SDN Module 3.2 Lab: SRv6

Objectives

The objectives of this lab are to understand the operation of Segment Routing over an IPv6 data plane (SRv6). This lab exercise will comprise four different parts as follows:

1. Analysis of the base IPv6 routing setup of the lab by verifying connectivity, routing tables and configuration files in the context of two different IPv6 routing implementations (Ubuntu Linux using the BIRD routing daemon and the Cisco IOS XRv implementation).
2. Base configuration to support SRv6 test cases.
3. Configuration and analysis of an overlay IPv4 service over an IPv6 underlay using SRv6 encapsulation. For this first case, shortest-path routing will be used in the IPv6 underlay.
4. Configuration and analysis of an overlay IPv4 service over an IPv6 underlay using SRv6 encapsulation. For this second case, a traffic-engineered path will be used in the IPv6 underlay.

Prerequisites:

Knowledge of IPv6, IGP, MPLS and Segment Routing (MPLS) is required.

Lab Setup

The base SRv6 lab topology comprises of 8 routers: R1, R2, R3, R4, R5, R6, R7 and R8. Multiple sets of identical topologies will be used. Workshop attendees will be split up into groups of 6 where each group member will be responsible for the configuration of a specific router as advised by your instructors.

Your instructors will provide details of the IP addresses and port numbers to be used to connect to each of these devices.

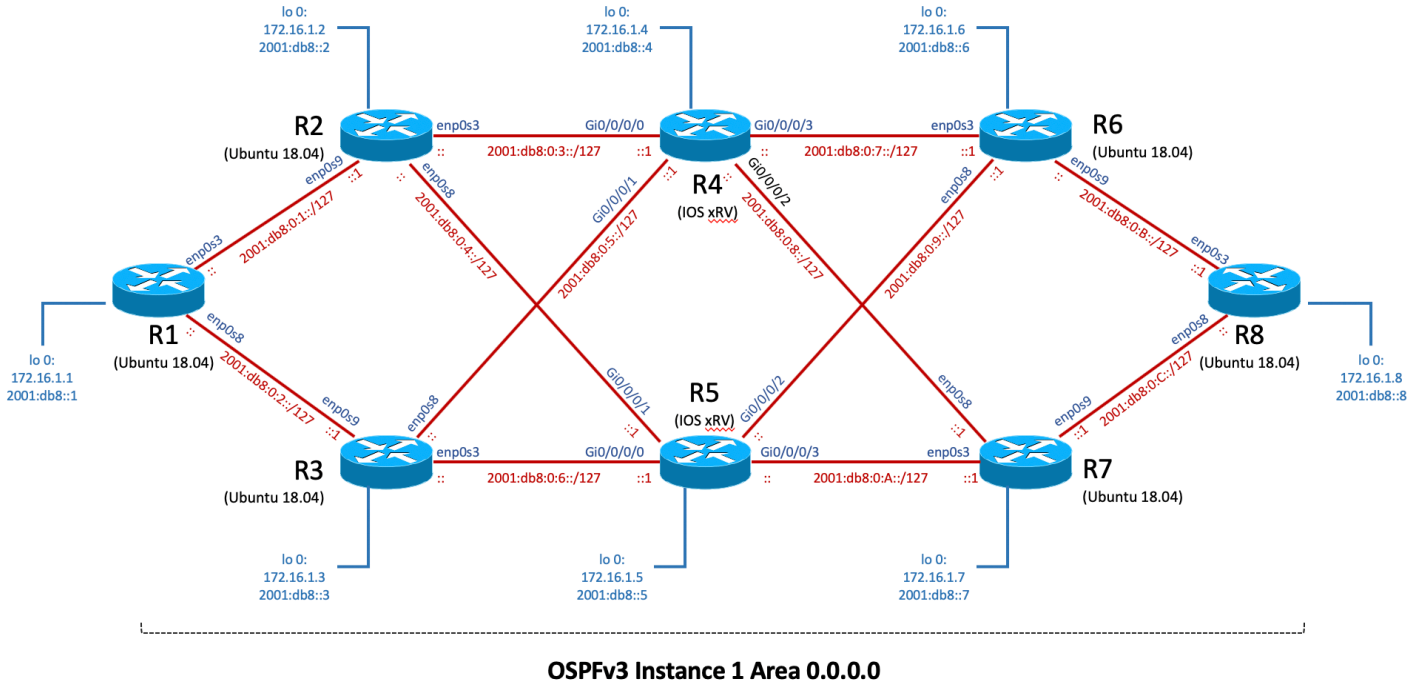
Routers R1, R2, R3, R6, R7 and R8 are Ubuntu 18.04 servers configured as routers. Routers R4 and R5 are running Cisco IOS-XRv. The entire routing setup is running within an GNS3 environment. You will be able to log into each of the routers but will not be able to access the GNS3 setup itself.

Login credentials for the routers are:

- Username: `apnic`
- Password: `apnic`

In the first part of this lab, you will spend some time getting familiar with the network topology and addressing plan before you start building the configuration on the routers. Note that the base IPv4 interface and routing configuration has already been done for you.

The following will be the common topology and IP address plan used for this lab.



Guide to Addressing

- Loopback addresses:
  - IPv4 range: 172.16.1.x/24 where 'x' is the router number
    NOTE: the only reason for configuring IPv4 loopback addresses is for using them as router IDs for OSPFv3.
  - IPv6 range: 2001:db8::x/64 where 'x' is the router number
  - Ubuntu Servers: the loopback interface is interface `lo`
  - Cisco IOS XRv routers: the loopback interface is `interface loopback 0`

- Link addresses:
  - IPv6 range: 2001:db8:x:0::/127 where 'x' is between 1 and C.
  - The router on the left is assigned the '::' address while the router on the right is assigned the '::1' address of the /127.
  - Ubuntu Servers: the following interfaces are configured for the router interconnections:
    - enp0s3
    - enp0s8
    - enp0s9
  - Cisco IOS XRv routers: the following interfaces are configured for the router interconnections:
    - GigabitEthernet0/0/0/0

- GigabitEthernet0/0/0/1
- GigabitEthernet0/0/0/2
- GigabitEthernet0/0/0/2

Part 1: Analysis of the base IPv6 routing setup

In this first part, you will analyse the base IPv6 routing configuration. While this is an important part of SRv6, the base IPv6 interface and routing configuration has already been done for you in order to save time.

Lab Exercise Part 1(a) - Establish connectivity to Ubuntu router

1. Telnet to your designated router's console port as shown below:

```
telnet <IP advised by instructor> 5000 [R1]
telnet <IP advised by instructor> 5001 [R2]
telnet <IP advised by instructor> 5002 [R3]
telnet <IP advised by instructor> 5005 [R6]
telnet <IP advised by instructor> 5006 [R7]
telnet <IP advised by instructor> 5007 [R8]
```

2. Login using the following credentials:

   - Username: `apnic`
   - Password: `apnic`

Lab Exercise Part 1(b) - Verify Ubuntu router addressing configuration and connectivity

1. We will start by viewing the network interface configuration of the router.

   Note: With version 18.04 (Bionic Beaver), Ubuntu has switched to Netplan for the configuration of network interfaces. This is a YAML based configuration system, which simplifies the configuration process. This new tool replaces the configuration file (`/etc/network/interfaces`) that had previously been used to configure the network interfaces on Ubuntu. The configuration files are now located as YAML files at `/etc/netplan/*.yaml`

   View the configuration file by issuing the following command:

```
apnic@R1-ubuntu1804:~$ cat /etc/netplan/50-cloud-init.yaml
```

   NOTE: In the sections that follow, all configuration commands and outptuts are provided from the perspective of R1. Be careful to modify your own configuration to suit the router assigned

to you.

As indicated above, all the routers are pre-configured with basic interface and OSPF configuration according to the topology diagram above. Your router should have a netplan configuration similar to the following, which shows the configuration for router **R1**:

```
apnic@R1-ubuntu1804:~$ cat /etc/netplan/50-cloud-init.yaml
# This file is generated from information provided by
# the datasource.  Changes to it will not persist across an instance.
# To disable cloud-init's network configuration capabilities, write a file
# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
# network: {config: disabled}
network:
    ethernets:
        lo:
            renderer: networkd
            match:
              name: lo
            addresses: [172.16.1.1/32,"2001:db8::1/128","fe80::1/128"]
        enp0s3:
            dhcp4: false
            dhcp6: false
            addresses: ["2001:db8:0:1::/127"]
        enp0s8:
            dhcp4: false
            dhcp6: false
            addresses: ["2001:db8:0:2::/127"]
        enp0s9:
            dhcp4: true
            dhcp6: false
        enp0s10:
            dhcp4: false
            dhcp6: false
    version: 2
apnic@R1-ubuntu1804:~$
```

Note carefully the following aspects of the configuration and ensure that it matches the topology above:

- the IPv4 and IPv4 address configuration of interface `lo`
- the correct link addresses for the other interfaces as per the addressing indicated in the topology diagram

While no changes to this file are required at this stage, if and when you do make changes to it, apply the configuration using the following command:

```
apnic@R1-ubuntu1804:~$ sudo netplan apply
```

