold is 45  

mpls ldp discovery hello interval 15  
mpls ldp discovery hello holdtime 45  

Session Keepalive is 60s and hold is 180 sec but we change it to hold time 90 (which mean 90/3 = 30s Keepalive)  

mpls ldp holdtime 90  

**LDP Authentication**  
Let’s say we want to have authentication between r1 and r2  
R2(config)# mpls ldp neighbor 1.1.1.1 password cisco  
R1(config)# mpls ldp neighbor 2.2.2.2 password cisco  

To force the use of these MD5 passwords we will need to apply the mpls ldp password required command.  
R1(config)# mpls ldp password required  

R1# show mpls ldp neighbor password current  

R3# show mpls ldp parameters  
Protocol version: 1  
*Downstream label generic region: min label: 300; max label: 399*  

Labels given on each router from 16 to 100000 but as we saw before we can change this range specially for learning purpose, when you change mpls label range on already running Mpls routers, you will need to reload
**Configure LDP conditional outbound label advertising**

To exclude links from getting advertised labels

```
no mpls ldp advertise-labels
mpls ldp advertise-labels for 1 to 2
```

access-list 1 deny 10.1.12.0 0.0.0.255
access-list 1 deny 10.1.23.0 0.0.0.255
access-list 1 deny 10.1.34.0 0.0.0.255
access-list 1 deny 10.1.35.0 0.0.0.255
access-list 1 deny 10.1.45.0 0.0.0.255
access-list 1 deny 10.1.56.0 0.0.0.255
access-list 1 deny 10.1.67.0 0.0.0.255
access-list 1 permit any
access-list 2 permit any

**LDP inbound filtering example:**

Let's say on R1 for the prefix 192.1.4.4.

```
mpls ldp neighbor 192.1.5.5 labels accept 1
access-list 1 permit 192.1.5.5
```

**Disabling MPLS TTL propagation**

MPLS routers copy the TTL of an IP packet when it enters a label-switched path (LSP), such that an IP packet with a TTL of 255 receives an MPLS label with a TTL of 255. By default, IOS routers will decrement the MPLS TTL of an MPLS-encapsulated packet in place of the IP TTL, at every label-switched hop. Cisco calls this behavior *TTL propagation*.

Let's test the effect of ttl propagation according to LDP Lab we used in pervious pages.

If I traceroute 5.5.5.5 I will notice that traceroute exposes all the links within provider network

```
 1 10.1.67.6 32 msec 24 msec 24 msec
 2 10.1.56.5 [MPLS: Label 16 Exp 0] 164 msec 132 msec 172 msec
 3 10.1.35.3 [MPLS: Label 16 Exp 0] 88 msec 96 msec 84 msec
 4 10.1.23.2 [MPLS: Label 16 Exp 0] 64 msec 60 msec 60 msec
 5 10.1.12.1 88 msec * 104 msec
```

Cisco IOS provides the option to disable MPLS TTL propagation, with the `no mpls ip propagate-ttl` command under global configuration. If applied, this command should be applied to all routers in the MPLS domain.

With TTL propagation disabled, the MPLS TTL is calculated independent of the IP TTL, and the IP TTL remains constant for the length of the LSP. Because the MPLS TTL never drops to zero, none of the LSP hops trigger an ICMP TTL exceeded message and consequently these hops are not recorded in the traceroute.
R7, R1 (non MPLS enabled Router)

R7# tracert 5.5.5.5

Type escape sequence to abort.
Tracing the route to 5.5.5.5

1 10.1.17.1 52 msec 44 msec 8 msec
2 10.1.12.2 [MPLS: Label 205 Exp 0] 132 msec 140 msec 180 msec
3 10.1.23.3 [MPLS: Label 300 Exp 0] 128 msec 120 msec 152 msec
4 10.1.34.4 [MPLS: Label 400 Exp 0] 136 msec 140 msec 152 msec
5 10.1.45.5 156 msec 112 msec 128 msec

R7#

R1# tracert 5.5.5.5

Type escape sequence to abort.
Tracing the route to 5.5.5.5

1 10.1.12.2 [MPLS: Label 205 Exp 0] 136 msec 140 msec 116 msec
2 10.1.23.3 [MPLS: Label 300 Exp 0] 100 msec 116 msec 108 msec
3 10.1.34.4 [MPLS: Label 400 Exp 0] 348 msec 372 msec 400 msec
4 10.1.45.5 300 msec 528 msec 284 msec

PE’s (MPLS enabled Routers)

R1(config)# no mpls ip propagate-ttl
R5(config)# no mpls ip propagate-ttl

R7# tracert 5.5.5.5

Type escape sequence to abort.
Tracing the route to 5.5.5.5

1 10.1.17.1 32 msec 52 msec 32 msec
2 10.1.45.5 160 msec 148 msec 148 msec

R7#

R1# tracert 5.5.5.5

Type escape sequence to abort.
Tracing the route to 5.5.5.5

1 10.1.45.5 124 msec 96 msec 108 msec

R1#

Command can be end with (forwarded) or (local)

Local will make network hidden when using traceroute from internal users only (users inside MPLS Cloud)

Forwarded will make network hidden when using traceroute from external users only (users outside MPLS Cloud)
If we want local routers only not see the trace

R1(config)#no mpls ip propagate-ttl local
R5(config)#no mpls ip propagate-ttl local

R1#traceroute 5.5.5.5
Type escape sequence to abort.
Tracing the route to 5.5.5.5

1 10.1.45.5 116 msec 132 msec 100 msec
R1#

BUT

R7#traceroute 5.5.5.5
Type escape sequence to abort.
Tracing the route to 5.5.5.5

1 10.1.17.1 32 msec 32 msec 28 msec
2 10.1.12.2 [MPLS: Label 205 Exp 0] 120 msec 140 msec 148 msec
3 10.1.23.3 [MPLS: Label 300 Exp 0] 152 msec 152 msec 144 msec
4 10.1.34.4 [MPLS: Label 400 Exp 0] 140 msec 152 msec 132 msec
5 10.1.45.5 156 msec 172 msec 136 msec
R7#

If we type
R1(config)#no mpls ip propagate-ttl forwarded
R5(config)#no mpls ip propagate-ttl forwarded

R1#traceroute 5.5.5.5
Type escape sequence to abort.
Tracing the route to 5.5.5.5

1 10.1.12.2 [MPLS: Label 205 Exp 0] 416 msec 432 msec 224 msec
2 10.1.23.3 [MPLS: Label 300 Exp 0] 348 msec 360 msec 304 msec
3 10.1.34.4 [MPLS: Label 400 Exp 0] 264 msec 340 msec 254 msec
4 10.1.45.5 140 msec 104 msec 92 msec
R7#

R7#traceroute 5.5.5.5
Type escape sequence to abort.
Tracing the route to 5.5.5.5

1 10.1.17.1 64 msec 24 msec 28 msec
2 10.1.45.5 156 msec 156 msec 148 msec
R7#
**OSPF & LDP (LDP Autoconfig , MPLS LDP-IGP Synchronization )**

Two things I would like to talk about here

**First** LDP Autoconfig all OSPF enabled interfaces or just those in a given area participate in MPLS.

```
router ospf 1
mpls ldp autoconfig area 0
```

This command will let LDP run on all interfaces belong to ospf area 0

**Second** MPLS LDP-IGP Synchronization issue

A problem with MPLS networks is that LDP and the IGP of the network are not synchronized. Synchronization means that the packet forwarding out of an interface happens only if both the IGP and LDP agree that this is the outgoing link to be used.

When the LDP session is broken on a link, the IGP still has that link as outgoing , This is not a big problem for networks that are running IPv4-over-MPLS only. However, this is a problem for more than just the IPv4-over-MPLS case. With MPLS VPN, AToM, Virtual Private LAN Switching (VPLS), or IPv6 over MPLS, the packets must not become unlabeled in the MPLS network. If they do become unlabeled, the LSR does not have the intelligence to forward the packets anymore and drops them.

The same problem can occur when LSRs restart. The IGP can be quicker in establishing the adjacencies than LDP can establish its sessions. This means that the IGP forwarding is already happening before the LFIB has the necessary information to start the correct label forwarding. The packets are incorrectly forwarded (unlabeled) or dropped until the LDP session is established.

The solution is MPLS LDP-IGP Synchronization. This feature ensures that the link is not used to forward (unlabeled) traffic when the LDP session across the link is down. Till today the only IGP that is supported with MPLS LDP-IGP Synchronization is OSPF.

**How MPLS LDP-IGP Synchronization Works?**

When the MPLS LDP-IGP synchronization is active for an interface, the IGP announces that link with maximum metric until the synchronization is achieved, or until the LDP session is running across that interface. The maximum link metric for OSPF is 65536 (hex 0xFFFF). No path through the interface where LDP is down is used unless it is the only path. (No other paths have a better metric.) After the LDP session is established and label bindings have been exchanged, the IGP advertises the link with its normal IGP metric. At that point, the traffic is label-switched across that interface. Basically, OSPF does not form an adjacency across a link if the LDP session is not established first across that link. (OSPF does not send out Hellos on the link.) Until the LDP session is established or until the synchronization Holddown timer has expired, the OSPF adjacency is not established.

```
Router ospf 1
Mpls ldp sync
```

Also We can enable or disable it under interface:

```
Int f0/0
No mpls ldp igp sync
```
To prevent OSPF from waiting indefinitely for LDP to come up, you can configure a Holdown Timer.

