

# ROSE-SRv6 tutorial on Linux - Part 1

## Manual creation of SRv6 tunnels in the data plane

### ROSE Project Technical Report

**Authored by:** Paolo Lungaroni, Andrea Mayer, Stefano Salsano

**Version:** 1.3

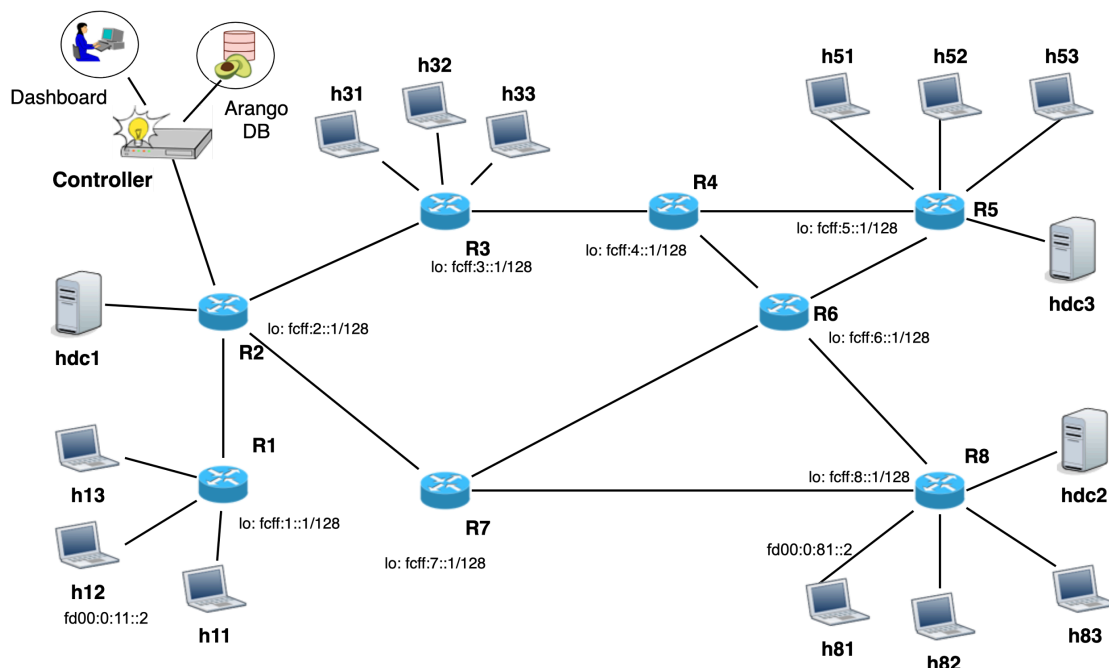
**Date:** 2020:11:07

This demo has also been run during the HPSR 2020 tutorial: "Segment Routing over IPv6 (SRV6) and the Network Programming Model" ([program here](#)).

This tutorial is based on the rose-srv6 VM that can be [downloaded here](#).

To give us feedback you can open issues on our [github repository](#), or contact the ROSE team [here](#).

Using Mininet, we deploy the topology depicted in the figure hereafter, then we will add a bidirectional SRv6 tunnel (actually, two unidirectional tunnels!). In the second part of this tutorial ([available here](#)) we will use the Controller to setup the tunnels.



Link to repository for the given topology  
(<https://github.com/netgroup/rose-srv6-tutorial.git>)

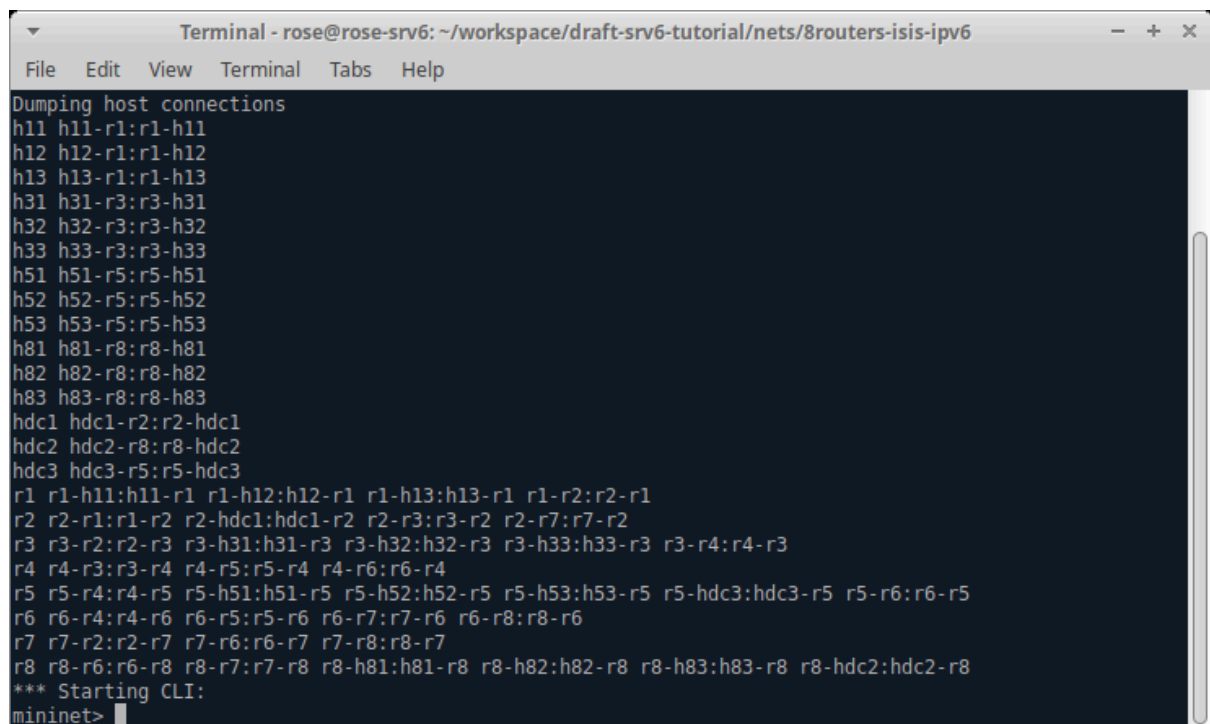
To start the mininet simulation, click the "Terminal Emulator" icon and enter in the tutorial folder typing:

```
$ cd ~/workspace/rose-srv6-tutorial/nets/8routers-isis-ipv6/
```

