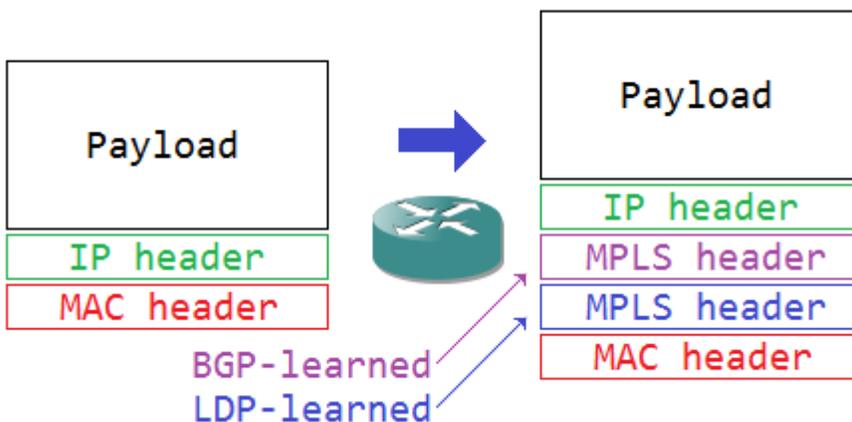


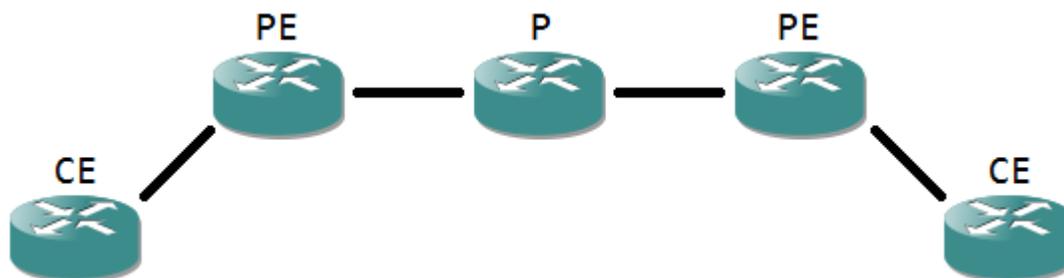
MPLS, PART II

Where [MPLS part I](#) explains the basics of labeling packets, it's not giving any advantage over normal routing, apart from faster table lookups. But extensions to MPLS allow for more. In this article I'll explain MPLS-VPN, and more specifically a **Virtual Private Routed Network (VPRN)**.

A VPRN is a routed (layer 3) network over an MPLS cloud, that is VRF-aware, or customer-aware. This means several different routing instances (VRFs, remember?) can share the same MPLS cloud. How is this achieved when there's only a point-to-point link between routers with one IP address? After all, an interface can only be assigned to one VRF. Solution: by adding a second MPLS label to the data: the 'outer' label (the one closest to the layer 2 header) is used to specify the destination router, the 'inner' label (closest to the original layer 3 header) is used to specify to which VRF a packet belongs. The outer label workings are identical to standard MPLS: these are learned by LDP and matched with a prefix in the routing table. But for the inner label, a VRF-aware process needs to run on each router that can handle label information and propagate it to other routers. That process is Multiprotocol-BGP, or MP-BGP.



The outer label is used to route the packet through the MPLS cloud, and the last router(s) use the inner label to see to which VRF a packet belongs. Let's look at the configuration to understand it more. First, the basic setup:



Notice the router names, as these are often used in MPLS terminology.

- The **C**ustomer **E**dge router is a router that directly connects to a customer network. It's usually the demarcation point, where the equipment governed by the MPLS provider begins. Contrary to the name, the CE itself is often managed by the provider as well.
- The **P**rovider **E**dge router is the 'first' router (seen from a customer site point of view) that has MPLS enabled interfaces. It's where the labels are applied for the first time.
- A **P**rovider router is a router completely internal to the MPLS cloud, having only MPLS enabled interfaces.