By default, if synchronization is not achieved, the IGP waits indefinitely to bring up the adjacency. You can change this with the global command `mpls ldp igp sync holddown msecs` which instructs the IGP to wait only for the configured time. After the synchronization Holdown timer expires, the IGP forms an adjacency across the link.

```
Router ospf 1
Mpls ldp sync
mpls ldp igp sync holddown 30000
show ip ospf mpls ldp interface serial 4/0
```

**MPLS LDP Session Protection**
A common problem in networks is flapping links which do have an important impact on the convergence of the network. Because the IGP adjacency and the LDP session are running across the link, they go down when the link goes down.

When the LDP session between two directly connected LSRs is protected, a targeted LDP session is built between the two LSRs. When the directly connected link does go down between the two LSRs, the targeted LDP session is kept up as long as an alternative path exists between the two LSRs. The LDP link adjacency is removed when the link goes down, but the targeted adjacency keeps the LDP session up. When the link comes back up, the LSR does not need to re-establish the LDP session.

```
mpls ldp session protection[vrf vpn-name] [for acl] [duration seconds]
```

The access list (acl) you can configure lets you specify the LDP peers that should be protected. It should hold the LDP Router Identifier of the LDP neighbors that need protection.

The duration is the time that the protection (the targeted LDP session) should remain in place after the LDP link adjacency has gone down. The default value is infinite.

If you issue the `mpls ldp session protection` command without the `duration` keyword, then session protection is enabled for 86400 seconds (24 hours) meaning that the LDP targeted hello adjacency is retained for 24 hours after a link is lost. This is the default timeout.

For the protection to work, you need to enable it on both the LSRs. If this is not possible, you can enable it on one LSR, and the other LSR can accept the targeted LDP Hellos by configuring the command `mpls ldp discovery targeted-hello accept`.

**LDP Transport Address**
LDP advertises its LDP Router ID as the transport address in LDP Discovery Hello messages sent from the interface.

The `mpls ldp discovery transport-address` command provides the means to modify the default behavior, which is useful with ATM interfaces.

```
Router(config)# interface pos2/0
Router(config-if)# mpls ldp discovery transport-address interface
Router(config)# interface pos3/1
Router(config-if)# mpls ldp discovery transport-address 145.22.0.56
```
The LDP session is a TCP connection that is established between two IP addresses of the LSRs. Usually these IP addresses are used to create the LDP router Identifier on each router. However, if you do not want to use this IP address to create the LDP session, you can change it. To change the IP address, configure the command `mpls ldp discovery transport-address {interface | ip-address}` on the interface of the router and specify an interface or IP address to be used to create the LDP session. This transport IP address is advertised in the LDP Hellos that are sent on the LDP-enabled interfaces.

**MPLS LDP Graceful Restart Feature**

**Before we go to understand MPLS VPN, we need to understand the concept of VRF & VRF Lite**

**VRF Lite (Multi-VRF CE)**

Virtual Routing and Forwarding (VRF) is a technology that allows multiple instances of a routing table to co-exist within the same router at the same time.

Assume I have R3 and I would like to have two routing tables one for each neighbor, in this case I will create a virtual routing table for each R1 & R2 and I will give each vrf a name, in the end I will have three tables:
- RIB general routing table we already familiar with and use `sh ip route` to display it
- VRF AS101 this virtual routing to use with R1
- VRF AS102 this virtual routing to use with R2

I will still have problem if I need one of R1 routes reachable by R2 or vice versa. And to solve this we use what we called VRF Route Leaking
Let's configure our routers to see exactly what I mean, *remember VRF LITE can be configured alone without MPLS*, but when implement MPLS VPN we will need to use the concept of vrf.

**R1**
```
ip vrf AS101
rd 1:1

interface Loopback100
ip vrf forwarding AS101
ip address 100.1.1.1 255.255.255.255

interface FastEthernet0/0
ip vrf forwarding AS101
ip address 10.10.101.6 255.255.255.0

router ospf 101 vrf AS101
log-adjacency-changes
network 0.0.0.0 255.255.255.255 area 0
```

**R2**
```
ip vrf AS102
rd 2:2

interface Loopback100
ip vrf forwarding AS102
ip address 100.2.2.2 255.255.255.255

interface FastEthernet0/1
ip vrf forwarding AS102
ip address 10.10.102.7 255.255.255.0

router ospf 102 vrf AS102
log-adjacency-changes
network 0.0.0.0 255.255.255.255 area 0
```

**R3**
```
ip vrf AS101
rd 1:1

ip vrf AS102
rd 2:2

interface FastEthernet0/0
ip vrf forwarding AS101
ip address 10.10.101.8 255.255.255.0

interface FastEthernet0/1
ip vrf forwarding AS102
ip address 10.10.102.8 255.255.255.0

router ospf 101 vrf AS101
log-adjacency-changes
network 0.0.0.0 255.255.255.255 area 0
default-information originate always
!
router ospf 102 vrf AS102
log-adjacency-changes
network 0.0.0.0 255.255.255.255 area 0
default-information originate always

ip route 10.10.101.6 255.255.255.255 f0/0
ip route 10.10.102.7 255.255.255.255 f0/1
ip route vrf AS101 100.2.2.2 255.255.255.255 10.10.102.7 global
ip route vrf AS102 100.1.1.1 255.255.255.255 10.10.101.6 global

R1#ping vrf AS101 100.2.2.2 source loopback 100

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 100.2.2.2, timeout is 2 seconds:
Packet sent with a source address of 100.1.1.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/62/68 ms

OR Don’t use static routes to do route leaking and lets use route-target with MP-BGP but we will talk about this option & also about route leaking later in this guide.
What is MPLS VPN?
Let’s check our first topology

VPN Tunnel Created between PE’s to send & receive customers vrf routes
So MPLS VPN mean providing connections between different company sites using SP infra
SP infra routers will communicate with each other’s using VPN to separate each customer traffic.

- In the MPLS VPN implementation, both P and PE routers run MPLS.
- A CE router does not need to run MPLS.
- A CE router has a direct Layer 3 connection with the PE router. and Because the CE and PE routers interact at Layer 3, they must run a routing protocol (or static routing) between them.
- All P and PE routers must have the complete routing table of every customer. so every P and PE router has a private routing table for each customer. Several processes of one routing protocol (one process per VPN) could be running on all the routers to distribute the VPN routes.

In the above topology SP PE1 will be connected to two customers (CBTME, TRAININGHOUSE) And he supposed to communicate to PE2 so he can send any traffic from CBTME1 to CBTME2

PE1, PE2 had one Routing table (Global) and he wants to separate routes coming from Different customers

Where MPLS LDP will be configured?
Configure Label Distribution Protocol (LDP) between all P and PE routers so that all IP traffic is label-switched between them.
**How PE1 will recognize & separate Different CE routes?**

By using virtual routing table called VRF for each customer, even separate CEF will work for each VRF table. A virtual routing/forwarding (VRF) is a VPN routing and forwarding instance. Because the routing should be separate and private for each customer (VPN) on a PE router, each VPN should have its own routing table. This private routing table is called the VRF routing table.

You create the VRF on the PE router with the `ip vrf` command. You use the `ip vrf forwarding` command to assign PE-CE interfaces on the PE router to a VRF. You can assign an interface to only one VRF, but you can assign several interfaces to the same VRF. The PE router then automatically creates a VRF routing table and CEF table.

The routing table as we used to know is RIB, from this point will now be referred to as the **global** or the default routing table.

**Commands for creating vrf table and assign it to interface for IPv4, IPv6**

IP VRF <Case Sensitive NAME>

Example:

```
ip vrf ce1
int s0/0
ip vrf forwarding ce1
```

*Note: when we type `ip vrf forwarding ce1` under serial 0/0 interface we will need to retype again interface ip address*

To create vrf for ipv4 and ipv6 we use different way we use the following commands instead.

