A small MPLS VPN tutorial

(by Alexandre Ribeiro, alexandregomesribeiro@gmail.com)

Overview

I'm now half way through the MPLS and VPN Architectures book and I decided to try to make some sense out of all the things I'm reading. As I said previously, I'm not impressed by this book, since it's badly structured and it skips some important troubleshooting points (like one I'm going to present in this tutorial).

The following tutorial assumes some familiarity with MPLS and considerable IP knowledge.

The topology I used for this small tutorial is simple. It consists of two clients, A and B, each of which has two sites, A1 and A2 for client A, and B1 and B2 for client B. Then there are three LSRs: two of these are PE routers, PE_A connects to A1 and B1 and PE_Z connects to A2 and B2.

To make things interesting I decided to make clients A and B to have totally overlapping IP networks (isn't this the point of a VPN? :-)). The figure below illustrates the scenario I've just described:

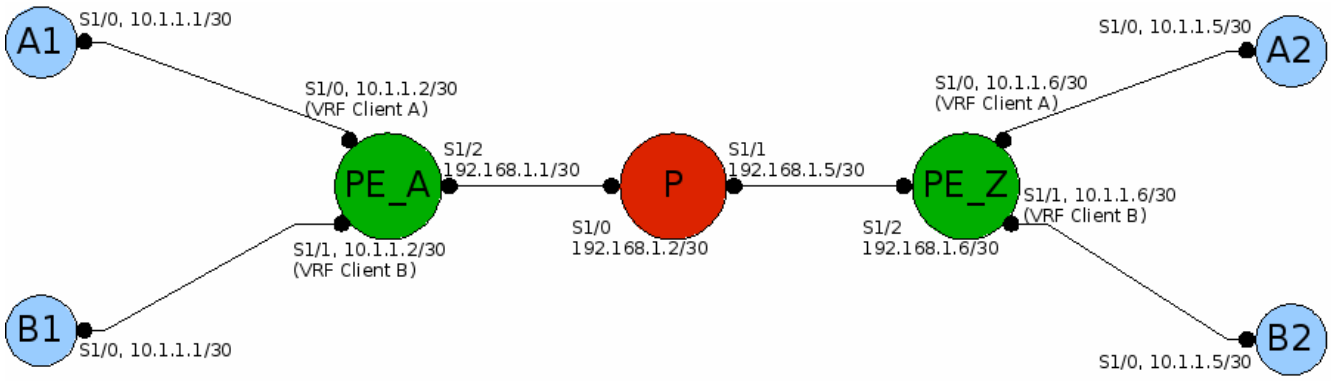

Figure 1: MPLS VPN scenario

Definitions

Before I begin rambling about configurations, I'll explain some important topics first, so that we're speaking the same language when I get down to configuring the routers. I'm assuming that you have a sound knowledge of IP (routing, IGPs, BGP, etc). If you don't exactly understand one of the definitions below, don't worry, it'll become very clear in a later part of this tutorial, when actually configuring the routers. So, let's get down to some definitions:

FEC – Forwarding equivalence class. A FEC is a group of packets that are routed the same way (i.e. to the same destination). In more practical terms, each entry of a router's routing table is a FEC.

Label – This is MPLS's foundation. A router will generate a label for each FEC it has. A Provider router (the "P" router in the figure above) will switch frames purely based on the label, without ever needing to go to L3 information on the frame.

LDP – Label distribution protocol. After generating the labels for the FECs, a router needs to inform its neighbors of the relationship between its FECs and the labels it has generated, so that neighbor routers may mark packets whose destination is the FEC with the respective label. LDP is used to disseminate label-to-FEC information.

The figure below will make the definitions above "click" together:

