Manual:MPLSVPLS

From MikroTik Wiki

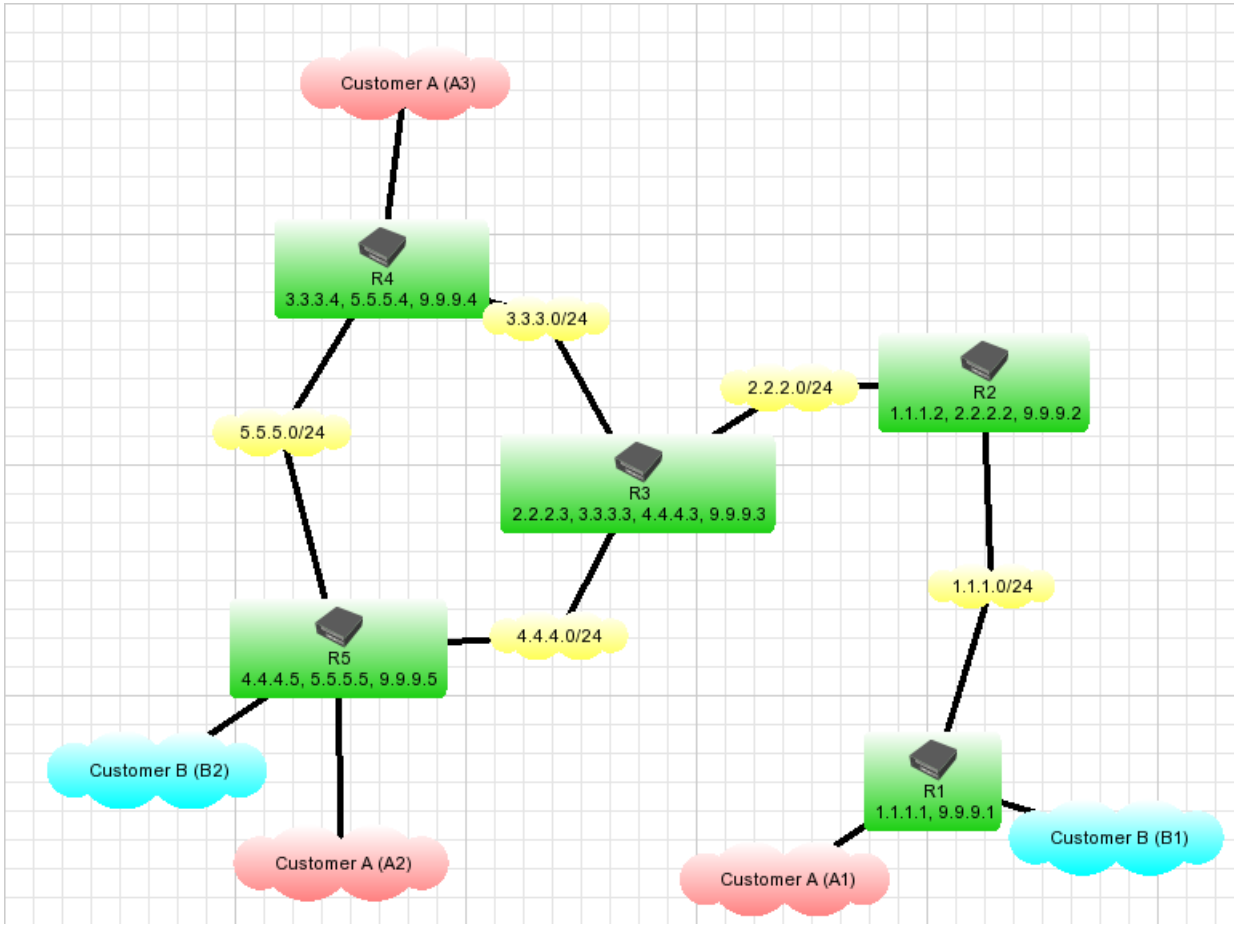
Contents

MPLS Overview

For MPLS overview and MPLS features that RouterOS supports see MPLS Overview

Example network

Consider network service provider that is connecting 3 remote sites of Customer A (A1,A2 and A3) and 2 remote sites of Customer B (B1 and B2) using its routed IP network core, consisting of routers (R1-R5):



Customers require transparent ethernet segment connection between sites. So far it has been implemented by means of bridging EoIP tunnels with physical ethernet interfaces.

Note that there are no IP addresses configured on R1, R4 and R5 interfaces that face customer networks.

Enabling MPLS forwarding can speed up packet forwarding process in such network. Using one of MPLS applications - VPLS can further increase efficency of ethernet frame forwarding by not having to encapsulate ethernet frames in IP frames, thus removing IP header overhead.

This guide gives step by step instructions that will lead to implementation of VPLS to achieve necessary service.