```
vrf definition vrf1
  !
  address-family ipv4
  !
  address-family ipv6

int s0/0
vrf forwarding ce1
```

Each customer sites use Private Ip address range, how PE will solve any ipv4 address overlapping between different customers?

By using **RD (Route Distinguisher)** to create VPNv4 Prefix to make routes unique in MP-BGP, Doesn’t have to be same on the sites of same customer

`vpnv4 prefix = ipv4 prefix 32bit + RD 64bit`

RD Written on one of two ways:

- **ASN:ID**  ex: 65000:1
- **IP address: ID**  ex: 10.1.1.1:1
- **RD** = 65000:1:10.0.0.0/8
An RD is a 64-bit field used to make the VRF prefixes unique when MP-BGP carries them. Each VRF instance on the PE router must have one RD assigned to it. This 64-bit value can have two formats: ASN:nn or IP-address:nn, where nn represents a number. The most commonly used format is ASN:nn, where ASN stands for autonomous system number.

The combination of the RD with the IPv4 prefix provides a vpnv4 prefix, of which the address is 96 bits long. The mask is 32 bits long, just as it is for an IPv4 prefix. If you take an IPv4 prefix 10.1.1.0/24 and an RD 1:1, the vpnv4 prefix becomes 1:1:10.1.1.0/24. One customer might use different RDs for the same IPv4 route.

**How PE will communicate with CE and getting routes from CE?**
We can use Static routes, IGP or eBGP between CE & PE to perform that.
OSPF can run multi process but other routing protocols run in one process only so this Single process MUST divide to two several instance (routing contexts) each instance has its own settings and own commands.

**How PE1 & PE2 will send customers routes to each other?**
By using MP-BGP which will create VPNv4 tunnel between PE1 & PE2 to send and receive VPNv4 prefixes and this will need additional MPLS label called VPN label to distinguish each.

The combination of the RD with the IPv4 prefix makes up the vpnv4 prefix. It is this vpnv4 prefix that iBGP needs to carry between the PE routers.

To support the Multiprotocol behavior of BGP in Cisco IOS, the BGP routing process has the Concept of address families. The four address families that are currently supported are IPv4, IPv6, vpnv4 (VPN-IPv4), and vpnv6 (VPN-IPv6). The subsequent address families that you can specify are unicast, multicast, and VRF.

You use the address family vpnv4 under the router bgp process to configure the vpnv4 BGP sessions and parameters, which the PE routers need.
You use the address family ipv4 vrf vrf-name under the router bgp process on the PE routers to configure the BGP sessions and parameters toward the CE routers, across the VRF interfaces.

**But How does the egress PE router know which VRF the packet belongs to?** This information is not in the IP header, and it cannot be derived from the IGP label, because this is used solely to forward the packet through the service provider network. The solution is to add another label in the MPLS label stack. This label indicates which VRF the packet belongs to. Therefore, all customer packets are forwarded with two labels: the IGP label as the top label and the VPN label as the bottom label. The VPN label must be put on by the ingress PE router to indicate to the egress PE router which VRF the packet belongs to. How does the egress PE router signal to the ingress PE router which label to use for a VRF prefix? Because MP-BGP is already used to advertise the vpnv4 prefix, it also signals the VPN label (also referred to as the BGP label) that is associated with the vpnv4 prefix.
How PE1 & PE2 MP-BGP will understand which routes belong to each customer? and how we will import or export these routes?

By using **RT (Route Target)**, will tell PE in which vrf this route will be put inside.

RT is just for VPNv4 table to know which routes belong to which customer

RT can be different from PE to another but most important is import /export the right RT

Simply, RT’s indicates to the PE routers if the route should be imported into a VRF

**Difference between route distinguisher and route target:**

To conclude, the route distinguisher and route target values perform two completely separate functions, and although in a lot of cisco press publications the values are the same (which they can be) it is confusing to someone learning MPLS for the first time as they assume they do the same thing.

The route distinguisher makes a unique VPNv4 address across the MPLS network

The route target defines which prefixes get imported and exported on the PE routers.

An RT is a BGP extended community (optional transitive attribute that is described in RFC 1997 ) that indicates which routes should be imported from MP-BGP into the VRF. Exporting an RT means that the exported vpnv4 route receives an additional BGP extended community—this is the RT

The command to configure RTs for a VRF is `route-target {import | export | both} route-target-ext-community`. The keyword both indicates both import and export.

The number of routes leaked from one VRF to another can be limited by configuring an import or export map under `ip vrf`, which uses a route map to further filter routes.

**Remember**, if we want to make sites for one customer talk to each other, we call this concept **Intranet VPN**. If we want to make sites for one customer talk to another site for another customer, we call this concept **extranet VPN**

**Configuring MP-BGP**

Only BGP extended communities are sent by default to the vpnv4 neighbor. If you want to use standard communities, too, please specify `send-community both` for the BGP neighbor.

```plaintext
Router bgp 65000
nei 10.200.254.2 remote-as 65001
nei 10.200.254.2 update-source loop0 < must be sourced from a Loopback 0 interface /32
address-family vpnv4
neighbor 10.200.254.2 activate < nei must be activated once we use address-families
neighbor 10.200.254.2 send-community both
```

BGP automatically in background create address family ipv4 for your configuration

You may disable the default behavior via the command `no bgp default ipv4-unicast`. 

The BGP, as you surely know, has a multi-protocol capability - in a single session, it is capable of carrying information about diverse routed protocols (IPv4 Unicast, IPv4 Multicast, IPv6 Unicast, IPv6 Multicast, VPNv4, CLNP), in BGP’s parlance called "address families".

Not having a neighbor listed under a particular address family means that we are not planning to exchange information from that address family with that neighbor.

For backward compatibility with older BGP versions that have not been multiprotocol-capable, the BGP implicitly assigns all defined neighbors to an invisible address-family ipv4 section. In other words, as soon as you define a neighbor, it is automatically being added to an invisible address-family ipv4 section so that you don't have to do it manually.

You can change it, however. First of all, if you enter the BGP configuration and issue the command `bgp upgrade-cli` you will find out that the BGP configuration has been fully converted to the address family style of configuration. Outside any address-family stanzas, only the basic neighbor settings are configured like their addresses, AS numbers, update sources. However, all remaining per-address-family commands will be automatically moved into address-family stanzas. The behavior or operations of BGP do not change with this new style of configuration, only the configuration format is changed.

Furthermore, if you enter the `no bgp default ipv4-unicast` command in the BGP configuration, you will prevent BGP from automatically assigning each newly defined neighbor into address-family ipv4 section. You will then be required to add every defined neighbor to each intended address family automatically - it won't be done automatically for you anymore.

To inject a particular VRF’s routes into BGP, you must activate the respective address-family under the BGP process and enable route redistribution (such as static or connected). All the respective routes belonging to that particular VRF will be injected into the BGP table with their RDs and have their VPN labels generated.

The import process is a bit more complicated and is based on the concept of “Route Targets. Routes with the same RD may eventually belong to multiple VRFs, when you share their routes.

By default, all prefixes redistributed from a VRF into a BGP process are tagged with the extended community `X:Y`specified under the 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 `route-target import X:Y`. The import process is based entirely on the route-targets, not the RDs.

**Configuring RT’s**

```
Ip vrf <NAME>
Route-target import  asn:id
Route-target export asid
```

**Other commands we may need:**

```
ip bgp-community new-format
```

is used to configure the local router to display BGP communities in the AA:NN format to conform with RFC-1997. This command only affects the format in which BGP communities are displayed; it does not affect the community or community exchange. However, expanded IP community lists that match locally
configured regular expressions may need to be updated to match on the AA:NN format instead of the 32-bit number.

RFC 1997, BGP Communities Attribute, specifies that a BGP community is made up of two parts that are each 2 bytes long. The first part is the autonomous system number and the second part is a 2-byte number defined by the network operator.

**bgp upgrade-cli**

To upgrade a Network Layer Reachability Information (NLRI) formatted router configuration file to the address-family identifier (AFI) format and set the router command-line interface (CLI) to use only AFI commands

The `bgp upgrade-cli` command is used to upgrade a router that is running in the NLRI formatted CLI to the AFI CLI format. The upgrade is automatic and does not require any further configuration by the network operator, and no configuration information is lost but you cannot return to the NLRI configuration because a no form does not exist for this command. Several NLRI-based commands do not exist under the AFI format but have equivalent commands under the AFI format.


Example:

```bash
ip bgp-community new-format
c router bgp 65000
bgp upgrade-cli
yes
no bgp default ipv4-unicast
 nei 22.22.22.22 remote-as 65000
 nei 22.22.22.22 update-source loop0
 add vpnv4
 nei 22.22.22.22 act
 nei 22.22.22.22 send-community both

sh ip bgp summ = sh bgp ipv4 uni sum

sh bgp vpnv4 uni all sum
```
Each Customer may use Same Private IPv4 address used by Another Customer, so we add RD 64 bit to 32 bit ipv4 address

**RT** is BGP extended communities
Will be used to import export Customer VRF routes Between PE’s

Configure Redistribution between MP-BGP and the method used to connect PE to CE (Static routes, IGP):

MP-BGP & PE-CE routing protocol must Redistribute mutual (in both direction)
But Remember if PE-CE routing protocol is EBGP so no need for Redistribution
**Configuring PE-CE connectivity**

**Connected Routes**

Strictly speaking, the connected routes are not a routing protocol. However, to ensure connectivity, it is best practice to redistribute the connected routes on the PE router into BGP. That way, when the user launches a ping from a CE router to the remote CE router, the return packet is routed back. By default, if the user sends a ping and does not specify the source IP address, it takes as the source IP address the IP address of the outgoing interface, which in the case of a CE router is an IP address from the subnet on the PE-CE link. As such, the return packet has this IP address as the destination IP address. Thus, this prefix must be known on the remote sites for the ping to succeed. You can choose not to distribute the connected subnets into BGP, but then you have to launch a ping from CE to CE by specifying a different source IP address on the CE router. Then you must include this IP address in the specific PE-CE routing protocol!

```
address-family ipv4 vrf cust-one
redistribute connected
neighbor 10.10.2.1 remote
```

**Static Routing**

```
ip route vrf cust-one 10.88.1.1 255.255.255.255 10.10.2.1
```

**RIPv2**

In Cisco IOS, RIPv2 is supported as a PE-CE routing protocol, but RIP version 1 is not. You can see the basic RIPv2 VRF configuration on a PE router in Example 7-24. Only one RIPv2 process exists on the PE router. The specific configuration needed per VRF is configured under the specific address family. Make sure the default-metric command is configured for RIP. Otherwise, no routes are distributed from BGP to RIP.

**OSPF**

All OSPF routes become external routes on the remote PE when the routes are redistributed back into OSPF. The result of this would be that all OSPF routes that transverse the MPLS VPN backbone would be less preferable than the routes that did not transverse the backbone but were sent via an intersite link (backdoor link) from one OSPF site to another.

To run OSPF for a VRF, you configure the OSPF process command with the VRF keyword. The syntax is router ospf process-id vrf vrf-name. Note that RIPv2 and EIGRP have only one routing process with an address family per VRF configured. OSPF has one separate OSPF process per VRF.