2. Verify that the address configuration in the netplan file has been correctly applied to the router by using the following command:

```
apnic@R1-ubuntu1804:~$ ip addr show
```

Your router should output something similar to the following, which shows the configuration for router **R1**:

```
apnic@R1-ubuntu1804:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet 172.16.1.1/32 scope global lo
       valid_lft forever preferred_lft forever
    inet6 2001:db8::1/128 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::1/128 scope link
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:cf:9d:c2 brd ff:ff:ff:ff:ff:ff
    inet6 2001:db8:0:1::/127 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fecf:9dc2/64 scope link
       valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:87:28:88 brd ff:ff:ff:ff:ff:ff
    inet6 2001:db8:0:2::/127 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe87:2888/64 scope link
       valid_lft forever preferred_lft forever
4: enp0s9: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
    link/ether 08:00:27:60:54:5f brd ff:ff:ff:ff:ff:ff
5: enp0s10: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
    link/ether 08:00:27:ed:5d:ba brd ff:ff:ff:ff:ff:ff
apnic@R1-ubuntu1804:~$
```

3. Verify that all your router loopback and all link addresses are reachable by pinging both the near-end and far-end IPv6 link addresses. Use the following syntax:

```
apnic@R1-ubuntu1804:~$ ping -6 -c 3 <IPv6 address>
```

Your router should output something similar to the following, which shows the outputs from router **R1**:

```
apnic@R1-ubuntu1804:~$
apnic@R1-ubuntu1804:~$ ping -6 -c 3 2001:db8::1
PING 2001:db8::1(2001:db8::1) 56 data bytes
64 bytes from 2001:db8::1: icmp_seq=1 ttl=64 time=0.111 ms
64 bytes from 2001:db8::1: icmp_seq=2 ttl=64 time=0.080 ms
64 bytes from 2001:db8::1: icmp_seq=3 ttl=64 time=0.077 ms

--- 2001:db8::1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 0.077/0.089/0.111/0.017 ms
apnic@R1-ubuntu1804:~$
apnic@R1-ubuntu1804:~$ ping -6 -c 3 2001:db8:0:1::
PING 2001:db8:0:1::(2001:db8:0:1::) 56 data bytes
64 bytes from 2001:db8:0:1::: icmp_seq=1 ttl=64 time=0.025 ms
64 bytes from 2001:db8:0:1::: icmp_seq=2 ttl=64 time=0.082 ms
64 bytes from 2001:db8:0:1::: icmp_seq=3 ttl=64 time=0.075 ms

--- 2001:db8:0:1:: ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2019ms
rtt min/avg/max/mdev = 0.025/0.060/0.082/0.026 ms
apnic@R1-ubuntu1804:~$
apnic@R1-ubuntu1804:~$ ping -6 -c 3 2001:db8:0:1::1
PING 2001:db8:0:1::1(2001:db8:0:1::1) 56 data bytes
64 bytes from 2001:db8:0:1::1: icmp_seq=1 ttl=64 time=2.76 ms
64 bytes from 2001:db8:0:1::1: icmp_seq=2 ttl=64 time=1.48 ms
64 bytes from 2001:db8:0:1::1: icmp_seq=3 ttl=64 time=1.56 ms

--- 2001:db8:0:1::1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.489/1.939/2.763/0.584 ms
apnic@R1-ubuntu1804:~$
apnic@R1-ubuntu1804:~$ ping -6 -c 3 2001:db8:0:2::
PING 2001:db8:0:2::(2001:db8:0:2::) 56 data bytes
64 bytes from 2001:db8:0:2::: icmp_seq=1 ttl=64 time=0.076 ms
64 bytes from 2001:db8:0:2::: icmp_seq=2 ttl=64 time=0.078 ms
64 bytes from 2001:db8:0:2::: icmp_seq=3 ttl=64 time=0.077 ms

--- 2001:db8:0:2:: ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 0.076/0.077/0.078/0.000 ms
apnic@R1-ubuntu1804:~$
apnic@R1-ubuntu1804:~$ ping -6 -c 3 2001:db8:0:2::1
PING 2001:db8:0:2::1(2001:db8:0:2::1) 56 data bytes
64 bytes from 2001:db8:0:2::1: icmp_seq=1 ttl=64 time=1.13 ms
64 bytes from 2001:db8:0:2::1: icmp_seq=2 ttl=64 time=1.51 ms
64 bytes from 2001:db8:0:2::1: icmp_seq=3 ttl=64 time=1.43 ms

--- 2001:db8:0:2::1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.135/1.361/1.518/0.166 ms
apnic@R1-ubuntu1804:~$
```

At the end of this step, you have verified that the interface address configuration on your router, as well

as your adjacent routers, is correct.

NOTE: Do not proceed further if the above is not the case. Request the assistance of an instructor in order to resolve the issue.

Lab Exercise Part 1(c) - Verify Ubuntu router routing configuration and connectivity

We are using the BIRD internet routing daemon on the Ubuntu servers in order to provide IPv6 routing protocol capabilities. With the version of BIRD that we are using, v1.6, two separate daemons are run for IPv4 and IPv6. These are `/usr/sbin/bird` and `/usr/sbin/bird6`, respectively.

1. We will familiarise ourselves with BIRD by viewing its configuration file and running some CLI commands. The BIRD configuration files are located in `/etc/bird` and require super-user privilege to access.

   View the bird6 configuration file by issuing the following command:

   ```
   apnic@R1-ubuntu1804:~$ sudo cat /etc/bird/bird6.conf
   ```

   NOTE: In the sections that follow, all configuration commands and outptuts are provided from the perspective of R1. Be careful to modify your own configuration to suit the router assigned to you.

   As indicated above, all the routers are pre-configured with basic interface and OSPF configuration according to the topology diagram above. Your router should have a BIRD configuration similar to the following, which shows the configuration for router **R1**:

```
apnic@R1-ubuntu1804:~$ sudo cat /etc/bird/bird6.conf
# This is a minimal configuration file, which allows the bird daemon to start
# but will not cause anything else to happen.
#
# Please refer to the documentation in the bird-doc package or BIRD User's
# Guide on http://bird.network.cz/ for more information on configuring BIRD and
# adding routing protocols.

# Change this into your BIRD router ID. It's a world-wide unique identification
# of your router, usually one of router's IPv4 addresses.
router id 172.16.1.1;

# The Kernel protocol is not a real routing protocol. Instead of communicating
# with other routers in the network, it performs synchronization of BIRD's
# routing tables with the OS kernel.
protocol kernel {
        learn;
        scan time 60;
        import all;
        export all;   # Actually insert routes into the kernel routing table
}

# The Device protocol is not a real routing protocol. It doesn't generate any
# routes and it only serves as a module for getting information about network
# interfaces from the kernel.
protocol device {
        scan time 60;
}

protocol ospf {
        import all;
        export all;

        area 0 {
                interface "enp0s3", "enp0s8", "enp0s9" {
                        type pointopoint;
                };
                interface "lo";
        };
}
apnic@R1-ubuntu1804:~$
```

Analysis of the bird6.conf file

Note carefully the following aspects of the configuration for your router and ensure that it matches the topology above. BIRD has an internal routing table that is referred to in the comments that follow.

- Configuration of the Router ID corresponding to your router `router id 172.16.1.1;`
- Configuration of the `protocol kernel` block:

```
protocol kernel {
    learn;
    scan time 60;
    import all;
    export all;
}
```

- `learn` allows routes such as statically configured routes to be imported into the Bird routing table
- `import all` enables the importing of routes from the kernel routing table to the BIRD routing table
- `export all` actually insert routes into the kernel routing table from the Bird routing table; this is needed so that ospf routes learnt from other routers can actually be used for forwarding

- Configuration of the `protocol ospf` block:

```
protocol ospf {
    import all;
    export all;

    area 0 {
        interface "enp0s3", "enp0s8", "enp0s9" {
            type pointopoint;
        };
        interface "lo";
    };
}
```

- `import all` enable all routes to be imported into the BIRD routing table from OSPF (default behaviour)
- `export all` enable all routes to be exported from the BIRD routing table into OSPF (disabled by default)
- `area 0` enables OSPF v3 on the configured router interfaces
- `type pointopoint` explicitly defines the OSPF type for these interfaces as point-to-point

2. BIRD provides a command-line client `birdc` (for IPv4) and `birdc6` to allow communication with the running BIRD daemon.