Prerequisites for MPLS

"Loopback" IP address

Although not a strict requirement, it is advisable to configure routers participating in MPLS network with "loopback" IP addresses (not attached to any real network interface) to be used by LDP to establish sessions.

This serves 2 purposes:

- as there is only one LDP session between any 2 routers, no matter how many links connect them, loopback IP address ensures that LDP session is not affected by interface state or address changes
- use of loopback address as LDP transport address ensures proper penultimate hop popping behaviour when multiple labels are attached to packet as in case of VPLS

In RouterOS "loopback" IP address can be configured by creating dummy bridge interface without any ports and adding address to it. For example, on R1 it is done with the following commands:

```
/interface bridge add name=lobridge
/ip address add address=9.9.9.1/32 interface=lobridge
```

The rest of routers are configured similar way.

IP connectivity

As LDP distributes labels for active routes, essential requirement is properly configured IP routing. LDP by default distributes labels for active IGP routes (that is - connected, static, and routing protocol learned routes, except BGP).

In given example setup OSPF is used to distribute routes. For example, on R5 OSPF is configured with the following commands:

```
/routing ospf instance set redistribute-connected=as-type-1
/routing ospf network add area=backbone network=4.4.4.0/24
/routing ospf network add area=backbone network=5.5.5.0/24
```

On other routers OSPF is configured in similar way.

This yields routing table on R5 like this:

```
[admin@R5] > ip route print
Flags: X - disabled, A - active, D - dynamic, C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY-STATE GATEWAY         DISTANCE             INTERFACE
 0 ADo  1.1.1.0/24                         reachable     4.4.4.3         110                  ether1
 1 ADo  2.2.2.0/24                         reachable     4.4.4.3         110                  ether1
 2 ADo  3.3.3.0/24                         reachable     4.4.4.3         110                  ether1
                                           reachable     5.5.5.4                              ether2
 3 ADC  4.4.4.0/24         4.4.4.5                                       0                    ether1
 4 ADC  5.5.5.0/24         5.5.5.5                                       0                    ether2
 5 ADo  9.9.9.1/32                         reachable     4.4.4.3         110                  ether1
 6 ADo  9.9.9.2/32                         reachable     4.4.4.3         110                  ether1
 7 ADo  9.9.9.3/32                         reachable     4.4.4.3         110                  ether1
 8 ADo  9.9.9.4/32                         reachable     5.5.5.4         110                  ether2
 9 ADC  9.9.9.5/32         9.9.9.5                                       0                    lobridge
```

and traceroute from R5 to R1 like this:

```
[admin@R5] > /tool traceroute 9.9.9.1
     ADDRESS                                  STATUS
   1         4.4.4.3 11ms 1ms 4ms
   2         2.2.2.2 23ms 3ms 2ms
   3         9.9.9.1 26ms 3ms 3ms
```

Configuring LDP

In order to distribute labels for routes, LDP should get enabled. On R1 this is done by commands (interface ether3 is facing network 1.1.1.0/24):

```
/mpls ldp set enabled=yes transport-address=9.9.9.1 lsr-id=9.9.9.1
/mpls ldp interface add interface=ether3
```

Note that transport-address gets set to 9.9.9.1. This makes router originate LDP session connections with this address and also advertise this address as transport address to LDP neighbors.

Other routers are configured in similar way - LDP is enabled on interfaces that connect routers and not enabled on interfaces that connect customer networks. For example, on R5:

```
[admin@R5] > /ip address print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0   4.4.4.5/24         4.4.4.0         4.4.4.255       ether1
 1   5.5.5.5/24         5.5.5.0         5.5.5.255       ether2
 2   9.9.9.5/32         9.9.9.5         9.9.9.5         lobridge
[admin@R5] > /mpls ldp interface print
Flags: I - invalid, X - disabled
 #   INTERFACE                                       HELLO-INTERVAL       HOLD-TIME
 0   ether1                                          5s                   15s
 1   ether2                                          5s                   15s
```

After LDP sessions are established, on R5 there are 2 LDP neighbors:

```
[admin@R5] > /mpls ldp neighbor print
Flags: X - disabled, D - dynamic, O - operational, T - sending-targeted-hello, V - vpls
 #      TRANSPORT       LOCAL-TRANSPORT PEER                        SEND-TARGETED ADDRESSES
 0 DO   9.9.9.4         9.9.9.5         9.9.9.4:0                   no            3.3.3.4
                                                                                  5.5.5.4
                                                                                  9.9.9.4
```

```
 1 DO   9.9.9.3         9.9.9.5         9.9.9.3:0                    no            2.2.2.3
                                                                                   3.3.3.3
                                                                                   4.4.4.3
                                                                                   9.9.9.3
```