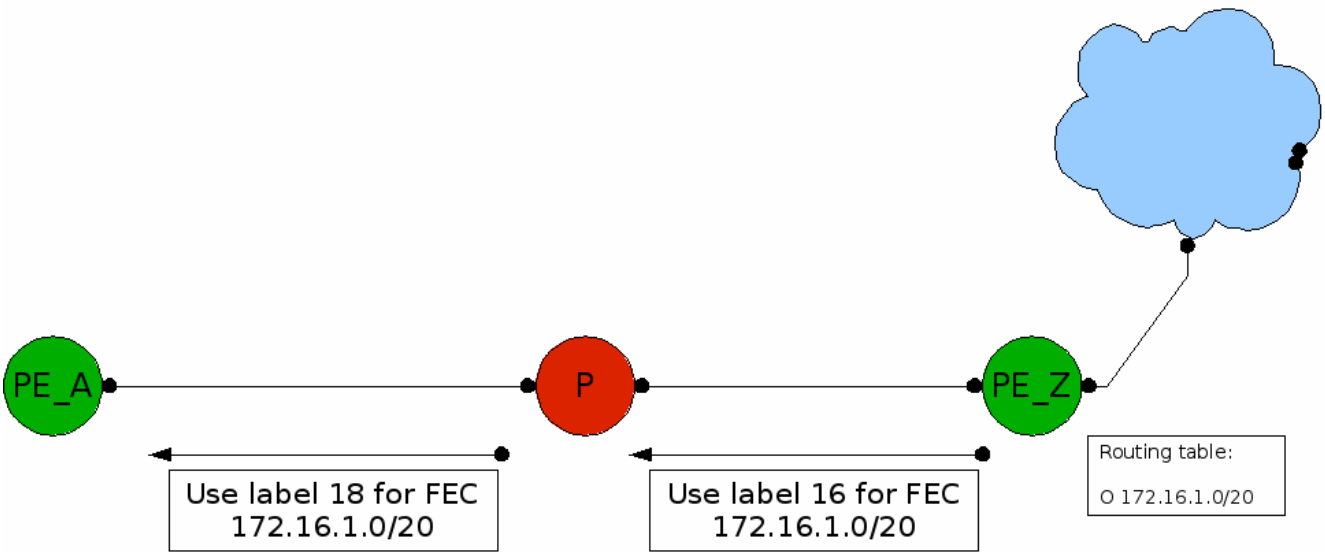

Figure 2: LDP at work

In figure 2 you can see router PE_Z communicating the label for FEC 172.16.1.0/20 to router P. Router P then does the same for router A. The labels assigned to FEC are locally significant. In the figure above router P could actually have used the same label as router PE_Z (label 16).

After the labels are known to all the routers, the P routers can switch frames purely based on labels. The figure below illustrates this:

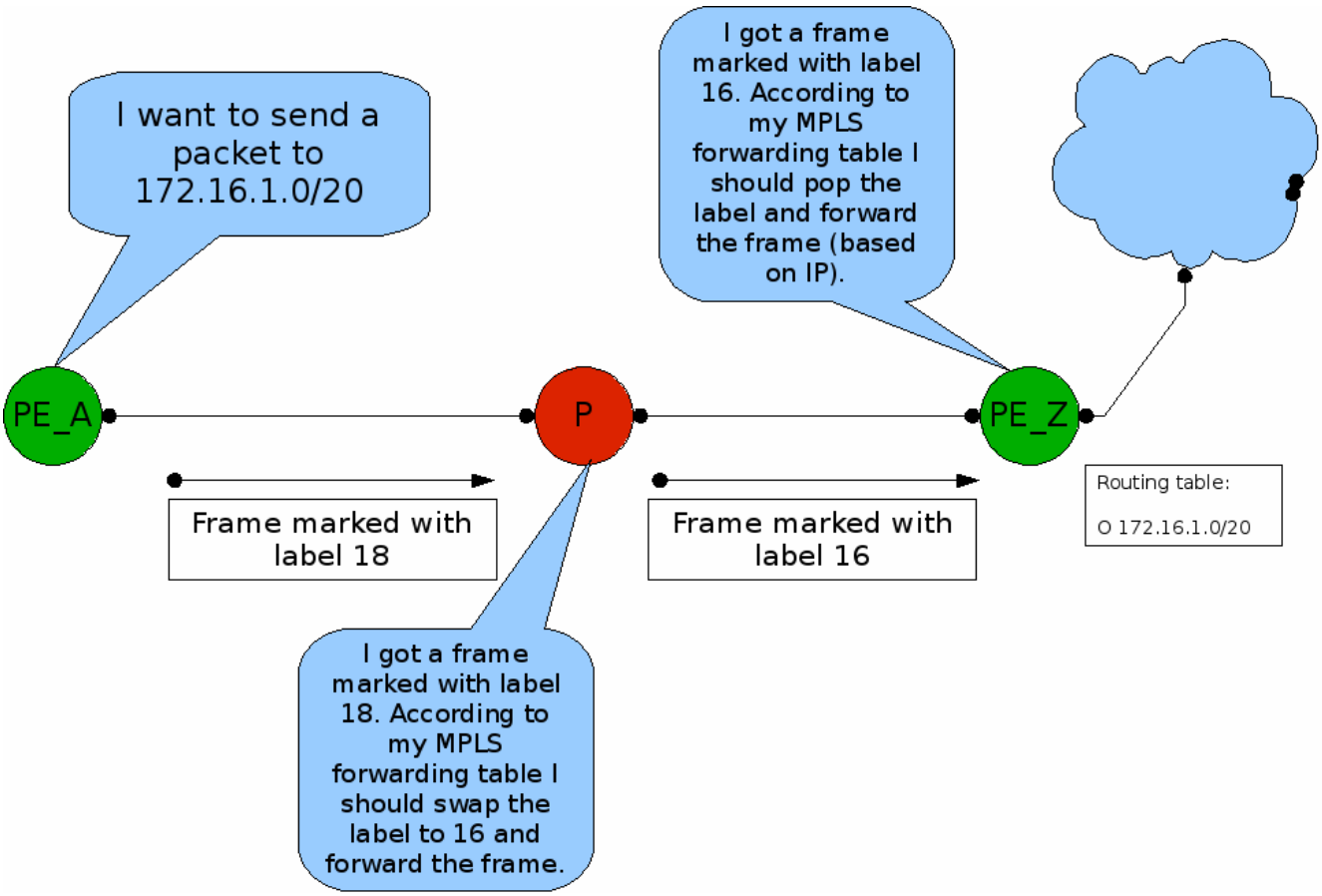

Figure 3: Frame forwarding with MPLS

The definitions and illustrations above define what “basic” MPLS is. Just with this it's possible to reap tremendous benefit in a carrier's BGP core (the explanation to this is way out of this small tutorial's scope).