   Invoke the birdc6 tool by issuing the following command:

```
apnic@R1-ubuntu1804:~$ sudo birdc6
```

   `birdc6` provides a number of useful commands that you can use to verify the operation of the

various routing protocols.

a. In the first instance, enter the following command to view the list of supported commands:

```
bird> ?
```

A sample output for **R1** follows:

```
apnic@R1-ubuntu1804:~$
apnic@R1-ubuntu1804:~$ sudo birdc6
BIRD 1.6.8 ready.
bird> ?
add roa ...                                 Add ROA record
configure ...                               Reload configuration
debug ...                                   Control protocol debugging via BIRD logs
delete roa ...                              Delete ROA record
disable (<protocol> | "<pattern>" | all) [message]  Disable protocol
down                                        Shut the daemon down
dump ...                                    Dump debugging information
echo ...                                    Control echoing of log messages
enable (<protocol> | "<pattern>" | all) [message]  Enable protocol
eval <expr>                                 Evaluate an expression
exit                                        Exit the client
flush roa [table <name>]                    Removes all dynamic ROA records
help                                        Description of the help system
mrtdump ...                                 Control protocol debugging via MRTdump files
quit                                        Quit the client
reload <protocol> | "<pattern>" | all       Reload protocol
restart (<protocol> | "<pattern>" | all) [message]  Restart protocol
restrict                                    Restrict current CLI session to safe commands
show ...                                    Show status information
bird>
bird>
```

b. Enter the following command to view status information about the BIRD daemon

```
bird> show status
```

A sample output for **R1** follows:

```
bird> show status
BIRD 1.6.8
Router ID is 172.16.1.1
Current server time is 2020-01-22 12:27:25
Last reboot on 2020-01-22 04:22:08
Last reconfiguration on 2020-01-22 10:45:53
Daemon is up and running
bird>
```

c. Enter the following command to view status information about the configured protocols:

```
bird> show protocols
```

A sample output for **R1** follows:

```
bird> show protocols
name     proto    table    state  since       info
device1  Device   master   up     04:22:07
ospf1    OSPF     master   up     04:22:07    Running
kernel1  Kernel   master   up     07:18:32
bird>
```

d. Enter the following command to view status information about the OSPF process:

```
bird> show ospf
```

A sample output for **R1** follows:

```
bird> show ospf
ospf1:
RFC1583 compatibility: disabled
Stub router: No
RT scheduler tick: 1
Number of areas: 1
Number of LSAs in DB:   23
        Area: 0.0.0.0 (0) [BACKBONE]
                Stub:   No
                NSSA:   No
                Transit:        No
                Number of interfaces:   3
                Number of neighbors:    2
                Number of adjacent neighbors:   2
bird>
```

e. Enter the following command to display the BIRD routing table:

```
bird> show route
```

A sample output for **R1** follows:

```
bird> show route
2001:db8:0:4::/127 via fe80::a00:27ff:fe83:f34a on enp0s3 [ospf1 22:16:31] * I (150/20) [172.16.1.2]
2001:db8::4/128    via fe80::a00:27ff:fe83:f34a on enp0s3 [ospf1 22:18:57] * I (150/20) [172.16.1.4]
2001:db8:0:5::/127 via fe80::a00:27ff:fefd:4a0b on enp0s8 [ospf1 22:16:39] * I (150/20) [172.16.1.3]
2001:db8::5/128    via fe80::a00:27ff:fe83:f34a on enp0s3 [ospf1 22:19:00] * I (150/20) [172.16.1.5]
2001:db8:0:6::/127 via fe80::a00:27ff:fefd:4a0b on enp0s8 [ospf1 22:16:39] * I (150/20) [172.16.1.3]
2001:db8::6/128    via fe80::a00:27ff:fe83:f34a on enp0s3 [ospf1 22:18:57] * I (150/21) [172.16.1.6]
2001:db8:0:7::/127 via fe80::a00:27ff:fe83:f34a on enp0s3 [ospf1 22:18:57] * I (150/21) [172.16.1.4]
2001:db8::7/128    via fe80::a00:27ff:fe83:f34a on enp0s3 [ospf1 22:19:02] * I (150/21) [172.16.1.7]
2001:db8::1/128    dev lo [ospf1 22:16:18] * I (150/0) [172.16.1.1]
2001:db8:0:1::/127 dev enp0s3 [ospf1 22:16:18] * I (150/10) [172.16.1.1]
2001:db8:0:2::/127 dev enp0s8 [ospf1 22:16:18] * I (150/10) [172.16.1.1]
2001:db8::2/128    via fe80::a00:27ff:fe83:f34a on enp0s3 [ospf1 22:16:31] * I (150/10) [172.16.1.2]
2001:db8:0:3::/127 via fe80::a00:27ff:fe83:f34a on enp0s3 [ospf1 22:16:31] * I (150/20) [172.16.1.2]
2001:db8::3/128    via fe80::a00:27ff:fefd:4a0b on enp0s8 [ospf1 22:16:39] * I (150/10) [172.16.1.3]
2001:db8:0:c::/127 via fe80::a00:27ff:fe83:f34a on enp0s3 [ospf1 22:19:02] * I (150/31) [172.16.1.7]
2001:db8:0:8::/127 via fe80::a00:27ff:fe83:f34a on enp0s3 [ospf1 22:18:57] * I (150/21) [172.16.1.4]
2001:db8::8/128    via fe80::a00:27ff:fe83:f34a on enp0s3 [ospf1 22:18:57] * I (150/31) [172.16.1.8]
2001:db8:0:9::/127 via fe80::a00:27ff:fe83:f34a on enp0s3 [ospf1 22:19:00] * I (150/21) [172.16.1.5]
2001:db8:0:a::/127 via fe80::a00:27ff:fe83:f34a on enp0s3 [ospf1 22:19:00] * I (150/21) [172.16.1.5]
2001:db8:0:b::/127 via fe80::a00:27ff:fe83:f34a on enp0s3 [ospf1 22:18:57] * I (150/31) [172.16.1.6]
bird>
```

The IPv6 routing table should have the following routes:

- 8 x IPv6 loopback addresses as follows:

```
2001:db8::1/128
2001:db8::2/128
2001:db8::3/128
2001:db8::4/128
2001:db8::5/128
2001:db8::6/128
2001:db8::7/128
2001:db:::8/128
```

- 12 x IPv6 link sunnets:

```
2001:db8:0:1::/127
2001:db8:0:2::/127
2001:db8:0:3::/127
2001:db8:0:4::/127
2001:db8:0:5::/127
2001:db8:0:6::/127
2001:db8:0:7::/127
2001:db8:0:8::/127
2001:db8:0:9::/127
2001:db8:0:A::/127
2001:db8:0:B::/127
2001:db8:0:C::/127
```

e. Exit out of birdc6:

```
bird> exit
```

1. Verify that you have basic reachability to all routers in your network by pinging their IPv6 loopback addresses. Use the following syntax:

   ```
   apnic@R1-ubuntu1804:~$ ping -6 -c 3 2001:db8::1
   apnic@R1-ubuntu1804:~$ ping -6 -c 3 2001:db8::2
   apnic@R1-ubuntu1804:~$ ping -6 -c 3 2001:db8::3
   apnic@R1-ubuntu1804:~$ ping -6 -c 3 2001:db8::4
   apnic@R1-ubuntu1804:~$ ping -6 -c 3 2001:db8::5
   apnic@R1-ubuntu1804:~$ ping -6 -c 3 2001:db8::6
   apnic@R1-ubuntu1804:~$ ping -6 -c 3 2001:db8::7
   apnic@R1-ubuntu1804:~$ ping -6 -c 3 2001:db8::8
   ```

   At the end of this step, you have verified that the routing configuration on your router is correct and that you have reachability to all other routers in your network.

   NOTE: Do not proceed further if the above is not the case. Request the assistance of an instructor in order to resolve the issue.

Lab Exercise Part 1(d) - Verify Cisco router configuration and connectivity

You will have noted that there are two Cisco IOS XRv routers, R4 and R5, that have not been assigned to any participant. The purpose for these routers will become clear as the lab proceeds. For the purpose of this part of the lab, you will log in to and verify the configuration and connectivity of these routers. Members of the group will need to take it in turns to access these routers. Alternatively, you may nominate two members of your group to work on these two routers and work together with them.

NOTE: there can be multiple simultaneous console connections to routers R4 and R5 but the same output is streamed to all of these connections. To avoid confusion, co-ordinate the connecticity to the routers between your team members.