```
Make sure you have the subnets keyword on the redistribute bgp Command under the router ospf process. Otherwise, only classful routes are redistributed. When you are redistributing OSPF into BGP, make sure to configure the appropriate match parameters on the redistribute command so that you can redistribute the proper OSPF type of routes.
```

**EIGRP**

When running EIGRP between PE & CE we need to consider Backdoor issues.
EIGRP PE-CE with Backdoor Links soo
when a route disappears, routing can take longer to reconverge, To help speed up the reconverging, you can use Site-of-Origin (SOO) for EIGRP. It can be defined on the PE routers on the VRF interfaces toward the CE routers and on the routers with a backdoor link.
When the router receives a route across the interface with this route map configured and the SOO of the route matches the configured SOO, the router rejects the route. When the PE router receives a vpnv4 update with the SOO set, it extracts the SOO and adds it to the EIGRP route when it is reconstructed.

The disadvantage of using the SOO for EIGRP on the PE and backdoor routers is that one part of the site cannot reach the other part of the site across the backdoor link and the MPLS VPN backbone if the site is split. The backdoor router or the PE router blocks the route that is needed to get to the other part of the site. To work around this problem, you can configure the sitemap for SOO only on the PE routers and not the backdoor routers.

EBGP
We will talk about in Lab next few pages.

Steps to configure MPLS VPN
- Configure IGP between PE’s & P’s
- Configure MPLS LDP between PE’s & P’s
- Configure VRF, RD, RT and assign VRF to PE interface facing CE
- Configure MP-BGP between PE’s
- Configure Static Route, IGP or BGP between PE’s & CE’s
- Configure Redistribution between MP-BGP and the method used to connect PE to CE (Static routes, IGP)
MPLS VPN Labs
As you can see below I have two companies (Cbtme & Traininghouse) and with MPLS Layer3 VPN I can connect each company sites to each other’s

Steps to configure MPLS L3 VPN
- Configure IGP between PE’s & P’s
- Configure MPLS LDP between PE’s & P’s
- Configure VRF, RD, RT and assign VRF to PE interface facing CE
- Configure MP-BGP between PE’s
- Configure Static Route, IGP or BGP between PE’s & CE’s
- Configure Redistribution between MP-BGP and the method used to connect PE to CE (Static routes, IGP, BGP)

Configure IGP between PE’s & P’s
We will run OSPF between P’s & PE’s (we can use static routes as well but this is not common)

P1
```
router ospf 1
router-id 1.1.1.1
log-adjacency-changes
network 1.1.1.1 0.0.0.0 area 0
network 12.12.12.0 0.0.0.255 area 0
network 100.100.100.0 0.0.0.255 area 0
```
P2
router ospf 1
router-id 2.2.2.2
log-adjacency-changes
network 2.2.2.2 0.0.0.0 area 0
network 12.12.12.0 0.0.0.255 area 0
network 200.200.200.0 0.0.0.255 area 0

PE1
router ospf 1
router-id 11.11.11.11
log-adjacency-changes
network 11.11.11.11 0.0.0.0 area 0
network 100.100.100.0 0.0.0.255 area 0

PE2
router ospf 1
router-id 22.22.22.22
log-adjacency-changes
network 22.22.22.22 0.0.0.0 area 0
network 200.200.200.0 0.0.0.255 area 0

Configure MPLS LDP between PE’s & P’s

P1
mpls label protocol ldp
mpls ldp router-id Loopback0 force
int f0/0
mpls ip
int f0/1
mpls ip

P2
mpls label protocol ldp
mpls ldp router-id Loopback0 force
int f0/0
mpls ip
int f0/1
mpls ip
PE1
mpls label protocol ldp
mpls ldp router-id Loopback0 force
int f0/0
mpls ip

PE2
mpls label protocol ldp
mpls ldp router-id Loopback0 force
int f0/0
mpls ip

Configure VRF, RD, RT and assign VRF to PE interface facing CE

According to the following table:

<table>
<thead>
<tr>
<th>Router</th>
<th>VRF name</th>
<th>RD</th>
<th>RT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE1</td>
<td>CBTME</td>
<td>1:1</td>
<td>1:1</td>
</tr>
<tr>
<td>PE1</td>
<td>TRAININGHOUSE</td>
<td>2:2</td>
<td>2:2</td>
</tr>
<tr>
<td>PE2</td>
<td>CBTME</td>
<td>1:1</td>
<td>1:1</td>
</tr>
<tr>
<td>PE2</td>
<td>TRAININGHOUSE</td>
<td>2:2</td>
<td>2:2</td>
</tr>
</tbody>
</table>

PE1
interface Serial0/0
ip vrf forwarding CBTME

interface Serial0/1
ip vrf forwarding TRAININGHOUSE

ip vrf CBTME
    rd 1:1
    route-target export 1:1
    route-target import 1:1
!
ip vrf TRAININGHOUSE
    rd 2:2
    route-target export 2:2
    route-target import 2:2
**PE2**
interface Serial0/0
ip vrf forwarding CBTME

interface Serial0/1
ip vrf forwarding TRAININGHOUSE

ip vrf CBTME
  rd 1:1
  route-target export 1:1
  route-target import 1:1

ip vrf TRAININGHOUSE
  rd 2:2
  route-target export 2:2
  route-target import 2:2

*(Once you assign interface to VRF, this interface ip address will be part of VRF routing table not the global routing table any more.)*

**Configure MP-BGP between PE’s (Creating VPNV4 tunnel)**

**PE1**
router bgp 65000
  no synchronization
  bgp log-neighbor-changes
  neighbor 22.22.22.22 remote-as 65000
  neighbor 22.22.22.22 update-source Loopback0
  no auto-summary

  address-family vpnv4
  neighbor 22.22.22.22 activate
  neighbor 22.22.22.22 send-community extended
  exit-address-family
PE2
router bgp 65000
no synchronization
bgp log-neighbor-changes
neighbor 11.11.11.11 remote-as 65000
neighbor 11.11.11.11 update-source Loopback0
no auto-summary
|
address-family vpnv4
neighbor 11.11.11.11 activate
neighbor 11.11.11.11 send-community extended
exit-address-family

From This Point we can have more than one scenario to configure PE-CE connectivity:

Configure Static Route, IGP or BGP between PE’s & CE’s: **Static Route**

(Our objectives here connect CBTME site1 with site 2 only)

PE1
ip route vrf CBTME 40.40.40.41 255.255.255.255 10.1.1.2
ip route vrf CBTME 40.40.40.42 255.255.255.255 10.3.3.2

PE2
ip route vrf CBTME 40.40.40.41 255.255.255.255 10.1.1.2
ip route vrf CBTME 40.40.40.42 255.255.255.255 10.3.3.2

---

Add 40.40.40.41 to routing table vrf CBTME using next hop 10.1.1.2 (CBTME 1)
Configure Redistribution between MP-BGP and Static Route

PE1
router bgp 65000
address-family ipv4 vrf CBTME
    redistribute connected <redistribute all connected routes in MP-BGP vrf CBTME
    redistribute static <redistribute all static routes in MP-BGP vrf CBTME
    no synchronization
    exit-address-family

PE2
address-family ipv4 vrf CBTME
    redistribute connected
    redistribute static
    no synchronization
    exit-address-family

Nothing to redistribute here on other direction since we use static route not a routing protocol, only a default route point to PE will be needed in CE.

CBTME 1
ip route 0.0.0.0 0.0.0.0 10.1.1.1

CBTME 2
ip route 0.0.0.0 0.0.0.0 10.3.3.1
Configure Static Route, IGP or BGP between PE’s & CE’s: **RIP**
(Our objectives here connect CBTME site1 with site 2 only)

**PE1**
```
router rip
version 2
no auto-summary

address-family ipv4 vrf CBTME
  network 10.0.0.0
  no auto-summary
  version 2
  exit-address-family
```

**PE2**
```
router rip
version 2
no auto-summary