To start the mininet simulation:

```
$ sudo bash starter.sh
```

After creating the network topology, mininet shows the prompt command line through which we are able to interact with the nodes.



```
Terminal - rose@rose-srv6: ~/workspace/draft-srv6-tutorial/nets/8routers-isis-ipv6
File Edit View Terminal Tabs Help
Dumping host connections
h11 h11-r1:r1-h11
h12 h12-r1:r1-h12
h13 h13-r1:r1-h13
h31 h31-r3:r3-h31
h32 h32-r3:r3-h32
h33 h33-r3:r3-h33
h51 h51-r5:r5-h51
h52 h52-r5:r5-h52
h53 h53-r5:r5-h53
h81 h81-r8:r8-h81
h82 h82-r8:r8-h82
h83 h83-r8:r8-h83
hdc1 hdc1-r2:r2-hdc1
hdc2 hdc2-r8:r8-hdc2
hdc3 hdc3-r5:r5-hdc3
r1 r1-h11:h11-r1 r1-h12:h12-r1 r1-h13:h13-r1 r1-r2:r2-r1
r2 r2-r1:r1-r2 r2-hdc1:hdc1-r2 r2-r3:r3-r2 r2-r7:r7-r2
r3 r3-r2:r2-r3 r3-h31:h31-r3 r3-h32:h32-r3 r3-h33:h33-r3 r3-r4:r4-r3
r4 r4-r3:r3-r4 r4-r5:r5-r4 r4-r6:r6-r4
r5 r5-r4:r4-r5 r5-h51:h51-r5 r5-h52:h52-r5 r5-h53:h53-r5 r5-hdc3:hdc3-r5 r5-r6:r6-r5
r6 r6-r4:r4-r6 r6-r5:r5-r6 r6-r7:r7-r6 r6-r8:r8-r6
r7 r7-r2:r2-r7 r7-r6:r6-r7 r7-r8:r8-r7
r8 r8-r6:r6-r8 r8-r7:r7-r8 r8-h81:h81-r8 r8-h82:h82-r8 r8-h83:h83-r8 r8-hdc2:hdc2-r8
*** Starting CLI:
mininet>
```

As soon as Mininet has successfully loaded the network topology (nodes, links, etc), ISIS routing daemons running on top of routers (r1,r2,..., r8) start to advertise routes with their neighbors.

We can take a look at the main routing table of the router r1 through the following command:

```
mininet> gterm r1
# ip -6 route show
```

```
root@rose-srv6:/home/rose/workspace/rose-srv6-tutorial/nets/8routers-isis-ipv6# ip -6 route show
fcf0:0:1:2::/64 dev r1-r2 proto kernel metric 256 pref medium
fcf0:0:2:3::/64 nhid 16 via fe80::f8b2:46ff:fe8a:9337 dev r1-r2 proto isis metric 20 pref medium
fcf0:0:2:7::/64 nhid 16 via fe80::f8b2:46ff:fe8a:9337 dev r1-r2 proto isis metric 20 pref medium
fcf0:0:3:4::/64 nhid 16 via fe80::f8b2:46ff:fe8a:9337 dev r1-r2 proto isis metric 20 pref medium
fcf0:0:4:5::/64 nhid 16 via fe80::f8b2:46ff:fe8a:9337 dev r1-r2 proto isis metric 20 pref medium
fcf0:0:4:6::/64 nhid 16 via fe80::f8b2:46ff:fe8a:9337 dev r1-r2 proto isis metric 20 pref medium
fcf0:0:5:6::/64 nhid 16 via fe80::f8b2:46ff:fe8a:9337 dev r1-r2 proto isis metric 20 pref medium
fcf0:0:6:7::/64 nhid 16 via fe80::f8b2:46ff:fe8a:9337 dev r1-r2 proto isis metric 20 pref medium
fcf0:0:6:8::/64 nhid 16 via fe80::f8b2:46ff:fe8a:9337 dev r1-r2 proto isis metric 20 pref medium
fcf0:0:7:8::/64 nhid 16 via fe80::f8b2:46ff:fe8a:9337 dev r1-r2 proto isis metric 20 pref medium
fcff:1::/32 dev lo proto kernel metric 256 pref medium
fcff:2:1::/48 nhid 16 via fe80::f8b2:46ff:fe8a:9337 dev r1-r2 proto isis metric 20 pref medium
fcff:2::/32 nhid 16 via fe80::f8b2:46ff:fe8a:9337 dev r1-r2 proto isis metric 20 pref medium
fcff:3::/32 nhid 16 via fe80::f8b2:46ff:fe8a:9337 dev r1-r2 proto isis metric 20 pref medium
fcff:4::/32 nhid 16 via fe80::f8b2:46ff:fe8a:9337 dev r1-r2 proto isis metric 20 pref medium
fcff:5:1::/48 nhid 16 via fe80::f8b2:46ff:fe8a:9337 dev r1-r2 proto isis metric 20 pref medium
fcff:5::/32 nhid 16 via fe80::f8b2:46ff:fe8a:9337 dev r1-r2 proto isis metric 20 pref medium
fcff:6::/32 nhid 16 via fe80::f8b2:46ff:fe8a:9337 dev r1-r2 proto isis metric 20 pref medium
fcff:7::/32 nhid 16 via fe80::f8b2:46ff:fe8a:9337 dev r1-r2 proto isis metric 20 pref medium
fcff:8:1::/48 nhid 16 via fe80::f8b2:46ff:fe8a:9337 dev r1-r2 proto isis metric 20 pref medium
fcff:8::/32 nhid 16 via fe80::f8b2:46ff:fe8a:9337 dev r1-r2 proto isis metric 20 pref medium
fd00:0:11::/64 dev r1-h11 proto kernel metric 256 pref medium
fd00:0:12::/64 dev r1-h12 proto kernel metric 256 pref medium
fd00:0:13::/64 dev r1-h13 proto kernel metric 256 pref medium
fe80::/64 dev r1-h12 proto kernel metric 256 pref medium
fe80::/64 dev r1-h13 proto kernel metric 256 pref medium
fe80::/64 dev r1-r2 proto kernel metric 256 pref medium
fe80::/64 dev r1-h11 proto kernel metric 256 pref medium
root@rose-srv6:/home/rose/workspace/rose-srv6-tutorial/nets/8routers-isis-ipv6#
```