Continuing with some definitions:

VPN - Virtual private network. Behaves like a physical private network, but it's "virtual" :-). There are two types of VPNs, peer-to-peer and overlay. The peer-to-peer VPN is an L3 VPN, where the CE and PE have to have L3 connectivity. In the overlay VPN the carrier will offer “emulated” L2 services to connect the VPN's sites.

Site - A site is a part of one or more VPNs, or the other way around, a VPN is a set of sites, where each site may belong to more than one VPN. In this tutorial's scenario, each site is only a member of one VPN (sites A1 and A2 are members of Client A's VPN and sites B1 and B2 are members of Client B's VPN). If you now had some sort of central resource (in a different VPN) that had to be access by A1 and B1, then those sites would also be members of another VPN, to be able to access that central resource.

VRF - Virtual Routing and forwarding instance. Composed of the routing table and the forwarding table. The VRF contains routes of one or more VPNs. In an extreme scenario you could have one VRF

per site and in another extreme one VRF per VPN. Usually something in the middle happens.

Route target – Think of this as a VPN Id. A VRF has to be somehow identified, so that its information may travel through an MP-BGP core to another PE router, where it'll be imported by VRF's in that PE router.

Route distinguisher – If there are overlapping IP networks in some VPN's, there has to be some way of distinguish them. Rd's are attached to each route, so that they become unique in the MP-BGP core.

MP-BGP – Multi protocol BGP. Each PE router has to communicate its routes to other PE routers. Since the routes it has are now 96 bits long (32 bits of the IP route plus 64 bits for the RD), BGP can't be used. MP-BGP has two extended attributes for VPNs: Site of origin (SOO) and Route target. The SOO is used for loop prevention (needed only in multi-homing scenarios).

Overall description of an MPLS VPN working

The problem being solved by an MPLS VPN is the isolation of different customer's traffic (which may or may not have overlapping IP addresses). Since the customer's traffic will be isolated, they could be totally unaware that they're using an MPLS VPN, since traffic will be forwarded in a transparent fashion.

So, you have the Customer Edge (CE) routers, which don't need any special configuration at all. They be connected to a PE router via IP, usually a point to point connection, using a /30 network for instance.

The PE routers is where the fun begins :-). In the PE routers you have to create VRF's that'll contain routes for one or more VPN. VRF's, as stated before, are usually created for a set of sites. Each VRF will contain the routes for the sites it pertains to. PE routers have to communicate their VRF's contents between themselves. To that end MP-BGP is used. Each PE routers has a MP-BGP connection (iMP-BGP) to every other PE router. Since MPLS is being used, each PE router has a label for every other PE router in the MPLS core. Furthermore, for each route inside each VRF, a label is assigned. This means that when a PE router get a packet, it will check the corresponding VRF, add the relevant label to the packet, then check the labels for the egress PE router and also add a label for that PE router. This means that the frame leaving the ingress PE router will actually have two labels. You can check a VRF's MPLS forwarding table with “sh mpls forwarding-table vrf *vrfname*”. The global MPLS forwarding table can be checked with the same command, without the vrf part at the end.

The P routers are highly efficient, since all they have to do is to switch frames based on the outer label. Remember that the frame being passed between PE routers have two labels, the inner one identifying a network for a specific VRF, and the outer one identifying the egress PE. The P router only switches based on the outer label, making the frame arrive to the egress PE router.

And this is it! It's actually quite simple :-) If this hasn't quite sinked in, be sure that it will after the next section, where I'll actually implement the scenario show in the first figure of this tutorial.

Configuring the routers

IGP configuration inside the MPLS backbone

I'm assuming that all ports are configured according to figure 1 (except the VRF part) and that PE_A, P and PE_Z have loopback ports configured with IP addresses 172.16.1.1/32, 172.16.1.2/32 and 172.16.1.3/32 respectively. I chose OSPF as the IGP to give basic routing connectivity inside the MPLS core. So, let's start:

```
PE_A(config)#router ospf 110
PE_A(config-router)#network 172.16.0.0 0.0.255.255 area 0
PE_A(config-router)#network 192.168.1.0 0.0.0.255 area 0
```

```
P(config)#router ospf 110
P(config-router)#network 172.16.0.0 0.0.255.255 area 0
P(config-router)#network 192.168.1.0 0.0.0.255 area 0
P(config-router)#
*Aug 26 11:56:05.579: %OSPF-5-ADJCHG: Process 110, Nbr 172.16.1.1 on Serial1/0 from
LOADING to FULL, Loading Done
```

```
PE_Z(config)#router ospf 110
PE_Z(config-router)#network 172.16.0.0 0.0.255.255 area 0
PE_Z(config-router)#network 192.168.1.0 0.0.0.255 area 0
PE_Z(config-router)#
*Aug 26 11:58:07.851: %OSPF-5-ADJCHG: Process 110, Nbr 172.16.1.2 on Serial1/2 from
LOADING to FULL, Loading Done
```

Now we have connectivity from PE_A to PE_Z:

```
PE_Z#ping 172.16.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/33/44 ms
```