1. Telnet to the router's console port as shown below:

   ```
   telnet <IP advised by instructor> 5003 [R4]
   telnet <IP advised by instructor> 5004 [R5]
   ```

2. Login using the following credentials:

   - Username: `apnic`
   - Password: `apnic`

3. View the router's configuration:

```
show running-config
```

All the routers are pre-configured with basic interface and OSPF configuration according to the topology diagram above. Your router should have a configuration similar to the following, which shows the configuration for router R4:

```
RP/0/0/CPU0:R4#show running-config
Thu Jan 23 08:55:33.270 UTC
Building configuration...
!! IOS XR Configuration 6.1.2
!! Last configuration change at Wed Jan 22 23:42:08 2020 by cisco
!
hostname R4
interface Loopback0
 ipv4 address 172.16.1.4 255.255.255.255
 ipv6 address 2001:db8::4/128
!
interface MgmtEth0/0/CPU0/0
 shutdown
!
interface GigabitEthernet0/0/0/0
 ipv6 nd suppress-ra
 ipv6 address 2001:db8:0:3::1/127
!
interface GigabitEthernet0/0/0/1
 ipv6 nd suppress-ra
 ipv6 address 2001:db8:0:5::1/127
!
interface GigabitEthernet0/0/0/2
 ipv6 nd suppress-ra
 ipv6 address 2001:db8:0:8::/127
!
interface GigabitEthernet0/0/0/3
 ipv6 nd suppress-ra
 ipv6 address 2001:db8:0:7::/127
!
router ospfv3 1
 router-id 172.16.1.4
 area 0.0.0.0
  network point-to-point
  interface Loopback0
  !
  interface GigabitEthernet0/0/0/0
  !
  interface GigabitEthernet0/0/0/1
  !
  interface GigabitEthernet0/0/0/2
  !
  interface GigabitEthernet0/0/0/3
  !
 !
!
end

RP/0/0/CPU0:R4#
```

4. View the routing table of the router and ensure that you see all router loopbacks and interface subnets.

```
show route ipv6
```

A sample output for **R4** follows:

```
RP/0/0/CPU0:R4#show route ipv6
Thu Jan 23 08:58:24.748 UTC

Codes: C - connected, S - static, R - RIP, B - BGP, (>) - Diversion path
       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, E - EGP
       i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local, G  - DAGR, l - LISP
       A - access/subscriber, a - Application route
       M - mobile route, r - RPL, (!) - FRR Backup path

Gateway of last resort is not set

O    2001:db8::1/128
      [110/11] via fe80::a00:27ff:fe18:5ba5, 00:39:23, GigabitEthernet0/0/0/0
      [110/11] via fe80::a00:27ff:fe7e:2a02, 00:39:23, GigabitEthernet0/0/0/1
O    2001:db8::2/128
      [110/1] via fe80::a00:27ff:fe18:5ba5, 00:39:23, GigabitEthernet0/0/0/0
O    2001:db8::3/128
      [110/1] via fe80::a00:27ff:fe7e:2a02, 00:39:28, GigabitEthernet0/0/0/1
L    2001:db8::4/128 is directly connected,
      00:39:36, Loopback0
O    2001:db8::5/128
      [110/11] via fe80::a00:27ff:fe18:5ba5, 00:39:18, GigabitEthernet0/0/0/0
      [110/11] via fe80::a00:27ff:fe7e:2a02, 00:39:18, GigabitEthernet0/0/0/1
      [110/11] via fe80::a00:27ff:fe0f:82dd, 00:39:18, GigabitEthernet0/0/0/2
      [110/11] via fe80::a00:27ff:fe02:4da, 00:39:18, GigabitEthernet0/0/0/3
O    2001:db8::6/128
      [110/1] via fe80::a00:27ff:fe02:4da, 00:39:28, GigabitEthernet0/0/0/3
O    2001:db8::7/128
      [110/1] via fe80::a00:27ff:fe0f:82dd, 00:39:23, GigabitEthernet0/0/0/2
O    2001:db8::8/128
      [110/11] via fe80::a00:27ff:fe0f:82dd, 00:39:23, GigabitEthernet0/0/0/2
      [110/11] via fe80::a00:27ff:fe02:4da, 00:39:23, GigabitEthernet0/0/0/3
O    2001:db8:0:1::/127
      [110/11] via fe80::a00:27ff:fe18:5ba5, 00:39:23, GigabitEthernet0/0/0/0
O    2001:db8:0:2::/127
      [110/11] via fe80::a00:27ff:fe7e:2a02, 00:39:28, GigabitEthernet0/0/0/1
C    2001:db8:0:3::/127 is directly connected,
      00:39:36, GigabitEthernet0/0/0/0
L    2001:db8:0:3::1/128 is directly connected,
      00:39:36, GigabitEthernet0/0/0/0
O    2001:db8:0:4::/127
      [110/11] via fe80::a00:27ff:fe18:5ba5, 00:39:23, GigabitEthernet0/0/0/0
C    2001:db8:0:5::/127 is directly connected,
      00:39:36, GigabitEthernet0/0/0/1
L    2001:db8:0:5::1/128 is directly connected,
      00:39:36, GigabitEthernet0/0/0/1
O    2001:db8:0:6::/127
      [110/11] via fe80::a00:27ff:fe7e:2a02, 00:39:28, GigabitEthernet0/0/0/1
C    2001:db8:0:7::/127 is directly connected,
      00:39:36, GigabitEthernet0/0/0/3
L    2001:db8:0:7::/128 is directly connected,
      00:39:36, GigabitEthernet0/0/0/3
C    2001:db8:0:8::/127 is directly connected,
      00:39:36, GigabitEthernet0/0/0/2
L    2001:db8:0:8::/128 is directly connected,
      00:39:36, GigabitEthernet0/0/0/2
O    2001:db8:0:9::/127
      [110/11] via fe80::a00:27ff:fe02:4da, 00:39:28, GigabitEthernet0/0/0/3
O    2001:db8:0:a::/127
      [110/11] via fe80::a00:27ff:fe0f:82dd, 00:39:23, GigabitEthernet0/0/0/2
O    2001:db8:0:b::/127
      [110/11] via fe80::a00:27ff:fe02:4da, 00:39:28, GigabitEthernet0/0/0/3
O    2001:db8:0:c::/127
      [110/11] via fe80::a00:27ff:fe0f:82dd, 00:39:23, GigabitEthernet0/0/0/2
RP/0/0/CPU0:R4#
```

The IPv6 routing table should have the following routes:

- 8 x IPv6 loopback addresses as follows:

```
2001:db8::1/128
2001:db8::2/128
2001:db8::3/128
2001:db8::4/128
2001:db8::5/128
2001:db8::6/128
2001:db8::7/128
2001:db:::8/128
```

- 12 x IPv6 link sunnets:

```
2001:db8:0:1::/127
2001:db8:0:2::/127
2001:db8:0:3::/127
2001:db8:0:4::/127
2001:db8:0:5::/127
2001:db8:0:6::/127
2001:db8:0:7::/127
2001:db8:0:8::/127
2001:db8:0:9::/127
2001:db8:0:A::/127
2001:db8:0:B::/127
2001:db8:0:C::/127
```

5. Verify that you have basic reachability to all routers in your network by pinging their IPv6 loopback addresses. Use the following syntax:

```
RP/0/0/CPU0:R4# ping ipv6 2001:db8::1
RP/0/0/CPU0:R4# ping ipv6 2001:db8::2
RP/0/0/CPU0:R4# ping ipv6 2001:db8::3
RP/0/0/CPU0:R4# ping ipv6 2001:db8::4
RP/0/0/CPU0:R4# ping ipv6 2001:db8::5
RP/0/0/CPU0:R4# ping ipv6 2001:db8::6
RP/0/0/CPU0:R4# ping ipv6 2001:db8::7
RP/0/0/CPU0:R4# ping ipv6 2001:db8::8
```

A sample output for **R4** follows:

```
RP/0/0/CPU0:R4#ping ipv6 2001:db8::1
Thu Jan 23 09:00:45.178 UTC
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:db8::1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/9 ms
RP/0/0/CPU0:R4#
```

NOTE: Before proceeding to the next part, ensure that all members of the group have completed this part.

The first part of this lab is now complete. Wait for your instructors to provide some interpretation of what you have observed so far before proceeding to the next part.

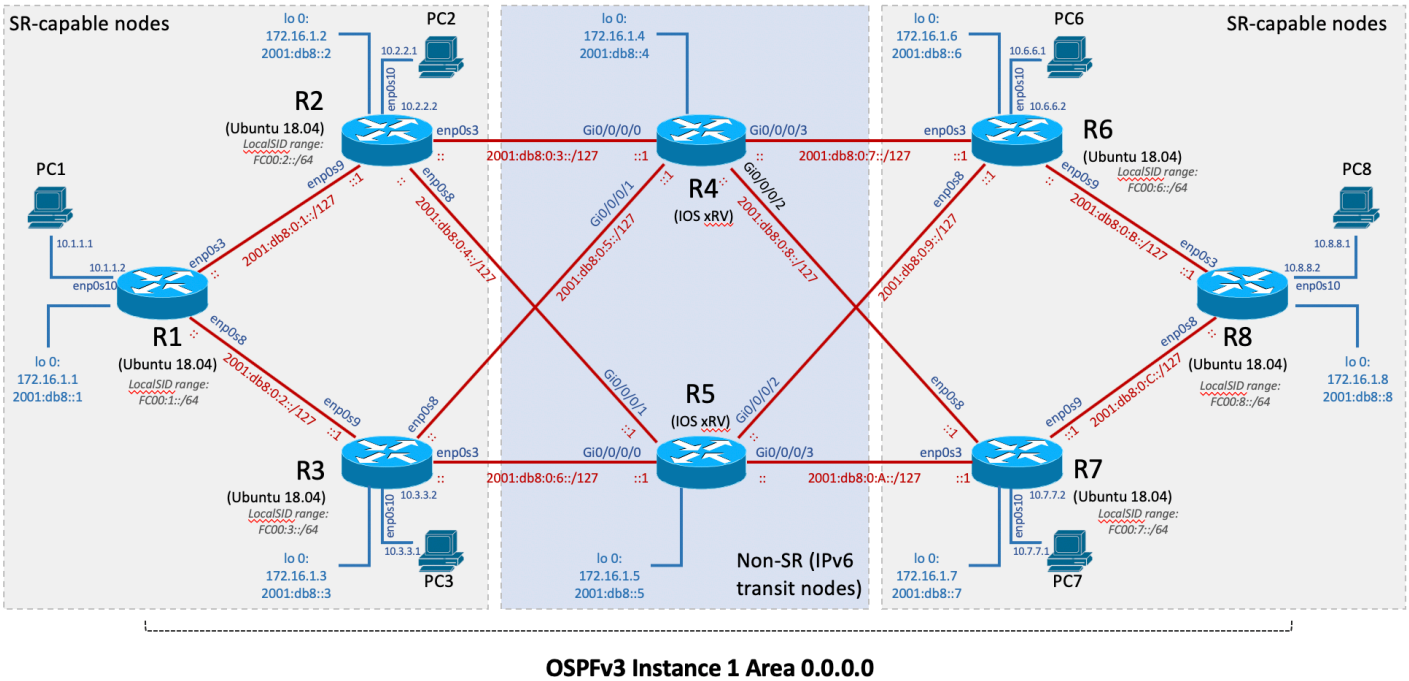
Part 2: Base SRv6 configuration

In part 2 of the lab, you will configure SRv6 processing capabilities on all routers so that you can establish some services over SRv6 infrastructure in subsequent labs. In addition, you will connect a PC to your router so that you can build overlay services.