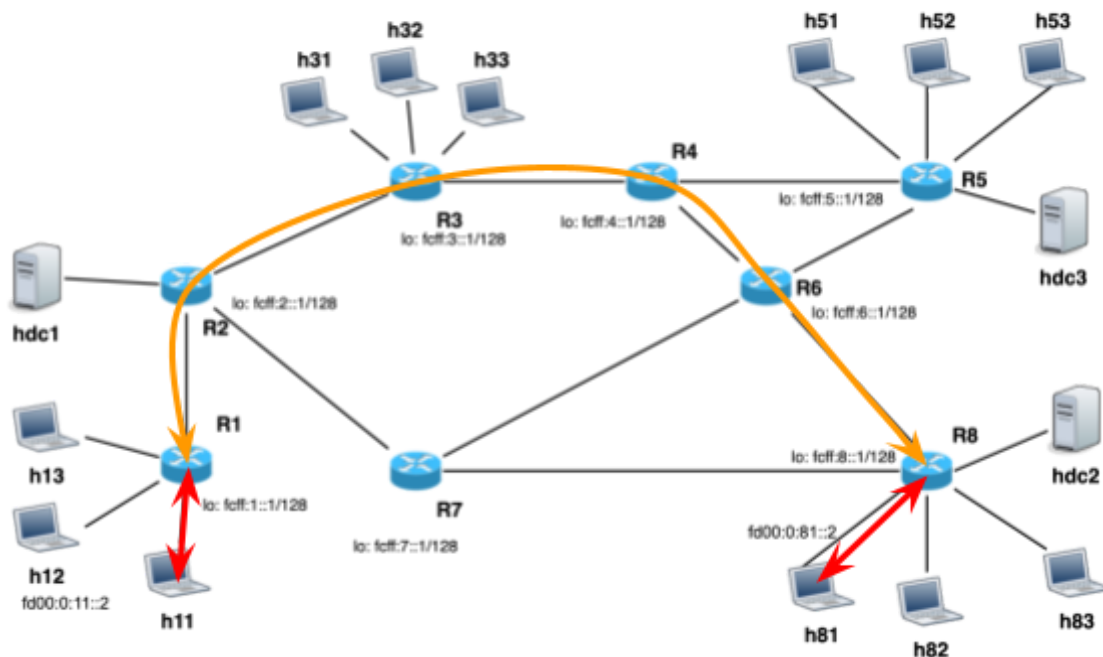
The figure shows the main routing table of the router r1. In particular, the routes with the addressing schema `fcf0:0:x:y::/64` are advertised from the routers (r2,r3,...,r8) using the ISIS protocol. Such addresses are associated with the interfaces of the routers that are used by the routers themselves to communicate with each other.

As you may notice, routes set by the ISIS demon are identified by the "proto isis" keyword.

Conversely, the `fcff:x::/32` represents the address of router x and it is associated with the loopback interface. The loopback interface allows you to reach the router avoiding to directly specify one of its interfaces.

Routes associated with local interfaces are marked with the "proto kernel" keyword because they are directly handled by the Linux kernel and not by the ISIS routing daemon.

Our purpose is to contact the h81 node from the h11 node as shown in the picture below:



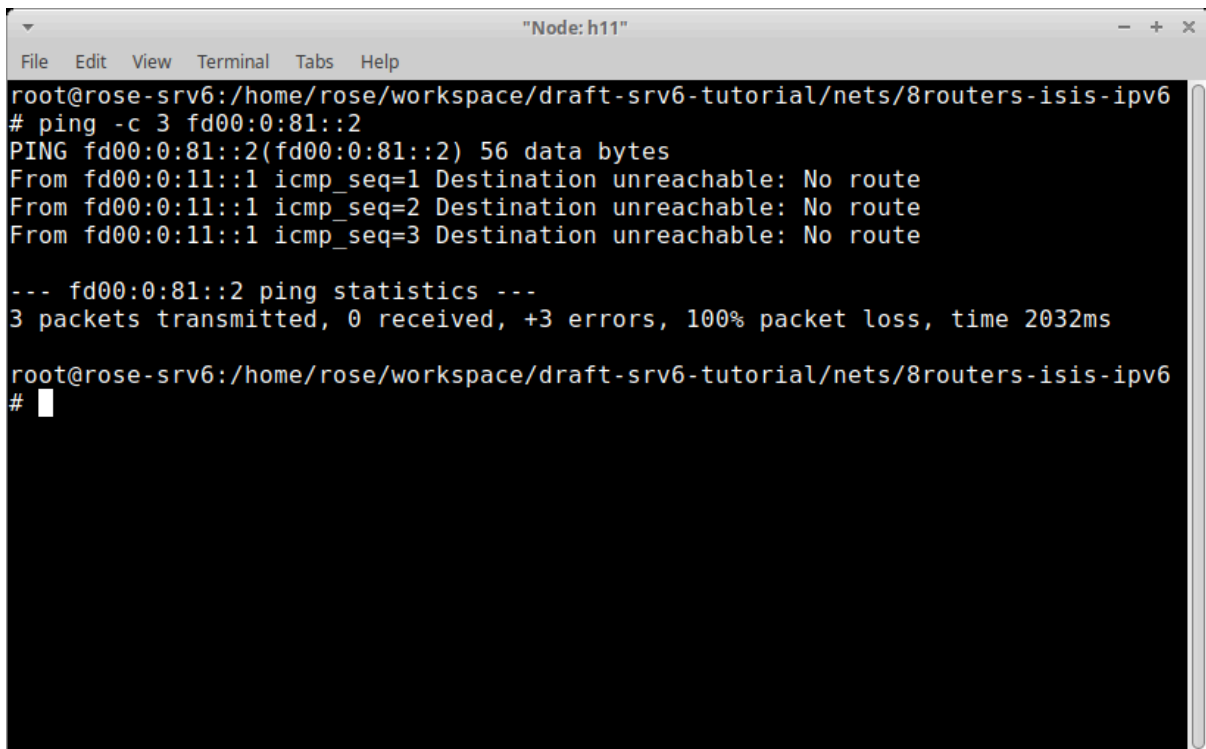
To enter in the console of h11 node, run:

```
mininet> gterm h11
```

```
"Node: h11"
File Edit View Terminal Tabs Help
root@rose-srv6:/home/rose/workspace/draft-srv6-tutorial/nets/8routers-isis-ipv6
#
```

To ping the h81 node (ip address: fd00:0:81::2), enter the following command:

```
# ping -c 3 fd00:0:81::2
```

A terminal window titled "Node: h11" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows a user at the prompt root@rose-srv6:/home/rose/workspace/draft-srv6-tutorial/nets/8routers-isis-ipv6. They run the command # ping -c 3 fd00:0:81::2. The output shows three failed ping attempts from fd00:0:11::1 to fd00:0:81::2, each with the message "Destination unreachable: No route". It then shows ping statistics: 3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2032ms. The prompt returns to root@rose-srv6:/home/rose/workspace/draft-srv6-tutorial/nets/8routers-isis-ipv6 #.

```
root@rose-srv6:/home/rose/workspace/draft-srv6-tutorial/nets/8routers-isis-ipv6
# ping -c 3 fd00:0:81::2
PING fd00:0:81::2(fd00:0:81::2) 56 data bytes
From fd00:0:11::1 icmp_seq=1 Destination unreachable: No route
From fd00:0:11::1 icmp_seq=2 Destination unreachable: No route
From fd00:0:11::1 icmp_seq=3 Destination unreachable: No route

--- fd00:0:81::2 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2032ms

root@rose-srv6:/home/rose/workspace/draft-srv6-tutorial/nets/8routers-isis-ipv6
#
```

As we can see, the 3 ping requests fail. The reason is that node h11 does not know how to reach the node h81 which is located in a different network that is not directly accessible.

