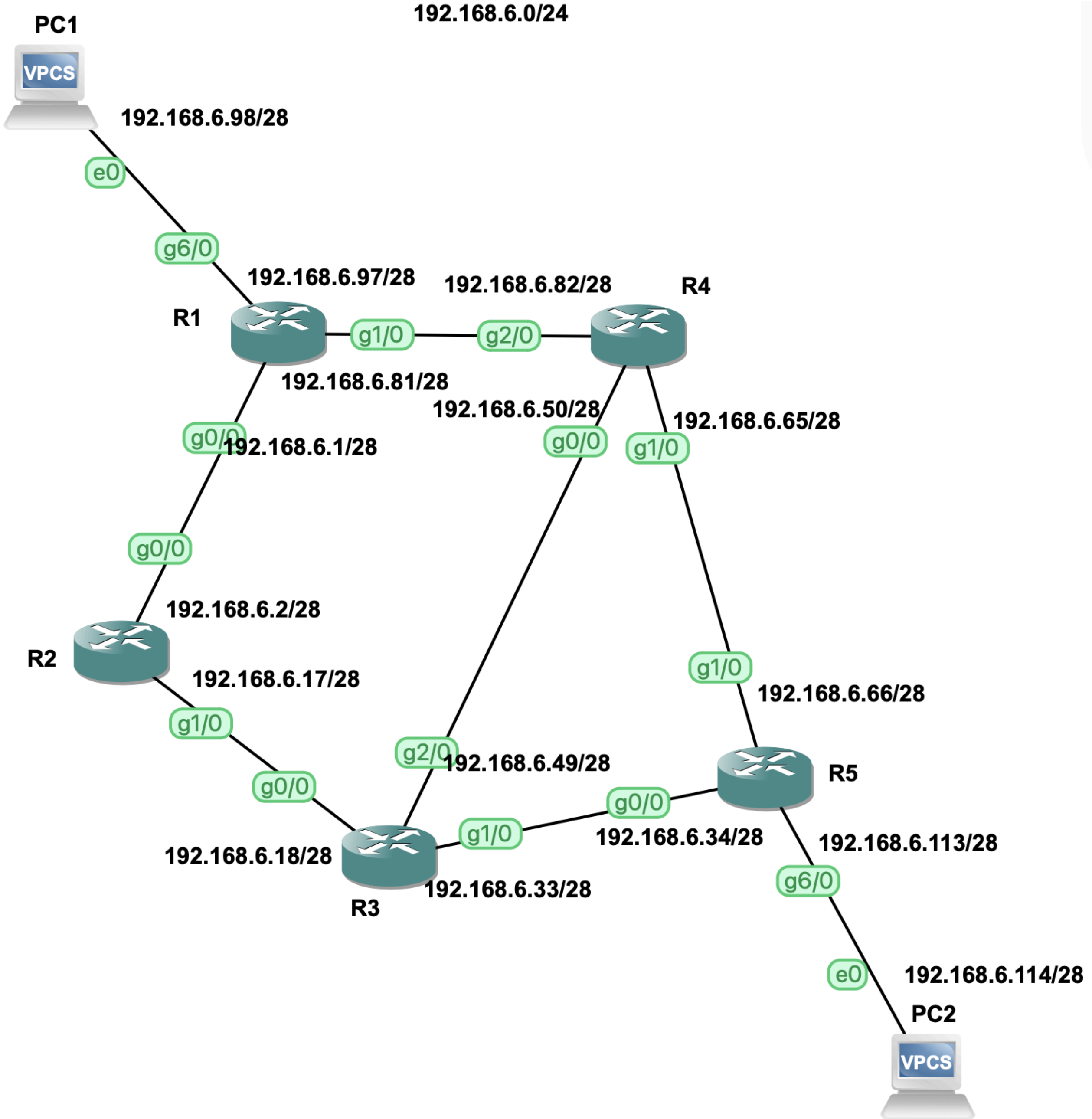
Traffic Engineering

Lab 2 Report

Group 06

João Pargana, 93592
Samuel Barata, 94230

Network Topology



Lab 2a

We decided to split the network into /28 segments since it left more subnets for future expansion and also enough hosts to put on each network. This allowed us to have up to 14 hosts per network and up to 16 networks.
For routing, we implemented RIP since it is the easiest protocol.
Since RIP takes up to 3 minutes to converge with the default setting on the routers we're using, we changed the RIP default timers so that it converges in less than 6 seconds.