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

Gateway of last resort is not set

     172.16.0.0/32 is subnetted, 3 subnets
O       172.16.1.1 [110/129] via 192.168.1.5, 00:01:13, Serial1/2
C       172.16.1.3 is directly connected, Loopback0
O       172.16.1.2 [110/65] via 192.168.1.5, 00:01:13, Serial1/2
     192.168.1.0/30 is subnetted, 2 subnets
O       192.168.1.0 [110/128] via 192.168.1.5, 00:01:13, Serial1/2
C       192.168.1.4 is directly connected, Serial1/2
```

Basic MPLS configuration

We can now enable MPLS to start to see some labels being exchanged. We have to enable MPLS globally and in each interface that's going to use it (the interfaces inside the MPLS cloud, excepting those that connect to CEs):

```
PE_A(config)#mpls ip
PE_A(config)#int s1/2
PE_A(config-if)#mpls ip

P(config)#mpls ip
P(config)#int s1/0
P(config-if)#mpls ip
P(config-if)#int s1/1
P(config-if)#mpls ip
*Aug 26 12:03:08.491: %LDP-5-NBRCHG: LDP Neighbor 172.16.1.1:0 (1) is UP

PE_Z(config)#mpls ip
PE_Z(config)#int s1/2
PE_Z(config-if)#mpls ip
PE_Z(config-if)#
*Aug 26 12:04:42.739: %LDP-5-NBRCHG: LDP Neighbor 172.16.1.2:0 (1) is UP
```

When you activate MPLS, LDP is automatically turned on and labels start being advertised (default mode: downstream unsolicited).

We now can see the MPLS forwarding table:

```
PE_Z#sh mpls for
Local   Outgoing     Prefix              Bytes tag   Outgoing     Next Hop
tag     tag or VC    or Tunnel Id        switched    interface
16      16           172.16.1.1/32       0           Se1/2        point2point
17      Pop tag      172.16.1.2/32       0           Se1/2        point2point
18      Pop tag      192.168.1.0/30      0           Se1/2        point2point
```

Notice that each FEC got a label, that's being advertised through LDP. You can also see an example of pure MPLS switching in this table. When this router receives a frame that has label 16, it swaps the label with 16 (the same label by coincidence) and sends it out S1/2.
The local tag is the tag locally assigned by this router to the FEC. It's the label that's advertised to the neighbors through LDP. The outgoing tag is the tag that's placed on a frame for a packet that fits the Prefix.

You can see the actual labels in the database (LIB – Label information base), instead of seeing the MPLS forwarding table (LFIB – Label Forwarding Information Base):

```
PE_Z#sh mpls ldp bind
  tib entry: 172.16.1.1/32, rev 2
        local binding:  tag: 16
        remote binding: tsr: 172.16.1.2:0, tag: 16
  tib entry: 172.16.1.2/32, rev 6
        local binding:  tag: 17
        remote binding: tsr: 172.16.1.2:0, tag: imp-null
  tib entry: 172.16.1.3/32, rev 4
        local binding:  tag: imp-null
        remote binding: tsr: 172.16.1.2:0, tag: 17
  tib entry: 192.168.1.0/30, rev 8
        local binding:  tag: 18
        remote binding: tsr: 172.16.1.2:0, tag: imp-null
  tib entry: 192.168.1.4/30, rev 10
        local binding:  tag: imp-null
        remote binding: tsr: 172.16.1.2:0, tag: imp-null
```

Directly connected networks get an imp-null label, which is actually label 3. This way a neighboring router knows that it has to pop the tag.

Let's see the MPLS forwarding table on the P router:

```
P#sh mpls forwarding-table
Local   Outgoing     Prefix              Bytes tag   Outgoing     Next Hop
tag     tag or VC    or Tunnel Id        switched    interface
16      Pop tag      172.16.1.1/32       0           Se1/0        point2point
17      Pop tag      172.16.1.3/32       0           Se1/1        point2point
```

As you can see when the P router gets a frame with label 16, it pops the label and forwards the frame to S1/0 (where PE_A, with IP 172.16.1.1/32 is). When it gets a frame with label 17, it pops the label and forwards the frame to s1/1 (where PE_Z is).

VRF configuration

Looking at figure 1 you can see that both s1/0 and s1/1 have the same IP address. This was done on purpose to show what a VPN is all about. Since client's A and B have overlapping networks, I also wanted overlapped IP addresses on the interfaces. To configure VRFs do the following:

```
PE_A(config)#ip vrf ClientA
PE_A(config-vrf)#route-target 64999:1
PE_A(config-vrf)#rd 999:1
PE_A(config-vrf)#ip vrf ClientB
PE_A(config-vrf)#route-target 64999:2
PE_A(config-vrf)#rd 999:2
```

This configures two VRFs, ClientA and ClientB. ClientA's VRF will contain the routes for the VPN ClientA is in. I defined a route target (aka VPN id) of 64999:1 and a route distinguisher (so that overlapping routes can be distinguished on the MP-BGP backbone) of 999:1. I could have made RT and RD the same, but I wanted to have different values, just to show you that you can have different values. ClientB's VRF configuration is similar, with 64999:2 and 999:2 as RT and RD respectively.

After having defined VRFs, you should assign them to an interface, so that traffic coming into that interface can use that VRF's routing table:

```
PE_A(config-vrf)#int s1/0
PE_A(config-if)#ip vrf forwarding ClientA
PE_A(config-if)#ip address 10.1.1.2 255.255.255.252
PE_A(config-if)#no shut
PE_A(config-if)#int s1/1
PE_A(config-if)#ip vrf forwarding ClientB
PE_A(config-if)#ip address 10.1.1.2 255.255.255.252
PE_A(config-if)#no shut
```

A similar configuration has to be done for PE_Z (the only difference being the IP address, which is 10.1.1.6/30).

MP-BGP configuration

Now that VRFs are configured, we have to have a way of communicating the routes they contain throughout the backbone. Since routes contained in VRFs have 96 bits each (32 for IP + 64 for RD), a multi protocol routing protocol has to be used. Alternatives here are IS-IS or MP-BGP. I chose MP-BGP since it's designed for the huge amount of routes you may have to handle in a VPN.

Configuring MP-BGP is surprisingly simple:

```
PE_A(config)#router bgp 64999
PE_A(config-router)#no bgp default ipv4-unicast
```

Just a small explanation before continuing: BGP by default exchanges IPv4 routes between neighbors, and starts that exchange as soon as the neighbors are identified. Since we're going to exchange VPN addresses, we have to disable this default behavior. This is done by issuing the *no bgp default ipv4-unicast* command.

```
PE_A(config-router)#neighbor 172.16.1.3 remote-as 64999
PE_A(config-router)#neighbor 172.16.1.3 update-source lo0
```

Above are your standard BGP neighbors enabling commands. Note that the BGP connection will not be established at this moment, since we've deactivated BGP's default behavior.

Now we define that we want to exchange vpnv4 routes (32+64 bits), and we activate the neighbor:

```
PE_A(config-router)#address-family vpnv4
PE_A(config-router-af)#neighbor 172.16.1.3 activate
```

After doing the same at PE_Z you get:

```
*Aug 26 15:58:41.131: %BGP-5-ADJCHANGE: neighbor 172.16.1.1 Up
```

Now we have an MP-BGP neighborship established :-)

```
PE_Z#sh ip bgp neigh 172.16.1.1
BGP neighbor is 172.16.1.1,   remote AS 64999, internal link
  BGP version 4, remote router ID 172.16.1.1
  BGP state = Established, up for 00:02:33
  Last read 00:00:33, last write 00:00:33, hold time is 180, keepalive interval is 60 seconds
  Neighbor capabilities:
    Route refresh: advertised and received(old & new)
    Address family VPNv4 Unicast: advertised and received
  Message statistics:
    InQ depth is 0
    OutQ depth is 0
                         Sent       Rcvd
    Opens:                  1          1
    Notifications:          0          0
    Updates:                0          0
    Keepalives:             5          5
    Route Refresh:          0          0
    Total:                  6          6
```

Default minimum time between advertisement runs is 0 seconds

RIP configuration

We now have to configure an IGP to exchange routes with our clients. We could do it via static routing, but having an IGP reduces OPEX :-)

I'm assuming that the necessary configuration has been done at the clients, something like:

```
router rip
version 2
network 10.0.0.0
no auto-summary
```

Now we go to the PEs, to configure RIP per VRF:

```
PE_A(config)#router rip
PE_A(config-router)#version 2
```

RIP can run a different « context » per VRF. To do it we must configure it per VRF, with the now familiar address-family command:

```
PE_A(config-router)#address-family ipv4 vrf ClientA
PE_A(config-router-af)#version 2
PE_A(config-router-af)#network 10.0.0.0
PE_A(config-router-af)#no auto-summary
```

We do the same for vrf ClientB, and we do a similar configuration at PE_Z.

Now we have RIP running between the PEs and CEs, and MP-BGP running between the PEs. We have ships in the night :-). Let's redistribute between RIP and BGP and vice-versa:

```
PE_A(config)#router bgp 64999
PE_A(config-router)#address-family ipv4 vrf ClientA
PE_A(config-router-af)#redistribute rip metric 1
PE_A(config-router-af)#address-family ipv4 vrf ClientB
PE_A(config-router-af)#redistribute rip metric 1
```

```
PE_A(config-router-af)#router rip
PE_A(config-router)#address-family ipv4 vrf ClientA
PE_A(config-router-af)#redistribute bgp 64999 metric 1
PE_A(config-router-af)#address-family ipv4 vrf ClientB
PE_A(config-router-af)#redistribute bgp 64999 metric 1
```

We do the same for PE_Z. Routes are now being redistributed into VRFs:

```
PE_Z#sh ip bgp vpnv4 all
BGP table version is 9, local router ID is 172.16.1.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 999:1 (default for vrf ClientA)
*>i10.1.1.0/30      172.16.1.1               0    100      0 ?
*> 10.1.1.4/30      0.0.0.0                  0             32768 ?
Route Distinguisher: 999:2 (default for vrf ClientB)
*>i10.1.1.0/30      172.16.1.1               0    100      0 ?
*> 10.1.1.4/30      0.0.0.0                  0             32768 ?
```

We're done!

A1 can now reach A2, and B1 can now reach B2.