address-family ipv4 vrf CBTME
  network 10.0.0.0
  no auto-summary
  version 2
  exit-address-family
```

**CBTME 1**
```
router rip
version 2
network 10.0.0.0
network 40.0.0.0
no auto-summary
```

RIP, EIGRP & BGP run as one process on any router, since we would connect more than one customer to this router such as TRAININGHOUSE, we will need to create address-family for each customer (Context), in this case we have one customer only so we will need one address-family for CBTME under Main single RIP process.
CBTME 2
router rip
version 2
network 10.0.0.0
network 40.0.0.0
no auto-summary

**Configure Redistribution between MP-BGP and RIP**

PE1
router bgp 65000
address-family ipv4 vrf CBTME
  redistribute rip
    no synchronization
    exit-address-family

router rip
address-family ipv4 vrf CBTME
  redistribute bgp 65000 metric 5

PE2
router bgp 65000
address-family ipv4 vrf CBTME
  redistribute rip
    no synchronization
    exit-address-family

router rip
address-family ipv4 vrf CBTME
  redistribute bgp 65000 metric 5

*Note in both PE1 & PE2 we can redistribute connected as well :*
router bgp 65000
address-family ipv4 vrf CBTME
redistribute connected
Configure Static Route, IGP or BGP between PE’s & CE’s: **OSPF**

(Our objectives here connect CBTME site1 with site 2, connect TRAININGHOUSE site1 with site 2)

For simplicity will advertise ospf under interfaces instead of using network command, you still can use network command under each ospf process

PE1

- `router ospf 2 vrf CBTME`
- `router-id 10.1.1.1`
- `log-adjacency-changes`

- `router ospf 3 vrf TRAININGHOUSE`
- `router-id 10.2.2.1`
- `log-adjacency-changes`
- `int s0/1`
- `ip ospf 3 area 0`
- `int s0/0`
- `ip ospf 2 area 0`

PE2

- `router ospf 2 vrf CBTME`
- `router-id 10.3.3.1`
- `log-adjacency-changes`

- `router ospf 3 vrf TRAININGHOUSE`
- `router-id 10.4.4.1`
- `log-adjacency-changes`
- `int s0/1`
- `ip ospf 3 area 0`
int s0/0
ip ospf 2 area 0

**CBTME 1**
router ospf 1
  router-id 10.1.1.2
  log-adjacency-changes
  int loop 0
  ip ospf 1 area 0
  int s0/0
  ip ospf 1 area 0

**CBTME 2**
router ospf 1
  router-id 10.3.3.2
  log-adjacency-changes
  int loop 0
  ip ospf 1 area 0
  int s0/0
  ip ospf 1 area 0

**TRAININGHOUSE 1**
router ospf 1
  router-id 10.2.2.2
  log-adjacency-changes
  int loop 0
  ip ospf 1 area 0
  int s0/1
  ip ospf 1 area 0

**TRAININGHOUSE 2**
router ospf 1
  router-id 10.4.4.2
  log-adjacency-changes
  int loop 0
  ip ospf 1 area 0
  int s0/1
  ip ospf 1 area 0

**Configure Redistribution between MP-BGP and OSPF**

**PE1**
routerr bgp 65000
  address-family ipv4 vrf TRAININGHOUSE
  redistribute ospf 3 vrf TRAININGHOUSE
  no synchronization
  exit-address-family

address-family ipv4 vrf CBTME
redistribute ospf 2 vrf CBTME
no synchronization
exit-address-family

router ospf 2 vrf CBTME
redistribute bgp 65000 subnets

router ospf 3 vrf TRAININGHOUSE
redistribute bgp 65000 subnets

PE2
router bgp 65000
address-family ipv4 vrf TRAININGHOUSE
redistribute ospf 3 vrf TRAININGHOUSE
no synchronization
exit-address-family

! address-family ipv4 vrf CBTME
redistribute ospf 2 vrf CBTME
no synchronization
exit-address-family

router ospf 2 vrf CBTME
redistribute bgp 65000 subnets

router ospf 3 vrf TRAININGHOUSE
redistribute bgp 65000 subnets

PE1#sh ip bgp VPNV4 vrf TRAININGHOUSE
BGP table version is 17, local router ID is 11.11.11.11
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
    r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

<table>
<thead>
<tr>
<th>Network</th>
<th>Next Hop</th>
<th>Metric</th>
<th>LocPrf</th>
<th>Weight</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Route Distinguisher: 2:2 (default for vrf TRAININGHOUSE)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>* &gt; 10.2.2.0/24</td>
<td>0.0.0.0</td>
<td>0</td>
<td>32768</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>* &gt; 10.4.4.0/24</td>
<td>22.22.22.22</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>* &gt; 50.50.50.51/32</td>
<td>10.2.2.2</td>
<td>65</td>
<td>32768</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>* &gt; 50.50.50.52/32</td>
<td>22.22.22.22</td>
<td>65</td>
<td>100</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

PE2#sh ip bgp vpnv4 vrf TRAININGHOUSE
BGP table version is 31, local router ID is 22.22.22.22
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
    r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
Notice if I remove exporting RT 2:2 in PE2

PE2
ip vrf TRAININGHOUSE
rd 2:2
no route-target export 2:2
route-target import 2:2

clear ip bgp *

I will lose these routes in PE1

PE1#sh ip bgp vpnv4 vrf TRAININGHOUSE
BGP routing table entry for 1:1:40.40.40.42/32, version 13
Paths: (1 available, best #1, table CBTME)
Flag: 0x820
Not advertised to any peer
Local
22.22.22.22 (metric 31) from 22.22.22.22 (22.22.22.22)
Origin incomplete, metric 65, localpref 100, valid, internal, best
Extended Community: RT:1:1 OSPF DOMAIN ID:0x0005:0x000000020200
OSPF RT:0.0.0.0:2:0 OSPF ROUTER ID:10.3.3.1:0
mpls labels in/out nolabel/22
PE2#sh ip bgp vpnv4 vrf CBTME 40.40.40.41
BGP routing table entry for 1:1:40.40.40.41/32, version 12
Paths: (1 available, best #1, table CBTME)
Flag: 0x820
Not advertised to any peer
Local
  11.11.11.11 (metric 31) from 11.11.11.11 (11.11.11.11)
  Origin incomplete, metric 65, localpref 100, valid, internal, best
Extended Community: RT:1:1 OSPF DOMAIN ID:0x0005:0x000000020200
  OSPF RT:0.0.0.0:2:0 OSPF ROUTER ID:10.1.1.1:0
  mpls labels in/out nolabel/22

CBTME1#sh ip route ospf
  40.0.0.0/32 is subnetted, 2 subnets
  O IA 40.40.40.42 [110/129] via 10.1.1.1, 00:30:13, Serial0/0
  10.0.0.0/24 is subnetted, 2 subnets
  O IA 10.3.3.0 [110/65] via 10.1.1.1, 00:30:13, Serial0/0

NOW Notice if we used
  router ospf 2 vrf CBTME in PE2
  router ospf 22 vrf CBTME in PE1

PE2#sh ip bgp vpnv4 vrf CBTME 40.40.40.41
BGP routing table entry for 1:1:40.40.40.41/32, version 12
Paths: (1 available, best #1, table CBTME)
Not advertised to any peer
Local
  11.11.11.11 (metric 31) from 11.11.11.11 (11.11.11.11)
  Origin incomplete, metric 65, localpref 100, valid, internal, best
Extended Community: RT:1:1 OSPF DOMAIN ID:0x0005:0x000000020200
  OSPF RT:0.0.0.0:2:0 OSPF ROUTER ID:10.1.1.1:0
  mpls labels in/out nolabel/22

CBTME2#sh ip route 40.40.40.41
Routing entry for 40.40.40.41/32
  Known via "ospf 1", distance 110, metric 65
  Tag Complete, Path Length == 1, AS 65000, type extern 2, forward metric 64
  Last update from 10.3.3.1 on Serial0/0, 00:11:01 ago
  Routing Descriptor Blocks:
  * 10.3.3.1, from 10.3.3.1, 00:11:01 ago, via Serial0/0
    Route metric is 65, traffic share count is 1
    Route tag 3489725928

CBTME2#sh ip route ospf
  40.0.0.0/32 is subnetted, 2 subnets
  O E2 40.40.40.41 [110/65] via 10.3.3.1, 00:11:31, Serial0/0
  10.0.0.0/24 is subnetted, 2 subnets
  O E2 10.1.1.0 [110/1] via 10.3.3.1, 00:11:31, Serial0/0

When we used different OSPF process, Domain id will not be the same and routes will be considered external O E2 instead of internal O IA
PE1#sh ip bgp vpnv4 vrf CBTME 40.40.40.42

Extended Community: RT:1:1 **OSPF DOMAIN ID:0x0005:0x000000020200**
OSPF RT:0.0.0.0:2:0 OSPF ROUTER ID:10.3.3.1:0

PE2#sh ip bgp vpnv4 vrf CBTME 40.40.40.41

Extended Community: RT:1:1 **OSPF DOMAIN ID:0x0005:0x00000160200**
**OSPF RT:0.0.0.0:2:0** OSPF ROUTER ID:10.1.1.1:0

Every domain id will begin with 0005 or 0105 or 0205 which identify the type of domain id format
0005 (16 bit) mean domain id format will be as the following
Global Administrator field  area number **0000 0016** (area 0 + process id 0016 in hexadecimal = 22 in decimal)
Local Administer Field  **0200** normally ignored

**OSPF RT:0.0.0.0:2:0**
Mean area 0 , internal ospf route is 2 , last 0 mean the route si neither external type 1 nor external type 2

If domain id match the routes consider type 3 LSAs

The routes that are in another OSPF process are showing up as external type 2 routes , due to redistribution of BGP routes into OSPF
Under normal OSPF design process id is only locally significant but in MPLS VPN is not
To solve this and let routes shown as O IA we can use one of two solutions

Solution 1 : use same process ID on all PE’s
Solution 2 : use domain-id command

**PE1**
router ospf 22 vrf CBTME
domain-d 0.0.0.2

**PE2**
router ospf 2 vrf CBTME
domain-id 0.0.0.2

Or just use same OSPF process id

**PE1**
router ospf 2 vrf CBTME