The following figure illustrates the SRv6 topology that will be used.

Note that the only SRv6-capable nodes are the Ubuntu routers R1,R2,R3,R6,R7 and R8. The Cisco routers R4 and R5 are **not** SRv6-capable.

Note also that there is a PC connected to each SRv6 router.



You will once again be working on the Ubuntu routers assigned to you. There will be no need to perform any configuration on the Cisco routers in this part.

1. Telnet to your designated router's console port as shown below:

```
telnet <IP advised by instructor> 5000 [R1]
telnet <IP advised by instructor> 5001 [R2]
telnet <IP advised by instructor> 5002 [R3]
telnet <IP advised by instructor> 5005 [R6]
telnet <IP advised by instructor> 5006 [R7]
telnet <IP advised by instructor> 5007 [R8]
```

2. Login using the following credentials:

   - Username: `apnic`
   - Password: `apnic`

Lab Exercise Part 2(a) - Enable SRH processing on Ubuntu router interfaces

1. Enable SRH processing on all interfaces that are expected to receive SRH packets. We do this by setting kernel parameters using the `sysctl` tool:

```
apnic@R1-ubuntu1804:~$ sudo sysctl -w net.ipv6.conf.all.seg6_enabled=1
apnic@R1-ubuntu1804:~$ sudo sysctl -w net.ipv6.conf.enp0s3.seg6_enabled=1
apnic@R1-ubuntu1804:~$ sudo sysctl -w net.ipv6.conf.enp0s8.seg6_enabled=1
apnic@R1-ubuntu1804:~$ sudo sysctl -w net.ipv6.conf.enp0s9.seg6_enabled=1
```

2. Verify that the sysctls have taken effect by issuing the following command:

```
apnic@R1-ubuntu1804:~$ sysctl -ar net.ipv6.conf.*.seg6_enabled
```

   Your router should output something similar to the following, which shows the output for router R1:

```
apnic@R1-ubuntu1804:~$ sysctl -ar net.ipv6.conf.*.seg6_enabled
net.ipv6.conf.all.seg6_enabled = 1
net.ipv6.conf.default.seg6_enabled = 0
net.ipv6.conf.enp0s10.seg6_enabled = 0
net.ipv6.conf.enp0s3.seg6_enabled = 1
net.ipv6.conf.enp0s8.seg6_enabled = 1
net.ipv6.conf.enp0s9.seg6_enabled = 1
net.ipv6.conf.lo.seg6_enabled = 0
apnic@R1-ubuntu1804:~$
```

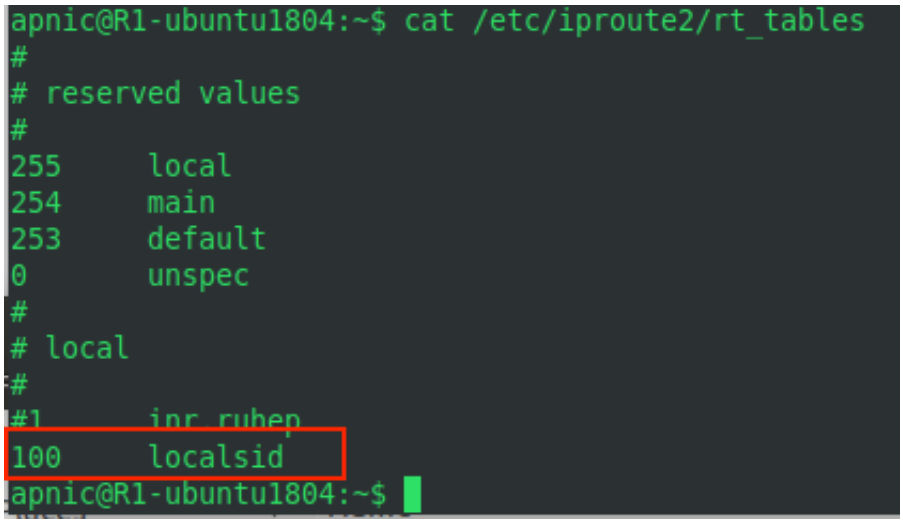
   NOTE: In the sections that follow, all configuration commands and outptuts are provided from the perspective of R1. Be careful to modify your own configuration to suit the router assigned to you.

3. A new routing table, called `localsid` has already been created for you on all routers. Verify its

existence using the following command:

```
apnic@R1-ubuntu1804:~$ cat /etc/iproute2/rt_tables
```

An example for **R1** is given below:



This routing table will be used to hold all local SIDs corresponding to SR endpoint actions.

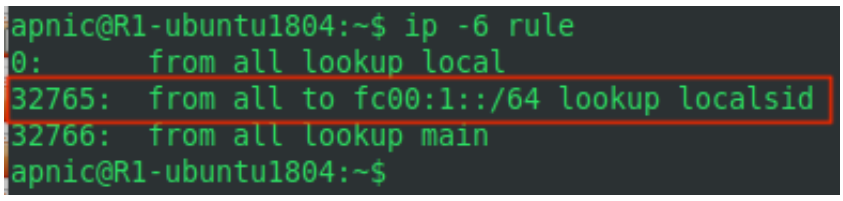
4. Create a new rule in the routing policy database to process all destinations in the localsid block (as per figure above) via the localsid routing table.

```
apnic@R1-ubuntu1804:~$ sudo ip -6 rule add to fc00:1::/64 lookup localsid
```

Verify that this rule has been successfully applied using the following command:

```
apnic@R1-ubuntu1804:~$ ip -6 rule
```

An example for **R1** is given below:



The lower priority value of the rule, 32765 (implying higher priority), means that it will direct all packets destined to the FC00:1::/64 network to the localsid routing table.

5. Create a blackhole route in the localsid routing table to drop all packets with destinations in the localsid range but without an explicit match.

```
apnic@R1-ubuntu1804:~$ sudo ip -6 route add blackhole default table localsid
```

Verify that this route has been successfully applied using the following command:

```
apnic@R1-ubuntu1804:~$ ip -6 route show table localsid
```

An example for **R1** is given below:

```
apnic@R1-ubuntu1804:~$ ip -6 route show table localsid
blackhole default dev lo metric 1024 error -22 pref medium
apnic@R1-ubuntu1804:~$
```

6. Create a route in the main routing table for the localsid block so that it can be advertised to the rest of the network:

```
apnic@R1-ubuntu1804:~$ sudo ip -6 route add blackhole fc00:1::/64
```

Verify that this route has been successfully applied using the following command:

```
apnic@R1-ubuntu1804:~$ ip -6 route
```

An example for **R1** is given below:

```
apnic@R1-ubuntu1804:~$ ip -6 route
local ::1 dev lo proto kernel metric 256 pref medium
unreachable 2001:db8::1 dev lo proto kernel metric 256 error -101 pref medium
2001:db8::2 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8::3 via fe80::a00:27ff:fefd:4a0b dev enp0s8 proto bird metric 1024 pref medium
2001:db8::4 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8::5 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8::6 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8::7 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8::8 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8:0:1::/127 dev enp0s3 proto kernel metric 256 pref medium
2001:db8:0:2::/127 dev enp0s8 proto kernel metric 256 pref medium
2001:db8:0:3::/127 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8:0:4::/127 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8:0:5::/127 via fe80::a00:27ff:fefd:4a0b dev enp0s8 proto bird metric 1024 pref medium
2001:db8:0:6::/127 via fe80::a00:27ff:fefd:4a0b dev enp0s8 proto bird metric 1024 pref medium
2001:db8:0:7::/127 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8:0:8::/127 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8:0:9::/127 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8:0:a::/127 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8:0:b::/127 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8:0:c::/127 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
blackhole fc00:1::/64 dev lo metric 1024 error -22 pref medium
fc00:8::/64 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
unreachable fe80::1 dev lo proto kernel metric 256 error -101 pref medium
fe80::/64 dev enp0s8 proto kernel metric 256 pref medium
fe80::/64 dev enp0s3 proto kernel metric 256 pref medium
fe80::/64 dev enp0s10 proto kernel metric 256 pref medium
apnic@R1-ubuntu1804:~$
```

Lab Exercise Part 2(b) - Configure link to attached PCs

As you can see from the topology diagram, a PC is connected to each SRv6 router. There is a one-to-one correspondence between the PCs and routers as follows:

```
PC1 <-> R1
PC2 <-> R2
PC3 <-> R3
PC6 <-> R6
PC7 <-> R7
PC8 <-> R8
```

Each router is connected to its PC via interface `enp0s10`.