To connect the two nodes we create a Segment Routing tunnel, which starts on the R1 (the gateway of h11) and ends on the R8 (the gateway of h81). Note that core network is configured so that each router X is able to reach any other router Y and vice versa.

SR tunnels are uni-directionals and this means that we need to configure the functions for the encap and decap operations on both R1 and R8.

For entering in the R1 console:

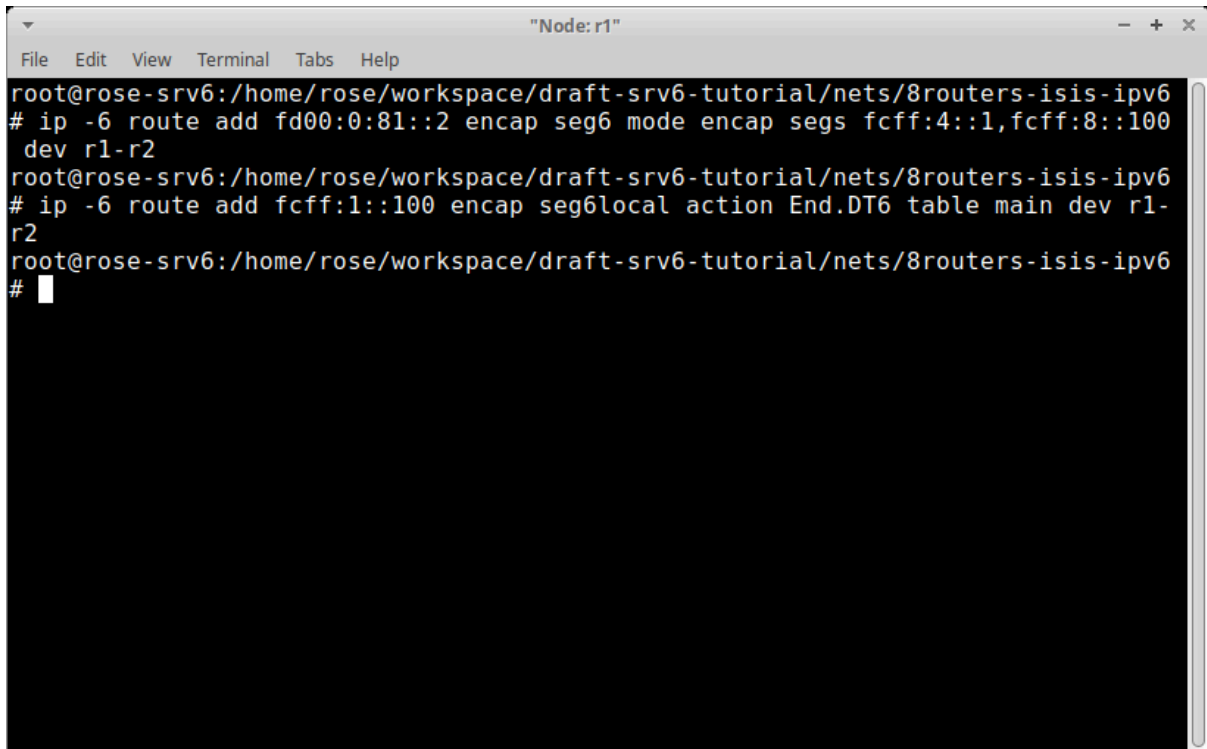
```
mininet> gterm r1
```

We configure the encap operation, typing in r1:

```
# ip -6 route add fd00:0:81::2 encap seg6 mode encap segs
fcff:4::1,fcff:8::100 dev r1-r2
```

And for the decap operation in r1:

```
# ip -6 route add fcff:1::100 encap seg6local action End.DT6  
table main dev r1-r2
```

A terminal window titled "Node: r1" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows the following commands and output:

```
root@rose-srv6:/home/rose/workspace/draft-srv6-tutorial/nets/8routers-isis-ipv6  
# ip -6 route add fd00:0:81::2 encap seg6 mode encap segs fcff:4::1,fcff:8::100  
dev r1-r2  
root@rose-srv6:/home/rose/workspace/draft-srv6-tutorial/nets/8routers-isis-ipv6  
# ip -6 route add fcff:1::100 encap seg6local action End.DT6 table main dev r1-  
r2  
root@rose-srv6:/home/rose/workspace/draft-srv6-tutorial/nets/8routers-isis-ipv6  
#
```