domain-d 0.0.0.2

**PE2**
router ospf 2 vrf CBTME
Configure Static Route, IGP or BGP between PE’s & CE’s: EIGRP
(Our objectives here connect CBTME site1 with site 2 ONLY)

PE1
    router eigrp 1
    no auto-summary
    
    address-family ipv4 vrf CBTME
    network 10.1.1.1 0.0.0.0
    auto-summary
    autonomous-system 100
    exit-address-family

PE2
    router eigrp 1
    no auto-summary
    
    address-family ipv4 vrf CBTME
    network 10.3.3.1 0.0.0.0
    auto-summary
    autonomous-system 100
    exit-address-family

CBTME 1
    router eigrp 100
    network 10.1.1.2 0.0.0.0
    network 40.40.41.0 0.0.0.0
    no auto-summary

CBTME 2
    
    router eigrp 100
    network 10.3.3.2 0.0.0.0
    network 40.40.42.0 0.0.0.0
    no auto-summary

Configure Redistribution between MP-BGP and EIGRP

PE1
    router eigrp 1
    address-family ipv4 vrf CBTME
    redistribute bgp 65000 metric 10000 100 250 1 1500
    exit-address-family

    router bgp 65000
    address-family ipv4 vrf CBTME
    redistribute eigrp 100
    no synchronization
PE2
router eigrp 1
address-family ipv4 vrf CBTME
  redistribute bgp 65000 metric 10000 100 250 1 1500
exit-address-family

router bgp 65000
address-family ipv4 vrf CBTME
  redistribute eigrp 100
  no synchronization
exit-address-family

Configure Static Route , IGP or BGP between PE’s & CE’s : EBGP
(our objectives here connect CBTME site1 with site 2 ONLY)

PE1
router bgp 65000
address-family ipv4 vrf CBTME
  neighbor 10.1.1.2 remote-as 65001
  neighbor 10.1.1.2 activate
  neighbor 10.1.1.2 as-overide
  no synchronization
exit-address-family

PE2
router bgp 65000
address-family ipv4 vrf CBTME
  neighbor 10.3.3.2 remote-as 65001
  neighbor 10.3.3.2 activate
  neighbor 10.3.3.2 as-overide
  no synchronization
exit-address-family

! CBTME 1
router bgp 65001
  no synchronization
  bgp log-neighbor-changes
  network 40.40.40.41 mask 255.255.255.255
  neighbor 10.1.1.1 remote-as 65000
  no auto-summary

! CBTME 2
router bgp 65001
  no synchronization
  bgp log-neighbor-changes
  network 40.40.40.42 mask 255.255.255.255
  neighbor 10.3.3.1 remote-as 65000
  no auto-summary
What is as-override & Allows-in?
CBTME1 & CBTME2 belong to same BGP AS# but they are divided on two sites
Because of the loop prevention mechanism, CBTME2 for instance will have to reject CBTME1 prefixes because it can see its own AS in the AS_PATH attribute.

To solve this we can use one of two sols:

Using Allows-in 1 in CE (CBTME1, CBTME2)
This allows CBTME sites to override the loop prevention mechanism by allowing an instance of AS 65001 to be in the AS_PATH

**CBTME1**
nei 10.1.1.1 allowas-in 1

**CBTME2**
nei 10.2.2.1 allowas-in 1

Using AS-override in PE (PE1, PE2)
This getting PE1 & PE2 to just strip AS 65001 from the BGP UPDATE before sending it to the CE (CBTME1 & CBTME2).

**PE1**
nei 10.1.1.2 as-override

**PE2**
nei 10.2.2.2 as-override

Configure Redistribution between MP-BGP and EBGP
When using EBGP NO NEED to Redistribution between MP-BGP & EBGP
Backup Link issues with OSPF, EIGRP, EBGP

First we will create Backup Link between CBTME 1 & CBTME2

We will always have two issues, possibility that backup link would be chosen as best path since its fastethernet and mpls cloud is serial
Second we would receive same routes from both PE’s which lead to have loop issue.

OSPF Backup Link & Sham Link

An MPLS link is not preferred in OSPF when there is a back door because intra-area routes are preferred over external routes. Routes that are advertised across a MPLS/VPN that are imported and exported into BGP pass the route information with it. This means upon redistribution out of BGP into OSPF, routes retain their external route marking. Therefore they are marked as external routes and no longer preferred by OSPF. They are a type 5 external LSA. The backdoor link becomes favored and subsequently used.

An OSPF sham-link can solve this problem. The OSPF sham link provides a logical link between two VRFs. It creates a link that makes the MPLS PE’s participating in the sham link appear as a point to point link within OSPF. These links are able to fool or trick routers in the OSPF domain that this is a better path thus preserving the LSAs as type 1 or type 3.

By using two loopbacks on the respective devices advertised into the BGP address family that corresponds with the customer VRF, OSPF can create a link that is more appealing. By using the command area <area-id> sham-link <source-address> <destination-address> cost <cost> it is possible to build this link.

This means all internal OSPF routes at one site can appear internal on the other side. The sham-link cost can be adjusted to be lower than the backdoor OSPF link and therefore traffic will prefer going over the MPLS core first.
CBTME1
int s0/0
ip ospf 1 area 0
int loop 0
ip ospf 1 area 0

interface FastEthernet0/0
ip address 199.199.199.1 255.255.255.0
ip ospf cost 131

router ospf 1
router-id 10.1.1.2
log-adjacency-changes
network 199.199.199.1 0.0.0.0 area 0

CBTME2
int s0/0
ip ospf 1 area 0
int loop 0
ip ospf 1 area 0

interface FastEthernet0/0
ip address 199.199.199.2 255.255.255.0
ip ospf cost 131

router ospf 1
router-id 10.3.3.2
log-adjacency-changes
network 199.199.199.2 0.0.0.0 area 0

PE1
interface Loopback73
ip vrf forwarding CBTME
ip address 73.73.73.1 255.255.255.255

router ospf 2 vrf CBTME
router-id 10.1.1.1
log-adjacency-changes
area 0 sham-link 73.73.73.1 73.73.73.2
redistribute bgp 65000 subnets

router bgp 65000
address-family ipv4 vrf CBTME
redistribute ospf 2 vrf CBTME
no synchronization
network 73.73.73.1 mask 255.255.255.255
exit-address-family
PE2

interface Loopback73
ip vrf forwarding CBTME
ip address 73.73.73.2 255.255.255.255
router ospf 2 vrf CBTME
router-id 10.3.3.1
log-adjacency-changes
area 0 sham-link 73.73.73.2 73.73.73.1
redistribute bgp 65000 subnets

router bgp 65000
address-family ipv4 vrf CBTME
redistribute ospf 2 vrf CBTME
no synchronization
network 73.73.73.2 mask 255.255.255.255
exit-address-family

EIGRP with Soo

Backdoor links are supported between EIGRP sites that are connected to the MPLS VPN backbone. However, when a route disappears, routing can take longer to reconverge, which is typical in the case of redistribution between routing protocols. The cause of the longer convergence is redistribution between EIGRP and BGP. To help speed up the re-converging, you can use Site-of-Origin (SOO) for EIGRP. It can be defined on the PE routers on the VRF interfaces toward the CE routers and on the routers with a backdoor link. You need to configure ip vrf sitemap on the interface, setting the extended community SOO. This route map sets the SOO on the EIGRP route, either on the PE or on the backdoor link router. When the router receives a route across the interface with this route map configured and the SOO of the route matches the configured SOO, the router rejects the route. When the PE router receives a
vpnv4 update with the SOO set, it extracts the SOO and adds it to the EIGRP route when it is reconstructed.

When no SOO for EIGRP is used anywhere, a count-to-infinity problem might exist across the EIGRP sites and across the MPLS VPN backbone. This means that when a route disappears, EIGRP routers see that the hop count slowly increases up to infinity. With EIGRP, infinity is a hop count of 100 by default. That means that it might take quite some time for the route to disappear, while in the meantime traffic is looped. You can lower the default maximum hop count of EIGRP by configuring the command metric maximum-hops hops. You must take care, however, not to configure this value too low. The value must be big enough for regular operation, but also in case the shortest path is unavailable and a longer path routes the traffic. The disadvantage of using the SOO for EIGRP on the PE and backdoor routers is that one part of the site cannot reach the other part of the site across the backdoor link and the MPLS VPN backbone if the site is split. The backdoor router or the PE router blocks the route that is needed to get to the other part of the site. To work around this problem, you can configure the sitemap for SOO only on the PE routers and not the backdoor routers. The count-to-infinity problem does not occur in this case, but the routing might take a bit longer to reconverge. Example 7-32 shows the SOO for an EIGRP route.