NOTE: In the sections that follow, all configuration commands and outputs are provided from the perspective of R1. Be careful to modify your own configuration to suit the router assigned to you.

1. Configure the router's IPv4 address for the `enp0s10` interface using the following command:

   ```
   apnic@R1-ubuntu1804:~$ sudo ip addr add 10.1.1.2/24 dev enp0s10
   ```

2. Verify that the address configuration in the netplan file has been correctly applied to the router by using the following command:

   ```
   apnic@R1-ubuntu1804:~$ ip addr show enp0s10
   ```

   Your router should output something similar to the following, which shows the configuration for router **R1**:

   ```
   apnic@R1-ubuntu1804:~$ ip addr show enp0s10
   5: enp0s10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group defaul
   t qlen 1000
       link/ether 08:00:27:ed:5d:ba brd ff:ff:ff:ff:ff:ff
       inet 10.1.1.2/24 brd 10.1.1.255 scope global enp0s10
          valid_lft forever preferred_lft forever
       inet6 fe80::a00:27ff:feed:5dba/64 scope link
          valid_lft forever preferred_lft forever
   apnic@R1-ubuntu1804:~$
   ```

3. Disable reverse path filtering for IPv4 packets. This setting is normally enabled (and recommended) but will pose problems for our lab since the network does not have IPv4 routing knowledge. Do this by setting kernel parameters using the `sysctl` tool:

   ```
   apnic@R1-ubuntu1804:~$ sudo sysctl -w net.ipv4.conf.all.rp_filter=0
   apnic@R1-ubuntu1804:~$ sudo sysctl -w net.ipv4.conf.enp0s3.rp_filter=0
   apnic@R1-ubuntu1804:~$ sudo sysctl -w net.ipv4.conf.enp0s8.rp_filter=0
   apnic@R1-ubuntu1804:~$ sudo sysctl -w net.ipv4.conf.enp0s9.rp_filter=0
   ```

4. Verify that the sysctls have taken effect by issuing the following command:

```
apnic@R1-ubuntu1804:~$ sysctl -ar "\.rp_filter"
```

Your router should output something similar to the following, which shows the configuration for router R1:

```
apnic@R1-ubuntu1804:~$ sysctl -ar "\.rp_filter"
net.ipv4.conf.all.rp_filter = 0
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.enp0s10.rp_filter = 1
net.ipv4.conf.enp0s3.rp_filter = 0
net.ipv4.conf.enp0s8.rp_filter = 0
net.ipv4.conf.enp0s9.rp_filter = 0
net.ipv4.conf.lo.rp_filter = 0
apnic@R1-ubuntu1804:~$
```

Lab Exercise Part 2(c) - Configure PCs

1. Telnet to your designated PC's console port as shown below:

```
telnet <IP advised by instructor> 5008 [PC1]
telnet <IP advised by instructor> 5010 [PC2]
telnet <IP advised by instructor> 5012 [PC3]
telnet <IP advised by instructor> 5014 [PC6]
telnet <IP advised by instructor> 5016 [PC7]
telnet <IP advised by instructor> 5018 [PC8]
```

2. There are no login credentials for getting access to the PC.

3. Assign the PCs WAN IP address and gateway using the following syntax:

```
PC1> ip <WAN IP> <MASK> <DEFAULT_GATEWAY>
```

An example for **PC1** is given below:

```
PC1>
PC1> ip 10.1.1.1 255.255.255.0 10.1.1.2
Checking for duplicate address...
PC1 : 10.1.1.1 255.255.255.0 gateway 10.1.1.2
```

4. Ping the near-end and far-end IP addresses of the link to ensure that the link is up.

An example for **PC1** is given below:

```
PC1> ping 10.1.1.1
10.1.1.1 icmp_seq=1 ttl=64 time=0.001 ms
10.1.1.1 icmp_seq=2 ttl=64 time=0.001 ms
10.1.1.1 icmp_seq=3 ttl=64 time=0.001 ms
10.1.1.1 icmp_seq=4 ttl=64 time=0.001 ms
10.1.1.1 icmp_seq=5 ttl=64 time=0.001 ms

PC1> ping 10.1.1.2
84 bytes from 10.1.1.2 icmp_seq=1 ttl=64 time=0.720 ms
84 bytes from 10.1.1.2 icmp_seq=2 ttl=64 time=1.046 ms
84 bytes from 10.1.1.2 icmp_seq=3 ttl=64 time=1.031 ms
^C
PC1> .
```

At the end of this step, you have ensured that your router is configured to support SRv6 processing and the attached PC is operational.

NOTE: Before proceeding to the next part, ensure that all members of the group have completed this part.

The second part of this lab is now complete. Wait for your instructors to provide some interpretation of what you have observed so far before proceeding to the next part.

Part 3: IPv4 overlay over SRv6 SPF path

In part 3 of the lab, you will configure an IPv4 overlay service to connect two PCs across the SRv6 infrastructure. Since the core network is not IPv4-enabled, simple routing does not work in this case. The IPv4 traffic between the PCs therefore needs to be tunneled across the network.

In this part, you will work in pairs in order to establish a point-to-point service. The following figures illustrate the pairings and the objectives. Take care to modify the instructions and commmands to suit your particular router and PC.

PC1 and PC8

R1 is configured with the following static route:
- *Route 10.8.8.0/24 via SRv6 encap policy {FC00:8::4}*

→

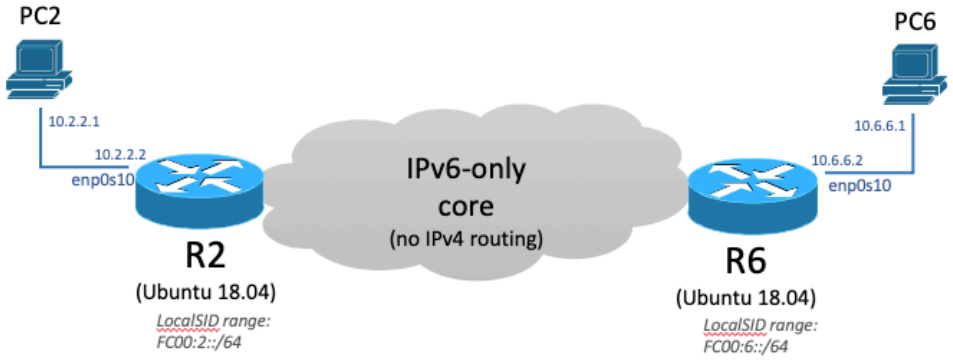
R8's Local SID table has the following entry:
- *FC00:8::4 -> Action: END.DX4 ipv4-next-hop: 10.8.8.1*

R1's Local SID table has the following entry:
- *FC00:1::4 -> Action: END.DX4 ipv4-next-hop: 10.1.1.1*

←

R8 is configured with the following static route:
- *Route 10.1.1.0/24 via SRv6 encap policy {FC00:1::4}*

PC2 and PC6



R2 is configured with the following static route:
- *Route 10.6.6.0/24 via SRv6 encap policy {FC00:6::4}*