If you want to check the routes installed in the node r1, you can look at the SRv6 tunnel routes:

```
# ip -6 route show
```

```
root@rose-srv6:/home/rose/workspace/draft-srv6-tutorial/nets/8routers-isis-ipv6# ip -6 route show
fcf0:0:1:2::/64 dev r1-r2 proto kernel metric 256 pref medium
fcf0:0:2:3::/64 nhid 16 via fe80::d01e:17ff:fe3e:9e65 dev r1-r2 proto isis metric 20 pref medium
fcf0:0:2:7::/64 nhid 16 via fe80::d01e:17ff:fe3e:9e65 dev r1-r2 proto isis metric 20 pref medium
fcf0:0:3:4::/64 nhid 16 via fe80::d01e:17ff:fe3e:9e65 dev r1-r2 proto isis metric 20 pref medium
fcf0:0:4:5::/64 nhid 16 via fe80::d01e:17ff:fe3e:9e65 dev r1-r2 proto isis metric 20 pref medium
fcf0:0:4:6::/64 nhid 16 via fe80::d01e:17ff:fe3e:9e65 dev r1-r2 proto isis metric 20 pref medium
fcf0:0:5:6::/64 nhid 16 via fe80::d01e:17ff:fe3e:9e65 dev r1-r2 proto isis metric 20 pref medium
fcf0:0:6:7::/64 nhid 16 via fe80::d01e:17ff:fe3e:9e65 dev r1-r2 proto isis metric 20 pref medium
fcf0:0:6:8::/64 nhid 16 via fe80::d01e:17ff:fe3e:9e65 dev r1-r2 proto isis metric 20 pref medium
fcf0:0:7:8::/64 nhid 16 via fe80::d01e:17ff:fe3e:9e65 dev r1-r2 proto isis metric 20 pref medium
fcff:1::100 encap seg6local action End.DT6 table 0 dev r1-r2 metric 1024 pref medium
fcff:1::/32 dev lo proto kernel metric 256 pref medium
fcff:2:1::/48 nhid 16 via fe80::d01e:17ff:fe3e:9e65 dev r1-r2 proto isis metric 20 pref medium
fcff:2::/32 nhid 16 via fe80::d01e:17ff:fe3e:9e65 dev r1-r2 proto isis metric 20 pref medium
fcff:3::/32 nhid 16 via fe80::d01e:17ff:fe3e:9e65 dev r1-r2 proto isis metric 20 pref medium
fcff:4::/32 nhid 16 via fe80::d01e:17ff:fe3e:9e65 dev r1-r2 proto isis metric 20 pref medium
fcff:5:1::/48 nhid 16 via fe80::d01e:17ff:fe3e:9e65 dev r1-r2 proto isis metric 20 pref medium
fcff:5::/32 nhid 16 via fe80::d01e:17ff:fe3e:9e65 dev r1-r2 proto isis metric 20 pref medium
fcff:6::/32 nhid 16 via fe80::d01e:17ff:fe3e:9e65 dev r1-r2 proto isis metric 20 pref medium
fcff:7::/32 nhid 16 via fe80::d01e:17ff:fe3e:9e65 dev r1-r2 proto isis metric 20 pref medium
fcff:8:1::/48 nhid 16 via fe80::d01e:17ff:fe3e:9e65 dev r1-r2 proto isis metric 20 pref medium
fcff:8::/32 nhid 16 via fe80::d01e:17ff:fe3e:9e65 dev r1-r2 proto isis metric 20 pref medium
fd00:0:11::/64 dev r1-h11 proto kernel metric 256 pref medium
fd00:0:12::/64 dev r1-h12 proto kernel metric 256 pref medium
fd00:0:13::/64 dev r1-h13 proto kernel metric 256 pref medium
fd00:0:81::2 encap seg6 mode encap segs 2 [ fcff:4::1 fcff:8::100 ] dev r1-r2 metric 1024 pref medium
fe80::/64 dev r1-h12 proto kernel metric 256 pref medium
fe80::/64 dev r1-h13 proto kernel metric 256 pref medium
fe80::/64 dev r1-r2 proto kernel metric 256 pref medium
fe80::/64 dev r1-h11 proto kernel metric 256 pref medium
root@rose-srv6:/home/rose/workspace/draft-srv6-tutorial/nets/8routers-isis-ipv6#
```

## Virtual Routing and Forwarding (VRF) in Linux



Virtual Routing and Forwarding (VRF) is a technology in Linux that allows multiple instances of a routing table to coexist within the same router. This enables separation of routing domains for different tenants or services, allowing them to have overlapping IP addresses and routes without interference.

Each VRF can maintain its own set of routes and IP addresses, making it ideal for scenarios where isolation is required, such as in multi-tenant environments.

### Creating a VRF

To create a VRF device in Linux, you can use the following command:

```
# ip link add vrf100 type vrf table 100
```

This command creates a VRF named vrf100 and associates it with routing table 100.

### Enslave network device to a VRF

In the Linux kernel, the concept of enslaving a network device to a VRF (Virtual Routing and Forwarding) allows for the segregation of network traffic and routing tables. When a network device is enslaved to a VRF, it becomes part of that specific routing domain, meaning that any incoming or outgoing packets on that device will be routed based on the

routes defined in the VRF's associated routing table. If a network device is not enslaved to a VRF, it will not perform automatic routing lookups against the master VRF. Consequently, traffic received on that device won't be routed according to the routes defined in the VRF, leading to potential misrouting or dropped packets.

To enslave a network device to a master VRF using the `iproute2` utility, you can use the following command:

```
# ip link set dev h11-r1 master vrf100
```

This command effectively associates the specified network device (h11-r1) with the designated VRF (vrf100), enabling proper routing based on the VRF's routing table.

### **End.DT6 in VRF Mode**

The SRv6 End.DT6 behavior can be configured to utilize the routing table associated with a specific VRF. However, it is crucial that the VRF is set to "Strict Mode" for this operation.

#### **Strict Mode**

Strict Mode enforces a one-to-one relationship between VRFs and routing tables. This means:

- 1) A specific routing table can only be associated with one VRF;
- 2) A specific VRF cannot share its routing table with any other VRF.

This mode is important for ensuring that routing decisions are unambiguous and that there's no overlap between VRFs.

#### **Enabling Strict Mode**

The strict mode can be enabled or disabled by modifying the `strict_mode` parameter through the `sysctl` command:

```
sysctl -w net.vrf.strict_mode=1
```

### **Deploying SRv6 End.DT6 in VRF Mode**

Once the VRF is configured with Strict Mode enabled, you can deploy the SRv6 End.DT6 in VRF mode with the following command:

```
# ip -6 route add fcff:1::100 encap seg6local action End.DT6 \
  vrftable 100 dev vrf100
```