```
A1>sh ip ro
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/30 is subnetted, 2 subnets
C       10.1.1.0 is directly connected, Serial1/0
R       10.1.1.4 [120/1] via 10.1.1.2, 00:00:22, Serial1/0
```

```
A1>ping 10.1.1.5

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.1.5, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 60/71/80 ms
```

Troubleshooting

I'm not going to state here typical troubleshooting techniques. Instead I'm going to talk about something that took me over 6 hours to figure out…

I did everything exactly like described above, with one very small difference: all the loopback addresses were /24 instead of /32.

When I enabled BGP in the PEs I got the following error/warning:

*Aug 26 06:59:34.559: %BGP-5-ADJCHANGE: neighbor 172.16.1.1 Up
*Aug 26 06:59:34.567: %BGP-4-VPNV4NH_MASK: Nexthop 172.16.1.3 may not be reachable from neigbor 172.16.1.1 - not /32 mask

I pinged 172.16.1.3 from PE_A and everything seemed to be fine, so I ignored this warning. Basic MPLS connectivity also seemed to be fine, since a "debug mpls packets" at the P router showed frames being switched.

After setting up the VPN, I pinged from A1 to A2 and… nothing. Since I was using Dynamips, I quickly blamed it on a Dynamips bug, and I set up this scenario in actual routers… still the same behavior. I then used Dynamips to capture traffic at various points and reached the conclusion that traffic was being dropped at the P router. I confirmed this with "debug mpls drops":

P#debug mpls drops
MPLS drops debugging is on
P#
*Aug 26 07:30:52.287: tagsw_replace_header: Pkt drop -- EoS conflict, incg label 16 hwinput Se1/1
*Aug 26 07:30:54.275: tagsw_replace_header: Pkt drop -- EoS conflict, incg label 16 hwinput Se1/1
*Aug 26 07:30:56.291: tagsw_replace_header: Pkt drop -- EoS conflict, incg label 16 hwinput Se1/1
*Aug 26 07:30:58.315: tagsw_replace_header: Pkt drop -- EoS conflict, incg label 16 hwinput Se1/1
*Aug 26 07:31:00.295: tagsw_replace_header: Pkt drop -- EoS conflict, incg label 16 hwinput Se1/1

I searched for these errors in Google and I got nothing!!! Has no one in the world ever experienced this!?

"debug mpls packets" confirmed that traffic was not exiting the P router:

P#debug mpls packets
MPLS packet debugging is on
P#
*Aug 26 07:32:22.599: MPLS: Se1/1: recvd: CoS=0, TTL=254, Label(s)=16/16
*Aug 26 07:32:24.579: MPLS: Se1/1: recvd: CoS=0, TTL=254, Label(s)=16/16
*Aug 26 07:32:26.559: MPLS: Se1/1: recvd: CoS=0, TTL=254, Label(s)=16/16
*Aug 26 07:32:28.591: MPLS: Se1/1: recvd: CoS=0, TTL=254, Label(s)=16/16
*Aug 26 07:32:30.595: MPLS: Se1/1: recvd: CoS=0, TTL=254, Label(s)=16/16

Armed with an extremely obsessive behavior (that's me :-)), I tried EVERYTHING. From changing the MTU between PE and P links, to changing the encapsulation being used.

Then I remembered that "smallish" error and decided to search for it in Google. I got this from Cisco's website:

%BGP-4-VPNV4NH_MASK : Nexthop [IP_address] may not be reachable from neigbor [IP_address] - not /32 mask

Explanation A VPNv4 route is being sent to the IBGP neighbor. The address of the next hop is a loopback interface that does not have a /32 mask defined. OSPF is being used on this loopback interface, and the OSPF network type of this interface is LOOPBACK. OSPF advertises this IP address as a host route (with mask /32), regardless of what mask is configured. This advertising conflicts with TDP, which uses configured masks, so the TDP neighbors may not receive a tag for the route indicated in this error message. This condition could break connectivity between sites that belong to the same VPN.

Recommended Action Configure the loopback that is being used as the next-hop loopback to use a 32-bit network mask (/32), or set the network type to point-to-point by entering the ip ospf network point-to-point command.

The explanation is quite simple: OSPF announced the loopback IP addresses as host routes (/32). LDP was expecting to find a /24 address at the routing table. Since it couldn't find it, it didn't advertise a label for this FEC!!

At the P router it's easy to see what's happening:

With /32 loopback addresses:

P#sh mpls for

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
18	Pop tag	172.16.1.1/32	214	Se1/0	point2point
19	Pop tag	172.16.1.3/32	126	Se1/1	point2point

With /24 loopback addresses:

P#sh mpls for

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
18	Untagged	172.16.1.1/32	535	Se1/0	point2point
19	Untagged	172.16.1.3/32	315	Se1/1	point2point

With /32 loopback addresses, LDP is able to announce the FEC for 172.16.1.x/32. It announces it with

an imp-null label (label 3), meaning that the penultimate router should pop the label.
With /24 loopback addresses, LDP doesn't announce anything regarding 172.16.1.x/32, since it's looking for 172.16.1.x/24 at the routing table. Consequently the P router doesn't get any labels for these FECs and tries to untag the frames coming in with the relevant labels.

Pop vs Untag

Pop purely pops the top label from the frame and forwards it to the relevant interface. This is the behavior I wanted for this scenario, since frames going through the P router have two labels, one for the VPN routes and another for the egress PE router.

Untag also pops the label, but it that label doesn't have the S bit (bottom of stack) set to 1, it drops the frame!!! AHA!! That was it!

I'm actually happy that this happened, since it got me to really understand MPLS and MPLS VPNs (I'm still half way through the book though, so I should learn a bit more). You should be happy too, since because of this you got to read this wonderful tutorial :-)

Please leave a comment if you found this tutorial useful. Consider it my payment for having spent 6 hours doing this.

Router configs and dynamips file

A1

```
A1#sh run
Building configuration...