For testing, we opened wireshark to check the route from PC1 to PC2 and we started shutting down those links and turning them back up. As expected, the network was able to adapt quickly without dropping packets.

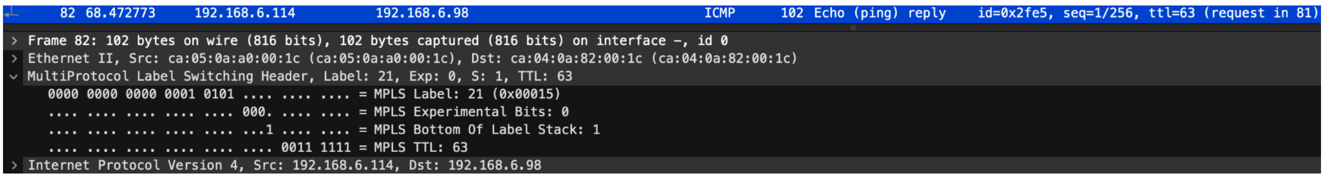
Lab 2b

Configuration of MPLS

To configure mpls on the core network we used the command `ip mpls` on each of the interfaces connected to the other routers.
To test if mpls was working between all routers we checked the mpls forwarding-table:

```
R1#show mpls forwarding-table
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
19         Pop Label  192.168.6.16/28  0             Gi0/0      192.168.6.2
20         20         192.168.6.32/28  0             Gi0/0      192.168.6.2
           22         192.168.6.32/28  0             Gi1/0      192.168.6.82
21         Pop Label  192.168.6.48/28  0             Gi1/0      192.168.6.82
22         Pop Label  192.168.6.64/28  0             Gi1/0      192.168.6.82
23         23         192.168.6.112/28 0             Gi1/0      192.168.6.82
```

As we can see, there is a route to every subnet.
For further testing we used wireshark to capture a ping between both PCs. Looking at the image below we can see that MPLS is applying labels to this traffic.



Default LSP

The default LSP (Label Switched Path) is the path that packets take through the MPLS network when no explicit path is configured.
Using a traceroute between PC1 and PC2 we can observe this path

```
PC1> trace 192.168.6.114 -P 1
trace to 192.168.6.114, 8 hops max (ICMP), press Ctrl+C to stop
 1   192.168.6.97    9.403  ms  9.929  ms  9.397 ms
 2   192.168.6.82    29.967 ms  29.438 ms  30.036 ms
 3   192.168.6.66    39.567 ms  39.584 ms  39.557 ms
 4   192.168.6.114   59.648 ms  39.469 ms  39.439 ms
```

As we can see, the default LSP is PC1 -> R1 -> R4 ->R5 -> PC2

Routing Table

```
R1#sh ip route
      192.168.6.0/24 is variably subnetted, 11 subnets, 2 masks
C        192.168.6.0/28 is directly connected, GigabitEthernet0/0
L        192.168.6.1/32 is directly connected, GigabitEthernet0/0
R        192.168.6.16/28 [120/1] via 192.168.6.2, 00:00:17, GigabitEthernet0/0
R        192.168.6.32/28
           [120/2] via 192.168.6.82, 00:00:05, GigabitEthernet1/0
           [120/2] via 192.168.6.2, 00:00:17, GigabitEthernet0/0
R        192.168.6.48/28
           [120/1] via 192.168.6.82, 00:00:05, GigabitEthernet1/0
R        192.168.6.64/28
           [120/1] via 192.168.6.82, 00:00:05, GigabitEthernet1/0
C        192.168.6.80/28 is directly connected, GigabitEthernet1/0
L        192.168.6.81/32 is directly connected, GigabitEthernet1/0
C        192.168.6.96/28 is directly connected, GigabitEthernet6/0
L        192.168.6.97/32 is directly connected, GigabitEthernet6/0
R        192.168.6.112/28
           [120/2] via 192.168.6.82, 00:00:05, GigabitEthernet1/0
```