The connection between CE and PE is a point-to-point so a /30 subnet is logical. Routing between CE and PE is done using a simple routing protocol, like RIP, OSPF, EIGRP or even static or standard BGP. The only notable part is that the PE router has to use a VRF to place the customer in. Below an example configuration:

```

Router-CE(config)#interface G0/1
Router-CE(config-if)#ip address 10.1.0.1 255.255.255.252
Router-CE(config-if)#exit
Router-CE(config)#router ospf 1
Router-CE(config-router)#network 10.1.0.0 0.0.0.3 area 0
Router-PE(config)#vrf definition VRF
Router-PE(config-vrf)#address-family ipv4
Router-PE(config-vrf-af)#exit
Router-PE(config-vrf)#exit
Router-PE(config)#interface G0/1
Router-PE(config-if)#vrf forwarding VRF
Router-PE(config-if)#ip address 10.1.0.2 255.255.255.252
Router-PE(config-if)#exit
Router-PE(config)#router ospf 2 vrf VRF
Router-PE(config-router)#network 10.1.0.0 0.0.0.3 area 0

```

```

*Mar 1 00:06:20.991: %OSPF-5-ADJCHG: Process 2, Nbr 10.1.0.1 on GigabitEthernet0/1 from
LOADING to FULL, Loading Done

```

If you happen to run a router on an IOS before 15.0, the commands for the VRF change: it becomes 'ip vrf VRF' to define a VRF, without the need to specify an address family, as 12.x IOS versions aren't VRF aware for IPv6. On the interface, the command is 'ip vrf forwarding VRF'.

So far so good. Now on to activating MPLS between the PE and the P router, and making sure the routers learn the MPLS topology:

```
Router-PE(config)#interface G0/2
Router-PE(config-if)#mpls ip
Router-PE(config-if)#ip address 10.0.0.1 255.255.255.252
Router-PE(config-if)#ip ospf network point-to-point
Router-PE(config-if)#exit
Router-PE(config)#interface Loopback0
Router-PE(config-if)#ip address 10.0.1.1
Router-PE(config-if)#exit
Router-PE(config)#router ospf 1
Router-PE(config-router)#network 10.0.0.0 0.0.1.255 area 0
Router-P(config)#interface G0/1
Router-P(config-if)#mpls ip
Router-P(config-if)#ip address 10.0.0.2 255.255.255.252
Router-P(config-if)#ip ospf network point-to-point
Router-P(config-if)#exit
Router-P(config)#interface Loopback0
Router-P(config-if)#ip address 10.0.1.2 255.255.255.255
Router-P(config-if)#exit
Router-P(config)#router ospf 1
Router-P(config-router)#network 10.0.0.0 0.0.1.255 area 0
*Mar 1 00:02:29.023: %OSPF-5-ADJCHG: Process 1, Nbr 10.0.0.1 on GigabitEthernet0/2 from
LOADING to FULL, Loading Done
*Mar 1 00:02:33.127: %LDP-5-NBRCHG: LDP Neighbor 10.0.0.1:0 (1) is UP
```

The 'ip ospf network point-to-point' is not really needed but used to reduce OSPF overhead. The loopbacks are needed for BGP later on.

Up until this point, we have MPLS in the default VRF and a separate VRF per customer for routing, but no routing of the VRFs over the MPLS. To exchange the inner labels needed to specify the VRF, MP-BGP between PE and P is configured:

```
Router-PE(config)#router bgp 65000
Router-PE(config-router)#neighbor 10.0.1.2 remote-as 65000
Router-PE(config-router)#neighbor 10.0.1.2 update-source Loopback0
Router-PE(config-router)#no address-family ipv4
Router-PE(config-router)#address-family vpnv4
Router-PE(config-router-af)#neighbor 10.0.1.2 activate
Router-PE(config-router-af)#neighbor 10.0.1.2 send-community extended
Router-P(config)#router bgp 65000
Router-P(config-router)#neighbor 10.0.1.1 remote-as 65000
Router-P(config-router)#neighbor 10.0.1.1 update-source Loopback0
Router-P(config-router)#no address-family ipv4
Router-P(config-router)#address-family vpnv4
Router-P(config-router-af)#neighbor 10.0.1.1 activate
Router-P(config-router-af)#neighbor 10.0.1.1 send-community extended
*Mar 1 00:11:31.879: %BGP-5-ADJCHANGE: neighbor 10.0.1.1 Up
```

Again some explanation. First off, BGP neighbors always need to be defined under the main process, after which they need to be activated for a specific address-family. The 'no address-family ipv4' command means that no conventional routing information for the default VRF will be exchanged (we already have OSPF for that). The 'address-family vpnv4' activates the VPRN capability, and the label exchange for VRFs. In this process, the neighbor is activated. The 'send-community extended' means BGP will exchange the community PathAttributes (PA), in which the label information is present. The loopbacks are used to connect to each other, not only for redundancy in case a physical interface should go down, but also because LDP does not exchange labels with another router on a connected subnet for that subnet. This means that if the directly connected interfaces are used as BGP neighbors, the BGP process can't figure out the labeling properly.