Current configuration : 1441 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname A1
!
boot-start-marker
boot-end-marker
!
!
no aaa new-model
!
resource policy
!
```

```
ip cef
!
!
!
interface FastEthernet0/0
 no ip address
 shutdown
 duplex half
!
interface Serial1/0
 ip address 10.1.1.1 255.255.255.252
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/1
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/2
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/3
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/4
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/5
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/6
 no ip address
 shutdown
```

```
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/7
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
router rip
 version 2
 network 10.0.0.0
 no auto-summary
!
no ip http server
no ip http secure-server
!
!
!
logging alarm informational
!
!
control-plane
!
!
gatekeeper
 shutdown
!
!
line con 0
 stopbits 1
line aux 0
 stopbits 1
line vty 0 4
 login
!
!
end
```

B1

```
B1#sh run
Building configuration...

Current configuration : 1441 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname B1
!
boot-start-marker
boot-end-marker
!
!
no aaa new-model
!
resource policy
!
ip cef
!
!
!
interface FastEthernet0/0
 no ip address
 shutdown
 duplex half
!
interface Serial1/0
 ip address 10.1.1.1 255.255.255.252
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/1
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/2
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
```

```
!
interface Serial1/3
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/4
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/5
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/6
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/7
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
router rip
 version 2
 network 10.0.0.0
 no auto-summary
!
no ip http server
no ip http secure-server
!
!
!
logging alarm informational
!
!
control-plane
!
!
```

```
gatekeeper
 shutdown
!
!
line con 0
 stopbits 1
line aux 0
 stopbits 1
line vty 0 4
 login
!
!
End
```

PE_A

```
PE_A#sh run
Building configuration...

Current configuration : 2604 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname PE_A
!
boot-start-marker
boot-end-marker
!
!
no aaa new-model
!
resource policy
!
ip cef
!
!
!
```

```
ip vrf ClientA
 rd 999:1
 route-target export 64999:1
 route-target import 64999:1
!
ip vrf ClientB
 rd 999:2
 route-target export 64999:2
 route-target import 64999:2
!
!
!
interface Loopback0
 ip address 172.16.1.1 255.255.255.255
!
interface FastEthernet0/0
 no ip address
 shutdown
 duplex half
!
interface Serial1/0
 ip vrf forwarding ClientA
 ip address 10.1.1.2 255.255.255.252
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/1
 ip vrf forwarding ClientB
 ip address 10.1.1.2 255.255.255.252
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/2
 ip address 192.168.1.1 255.255.255.252
 mpls ip
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/3
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/4
 no ip address
 shutdown
```

```
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/5
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/6
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/7
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
router ospf 110
 log-adjacency-changes
 network 172.16.0.0 0.0.255.255 area 0
 network 192.168.1.0 0.0.0.255 area 0
!
router rip
 version 2
 !
 address-family ipv4 vrf ClientB
 redistribute bgp 64999 metric 1
 network 10.0.0.0
 no auto-summary
 version 2
 exit-address-family
 !
 address-family ipv4 vrf ClientA
 redistribute bgp 64999 metric 1
 network 10.0.0.0
 no auto-summary
 version 2
 exit-address-family
!
router bgp 64999
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 neighbor 172.16.1.3 remote-as 64999
```

```
 neighbor 172.16.1.3 update-source Loopback0
 !
 address-family vpnv4
 neighbor 172.16.1.3 activate
 neighbor 172.16.1.3 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf ClientB
 redistribute rip metric 1
 no synchronization
 exit-address-family
 !
 address-family ipv4 vrf ClientA
 redistribute rip metric 1
 no synchronization
 exit-address-family
!
no ip http server
no ip http secure-server
!
!
logging alarm informational
!
!
control-plane
!
!
gatekeeper
 shutdown
!
!
line con 0
 stopbits 1
line aux 0
 stopbits 1
line vty 0 4
 login
!
!
End
```

P

```
P#sh run
Building configuration...

Current configuration : 1596 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname P
!
boot-start-marker
boot-end-marker
!
!
no aaa new-model
!
resource policy
!
ip cef
!
!
!
interface Loopback0
 ip address 172.16.1.2 255.255.255.255
!
interface FastEthernet0/0
 no ip address
 shutdown
 duplex half
!
interface Serial1/0
 ip address 192.168.1.2 255.255.255.252
 mpls ip
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/1
 ip address 192.168.1.5 255.255.255.252
 mpls ip
```

```
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/2
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/3
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/4
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/5
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/6
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/7
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
router ospf 110
 log-adjacency-changes
 network 172.16.0.0 0.0.255.255 area 0
 network 192.168.1.0 0.0.0.255 area 0
!
no ip http server
no ip http secure-server
!
```

!
!
logging alarm informational
!
!
control-plane
!
!
gatekeeper
 shutdown
!
!
line con 0
 stopbits 1
line aux 0
 stopbits 1
line vty 0 4
 login
!
!
End

PE_Z

PE_Z#sh run
Building configuration...

Current configuration : 2604 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname PE_Z
!
boot-start-marker
boot-end-marker
!
!
no aaa new-model
!
resource policy