LIB

```
R1#show mpls ldp bindings
  lib entry: 192.168.6.0/28, rev 7
    local binding:  label: imp-null
    remote binding: lsr: 192.168.6.17:0, label: imp-null
    remote binding: lsr: 192.168.6.82:0, label: 19
  lib entry: 192.168.6.16/28, rev 11
    local binding:  label: 19
    remote binding: lsr: 192.168.6.17:0, label: imp-null
    remote binding: lsr: 192.168.6.82:0, label: 20
  lib entry: 192.168.6.32/28, rev 13
    local binding:  label: 20
    remote binding: lsr: 192.168.6.17:0, label: 20
    remote binding: lsr: 192.168.6.82:0, label: 22
  lib entry: 192.168.6.48/28, rev 15
    local binding:  label: 21
    remote binding: lsr: 192.168.6.17:0, label: 21
    remote binding: lsr: 192.168.6.82:0, label: imp-null
  lib entry: 192.168.6.64/28, rev 17
    local binding:  label: 22
    remote binding: lsr: 192.168.6.82:0, label: imp-null
```

```
    remote binding: lsr: 192.168.6.17:0, label: 22
  lib entry: 192.168.6.80/28, rev 8
    local binding:  label: imp-null
    remote binding: lsr: 192.168.6.17:0, label: 18
        remote binding: lsr: 192.168.6.82:0, label: imp-null
  lib entry: 192.168.6.96/28, rev 9
    local binding:  label: imp-null
    remote binding: lsr: 192.168.6.17:0, label: 19
    remote binding: lsr: 192.168.6.82:0, label: 21
  lib entry: 192.168.6.112/28, rev 19
    local binding:  label: 23
    remote binding: lsr: 192.168.6.82:0, label: 23
    remote binding: lsr: 192.168.6.17:0, label: 23
```

LFIB

```
R1#show mpls forwarding
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
19         Pop Label  192.168.6.16/28  0             Gi0/0      192.168.6.2
20         20         192.168.6.32/28  0             Gi0/0      192.168.6.2
           22         192.168.6.32/28  0             Gi1/0      192.168.6.82
21         Pop Label  192.168.6.48/28  0             Gi1/0      192.168.6.82
22         Pop Label  192.168.6.64/28  0             Gi1/0      192.168.6.82
23         23         192.168.6.112/28 0             Gi1/0      192.168.6.82
```

Introducing a Fault

We decided to shutdown link between router R4 and R5. After 3 seconds of network instability, packets from PC1 started reaching PC2 correctly: PC1 -> R1 -> R4 -> R3 -> R5 -> PC2 and fot the reply the packets started following the path PC2 -> R5 -> R3 -> R2 -> R1 -> PC1.

We've checked the routing table of R3 and R5;
For traffic from PC2 to reach PC1, R5 would forward it to R3, and R3 would either forward it to R4 or R2 since they have the same preference. [120/2].

LIB

```
R5#show mpls ldp bindings
  lib entry: 192.168.6.0/28, rev 11
    local binding:  label: 19
    remote binding: lsr: 192.168.6.49:0, label: 19
  lib entry: 192.168.6.16/28, rev 13
    local binding:  label: 20
    remote binding: lsr: 192.168.6.49:0, label: imp-null
  lib entry: 192.168.6.32/28, rev 7
    local binding:  label: imp-null
    remote binding: lsr: 192.168.6.49:0, label: imp-null
  lib entry: 192.168.6.48/28, rev 15
    local binding:  label: 21
    remote binding: lsr: 192.168.6.49:0, label: imp-null
  lib entry: 192.168.6.64/28, rev 8
    local binding:  label: imp-null
    remote binding: lsr: 192.168.6.49:0, label: 20
  lib entry: 192.168.6.80/28, rev 17
    local binding:  label: 22
    remote binding: lsr: 192.168.6.49:0, label: 21
  lib entry: 192.168.6.96/28, rev 19
    local binding:  label: 23
    remote binding: lsr: 192.168.6.49:0, label: 22
  lib entry: 192.168.6.112/28, rev 9
    local binding:  label: imp-null
        remote binding: lsr: 192.168.6.49:0, label: 23
```