/mpls local-bindings shows labels that this router has assigned to routes and peers it has distributed the label to. It shows that R5 has distributed labels for all its routes to both of its neighbors - R3 and R4

```
[admin@R5] > /mpls local-bindings print
Flags: X - disabled, A - advertised, D - dynamic, L - local-route, G - gateway-route, e - egress
 #      DST-ADDRESS        LABEL      PEERS
 0 ADLe 4.4.4.0/24         impl-null  9.9.9.4:0
                                      9.9.9.3:0
 1 ADLe 9.9.9.5/32         impl-null  9.9.9.4:0
                                      9.9.9.3:0
 2 ADG  9.9.9.4/32         17         9.9.9.4:0
                                      9.9.9.3:0
 3 ADLe 5.5.5.0/24         impl-null  9.9.9.4:0
                                      9.9.9.3:0
 4 ADG  1.1.1.0/24         18         9.9.9.4:0
                                      9.9.9.3:0
 5 ADG  2.2.2.0/24         19         9.9.9.4:0
                                      9.9.9.3:0
 6 ADG  9.9.9.1/32         20         9.9.9.4:0
                                      9.9.9.3:0
 7 ADG  9.9.9.2/32         21         9.9.9.4:0
                                      9.9.9.3:0
 8 ADG  9.9.9.3/32         22         9.9.9.4:0
                                      9.9.9.3:0
 9 ADG  3.3.3.0/24         23         9.9.9.4:0
                                      9.9.9.3:0
```

/mpls remote-bindings shows labels that are allocated for routes by neighboring routers and advertised to this router:

```
[admin@R5] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
 0  D 4.4.4.0/24                         16         9.9.9.4:0
 1 AD 3.3.3.0/24         5.5.5.4         impl-null  9.9.9.4:0
 2  D 9.9.9.5/32                         17         9.9.9.4:0
 3 AD 9.9.9.4/32         5.5.5.4         impl-null  9.9.9.4:0
 4  D 5.5.5.0/24                         impl-null  9.9.9.4:0
 5  D 1.1.1.0/24                         18         9.9.9.4:0
 6  D 2.2.2.0/24                         19         9.9.9.4:0
 7  D 9.9.9.1/32                         20         9.9.9.4:0
 8  D 9.9.9.2/32                         21         9.9.9.4:0
 9  D 9.9.9.3/32                         22         9.9.9.4:0
10 AD 1.1.1.0/24         4.4.4.3         16         9.9.9.3:0
11 AD 2.2.2.0/24         4.4.4.3         impl-null  9.9.9.3:0
12  D 4.4.4.0/24                         impl-null  9.9.9.3:0
13  D 3.3.3.0/24                         impl-null  9.9.9.3:0
14 AD 9.9.9.1/32         4.4.4.3         17         9.9.9.3:0
15 AD 9.9.9.3/32         4.4.4.3         impl-null  9.9.9.3:0
16  D 9.9.9.4/32                         18         9.9.9.3:0
17  D 5.5.5.0/24                         19         9.9.9.3:0
18 AD 9.9.9.2/32         4.4.4.3         20         9.9.9.3:0
19  D 9.9.9.5/32                         21         9.9.9.3:0
```

Here we can observe that R5 has received label bindings for all routes from both its neighbors - R3 and R4, but only the ones for whom particular neighbor is next hop are active. For example:

```
[admin@R5] > /ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY         DISTANCE             INTERFACE
...
 5 ADo  9.9.9.1/32                        r 4.4.4.3         110                  ether1
...
```

```
[admin@R5] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
...
 7  D 9.9.9.1/32                         20         9.9.9.4:0
...
14 AD 9.9.9.1/32         4.4.4.3         17         9.9.9.3:0
...
```

From the above we see that R3, which is next hop for network 9.9.9.1/32 from R5 perspective, has assigned label 17 for traffic going to 9.9.9.1/32. This implies that when R5 will be routing traffic to this network, will impose label 17.

Label switching rules can be seen in /mpls forwarding-table. For example, on R3 it looks like this:

```
[admin@R3] > /mpls forwarding-table print
 # IN-LABEL             OUT-LABELS           DESTINATION        INTERFACE            NEXTHOP
...
 2 17                   17                   9.9.9.1/32         ether1               2.2.2.2
...
```

This rule says that R3 receiving packet with label 17 will change it to label 17 assigned by R2 for network 9.9.9.1/32 (R2 is next hop for 9.9.9.1/32 from R3 perspective):

```
[admin@R2] > /mpls local-bindings print
Flags: X - disabled, A - advertised, D - dynamic, L - local-route, G - gateway-route, e - egress
 #      DST-ADDRESS        LABEL      PEERS
...
 3 ADG  9.9.9.1/32         17         9.9.9.1:0
                                      9.9.9.3:0
...
```