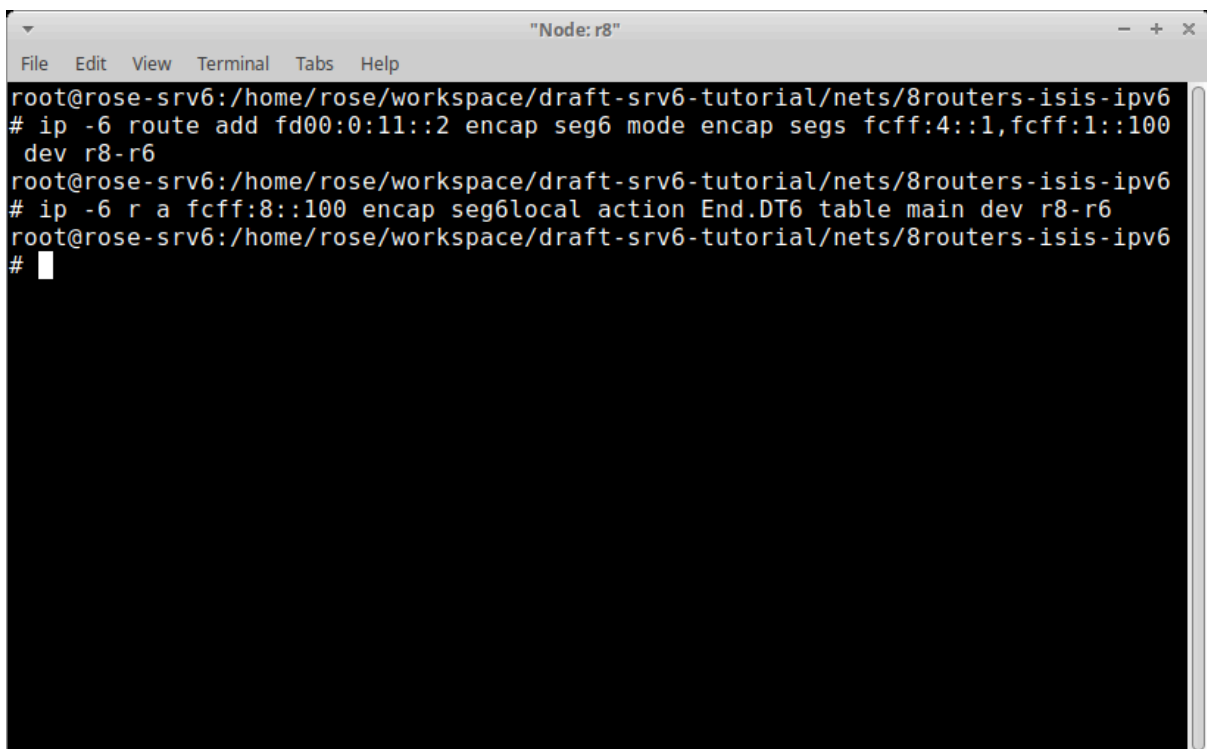


Vice-versa, we configure the R8 router entering into the R8 console by typing:

```
mininet> gterm r8
```

and then:

```
# ip -6 route add fd00:0:11::2 encap seg6 mode encap segs  
fcff:4::1,fcff:1::100 dev r8-r6  
  
# ip -6 route add fcff:8::100 encap seg6local action End.DT6  
table main dev r8-r6
```



```
"Node: r8"  
File Edit View Terminal Tabs Help  
root@rose-srv6:/home/rose/workspace/draft-srv6-tutorial/nets/8routers-isis-ipv6  
# ip -6 route add fd00:0:11::2 encap seg6 mode encap segs fcff:4::1,fcff:1::100  
dev r8-r6  
root@rose-srv6:/home/rose/workspace/draft-srv6-tutorial/nets/8routers-isis-ipv6  
# ip -6 r a fcff:8::100 encap seg6local action End.DT6 table main dev r8-r6  
root@rose-srv6:/home/rose/workspace/draft-srv6-tutorial/nets/8routers-isis-ipv6  
#
```

Hereafter, the installed routes in the R8 router:

```
Node: r8
File Edit View Terminal Tabs Help
fcf0:0:1:2::/64 nhid 26 via fe80::946a:b7ff:fe23:e69b dev r8-r7 proto isis metric 20 pref medium
fcf0:0:2:3::/64 nhid 26 via fe80::946a:b7ff:fe23:e69b dev r8-r7 proto isis metric 20 pref medium
fcf0:0:2:7::/64 nhid 26 via fe80::946a:b7ff:fe23:e69b dev r8-r7 proto isis metric 20 pref medium
fcf0:0:3:4::/64 nhid 22 via fe80::3032:3bff:febf:8ddf dev r8-r6 proto isis metric 20 pref medium
fcf0:0:4:5::/64 nhid 22 via fe80::3032:3bff:febf:8ddf dev r8-r6 proto isis metric 20 pref medium
fcf0:0:4:6::/64 nhid 22 via fe80::3032:3bff:febf:8ddf dev r8-r6 proto isis metric 20 pref medium
fcf0:0:5:6::/64 nhid 22 via fe80::3032:3bff:febf:8ddf dev r8-r6 proto isis metric 20 pref medium
fcf0:0:6:7::/64 nhid 27 proto isis metric 20
    nexthop via fe80::3032:3bff:febf:8ddf dev r8-r6 weight 1
    nexthop via fe80::946a:b7ff:fe23:e69b dev r8-r7 weight 1 pref medium
fcf0:0:6:8::/64 dev r8-r6 proto kernel metric 256 pref medium
fcf0:0:7:8::/64 dev r8-r7 proto kernel metric 256 pref medium
fcff:1::/32 nhid 26 via fe80::946a:b7ff:fe23:e69b dev r8-r7 proto isis metric 20 pref medium
fcff:2:1::/48 nhid 26 via fe80::946a:b7ff:fe23:e69b dev r8-r7 proto isis metric 20 pref medium
fcff:2::/32 nhid 26 via fe80::946a:b7ff:fe23:e69b dev r8-r7 proto isis metric 20 pref medium
fcff:3::/32 nhid 27 proto isis metric 20
    nexthop via fe80::3032:3bff:febf:8ddf dev r8-r6 weight 1
    nexthop via fe80::946a:b7ff:fe23:e69b dev r8-r7 weight 1 pref medium
fcff:4::/32 nhid 22 via fe80::3032:3bff:febf:8ddf dev r8-r6 proto isis metric 20 pref medium
fcff:5:1::/48 nhid 22 via fe80::3032:3bff:febf:8ddf dev r8-r6 proto isis metric 20 pref medium
fcff:5::/32 nhid 22 via fe80::3032:3bff:febf:8ddf dev r8-r6 proto isis metric 20 pref medium
fcff:6::/32 nhid 22 via fe80::3032:3bff:febf:8ddf dev r8-r6 proto isis metric 20 pref medium
fcff:7::/32 nhid 26 via fe80::946a:b7ff:fe23:e69b dev r8-r7 proto isis metric 20 pref medium
fcff:8::100 encap seg6local action End.DT6 table 0 dev r8-r6 metric 1024 pref medium
fcff:8:1::/48 dev r8-hdc2 proto kernel metric 256 pref medium
fcff:8::/32 dev lo proto kernel metric 256 pref medium
fd00:0:11::2 encap seg6 mode encap segs 2 [ fcff:4::1 fcff:1::100 ] dev r8-r6 metric 1024 pref medium
fd00:0:81::/64 dev r8-h81 proto kernel metric 256 pref medium
fd00:0:82::/64 dev r8-h82 proto kernel metric 256 pref medium
fd00:0:83::/64 dev r8-h83 proto kernel metric 256 pref medium
fe80::/64 dev r8-r6 proto kernel metric 256 pref medium
fe80::/64 dev r8-r7 proto kernel metric 256 pref medium
fe80::/64 dev r8-h81 proto kernel metric 256 pref medium
fe80::/64 dev r8-h82 proto kernel metric 256 pref medium
fe80::/64 dev r8-h83 proto kernel metric 256 pref medium
fe80::/64 dev r8-hdc2 proto kernel metric 256 pref medium
root@rose-srv6:/home/rose/workspace/draft-srv6-tutorial/nets/8routers-isis-ipv6#
```