LFIB

```
R3#show mpls forwarding
Local      Outgoing   Prefix           Bytes Label   Outgoing    Next Hop
Label      Label      or Tunnel Id     Switched      interface
19         Pop Label  192.168.6.0/28   0             Gi0/0       192.168.6.17
20         Pop Label  192.168.6.64/28  0             Gi1/0       192.168.6.34
21         Pop Label  192.168.6.80/28  54            Gi2/0       192.168.6.50
22         19         192.168.6.96/28  68544         Gi0/0       192.168.6.17
           21         192.168.6.96/28  0             Gi2/0       192.168.6.50
23         Pop Label  192.168.6.112/28 77446         Gi1/0       192.168.6.34
```

After restoring the link, the original path was restored.

Penultimate Hop Popping

In MPLS the labels are popped before they reach the router that has a direct connection to the destination network.
We can verify this using wireshark and the 2 PCs on both edges of the network.
We'll ping PC2 from PC1 and analyse the tags throughout the network.

PC1 -> PC2

PC1 sends a normal ICMP request to R1 since it's not part of the MPLS core, R1 forwards the packet to R3 over MPLS using the label 23.

```
2428  1314.219702    192.168.6.98             192.168.6.114                            ICMP      102  Echo (ping) request
> Frame 2428: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface -, id 0
> Ethernet II, Src: ca:01:0a:28:00:1c (ca:01:0a:28:00:1c), Dst: ca:04:0a:82:00:38 (ca:04:0a:82:00:38)
v MultiProtocol Label Switching Header, Label: 23, Exp: 0, S: 1, TTL: 63
    0000 0000 0000 0001 0111 .... .... .... = MPLS Label: 23 (0x00017)
    .... .... .... .... .... 000. .... .... = MPLS Experimental Bits: 0
    .... .... .... .... .... ...1 .... .... = MPLS Bottom Of Label Stack: 1
    .... .... .... .... .... .... 0011 1111 = MPLS TTL: 63
> Internet Protocol Version 4, Src: 192.168.6.98, Dst: 192.168.6.114
> Internet Control Message Protocol
```

In R4, if we see the MPLS forwarding table we can verify that the tag 23 will be popped and the packet forwarded to 192.168.6.66 (R5).

```
R4#show mpls forwarding
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
19         Pop Label  192.168.6.0/28   0             Gi2/0      192.168.6.81
20         Pop Label  192.168.6.16/28  0             Gi0/0      192.168.6.49
21         Pop Label  192.168.6.96/28  27784         Gi2/0      192.168.6.81
22         Pop Label  192.168.6.32/28  0             Gi0/0      192.168.6.49
           Pop Label  192.168.6.32/28  0             Gi1/0      192.168.6.66
23         Pop Label  192.168.6.112/28 23422         Gi1/0      192.168.6.66
```

In the wireshark capture we can see that the label is not forwarded to R5.

```
925  331.561383     192.168.6.98             192.168.6.114                            ICMP      98  Echo (ping) request
> Frame 925: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface -, id 0
> Ethernet II, Src: ca:04:0a:82:00:1c (ca:04:0a:82:00:1c), Dst: ca:05:0a:a0:00:1c (ca:05:0a:a0:00:1c)
> Internet Protocol Version 4, Src: 192.168.6.98, Dst: 192.168.6.114
> Internet Control Message Protocol
```

R5 won't need a label since it's directly connected to the destination network, it won't need further instructions.