R2 MPLS forwarding table tells:

```
[admin@R2] > /mpls forwarding-table print
 # IN-LABEL             OUT-LABELS           DESTINATION        INTERFACE            NEXTHOP
...
 1 17                                        9.9.9.1/32         ether1               1.1.1.1
...
```

Notice, that forwarding rule does not have any out-labels. The reason for this is that R2 is doing penultimate hop popping for this network. R1 does not assign any real label for 9.9.9.1/32 network, because it is known that R1 is egress point for 9.9.9.1/32 network (router is egress point for networks that are directly connected to it, because next hop for traffic is not MPLS router), therefore it advertises "implicit null" label for this route:

```
[admin@R2] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
...
13 AD 9.9.9.1/32         1.1.1.1         impl-null  9.9.9.1:0
...
```

This tells R2 to forward traffic for 9.9.9.1/32 to R1 unlabelled, which is exactly what R2 mpls forwarding-table entry tells. Penultimate hop popping ensures that routers do not have to do unnecessary label lookup when it is known in advance that router will have to route packet.

Using traceroute in MPLS networks

RFC4950 introduces extensions to ICMP protocol for MPLS. The basic idea is that some ICMP messages may carry MPLS "label stack object" (list of labels that were on packet when it caused particular ICMP message). ICMP messages of interest for MPLS are Time Exceeded and Need Fragment.

MPLS label carries not only label value, but also TTL field. When imposing label on IP packet, MPLS TTL is set to value in IP header, when last label is removed from IP packet, IP TTL is set to value in MPLS TTL. Therefore MPLS switching network can be diagnosed by means of traceroute tool that supports MPLS extension.

For example, traceroute from R5 to R1 looks like this:

```
[admin@R5] > /tool traceroute 9.9.9.1 src-address=9.9.9.5
     ADDRESS                                  STATUS
   1         4.4.4.3 15ms 5ms 5ms
                       mpls-label=17
   2         2.2.2.2 5ms 3ms 6ms
                       mpls-label=17
   3         9.9.9.1 6ms 3ms 3ms
```

Traceroute results show MPLS labels on packet when it produced ICMP Time Exceeded. The above means: when R3 received packet with MPLS TTL 1, it had label 17 on. This matches label advertised by R3 for 9.9.9.1/32. The same way R2 observed label 17 on packet on next traceroute iteration - R3 switched label 17 to label 17, as explaned above. R1 received packet without labels - R2 did penultimate hop popping as explaned above.

Drawbacks of using traceroute in MPLS network

Label switching ICMP errors

One of drawbacks of using traceroute in MPLS networks is the way MPLS handles produced ICMP errors. In IP networks ICMP errors are simply routed back to source of packet that caused the error. In MPLS network it is possible that router that produces error message does not even have route to source of IP packet (for example in case of assymetric label switching paths or some kind of MPLS tunneling, e.g. to transport MPLS VPN traffic).

Due to this produced ICMP errors are not routed to the source of packet that caused the error, but switched further along label switching path, assuming that when label switching path endpoint will receive ICMP error, it will know how to properly route it back to source.

This causes the situation, that traceroute in MPLS network can not be used the same way as in IP network - to determine failure point in the network. If label switched path is broken anywhere in the middle, no ICMP replies will come back, because they will not make it to the far endpoint of label switching path.

Penultimate hop popping and traceroute source address

Thorough understanding of pen ultimate hop behaviour and routing is necessary to understand and avoid problems that penultimate hop popping causes to traceroute.

In the example setup, regular traceroute from R5 to R1 would yield the following results:

```
[admin@R5] > /tool traceroute 9.9.9.1
     ADDRESS                                  STATUS
   1         0.0.0.0 timeout timeout timeout
   2         2.2.2.2 37ms 4ms 4ms
                      mpls-label=17
   3         9.9.9.1 4ms 2ms 11ms
```

compared to:

```
[admin@R5] > /tool traceroute 9.9.9.1 src-address=9.9.9.5
     ADDRESS                                  STATUS
   1         4.4.4.3 15ms 5ms 5ms
                      mpls-label=17
   2         2.2.2.2 5ms 3ms 6ms
                      mpls-label=17
   3         9.9.9.1 6ms 3ms 3ms
```

The reason why first traceroute does not get response from R3 is that by default traceroute on R5 uses source address 4.4.4.5 for its probes, because it is preferred source for route over which next hop to 9.9.9.1/32 is reachable:

```
[admin@R5] > /ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY         DISTANCE            INTERFACE
 ...
 3 ADC  4.4.4.0/24         4.4.4.5                           0                   ether1
 ...
 5 ADo  9.9.9.1/32                         r 4.4.4.3         110                 ether1
 ...
```

When first traceroute probe is transmitted (source: 4.4.4.5, destination 9.9.9.1), R3 drops it and produces ICMP error message (source 4.4.4.3 destination 4.4.4.5) that is switched all the way to R1. R1 then sends ICMP error back - it gets switched along label switching path to 4.4.4.5.