→

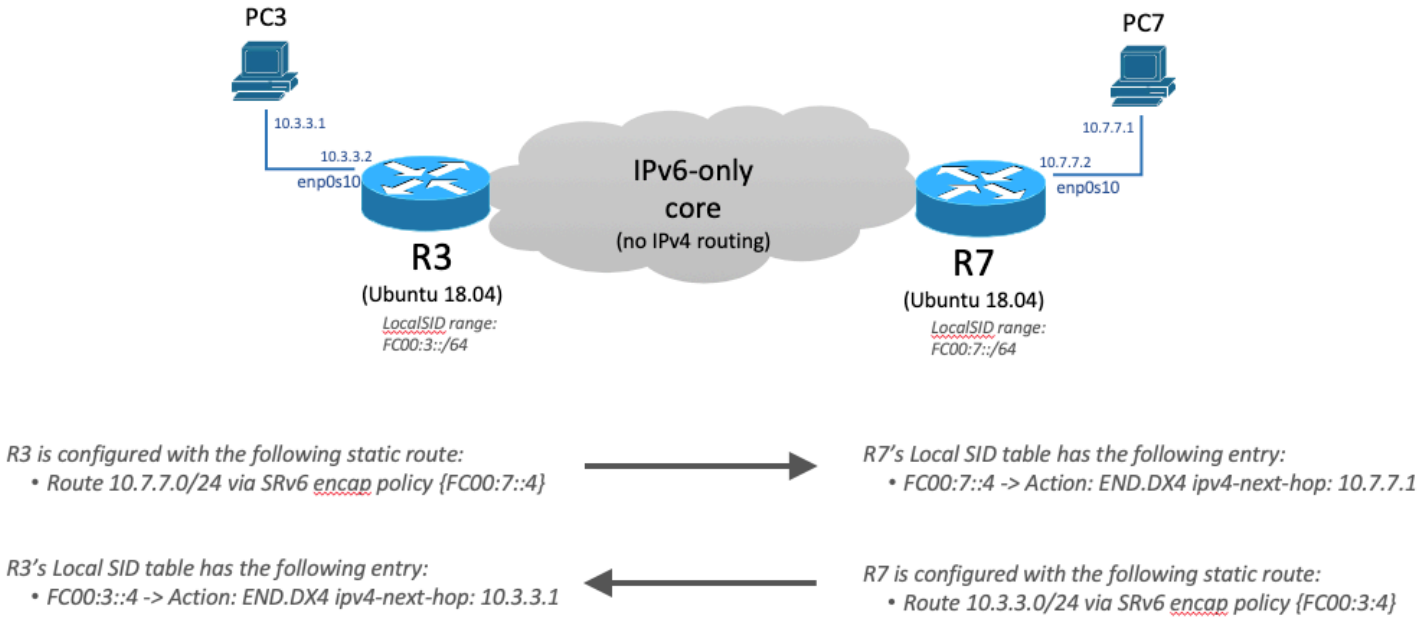
R6's Local SID table has the following entry:
- *FC00:6::4 -> Action: END.DX4 ipv4-next-hop: 10.6.6.1*

R2's Local SID table has the following entry:
- *FC00:2::4 -> Action: END.DX4 ipv4-next-hop: 10.2.2.1*

←

R6 is configured with the following static route:
- *Route 10.2.2.0/24 via SRv6 encap policy {FC00:2::4}*

PC3 and PC7



NOTE: In the sections that follow, all configuration commands and outputs are provided from the perspective of R1. Be careful to modify your own configuration to suit the router assigned to you.

Lab Exercise Part 3(a) - Routing configuration

1. We will commence this part by ensuring that each router can see the IPv6 address block corresponding to the localsid table of the paired router. In particular:

    - R1 should have visibility of R8's localsid block, FC00:8::/64
    - R8 should have visibility of R1's localsid block, FC00:1::/64
    - R2 should have visibility of R6's localsid block, FC00:6::/64
    - R6 should have visibility of R2's localsid block, FC00:2::/64
    - R3 should have visibility of R7's localsid block, FC00:7::/64
    - R7 should have visibility of R3's localsid block, FC00:3::/64

    Do this by issuing the following command on your router:

    ```
    apnic@R1-ubuntu1804:~$ ip -6 route
    ```

    An example for **R1** is given below:

```
apnic@R1-ubuntu1804:~$ ip -6 route
local ::1 dev lo proto kernel metric 256 pref medium
unreachable 2001:db8::1 dev lo proto kernel metric 256 error -101 pref medium
2001:db8::2 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8::3 via fe80::a00:27ff:fefd:4a0b dev enp0s8 proto bird metric 1024 pref medium
2001:db8::4 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8::5 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8::6 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8::7 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8::8 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8:0:1::/127 dev enp0s3 proto kernel metric 256 pref medium
2001:db8:0:2::/127 dev enp0s8 proto kernel metric 256 pref medium
2001:db8:0:3::/127 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8:0:4::/127 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8:0:5::/127 via fe80::a00:27ff:fefd:4a0b dev enp0s8 proto bird metric 1024 pref medium
2001:db8:0:6::/127 via fe80::a00:27ff:fefd:4a0b dev enp0s8 proto bird metric 1024 pref medium
2001:db8:0:7::/127 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8:0:8::/127 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8:0:9::/127 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8:0:a::/127 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8:0:b::/127 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
2001:db8:0:c::/127 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
blackhole fc00:1::/64 dev lo metric 1024 error -22 pref medium
fc00:8::/64 via fe80::a00:27ff:fe83:f34a dev enp0s3 proto bird metric 1024 pref medium
unreachable fe80::1 dev lo proto kernel metric 256 error -101 pref medium
fe80::/64 dev enp0s8 proto kernel metric 256 pref medium
fe80::/64 dev enp0s3 proto kernel metric 256 pref medium
fe80::/64 dev enp0s10 proto kernel metric 256 pref medium
apnic@R1-ubuntu1804:~$
```

As you can see from the above output, R1 has visibility of R8's localsid block, FC00:8::0/64.

2. Add a local SID to your localsid table with the appropriate endpoint action. For example, in the case of R1, this entails adding a route matching FC00:1::4/128 with an End.DX4 action pointing to the IPv4 next-hop 10.1.1.1 (PC1). The syntax for this command, as applicable to R1, is as follows:

```
apnic@R1-ubuntu1804:~$ sudo ip -6 route add fc00:1::4 encap seg6local action E
nd.DX4 nh4 10.1.1.1 dev enp0s10 table localsid
```

The purpose of this route is to apply the End.X action with a next-hop of 10.1.1.1 for packets with a SID (IPv6 DA) value of FC00:1::4.

Verify that this route has been successfully applied using the following command:

```
apnic@R1-ubuntu1804:~$ ip -6 route show table localsid
```

An example for **R1** is given below:

```
apnic@R1-ubuntu1804:~$ ip -6 route show table localsid
fc00:1::4  encap seg6local action End.DX4 nh4 10.1.1.1 dev enp0s10 metric 1024 pref medium
blackhole default dev lo metric 1024 error -22 pref medium
apnic@R1-ubuntu1804:~$
```

3. Add a static route for your paired PC. In the case of PC1, this will be PC8 and vice versa. For PC2, it will be PC6 and vice versa. For PC3, it will be PC7 and vice versa. The purpose of the route is to take

incoming traffic from the connected PC and encapsulate it in an IPv6 packet with a Segment Routing Header (SRH). At the far-end router, it will be decapsulated and forwarded on to its connected PC. The syntax for this command, as applicable to R1, is as follows:

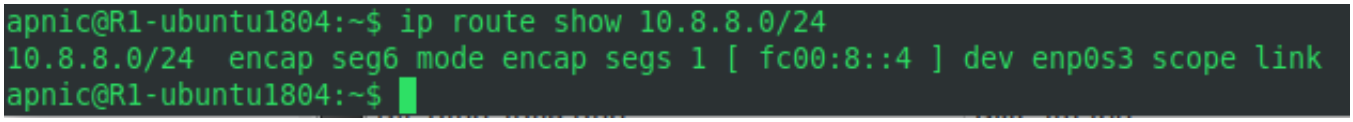
```
apnic@R1-ubuntu1804:~$ sudo ip route add 10.8.8.0/24 encap seg6 mode encap seg
s FC00:8::4 dev enp0s3
```

The purpose of this route is to encapsulate packets destined for 10.8.8.0/24 in an IPv6 packet with an SRH with a single segment FC00:8::4.

Verify that this route has been successfully applied using the following command:

```
apnic@R1-ubuntu1804:~$ ip route show 10.8.8.0/24
```

An example for **R1** is given below:



NOTE: Before proceeding to the next part, ensure that your partner has completed this part.

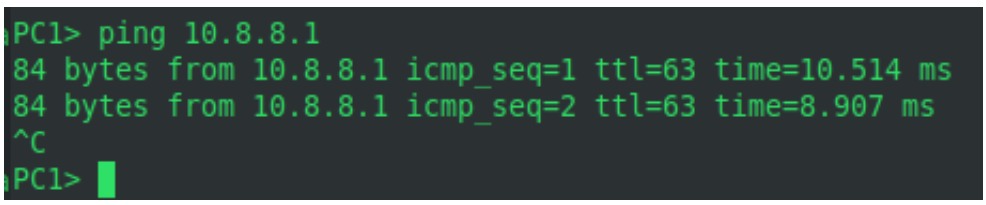
Lab Exercise Part 3(b) - Verification

1. Telnet to your designated PC's console port as shown below:

```
telnet <IP advised by instructor> 5008 [PC1]
telnet <IP advised by instructor> 5010 [PC2]
telnet <IP advised by instructor> 5012 [PC3]
telnet <IP advised by instructor> 5014 [PC6]
telnet <IP advised by instructor> 5016 [PC7]
telnet <IP advised by instructor> 5018 [PC8]
```

2. Ping your paired PC.

   An example for **PC1** is given below:



3. On your Ubuntu router, capture packets and analyse them using tshark.

   Enable capturing of ICMP packets via tshark using the following command (replace `enp0s3` with the

interface on which you want to capture packets):

```
apnic@R1-ubuntu1804:~$ sudo tshark -i enp0s3 -Y "icmp" -V > srv6_capture
```

The above command will pipe the capture output to a file named `srv6_capture` which you can then examine later.