PC2 -> PC1

In the return path of the ICMP packets the same occurs.
In R5 we can see the bindings it will preform

```
R5#show mpls ldp bindings
  lib entry: 192.168.6.0/28, rev 11
    local binding:  label: 19
    remote binding: lsr: 192.168.6.49:0, label: 19
    remote binding: lsr: 192.168.6.82:0, label: 19
  lib entry: 192.168.6.16/28, rev 13
    local binding:  label: 20
    remote binding: lsr: 192.168.6.49:0, label: imp-null
    remote binding: lsr: 192.168.6.82:0, label: 20
  lib entry: 192.168.6.32/28, rev 7
    local binding:  label: imp-null
    remote binding: lsr: 192.168.6.49:0, label: imp-null
    remote binding: lsr: 192.168.6.82:0, label: 22
  lib entry: 192.168.6.48/28, rev 15
    local binding:  label: 21
    remote binding: lsr: 192.168.6.49:0, label: imp-null
    remote binding: lsr: 192.168.6.82:0, label: imp-null
  lib entry: 192.168.6.64/28, rev 8
    local binding:  label: imp-null
    remote binding: lsr: 192.168.6.49:0, label: 20
```

```
    remote binding: lsr: 192.168.6.82:0, label: imp-null
  lib entry: 192.168.6.80/28, rev 17
    local binding:  label: 22
    remote binding: lsr: 192.168.6.49:0, label: 21
        remote binding: lsr: 192.168.6.82:0, label: imp-null
  lib entry: 192.168.6.96/28, rev 19
    local binding:  label: 23
    remote binding: lsr: 192.168.6.49:0, label: 22
    remote binding: lsr: 192.168.6.82:0, label: 21
  lib entry: 192.168.6.112/28, rev 9
    local binding:  label: imp-null
    remote binding: lsr: 192.168.6.49:0, label: 23
    remote binding: lsr: 192.168.6.82:0, label: 23
```

The router will forward the packet to network 192.168.6.96/28 network through router R4, this means it will apply the label 21. We can verify this using wireshark

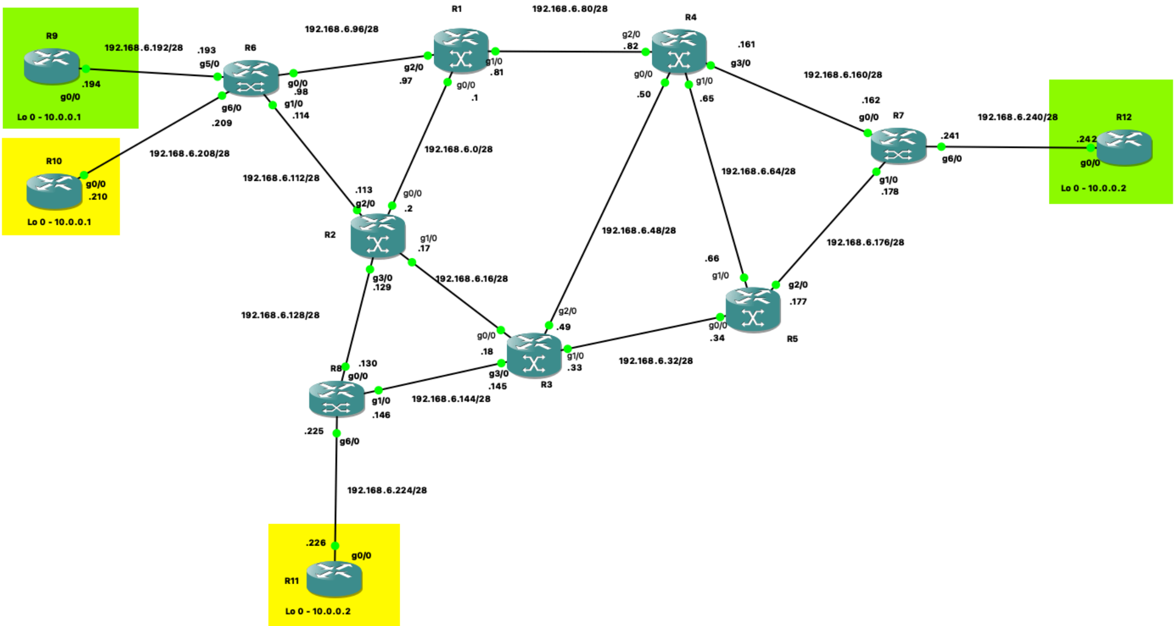
```
1660 593.006552   192.168.6.114   192.168.6.98   ICMP   102 Echo (ping) reply
> Frame 1660: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface -, id 0
> Ethernet II, Src: ca:05:0a:a0:00:1c (ca:05:0a:a0:00:1c), Dst: ca:04:0a:82:00:1c (ca:04:0a:82:00:1c)
v MultiProtocol Label Switching Header, Label: 21, Exp: 0, S: 1, TTL: 63
    0000 0000 0000 0001 0101 .... .... .... = MPLS Label: 21 (0x00015)
    .... .... .... .... .... 000. .... .... = MPLS Experimental Bits: 0
    .... .... .... .... .... ...1 .... .... = MPLS Bottom Of Label Stack: 1
    .... .... .... .... .... .... 0011 1111 = MPLS TTL: 63
> Internet Protocol Version 4, Src: 192.168.6.114, Dst: 192.168.6.98
```