R2 is penultimate hop popping router for network 4.4.4.0/24, because 4.4.4.0/24 is directly connected to R3. Therefore R2 removes last label and sends ICMP error to R3 unlabelled:

```
[admin@R2] > /mpls forwarding-table print
 # IN-LABEL             OUT-LABELS           DESTINATION        INTERFACE           NEXTHOP
 ...
 3 19                                        4.4.4.0/24         ether2              2.2.2.3
 ...
```

R3 drops received IP packet, because it receives packet with its own address as source address. ICMP errors produced by following probes come back correctly, because R3 receives unlabelled packets with source addresses 2.2.2.2 and 9.9.9.1, which are acceptable to route.

Command:

```
[admin@R5] > /tool traceroute 9.9.9.1 src-address=9.9.9.5
 ...
```

produces expected results, because source address of traceroute probes is 9.9.9.5. When ICMP errors are travelling back from R1 to R5, penultimate hop popping for 9.9.9.5/32 network happens at R3, therefore it never gets to route packet with its own address as source address.

Configuring VPLS

Configuring VPLS interfaces

VPLS interface can be considered tunnel interface just like EoIP interface. To achieve transparent ethernet segment forwarding between customer sites the following tunnels need to be established:

- R1-R5 (customer A)
- R1-R4 (customer A)
- R4-R5 (customer A)
- R1-R5 (customer B)

Note that each tunnel setup involves creating VPLS interfaces on both endpoints of tunnel.

Negotiation of VPLS tunnels is done by LDP protocol - both endpoints of tunnel exchange labels they are going to use for tunnel. Data forwarding in tunnel then happens by imposing 2 labels on packets: tunnel label and transport label - label that ensures traffic delivery to the other endpoint of tunnel.

VPLS tunnels are configured in /interface vpls menu. vpls-id parameter identifies VPLS tunnel and must be unique for every tunnel between this and remote peer.

The necessary setup:

- on R1:

```
/interface vpls add name=A1toA2 remote-peer=9.9.9.5 mac-address=00:00:00:00:00:a1 vpls-id=10 disabled=no
/interface vpls add name=A1toA3 remote-peer=9.9.9.4 mac-address=00:00:00:00:00:a1 vpls-id=10 disabled=no
/interface vpls add name=B1toB2 remote-peer=9.9.9.5 mac-address=00:00:00:00:00:b1 vpls-id=11 disabled=no
```

- on R4:

```
/interface vpls add name=A3toA1 remote-peer=9.9.9.1 mac-address=00:00:00:00:00:a3 vpls-id=10 disabled=no
/interface vpls add name=A3toA2 remote-peer=9.9.9.5 mac-address=00:00:00:00:00:a3 vpls-id=10 disabled=no
```

- on R5:

```
/interface vpls add name=A2toA1 remote-peer=9.9.9.1 mac-address=00:00:00:00:00:a2 vpls-id=10 disabled=no
/interface vpls add name=A2toA3 remote-peer=9.9.9.4 mac-address=00:00:00:00:00:a2 vpls-id=10 disabled=no
/interface vpls add name=B2toB1 remote-peer=9.9.9.1 mac-address=00:00:00:00:00:b2 vpls-id=11 disabled=no
```

Configuring VPLS tunnel causes dynamic LDP neighbor to be created and "targeted" LDP session to be established. Targeted LDP session is session that is established between two routers that are not direct neighbors. After this setup R1 LDP neighbors are:

```
[admin@R1] > mpls ldp neighbor pr
Flags: X - disabled, D - dynamic, O - operational, T - sending-targeted-hello, V - vpls
 #      TRANSPORT       LOCAL-TRANSPORT PEER                      SEND-TARGETED ADDRESSES
 0 DO   9.9.9.2         9.9.9.1         9.9.9.2:0                 no            1.1.1.2
                                                                                2.2.2.2
                                                                                9.9.9.2
 1 DOTV 9.9.9.5         9.9.9.1         9.9.9.5:0                 yes           4.4.4.5
                                                                                5.5.5.5
                                                                                9.9.9.5
 2 DOTV 9.9.9.4         9.9.9.1         9.9.9.4:0                 yes           3.3.3.4
                                                                                5.5.5.4
                                                                                9.9.9.4
```

Note that labels for IP routes are also exchanged between VPLS peers, although there is no chance any of them will be used. For example, without adding additional links, R4 will not become next hop for any route on R1, so labels learned from R4 are not likely to be ever used. Still routers maintain all labels exchanged so that they are ready for use immediately if needed. This default behaviour can be overridden by filtering which is discussed later.