At the end of the configuration process, the two tunnels are established and we can check if h11 reaches the node h81 using the ping command:

```
# ping -c 3 fd00:0:81::2
```

```
"Node: h11"
File Edit View Terminal Tabs Help
root@rose-srv6:/home/rose/workspace/draft-srv6-tutorial/nets/8routers-isis-ipv6
# ping -c 3 fd00:0:81::2
PING fd00:0:81::2(fd00:0:81::2) 56 data bytes
64 bytes from fd00:0:81::2: icmp_seq=1 ttl=63 time=0.226 ms
64 bytes from fd00:0:81::2: icmp_seq=2 ttl=63 time=0.119 ms
64 bytes from fd00:0:81::2: icmp_seq=3 ttl=63 time=0.103 ms

--- fd00:0:81::2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2046ms
rtt min/avg/max/mdev = 0.103/0.149/0.226/0.054 ms
root@rose-srv6:/home/rose/workspace/draft-srv6-tutorial/nets/8routers-isis-ipv6
# █
```

And now... It works!

We can capture the traffic on R8 router using the wireshark packet analyzer by typing:

```
mininet> gterm r8
```

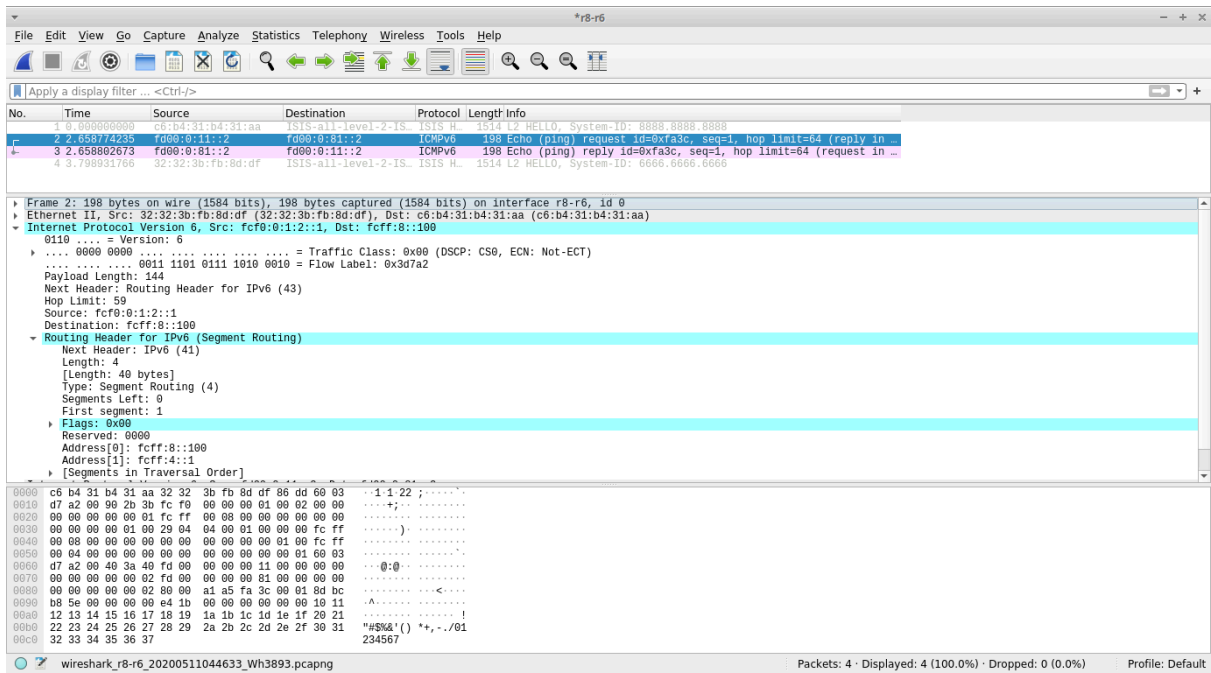
and then:

```
# wireshark 2>/dev/null &
```

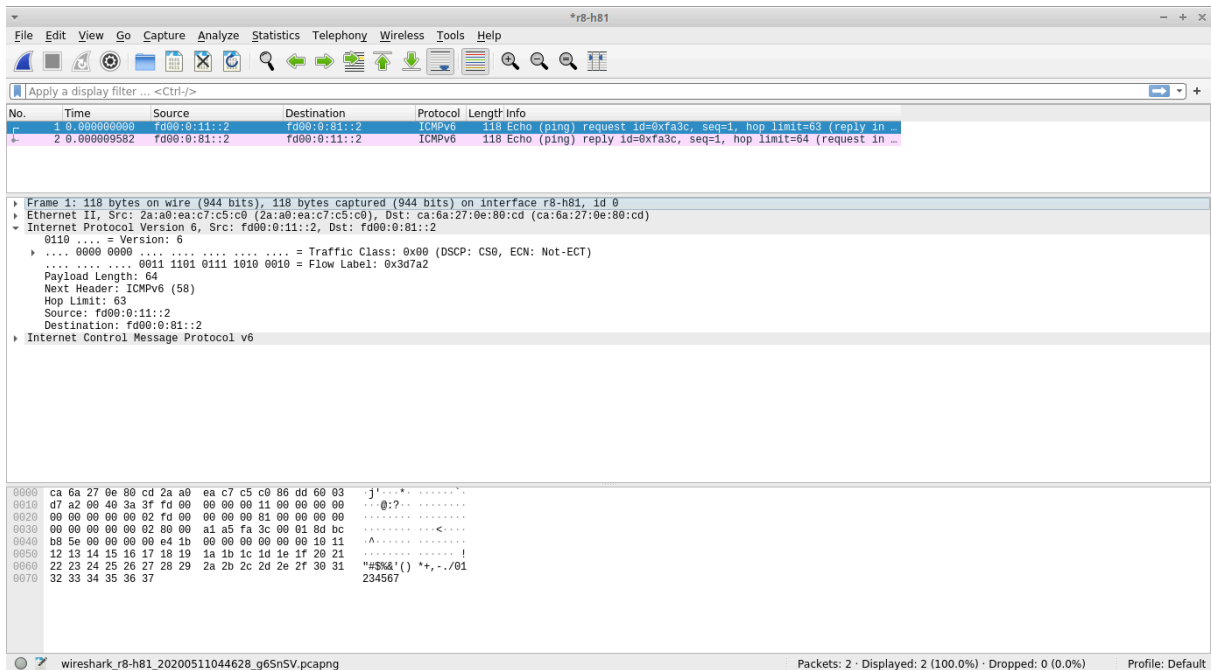
For the curious: i) `2>/dev/null` redirects *stderr* (fd 2) to the black hole (discard any output error of this command) and ii) `&` makes the command run in the background.