And as before, checking the MPLS forwarding table on R4 we can see that label 21 will be forwarded to R1 and its label popped.

```
R4#show mpls forwarding
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
19         Pop Label  192.168.6.0/28   0             Gi2/0      192.168.6.81
20         Pop Label  192.168.6.16/28  0             Gi0/0      192.168.6.49
21         Pop Label  192.168.6.96/28  27784         Gi2/0      192.168.6.81
22         Pop Label  192.168.6.32/28  0             Gi0/0      192.168.6.49
           Pop Label  192.168.6.32/28  0             Gi1/0      192.168.6.66
23         Pop Label  192.168.6.112/28 23422         Gi1/0      192.168.6.66
```

Lab 2c

Network Topology



We added three Label Edge Routers and 2 clients, each client present in 2 remote locations.

Configuration

We started by reconfiguring the MPLS Core to use OSPF using `area 0` instead of RIP as we thought this would facilitate the rest of the lab.

To configure this network, in the Client Edges routers we simply configured OSPF using `area 2` with the Provider Edges routers so they could learn the routes from the other locations.

The configuration of the Provider Edges was a bit more complex. It shares the configuration from the core MPLS network (OSPF with the core and MPLS on the provider interfaces). For the VPN to work, we created BGP sessions between all Provider Edge routers, this means they can exchange label information and OSPF routes for the connected clients. We also created 2 VRFs, one for the green and one for the yellow, we set the interfaces for each router with that VRF and a specific OSPF process. We redistribute the OSPF into the BGP and the BGP into the OSPF so the routes and labels get distributed to the other provider edges routers.

```
ip vrf GREEN
 rd 9:9
 route-target export 9:9
 route-target import 9:9
!
ip vrf YELLOW
 rd 10:10
 route-target export 10:10
 route-target import 10:10
!
```

```
interface Loopback0
 ip address 6.6.6.6 255.255.255.255
 ip ospf 1 area 0
!
interface GigabitEthernet5/0
 ip vrf forwarding GREEN
 ip address 192.168.6.193 255.255.255.240
 ip ospf 2 area 2
!
interface GigabitEthernet6/0
 ip vrf forwarding YELLOW
 ip address 192.168.6.209 255.255.255.240
 ip ospf 3 area 2
!
router ospf 2 vrf GREEN
 redistribute bgp 1 subnets
!
router ospf 3 vrf YELLOW
 redistribute bgp 1 subnets
!
router ospf 1
 mpls ldp autoconfig
 router-id 6.6.6.6
!
router bgp 1
 bgp log-neighbor-changes
 neighbor 7.7.7.7 remote-as 1
 neighbor 7.7.7.7 update-source Loopback0
 neighbor 8.8.8.8 remote-as 1
 neighbor 8.8.8.8 update-source Loopback0
 !
 address-family vpnv4
  neighbor 7.7.7.7 activate
  neighbor 7.7.7.7 send-community extended
  neighbor 8.8.8.8 activate
  neighbor 8.8.8.8 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf GREEN
  redistribute ospf 2
 exit-address-family
 !
 address-family ipv4 vrf YELLOW
  redistribute ospf 3
 exit-address-family
!
```

This configuration has different OSPF processes, which means that the green client's routes will not get sent to the yellow clients and vice-versa.