By monitoring state of VPLS interface its related information can be viewed:

```
[admin@R1] /interface vpls> monitor A1toA3 once
    remote-label: 24
     local-label: 27
   remote-status:
      igp-prefix: 9.9.9.4/32
     igp-nexthop: 1.1.1.2
  imposed-labels: 21,24
```

Here we can see that R1 has assigned label 27 for tunnel between R1 and R4. MPLS forwarding table shows that this label is recognized and instead of fowarding to some next hop, received over this tunnel:

```
[admin@R1] > /mpls forwarding-table print
 # IN-LABEL             OUT-LABELS           DESTINATION        INTERFACE          NEXTHOP
 ...
11 27                                        A1toA3
 ...
```

In turn remote endpoint (R4) has assigned label 24.

igp-prefix shows route that is used to get to remote endpoint of tunnel. This implies that when forwarding traffic to remote endpoint of tunnel this router will impose transport label - label distributed by next hop (which is shown as igp-nexthop) to 9.9.9.4/32 for 9.9.9.4/32 route. This can be confirmed on R2:

```
[admin@R2] > /mpls forwarding-table print
 # IN-LABEL             OUT-LABELS           DESTINATION        INTERFACE          NEXTHOP
 ...
 5 21                   18                   9.9.9.4/32         ether2             2.2.2.3
 ...
```

Tunnel label imposed on packets will be as assigned by remote router (R4) for this tunnel.

imposed-labels reflect this setup: packets produced by tunnel will have 2 labels on them: 21 and 24.

Penultimate hop popping effects on VPLS tunnels

Penultimate hop popping of transport label causes packets to arrive at VPLS tunnel endpoint with just one tag - tunnel tag. This makes VPLS tunnel endpoint do just one label lookup to find out what to do with packet. Transport label behaviour can be observed by traceroute tool between tunnel endpoints. For example, traceroute from R1 to R4 looks like this:

```
[admin@R1] > /tool traceroute 9.9.9.4 src-address=9.9.9.1
     ADDRESS                                  STATUS
   1         1.1.1.2 7ms 5ms 3ms
                     mpls-label=21
   2         2.2.2.3 5ms 4ms 18ms
                     mpls-label=18
   3         9.9.9.4 4ms 5ms 3ms
```

traceroute output shows, that endpoint of tunnel is receiving probe without label. The same happens with VPLS tunnel traffic - at R3 transport label (18) is popped and packet is switched with just tunnel label on.

The requirement to deliver packet with tunnel label to endpoint of tunnel explains configuration advice to use "loopback" IP addresses as tunnel endpoints. If in this case R4 was establishing LDP sessions with its address 3.3.3.4, penultimate hop popping would happen not at R3, but at R2, because R3 has network 3.3.3.0/24 as its connected network (and therefore advertises implicit null label). This would cause R3 (and not R4) to receive packet with just tunnel label on, yielding unpredicted results - either dropping frame if R3 does not recognize the packet or forwarding it the wrong way.

Another issue is having VPLS tunnel endpoints directly connected, as in case of R4 and R5. There are no transport labels they can use between themselves, because they both instruct other one to be penultimate hop popping router for their tunnel endpoint address. For example on R5:

```
[admin@R5] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
 ...
 3 AD 9.9.9.4/32         5.5.5.4         impl-null  9.9.9.4:0
 ...
```

This causes VPLS tunnel to use only tunnel label when sending packets:

```
[admin@R5] > /int vpls monitor A2toA3 once
    remote-label: 23
     local-label: 27
   remote-status:
      igp-prefix: 9.9.9.4/32
     igp-nexthop: 5.5.5.4
  imposed-labels: 23
```

Bridging ethernet segments with VPLS

VPLS tunnels provide virtual ethernet link between routers. To transparrently connect two physical ethernet segments, they must be bridged with VPLS tunnel. In general it gets done the same way as with EoIP interfaces.

So to transparently bridge customer B networks connected to R1 and R5 the following commands are used on R1:

```
/interface bridge add name=B
/interface bridge port add bridge=B interface=ether1
/interface bridge port add bridge=B interface=B1toB2
```

and on R5:

```
/interface bridge add name=B
/interface bridge port add bridge=B interface=ether3
/interface bridge port add bridge=B interface=B2toB1
```

Note that there is not need to run (R)STP protocol on bridge as there are links between segments B1 and B2 except single VPLS tunnel between R1 and R5.