Using wireshark, we can observe:

- 1) the R8 incoming traffic sent by the node r11 which has been encapsulated by the router R1 in a Segment Routing packet; (run the capture on the interface r8-r6)



2) the original plain IPv6 packet (sent by the node h11) which has been extracted by the Segment Routing packet and sent to the destination h81 (start the capture on the interface r8-h81).



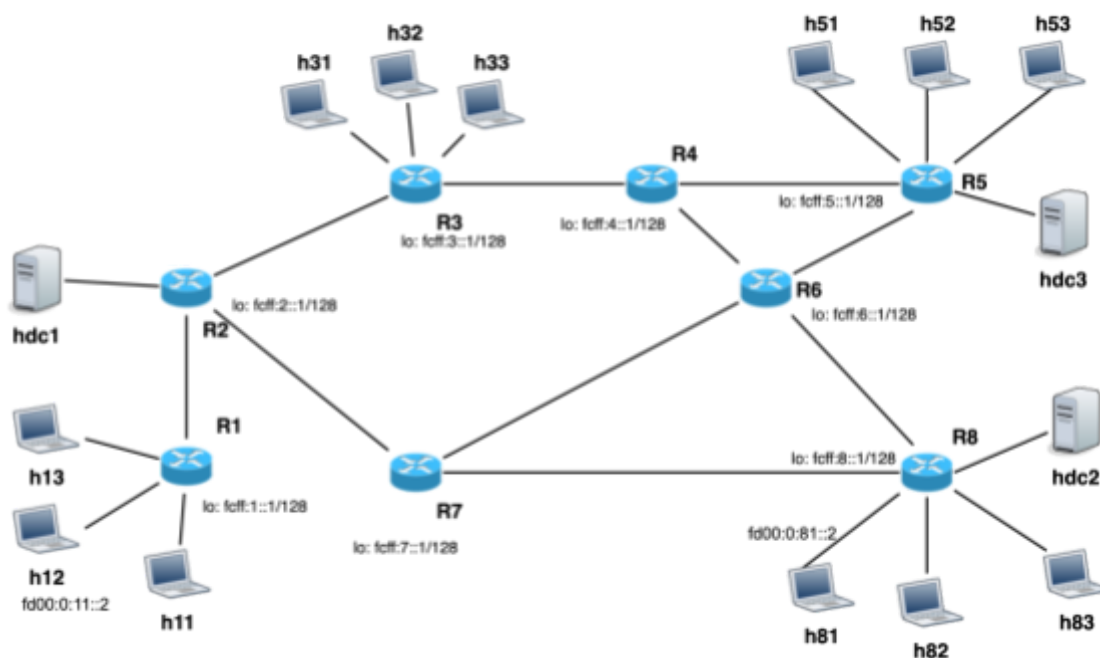
When the packet is received by the r81 node, it replies with a “ping reply” which is encapsulated by the R8 gateway in a SRv6 packet using the SRv6 reverse tunnel.

The packet reaches the router R1 which decaps the inner “ping reply” forwarding it to the h11.

# Porting 8routers-isis testbed to named network nets

The 8-router ISIS testbed was carefully reconstructed while ensuring that all dependencies on Zebra and IS-IS dynamic routing daemons were eliminated. The objective of this reconstruction was to establish a more controlled and reproducible self-contained testing environment.

In order to enhance configurability and streamline the setup process, the provided script incorporates adjustable setting variables. These variables facilitate the implementation of specific SRv6 policies, which are essential for establishing Layer 3 Virtual Private Networks (L3VPNs) between host nodes.



## To log-in to the testbed machine

```
ssh ubuntu@160.80.105.58
ubuntu@pal-r3-s08:~$ lxc exec hawaii-tutorial bash
root@hawaii-tutorial:~# screen -x
root@hawaii-tutorial:~#
root@hawaii-tutorial:~# cd ~/hawaii/rose-srv6-tutorial/nets/8routers-named-netns/
root@hawaii-tutorial:~/[...]/8routers-named-netns# ./named-ns-8r.sh
```

Implemented L3VPN services as follows:

| src ↔ dst | Policy                         | encap | topology   |
|-----------|--------------------------------|-------|--|
| h11 ↔ h81 | "r1,h81,srh,r3 r5,r8"          | plain | h11 - <b>green</b> - r2 - <b>red</b> - r4 - <b>red</b> - r6 - <b>blue</b> - h81                      |
| h81 ↔ h11 | "r8,h11,srh,,r1"               | plain | h81 - <b>green</b> - r6 - r4 - r3 - r2 - <b>blue</b> - h11   |
| h12 ↔ h82 | "r1,h82,red,r5 r7,r8"          | csid  | h12 - <b>green</b> - r2 - r3 - r4 - <b>red</b> - r6 - <b>blue</b> - h82                              |
| h82 ↔ h12 | "r8,h12,red,,r1"               | csid  | h82 - <b>green</b> - r6 - r4 - r3 - r2 - <b>blue</b> - h12   |
| h13 ↔ h83 | "r1,h83,red,r2 r7 r4 r5,r8"    | csid  | h13 - <b>green</b> - <b>red</b> - <b>red</b> - r6 - <b>red</b> - <b>red</b> - r6 - <b>blue</b> - h83 |
| h83 ↔ h13 | "r8,h13,srh,r7 r6 r5 r3 r2,r1" | plain | h83 - <b>green</b> - <b>red</b> - <b>red</b> - r4 - <b>red</b> - <b>red</b> - <b>blue</b> - h13      |

where in:

- **green** is an srv6 encap node
- **black** is an srv6 unaware node
- **red** is an srv6 aware node
- **blue** is an srv6 decap node