To better understand how this worked we captured a BGP packet with the information for the green clients.

```
      1 0.000000       7.7.7.7                6.6.6.6               BGP       189 UPDATE Message

▸ Frame 1: 189 bytes on wire (1512 bits), 189 bytes captured (1512 bits) on interface -, id 0
▸ Ethernet II, Src: ca:07:08:f7:00:08 (ca:07:08:f7:00:08), Dst: ca:04:08:9d:00:54 (ca:04:08:9d:00:54)
▸ MultiProtocol Label Switching Header, Label: 22, Exp: 6, S: 1, TTL: 255
▸ Internet Protocol Version 4, Src: 7.7.7.7, Dst: 6.6.6.6
▸ Transmission Control Protocol, Src Port: 57798, Dst Port: 179, Seq: 1, Ack: 1, Len: 131
▾ Border Gateway Protocol - UPDATE Message
    Marker: ffffffffffffffffffffffffffffffff
    Length: 131
    Type: UPDATE Message (2)
    Withdrawn Routes Length: 0
    Total Path Attribute Length: 108
  ▾ Path attributes
    ▸ Path Attribute - ORIGIN: INCOMPLETE
    ▸ Path Attribute - AS_PATH: empty
    ▸ Path Attribute - MULTI_EXIT_DISC: 2
    ▸ Path Attribute - LOCAL_PREF: 100
    ▾ Path Attribute - EXTENDED_COMMUNITIES
      ▸ Flags: 0xc0, Optional, Transitive, Complete
        Type Code: EXTENDED_COMMUNITIES (16)
        Length: 32
      ▾ Carried extended communities: (4 communities)
        ▸ Route Target: 9:9 [Transitive 2-Octet AS-Specific]
        ▸ OSPF Domain Identifier: 0:131584 [Transitive 2-Octet AS-Specific]
        ▸ OSPF Route Type: Area: 0.0.0.2, Type: Network [Generic Transitive Experimental Use]
        ▸ OSPF Router ID: 192.168.6.241 [Generic Transitive Experimental Use]
    ▾ Path Attribute - MP_REACH_NLRI
      ▸ Flags: 0x80, Optional, Non-transitive, Complete
        Type Code: MP_REACH_NLRI (14)
        Length: 49
        Address family identifier (AFI): IPv4 (1)
        Subsequent address family identifier (SAFI): Labeled VPN Unicast (128)
      ▸ Next hop:  RD=0:0 IPv4=7.7.7.7
        Number of Subnetwork points of attachment (SNPA): 0
      ▾ Network Layer Reachability Information (NLRI)
        ▾ BGP Prefix
            Prefix Length: 120
            Label Stack: 25 (bottom)
            Route Distinguisher: 9:9
            MP Reach NLRI IPv4 prefix: 10.0.0.2
        ▾ BGP Prefix
            Prefix Length: 120
            Label Stack: 24 (bottom)
            Route Distinguisher: 9:9
            MP Reach NLRI IPv4 prefix: 12.12.12.12
```

As we can see, in the MP_REACH_NLRI, BGP sends the route distinguisher (9:9 is the green client), the ipv4 network, and the label it expects to receive to deliver packets to this specific client.

To test the VPN we added 2 loopback interfaces to each Client Edge:

- R10 (Yellow) - 10.10.10.10/32 & 10.0.0.1/32
- R11 (Yellow) - 11.11.11.11/32 & 10.0.0.2/32
- R9 (Green) - 9.9.9.9/32 & 10.0.0.1/32
- R12 (Green) - 12.12.12.12/32 & 10.0.0.2/32

We tested the connectivity using pings, as we can see in the image below, R9 can ping the other green clients, but not the yellow ones.

```
R9#ping 10.10.10.10
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.10.10, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
R9#ping 12.12.12.12
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 12.12.12.12, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 48/50/52 ms
```