```
CBTME1
interface FastEthernet0/0
  ip vrf sitemap SOO
  ip address 13.13.13.1 255.255.255.0
  delay 100000

router eigrp 100
  network 10.1.1.2 0.0.0.0
  network 13.13.13.1 0.0.0.0
  network 40.40.40.41 0.0.0.0
  no auto-summary

route-map SOO permit 10
  set extcommunity soo 10:11
```
CBTME2
interface FastEthernet0/0
   ip vrf sitemap SOO
   ip address 13.13.13.2 255.255.255.0
delay 100000

router eigrp 100
   network 10.3.3.2 0.0.0.0
   network 13.13.13.2 0.0.0.0
   network 40.40.40.42 0.0.0.0
   no auto-summary

route-map SOO permit 10
   set extcommunity soo 10:10

PE1
interface Serial0/0
   ip vrf forwarding CBTME
   ip vrf sitemap SOO
   ip address 10.1.1.1 255.255.255.0

router eigrp 1
   no auto-summary
   !
   address-family ipv4 vrf CBTME
      redistribute bgp 65000 metric 10000 100 250 1 1500
      network 10.1.1.1 0.0.0.0
      auto-summary
      autonomous-system 100

router bgp 65000
   address-family ipv4 vrf CBTME
      redistribute eigrp 100
      no synchronization
      exit-address-family
      !
      route-map SOO permit 10
      set extcommunity soo 10:11

PE2
interface Serial0/0
   ip vrf forwarding CBTME
   ip vrf sitemap SOO
   ip address 10.3.3.1 255.255.255.0

router eigrp 1
   no auto-summary
address-family ipv4 vrf CBTME
redistribute bgp 65000 metric 10000 100 250 1 1500
network 10.3.3.1 0.0.0.0
auto-summary
autonomous-system 100
exit-address-family

router bgp 65000
address-family ipv4 vrf CBTME
redistribute eigrp 100
no synchronization
exit-address-family

**route-map SOO permit 10**
**set extcommunity soo 10:10**

PE1#sh ip bgp vpnv4 all 40.40.40.42
BGP routing table entry for 1:1:40.40.40.42/32, version 41
Paths: (1 available, best #1, table CBTME)
Flag: 0x820
Not advertised to any peer
Local
22.22.22.22 (metric 31) from 22.22.22.22 (22.22.22.22)
Origin incomplete, metric 2297856, localpref 100, valid, internal, best
Extended Community: SoO:10:10 RT:1:1
Cost:pre-bestpath:128:2297856 (default-2145185791) 0x8800:32768:0
0x8801:100:640000 0x8802:65281:1657856 0x8803:65281:1500
mpls labels in/out nolabel/30

PE1#sh ip bgp vpnv4 all 40.40.40.41
BGP routing table entry for 1:1:40.40.40.41/32, version 35
Paths: (1 available, best #1, table CBTME)
Flag: 0x820
Advertised to update-groups:
1
Local
10.1.1.2 from 0.0.0.0 (11.11.11.11)
Origin incomplete, metric 2297856, localpref 100, weight 32768, valid, sourced, best
Extended Community: SoO:10:11 RT:1:1
Cost:pre-bestpath:128:2297856 (default-2145185791) 0x8800:32768:0
0x8801:100:640000 0x8802:65281:1657856 0x8803:65281:1500
mpls labels in/out 29/nolabel

**BGP had same concept BGP Soo and doing the same**
Selective import/export Map

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 an export route-map are not exported into the BGP process. The export route-map may also be used to set the extended-community attribute 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 uses. First, it allows controlling all routes imported into VRF from BGP based on prefix-lists, access-lists, or extended/standard communities.

Notice that by default, all prefixes not permitted with the import-map are implicitly denied and not imported.

Selective VRF Import

Selective route import uses a route map that can filter the routes selected by the RT import filter. The routes imported into a VRF are BGP routes, so you can use match conditions in a route map to match any BGP attribute of a route. The import route map is deployed in the receiving VRF.

A route has to pass the RT import filter first and then the import route map. First, at least one of the RTs attached to the route needs to match one of the import RTs configured in the VRF. Second, the route is permitted by the import route map.

import map route-map-name attaches a route map to the VRF import process. A route is imported into the VRF only if at least one RT attached to the route matches one RT configured in the VRF AND the route is accepted by the route map.

Let's say CBTME 1 had the following additional loopbacks:

```plaintext
int loop 1
ip add 198.169.169.1 255.255.255.255
ip ospf 1 area 0

int loop 2
ip add 197.169.169.1 255.255.255.255
ip ospf 1 area 0
```

But we want CBTME 2 receive only 198.169.169.1 & loop 0 we already used in previous examples 40.40.40.41

```plaintext
PE2
ip access-list stand 10
permit 198.169.169.0 0.0.0.255
permit 40.40.40.0 0.0.0.255

route-map koko 10
match ip add 10

ip vrf CBTME
import map koko
```
Let’s check PE2 Before applying this import map:

PE2#sh ip bgp vpnv4 vrf CBTME
BGP table version is 21, local router ID is 22.22.22.22
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

<table>
<thead>
<tr>
<th>Network</th>
<th>Next Hop</th>
<th>Metric LocPrf Weight Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>*&gt;10.1.1.0/24</td>
<td>11.1.1.11</td>
<td>0 100 0 ?</td>
</tr>
<tr>
<td>*&gt; 10.3.3.0/24</td>
<td>0.0.0.0</td>
<td>0 32768 ?</td>
</tr>
<tr>
<td>*&gt;i40.40.40.41/32</td>
<td>11.1.1.11</td>
<td>65 100 0 ?</td>
</tr>
<tr>
<td>*&gt; 40.40.40.42/32</td>
<td>10.3.3.2</td>
<td>65 32768 ?</td>
</tr>
<tr>
<td>*&gt;i197.169.169.1/32</td>
<td>11.1.1.11</td>
<td>65 100 0 ?</td>
</tr>
<tr>
<td>*&gt;i198.169.169.1/32</td>
<td>11.1.1.11</td>
<td>65 100 0 ?</td>
</tr>
</tbody>
</table>

Let’s check PE2 After applying this import map:

E2#sh ip bgp vpnv4 vrf CBTME
BGP table version is 19, local router ID is 22.22.22.22
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

<table>
<thead>
<tr>
<th>Network</th>
<th>Next Hop</th>
<th>Metric LocPrf Weight Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>*&gt; 10.3.3.0/24</td>
<td>0.0.0.0</td>
<td>0 32768 ?</td>
</tr>
<tr>
<td>*&gt;i40.40.40.41/32</td>
<td>11.1.1.11</td>
<td>65 100 0 ?</td>
</tr>
<tr>
<td>*&gt; 40.40.40.42/32</td>
<td>10.3.3.2</td>
<td>65 32768 ?</td>
</tr>
<tr>
<td>*&gt;i197.169.169.1/32</td>
<td>11.1.1.11</td>
<td>65 100 0 ?</td>
</tr>
<tr>
<td>*&gt;i198.169.169.1/32</td>
<td>11.1.1.11</td>
<td>65 100 0 ?</td>
</tr>
</tbody>
</table>

PE2#sh ip route vrf CBTME

198.169.169.0/32 is subnetted, 1 subnets
B 198.169.169.1 [200/65] via 11.1.1.11, 00:01:05
   40.0.0.0/32 is subnetted, 2 subnets
B 40.40.40.41 [200/65] via 11.1.1.11, 00:01:05
O 40.40.40.42 [110/65] via 10.3.3.2, 00:05:03, Serial0/0
   10.0.0.0/24 is subnetted, 1 subnets
C 10.3.3.0 is directly connected, Serial0/0
Selective VRF Export
Some advanced MPLS VPN topologies are easiest to implement if you can attach a variety of RTs to routes exported from the same VRF. This capability allows only a subset of the routes exported from a VRF to be imported into another VRF. The export route map is deployed in the originating VRF. A route map can be specified for each VRF to attach additional RTs to routes exported from that VRF. The export route map performs only the attachment of RT’s. It does not perform any filtering function. Attributes attached to a route with an export route map are combined with the export RT attributes. If you specify export RTs in a VRF and set RTs with an export route map, all specified RTs will be attached to the exported route.

set extcommunity rt extended-community-value [additive] sets the BGP extended community attribute for a RT.
Export map route-map-name attaches a route map to the VRF export process.

MPLS VPN Route Leaking

Route Leaking Between Different VRFs
We will Use RT to make CBTME & TRAININGHOUSE receive routes & ping from each other’s

PE1 & PE2
ip vrf CBTME
   rd 1:1
   route-target export 1:1
   route-target import 1:1
   route-target import 2:2
   |
ip vrf TRAININGHOUSE
   rd 2:2
   route-target export 2:2
   route-target import 2:2
   route-target import 1:1

To verify just ping 50.50.50.52 in TRAININGHOUSE 2 from CBTME 1

Route Leaking between P routers & CE VRFs
Let’s say we want P1 1.1.1.1 reach CBTME1 40.40.40.41 & vice versa.

PE1#sh ip route 1.1.1.1
Routing entry for 1.1.1.1/32
Known via "ospf 1", distance 110, metric 11, type intra area
Last update from 100.100.100.1 on FastEthernet0/0, 00:15:16 ago
Routing Descriptor Blocks:
* 100.100.100.1, from 1.1.1.1, 00:15:16 ago, via FastEthernet0/0
   Route metric is 11, traffic share count is 1
PE1#sh ip route vrf CBTME 1.1.1.1
% Network not in table
PE1#sh ip route 10.1.1.1
% Network not in table
PE1#sh ip route 10.1.1.2
% Network not in table

PE1
ip route 10.1.1.1 255.255.255.255 s0/0
ip route 40.40.40.41 255.255.255.255 10.1.1.1
ip route vrf CBTME 1.1.1.1 255.255.255.255 100.100.100.1 global

router ospf 1
redis static subnets

router ospf 2 vrf CBTME
redis static subnets

PE1#sh ip route 10.1.1.2
Routing entry for 10.1.1.2/32
    Known via "static", distance 1, metric 0 (connected)
    Redistributing via ospf 1
    Advertised by ospf 1 subnets
Routing Descriptor Blocks:
    * directly connected, via Serial0/0
    Route metric is 0, traffic share count is 1

PE1#sh ip route vrf CBTME 1.1.1.1
Routing entry for 1.1.1.1/32
    Known via "static", distance 1, metric 0 (connected)
    Redistributing via ospf 2
    Advertised by ospf 2 subnets
Routing Descriptor Blocks:
    * directly connected, via Serial0/0
    Route metric is 0, traffic share count is 1

PE1#sh ip route 40.40.40.41
Routing entry for 40.40.40.41/32
    Known via "static", distance 1, metric 0
    Redistributing via ospf 1
    Advertised by ospf 1 subnets
Routing Descriptor Blocks:
    * 10.1.1.2
    Route metric is 0, traffic share count is 1
## Internet Access from an MPLS VPN Using a Global Routing Table

First let's Create interface loopback 13 in P1, let's assume P1 connected to internet and LOOP13 is one of internet ip address