```
!
ip cef
!
!
!
!
ip vrf ClientA
 rd 999:1
 route-target export 64999:1
 route-target import 64999:1
!
ip vrf ClientB
 rd 999:2
 route-target export 64999:2
 route-target import 64999:2
!
!
interface Loopback0
 ip address 172.16.1.3 255.255.255.255
!
interface FastEthernet0/0
 no ip address
 shutdown
 duplex half
!
interface Serial1/0
 ip vrf forwarding ClientA
 ip address 10.1.1.6 255.255.255.252
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/1
 ip vrf forwarding ClientB
 ip address 10.1.1.6 255.255.255.252
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/2
 ip address 192.168.1.6 255.255.255.252
 mpls ip
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/3
 no ip address
 shutdown
 serial restart-delay 0
```

```
 no dce-terminal-timing-enable
!
interface Serial1/4
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/5
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/6
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/7
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
router ospf 110
 log-adjacency-changes
 network 172.16.0.0 0.0.255.255 area 0
 network 192.168.1.0 0.0.0.255 area 0
!
router rip
 version 2
 !
 address-family ipv4 vrf ClientB
 redistribute bgp 64999 metric 1
 network 10.0.0.0
 no auto-summary
 version 2
 exit-address-family
 !
 address-family ipv4 vrf ClientA
 redistribute bgp 64999 metric 1
 network 10.0.0.0
 no auto-summary
 version 2
 exit-address-family
```

```
!
router bgp 64999
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 neighbor 172.16.1.1 remote-as 64999
 neighbor 172.16.1.1 update-source Loopback0
 !
 address-family vpnv4
 neighbor 172.16.1.1 activate
 neighbor 172.16.1.1 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf ClientB
 redistribute rip metric 1
 no synchronization
 exit-address-family
 !
 address-family ipv4 vrf ClientA
 redistribute rip metric 1
 no synchronization
 exit-address-family
!
no ip http server
no ip http secure-server
!
!
!
logging alarm informational
!
!
control-plane
!
!
!
gatekeeper
 shutdown
!
!
line con 0
 stopbits 1
line aux 0
 stopbits 1
line vty 0 4
 login
!
!
end
```

A2

```
A2#sh run
Building configuration...

Current configuration : 1441 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname A2
!
boot-start-marker
boot-end-marker
!
!
no aaa new-model
!
resource policy
!
ip cef
!
!
interface FastEthernet0/0
 no ip address
 shutdown
 duplex half
!
interface Serial1/0
 ip address 10.1.1.5 255.255.255.252
 serial restart-delay 0
 no dce-terminal-timing-enable
!
```

```
interface Serial1/1
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/2
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/3
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/4
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/5
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/6
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/7
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
router rip
 version 2
 network 10.0.0.0
 no auto-summary
!
```

no ip http server
no ip http secure-server
!
!
!
logging alarm informational
!
!
!
control-plane
!
!
!
gatekeeper
 shutdown
!
!
line con 0
 stopbits 1
line aux 0
 stopbits 1
line vty 0 4
 login
!
!
End

B2

B2#sh run
Building configuration...

Current configuration : 1441 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname B2
!
boot-start-marker
boot-end-marker
!

```
!
no aaa new-model
!
resource policy
!
ip cef
!
!
!
interface FastEthernet0/0
 no ip address
 shutdown
 duplex half
!
interface Serial1/0
 ip address 10.1.1.5 255.255.255.252
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/1
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/2
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/3
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/4
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/5
 no ip address
 shutdown
 serial restart-delay 0
```

```
 no dce-terminal-timing-enable
!
interface Serial1/6
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
interface Serial1/7
 no ip address
 shutdown
 serial restart-delay 0
 no dce-terminal-timing-enable
!
router rip
 version 2
 network 10.0.0.0
 no auto-summary
!
no ip http server
no ip http secure-server
!
!
!
logging alarm informational
!
!
!
control-plane
!
!
!
gatekeeper
 shutdown
!
!
line con 0
 stopbits 1
line aux 0
 stopbits 1
line vty 0 4
 login
!
!
End
```

Dynamips file

```
# Simple lab

autostart = False


[localhost]

    [[7200]]
        #image = \Program Files\Dynamips\images\c7200-adventerprise.124-6.image
        # On Linux / Unix use forward slashes:
        image = /home/alex/cisco_images/c7200-advent.124-6.image
        mmap = False
        npe = npe-400
        ram = 256
        idlepc = 0x60713e28


    [[ROUTER A1]]
        s1/0 = PE_A s1/0
        model = 7200

    [[router B1]]
        model = 7200
        s1/0 = PE_A s1/1

    [[router PE_A]]
        model = 7200
        s1/2 = P s1/0

    [[router PE_Z]]
        model = 7200
        s1/2 = P s1/1

    [[router P]]
        model = 7200

    [[router A2]]
        model = 7200
        s1/0 = PE_Z s1/0

    [[router B2]]
        model = 7200
        s1/0 = PE_Z s1/1
```