Split horizon bridging

In the example setup there are 3 tunnels set up to connect segments A1, A2 and A3, establishing so called "full mesh" of tunnels between involved segments. If bridging without (R)STP was enabled, traffic loop would occur. There are a few solutions to this:

- enabling (R)STP to eliminate the loop. This approach has a drawback - (R)STP protocol would disable forwarding through one of tunnels and keep it just for backup purposes. That way traffic between 2 segments would have to go through 2 tunnels, making setup inefficent
- using bridge firewall to make sure that traffic does not get looped - involves firewall rule setup making bridging less efficent
- using bridge horizon feature

The basic idea of split horizon bridging is to make traffic arriving over some port never be sent out some set of ports. For VPLS purposes this would mean never sending packet that arrived over one VPLS tunnel over to other VPLS tunnel, as it is known in advance, that sender of packet has connection to target network itself.

For example, if device in A1 sent packet to broadcast or unknown MAC address (which causes bridges to flood all interfaces), it would get sent to both, R5 and R4 over VPLS tunnels. In regular setup e.g. R5 when receiving such packet over VPLS tunnel would send it in A2 connected to it and also over VPLS tunnel to R4. This way R4 would get 2 copies of the same packet and further cause traffic to loop.

Bridge horizon feature allows to configure bridge ports with horizon setting so that packet received over port with horizon value X is not forwarded or flooded to any port with the same horizon value X. So in case of full mesh of VPLS tunnels, each router must be configured with the same horizon value for VPLS tunnels that are bridged together.

For example, configuration commands for R1 to enable bridging for customer A are:

```
/interface bridge add name=A
/interface bridge port add bridge=A interface=A1toA2 horizon=1
/interface bridge port add bridge=A interface=A1toA3 horizon=1
```

In similar way bridge should be configured on R4 and R5. Note that physical ethernet port is not configured with horizon value. If it was, that would disabled bridge forwarding data at all.

Note that horizon value has meaning only locally - it does not get transmitted over network, therefore it does not matter if the same value is configured in all routers participating in bridged network.

Optimizing label distribution

Label binding filtering

During implementation of given example setup, it has become clear that not all label bindings are necessary. For example, there is no need to exchange IP route label bindings between R1 and R5 or R1 and R4, as there is no chance they will ever be used. Also, if given network core is providing connectivity only for mentioned customer ethernet segments, there is no real use to distribute labels for networks that connect routers between themselves, the only routes that matter are /32 routes to endpoints of VPLS tunnels.

Label binding filtering can be used to distribute only specified sets of labels to reduce resource usage and network load.

There are 2 types of label binding filters:

- which label bindings should be advertised to LDP neighbors, configured in /mpls ldp advertise-filter
- which label bindings should be accepted from LDP neighbors, configured in /mpls ldp accept-filter

Filters are organized in ordered list, specifying prefix that must include the prefix that is tested against filter and neighbor (or wildcard).

In given example setup all routers can be configured so that they advertise labels only for routes that allow to reach endpoints of tunnels. For this 2 advertise filters need to be configured on all routers:

```
/mpls ldp advertise-filter add prefix=9.9.9.0/24 advertise=yes
/mpls ldp advertise-filter add prefix=0.0.0.0/0 advertise=no
```

This filter causes routers to advertise only bindings for routes that are included by 9.9.9.0/24 prefix which covers tunnel endpoints (9.9.9.1/32, 9.9.9.4/32, 9.9.9.5/32). The second rule is necessary because default filter result, when no rule matches is to allow action in question.

In given setup there is no need to set up accept filter, because by convention introduced by 2 abovementioned rules no LDP router will distribute unnecessary bindings.

Note that filter changes do not affect existing mappings, so to take filter into effect, connections between neighbors need to be reset. This can get done by removing them:

```
[admin@R1] /mpls ldp neighbor> print
Flags: X - disabled, D - dynamic, O - operational, T - sending-targeted-hello, V - vpls
 #      TRANSPORT       LOCAL-TRANSPORT PEER                    SEND-TARGETED ADDRESSES
 0 DO   9.9.9.2         9.9.9.1         9.9.9.2:0               no            1.1.1.2
                                                                              2.2.2.2
                                                                              9.9.9.2
 1 DOTV 9.9.9.5         9.9.9.1         9.9.9.5:0               yes           4.4.4.5
                                                                              5.5.5.5
                                                                              9.9.9.5
 2 DOTV 9.9.9.4         9.9.9.1         9.9.9.4:0               yes           3.3.3.4
                                                                              5.5.5.4
                                                                              9.9.9.4
[admin@R1] /mpls ldp neighbor> remove [find]
```