We can also see the routing table of R9 and conclude it didn't learn the route to R10 or R11

```
      9.0.0.0/32 is subnetted, 1 subnets
C        9.9.9.9 is directly connected, Loopback0
      10.0.0.0/32 is subnetted, 2 subnets
C        10.0.0.1 is directly connected, Loopback1
O IA     10.0.0.2 [110/3] via 192.168.6.193, 00:03:41, GigabitEthernet0/0
      12.0.0.0/32 is subnetted, 1 subnets
O IA     12.12.12.12 [110/3] via 192.168.6.193, 00:03:41, GigabitEthernet0/0
      192.168.6.0/24 is variably subnetted, 3 subnets, 2 masks
C        192.168.6.192/28 is directly connected, GigabitEthernet0/0
L        192.168.6.194/32 is directly connected, GigabitEthernet0/0
O IA     192.168.6.240/28
           [110/2] via 192.168.6.193, 00:03:41, GigabitEthernet0/0
```

To test the same IPs on different clients we did a trace from R12 and R11 to R9 and R10 (which are connected to the same PE router)

```
R11#trace 10.0.0.1
Type escape sequence to abort.
Tracing the route to 10.0.0.1
VRF info: (vrf in name/id, vrf out name/id)
  1 192.168.6.225 8 msec 8 msec 8 msec
  2 192.168.6.129 [MPLS: Labels 21/31 Exp 0] 40 msec 36 msec 40 msec
  3 192.168.6.209 [MPLS: Label 31 Exp 0] 28 msec 32 msec 28 msec
  4 192.168.6.210 40 msec 52 msec 48 msec
```

```
R12#trace 10.0.0.1
Type escape sequence to abort.
Tracing the route to 10.0.0.1
VRF info: (vrf in name/id, vrf out name/id)
  1 192.168.6.241 20 msec 20 msec 8 msec
  2 192.168.6.161 [MPLS: Labels 22/29 Exp 0] 52 msec 72 msec 60 msec
  3 192.168.6.81 [MPLS: Labels 17/29 Exp 0] 68 msec 60 msec 60 msec
  4 192.168.6.193 [MPLS: Label 29 Exp 0] 60 msec 52 msec 36 msec
  5 192.168.6.194 68 msec 60 msec 52 msec
```

We can see the MPLS labels, the first label is the normal label for the MPLS core, and the second label is the route distinguisher label, that's how the PE router knows to which client the packet should be sent.

We can confirm this by looking at the Provider Edge mpls forwarding table:

```
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
16         20         8.8.8.8/32       0             Gi1/0      192.168.6.113
17         18         192.168.6.176/28 0             Gi0/0      192.168.6.97
           22         192.168.6.176/28 0             Gi1/0      192.168.6.113
18         19         192.168.6.64/28  0             Gi0/0      192.168.6.97
19         Pop Label  192.168.6.16/28  0             Gi1/0      192.168.6.113
20         Pop Label  192.168.6.0/28   0             Gi0/0      192.168.6.97
           Pop Label  192.168.6.0/28   0             Gi1/0      192.168.6.113
21         25         192.168.6.32/28  0             Gi1/0      192.168.6.113
22         25         192.168.6.160/28 0             Gi0/0      192.168.6.97
23         26         192.168.6.48/28  0             Gi0/0      192.168.6.97
           26         192.168.6.48/28  0             Gi1/0      192.168.6.113
24         29         192.168.6.144/28 0             Gi1/0      192.168.6.113
25         Pop Label  192.168.6.128/28 482           Gi1/0      192.168.6.113
26         Pop Label  192.168.6.80/28  0             Gi0/0      192.168.6.97
27         27         7.7.7.7/32       0             Gi0/0      192.168.6.97
28         No Label   9.9.9.9/32[V]    0             Gi5/0      192.168.6.194
29         No Label   10.0.0.1/32[V]   1242          Gi5/0      192.168.6.194
30         No Label   192.168.6.192/28[V]   \
                                       6186          aggregate/GREEN
31         No Label   10.0.0.1/32[V]   1998          Gi6/0      192.168.6.210
32         No Label   10.10.10.10/32[V]    \
                                       0             Gi6/0      192.168.6.210
33         No Label   192.168.6.208/28[V]   \
                                       1314          aggregate/YELLOW
```

Labels 29 and 31 are for the same IP prefix but are forwarded to different interfaces.