```plaintext
P1
int loop13
ip add 13.13.13.13 255.255.255.255
ip ospf net point-to-p
router ospf 1
net 13.13.13.13 0.0.0.0 are 0
```

**We can use one of two methods to provide internet access for CE using Global Routing Table**

**Method 1**
We want CBTME2 connect to Internet & ping 13.13.13.13 using loop 0 10.3.3.2 as source

```plaintext
P1#sh ip route 10.3.3.2
% Subnet not in table

CBTME2#sh ip route 13.13.13.13
% Network not in table
```

**PE1**

```plaintext
ip route 10.3.3.1 255.255.255.255 s0/0
ip route 10.3.3.2 255.255.255.255 10.3.3.1
ip route vrf CBTME 0.0.0.0 0.0.0.0 200.200.200.1 global

router ospf 2 vrf CBTME
default-information originate always

router ospf 1
redis static subnets
```

CBTME2#ping 13.13.13.13 source loop0
!!!!!

In order to give internet access to a VRF via the global table, you will need to leak the VRF routes into the global table as well as create a VRF default static route pointing to a global next hop using the 'global' keyword.
Method 2 Using GRE Tunnel

PE1
int tunnel 1
ip add 99.99.99.99 255.255.255.0
! tunnel source 10.1.1.1
tunnel dest 10.1.1.2
tunnel vrf CBTME

ip route 40.40.40.41 255.255.255.255 tunnel1

CBTME1
int tunnel 1
ip add 99.99.99.100 255.255.255.0
! tunnel source 10.1.1.2
tunnel dest 10.1.1.1
ip route 0.0.0.0 0.0.0.0 tunnel 1

CBTME1#ping 13.13.13.13 source loop 0
!!!!!

MPLS VPN performance tuning
Time utilized between P's & PE's for propagate topology changes must be short as much as we can

Times factors affect this are:

1-time takes for IGP update to be redistributed into MP-BGP , in the past we used to use bgp scan-interval but current IOS fixed this and make IGP-to-BGP redistribution instant.

2-time takes local BGP speaker and other BGP speakers to propagates updates , the default is 5 seconds and could be set to 0 seconds

   nei 11.11.11.11 advertisement-interval 0

3-time takes PE BGP to import MP-BGP VPNv4 prefixes into local VRF table ,the default is 15 seconds and could be set to 5 seconds as min

   bgp scan-time import 5 ( you can set it to min 5 sec  and up to 60 sec)
**MPLS VPN & NAT**

**CBTME1**
- int loop 1
- ip add 21.21.21.1 255.255.255.255
- int loop 2
- ip add 21.21.21.2 255.255.255.255
- int loop 3
- ip add 21.21.21.3 255.255.255.255

**CBTME2**
- int loop 1
- ip add 22.22.22.1 255.255.255.255
- int loop 2
- ip add 22.22.22.2 255.255.255.255
- int loop 3
- ip add 22.22.22.3 255.255.255.255

**PE1**
- ip route vrf CBTME 221.221.221.0 255.255.255.0 10.1.1.2
- router bgp 65000
- add ipv4 vrf CBTME
- redis connected
- redis static

**PE2**
- ip route vrf CBTME 222.222.222.0 255.255.255.0 10.3.3.2
- router bgp 65000
- add ipv4 vrf CBTME
- redis connected
- redis static

*If we will implement NAT on CE*

**CBTME1**
- int range loop 1 - 3
- ip nat inside
- int s0/0
- ip nat outside

- ip nat inside source static 21.21.21.1 221.221.221.1

- access-list 100 permit ip 21.21.21.0 0.0.0.255 222.222.222.0 0.0.0.255
- ip nat pool auda 221.221.221.2 221.221.221.3 prefix-length 24 type match-host

- ip nat inside source list 100 pool auda
CBTME2
int range loop 1 - 3
ip nat inside
int s0/0
ip nat outside

ip nat inside source static 22.22.22.1 222.222.222.1

access-list 100 permit ip 22.22.22.0 0.0.0.255 221.221.221.0 0.0.0.255
ip nat pool auda 222.222.222.2 222.222.222.3 prefix-length 24 type match-host
ip nat inside source list 100 pool auda

CBTME1#sh ip nat translations
Pro Inside global  Inside local  Outside local  Outside global
--- 221.221.221.1    21.21.21.1       ---       ---

CBTME1#ping 222.222.222.1 source loop 3
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 222.222.222.1, timeout is 2 seconds:
Packet sent with a source address of 21.21.21.3
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 396/484/580 ms

CBTME1#sh ip nat translations
Pro Inside global  Inside local  Outside local  Outside global
--- 221.221.221.1    21.21.21.1       ---       ---
icmp 221.221.221.3:1 21.21.21.3:1 222.222.222.1:1 222.222.222.1:1
--- 221.221.221.3  21.21.21.3       ---       ---

If we will implement on PE (same but we add VRFname):
PE1
int s0/0
ip nat inside
int f0/0
ip nat outside

ip nat inside source static 21.21.21.1 222.222.222.1 vrf CBTME

access-list 100 permit ip  21.21.21.0 0.0.0.255 222.222.222.0 0.0.0.255
ip nat pool auda 221.221.221.2 221.221.221.3 prefix-length 24 type match-host
ip nat inside source list 100 pool auda vrf CBTME

PE2
int s0/0
ip nat inside
int f0/0
ip nat outside
ip nat inside source static 22.22.22.1 222.222.222.1 vrf CBTME

access-list 100 permit ip 22.22.22.0 0.0.0.255 221.221.221.0 0.0.0.255
ip nat pool auda 222.222.222.2 222.222.222.3 prefix-length 24 type match-host
ip nat inside source list 100 pool auda vrf CBTME

**Protecting PE**

Any customer can generate many number of routes, using resources in the PE routers, therefore resources used by single customer have to be limited

*We can limit number of routes received from BGP neighbor:*

```
PE1(config-router-af)# nei 11.11.11.11 maximum-prefix 120 80
```

120 is max prefixes  
80 is threshold which mean when 80% of 120 reach a warning message is logged (default 75%)

Optional we can add **warning-only** keyword which specifies action on exceeding the maximum number (default is to drop)

*We can limit total number of routes imported in PE vrf:*

Routes imported in PE vrf are coming from CE routers or from other PE routers, if number exceeded additional routes will be rejected, Optionally we can also set to a syslog message warning message

```
PE1(config-vrf)# maximum routes 100 80 warn-only
```

100 is max number of routes  
80 is warn-threshold (80%)  
**Warn-only** is to create syslog error message

**Resources:**

Free Videos from IPexpert "Next Generation"
http://youtu.be/2K11sOeaLHs  
http://youtu.be/Yl1xk9Mx8C4

MPLS Label Distribution Protocol
CCIEv5 MPLS Guide (LDP, VRF Lite, MPLS VPN)  
By CCSI: Yasser Auda

MPLS Virtual Private Networks  

Multiprotocol BGP MPLS VPN  

Intro to VRF lite From Packetlife  
http://packetlife.net/blog/2009/apr/30/intro-vrf-lite/

Inter-VRF Routing with VRF Lite From Packetlife  

Getting to know MPLS From Packetlife  
http://packetlife.net/blog/2008/jul/16/getting-to-know-mpls/

Creating an MPLS VPN From Packetlife  
http://packetlife.net/blog/2011/may/16/creating-mpls-vpn/

Cisco Press  MPLS Fundamentals  ( Free Chapters Avalible )  
http://www.ciscopress.com/store/mpls-fundamentals-9781587051975

MPLS Topics From René - CCIE #41726  
http://networklessons.com/category/mpls/

MPLS Topics From INE  

BGP as-override vs allow-as-in  
http://ccieblog.co.uk/bgp/bgp-as-override-vs-allow-as-in

Route Leaking in MPLS/VPN Networks  

VRF-lite route leaking  

Good Luck

CCSI: Yasser Auda
https://www.facebook.com/YasserRamzyAuda
https://learningnetwork.cisco.com/people/yasser.r.a?view=documents
https://www.youtube.com/user/yasserramzyauda