So on R1, for example we get:

```
[admin@R1] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
 0  D 9.9.9.1/32                         30         9.9.9.5:0
 1  D 9.9.9.5/32                         impl-null  9.9.9.5:0
 2  D 9.9.9.4/32                         31         9.9.9.5:0
 3  D 9.9.9.2/32                         32         9.9.9.5:0
 4  D 9.9.9.3/32                         33         9.9.9.5:0
 5 AD 9.9.9.2/32         1.1.1.2         impl-null  9.9.9.2:0
 6  D 9.9.9.1/32                         24         9.9.9.2:0
 7 AD 9.9.9.3/32         1.1.1.2         25         9.9.9.2:0
 8 AD 9.9.9.4/32         1.1.1.2         26         9.9.9.2:0
 9 AD 9.9.9.5/32         1.1.1.2         27         9.9.9.2:0
10  D 9.9.9.1/32                         27         9.9.9.4:0
11  D 9.9.9.5/32                         28         9.9.9.4:0
12  D 9.9.9.4/32                         impl-null  9.9.9.4:0
13  D 9.9.9.2/32                         29         9.9.9.4:0
14  D 9.9.9.3/32                         30         9.9.9.4:0
```

There still are unnecessary bindings, this time - the bindings distributed due to establishing targeted LDP session with remote endpoints of VPLS tunnels (bindings from 9.9.9.5 and 9.9.9.4)

To filter out those, we configure routers to not distribute any IP bindings to any of tunnel endpoint routers. For example on R1, filter table should look like this:

```
[admin@R1] /mpls ldp advertise-filter> print
Flags: X - disabled
 #   PREFIX            NEIGHBOR        ADVERTISE
 0   0.0.0.0/0         9.9.9.4         no
 1   0.0.0.0/0         9.9.9.5         no
 2   9.9.9.0/24        all             yes
 3   0.0.0.0/0         all             no
```

This causes routers to have minimal label binding tables, for example on R1:

```
[admin@R1] > /mpls local-bindings print
Flags: X - disabled, A - advertised, D - dynamic, L - local-route, G - gateway-route, e - egress
 #      DST-ADDRESS        LABEL      PEERS
 0 ADLe 9.9.9.1/32         impl-null  9.9.9.2:0
 1 ADG  9.9.9.3/32         40         9.9.9.2:0
 2 ADG  9.9.9.4/32         41         9.9.9.2:0
 3 ADG  9.9.9.2/32         42         9.9.9.2:0
 4 ADG  9.9.9.5/32         43         9.9.9.2:0
[admin@R1] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
 0 AD 9.9.9.2/32         1.1.1.2         impl-null  9.9.9.2:0
 1  D 9.9.9.1/32                         24         9.9.9.2:0
 2 AD 9.9.9.3/32         1.1.1.2         25         9.9.9.2:0
 3 AD 9.9.9.4/32         1.1.1.2         26         9.9.9.2:0
 4 AD 9.9.9.5/32         1.1.1.2         27         9.9.9.2:0
```

Note that IP binding distribution should not be disabled between R4 and R5 although they are tunnel endpoints. Doing so would not harm regular case, because R4 and R5 does not need IP bindings to VPLS tunnel data, but in case link between R3 and R5 would go down, all traffic to R5 from R1 would have to be rerouted through R4. In such case R5 not distributing IP bindings to R4 would cause R4 to not be able to forward MPLS traffic to R5.

Effects of label binding filtering on data forwarding in network

Note the traceroute results after these changes. Traceroute from R1 to R5 using R1 loopback address as source address still behaves the same - each hop reports received labels:

```
[admin@R1] > tool traceroute 9.9.9.5 src-address=9.9.9.1
     ADDRESS                                  STATUS
   1         1.1.1.2 11ms 4ms 5ms
                      mpls-label=27
   2         2.2.2.3 4ms 4ms 4ms
                      mpls-label=25
   3         9.9.9.5 12ms 3ms 3ms
```

But in case of traceroute using R1 address that faces network:

```
[admin@R1] > tool traceroute 9.9.9.5 src-address=1.1.1.1
     ADDRESS                                  STATUS
   1         0.0.0.0 timeout timeout timeout
   2         0.0.0.0 timeout timeout timeout
   3         9.9.9.5 3ms 3ms 3ms
```

Now all hops except the last one do not respond. The reason for this is the fact, that there is no label switching path back from R5 to R1 (which this time uses address 1.1.1.1) because there are no label bindings distributed - so ICMP response is routed and routers on the way back (R3 and R2) receive packets with their own source address and drop them right away without routing.

On the other hand, traceroute from R1 to R5 using their non-loopback addresses:

```
[admin@R1] > tool traceroute 4.4.4.5 src-address=1.1.1.1
     ADDRESS                                  STATUS
   1         1.1.1.2 13ms 1ms 1ms
   2         2.2.2.3 3ms 2ms 2ms
   3         4.4.4.5 3ms 3ms 23ms
```

There is no label switching involved doing this traceroute and it works just like in network without MPLS at all.

See also

- BGP Based VPLS
- EXP_bit_behaviour