So VRF-aware MPLS is running now, but on each PE router it needs to be specified which VRF needs to be injected in the MPLS cloud. MP-BGP does this using import and export of routing tables. For MP-BGP, a route needs to be uniquely identified, and explained to which VRF it belongs. This is done with a **Route Distinguisher (RD)** and **Route Target (RT)**. Both are 64 bits, and usually in the format AS:nn, or the first 32 bits the AS number and the last 32 bits a unique chosen number.

- RD uniquely identifies a route. Inside MP-BGP, a route is prepended with its RD, e.g. 65000:1:192.168.1.0/24. This way, if the route exists twice (in different VRFs), it's still unique because the RD part of the prefix is different.
- RT specifies to which VRF a route belongs. It handles the import and export of routes from a VRF to the BGP process. In its basic form, it's the same number as the RD, and the same at all PE routers for a certain client.

Configuration of these parameters is done inside the VRF:

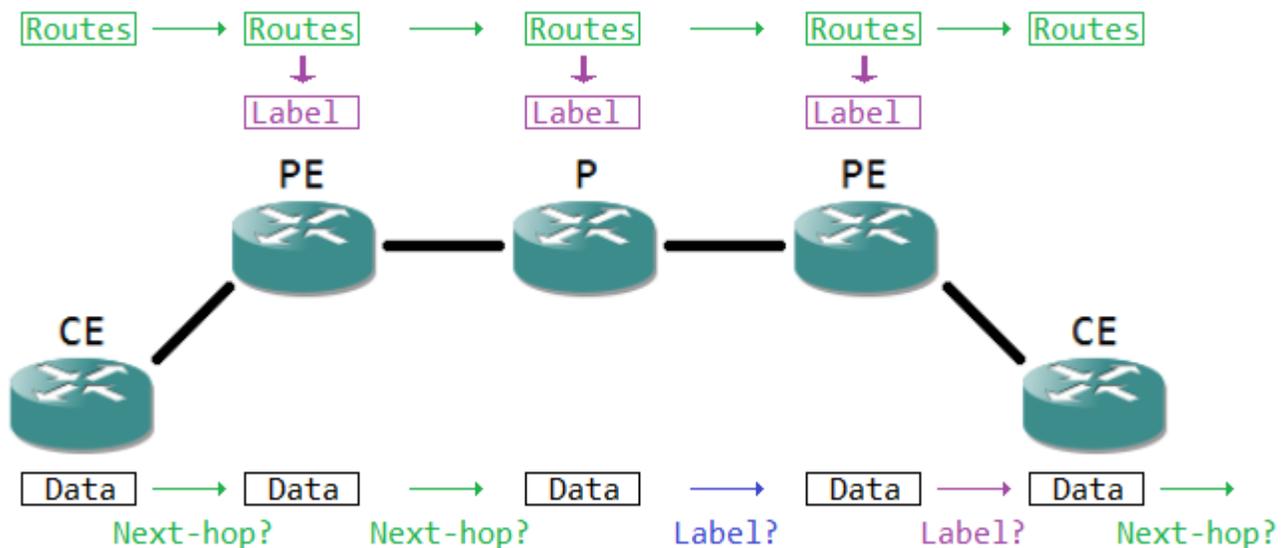
```
Router-PE(config)#vrf definition VRF
Router-PE(config-vrf)#rd 65000:1
Router-PE(config-vrf)#route-target both 65000:1
```

Now that the VRF can be used in the BGP process, it is imported in the process as following:

```
Router-PE(config)#router bgp 65000
Router-PE(config-router)#address-family ipv4 vrf VRF
Router-PE(config-router-af)#redistribute ospf 2
```

The redistribution, of course, needs to be mutual between OSPF and BGP, so a few more lines of configuration are needed to complete everything:

```
Router-PE(config)#router ospf 2 vrf VRF
Router-PE(config-router)#redistribute bgp 65000 subnets
```



And now everything is completed: the PE router learns routes from the P router by OSPF, and redistributes it into BGP to propagate them over the MPLS cloud. From this point on, configuration is modular: on another PE router, the configuration is likewise. Adding a P router isn't any different from the P router in this example, the processes and parameters are the same

each time. Do remember that routers don't advertise iBGP-learned routes to other iBGP peers, so the PE and P routers need to form a full mesh, unless you're using route reflectors or confederations.

Source : <http://reggle.wordpress.com/2013/03/17/mpls-part-ii-vrf-aware-mpls-vpn/>