4. Ping your paired PC from your PC in order to generate ICMP traffic.
5. On your Ubuntu router, stop the packet capture by hitting Ctrl-C. Then, examine and analyse the capture file using the following command:

```
apnic@R1-ubuntu1804:~$ more srv6_capture
```

Your instructor will provide a detailed analysis of some sample packet captures.

At the end of this step, you have ensured that your router is configured to support SRv6 processing and the attached PC is operational.

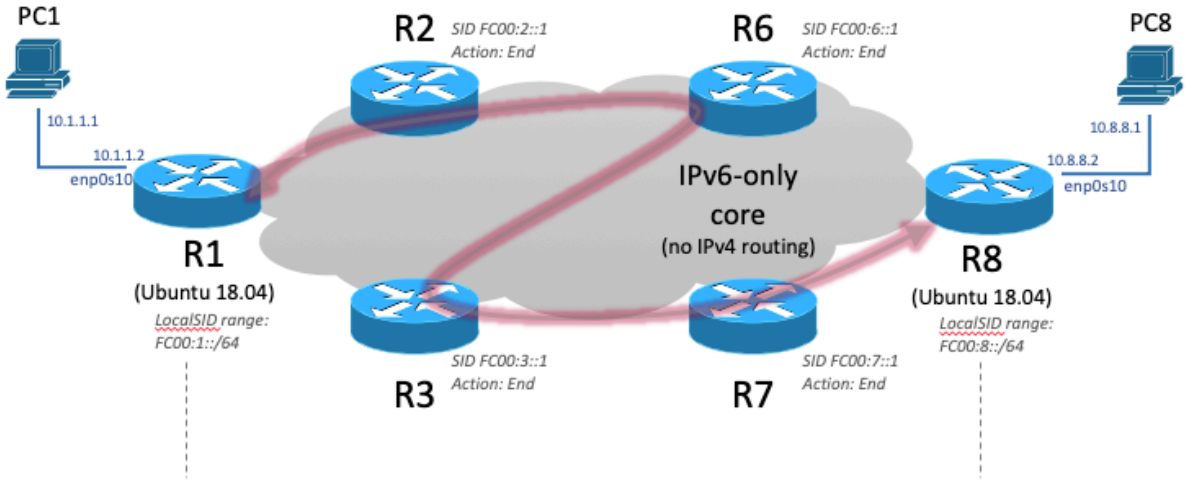
The third part of this lab is now complete. Wait for your instructors to provide some interpretation of what you have observed so far before proceeding to the next part.

Part 4: IPv4 overlay over SRv6 SPF-TE path

In part 4 of the lab, you will configure an IPv4 overlay service to connect two PCs across the SRv6 infrastructure. Since the core network is not IPv4-enabled, simple routing does not work in this case. The IPv4 traffic between the PCs therefore needs to be tunneled across the network. Unlike the previous part, you will define a traffic-engineered path within the SRv6 underlay network.

In this part, you will again work in pairs in order to establish a point-to-point service. The following figures illustrate the pairings and the objectives. Take care to modify the instructions and commmands to suit your particular router and PC.

PC1 and PC8



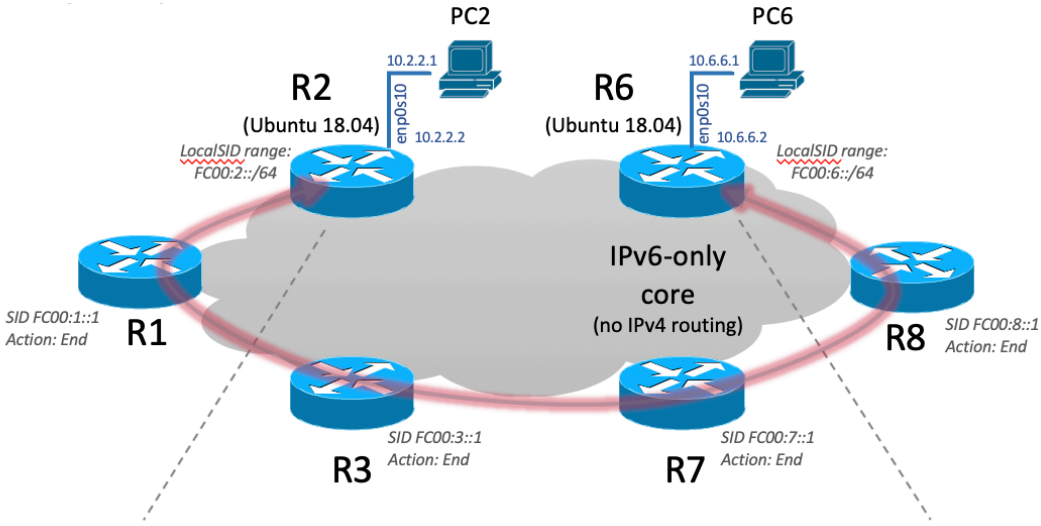
R1 is configured with the following static route:
- *Route 10.8.8.0/24 via SRv6 encap policy {FC00:2::1, FC00:6::1, FC00:3::1,FC00:7::1,FC00:8::4}*

R8's Local SID table has the following entry:
- *FC00:8::4 -> Action: END.DX4 ipv4-next-hop: 10.8.8.1*

R1's Local SID table has the following entry:
- *FC00:1::4 -> Action: END.DX4 ipv4-next-hop: 10.1.1.1*

R8 is configured with the following static route:
- *Route 10.1.1.0/24 via SRv6 encap policy {FC00:7::1, FC00:3::1, FC00:6::1,FC00:2::1, FC00:1::4}*

PC2 and PC6



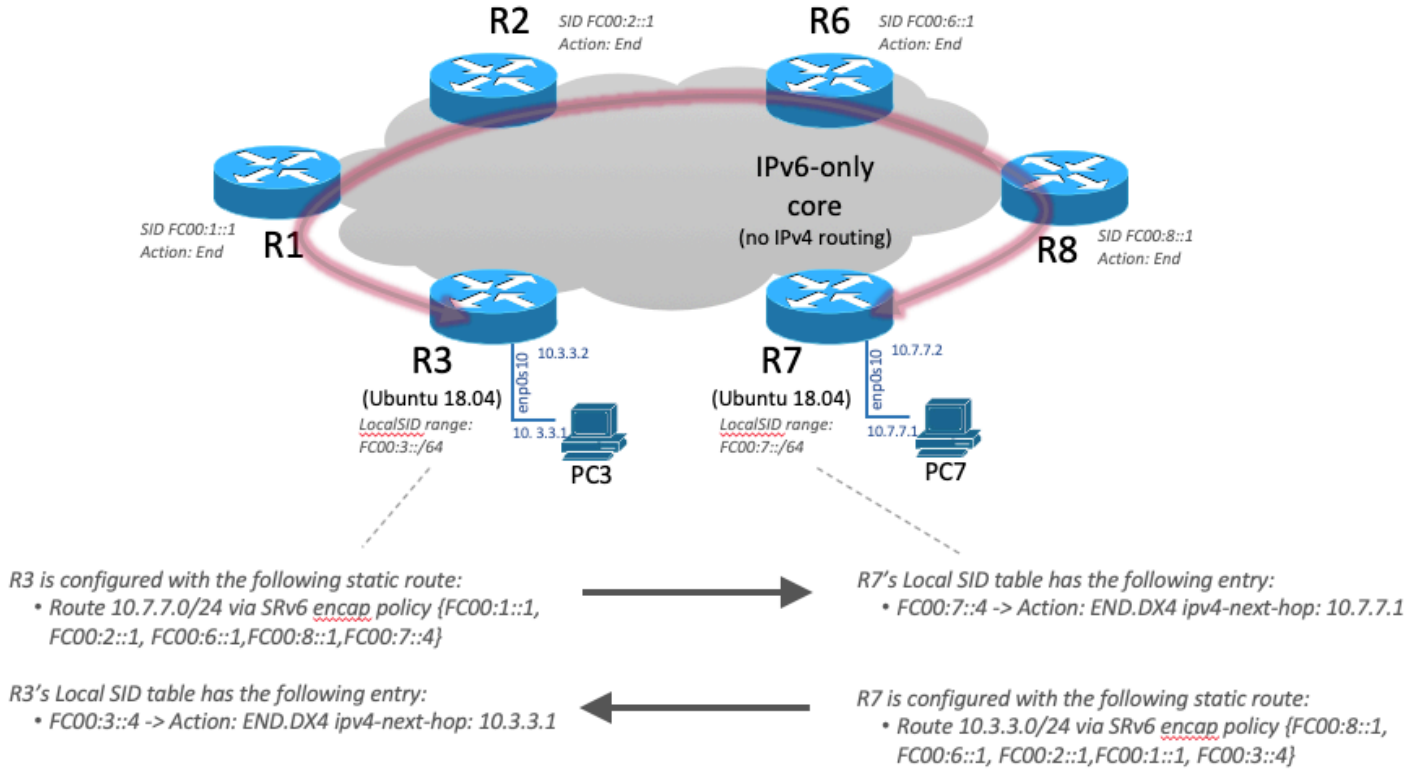
R2 is configured with the following static route:
- *Route 10.6.6.0/24 via SRv6 encap policy {FC00:1::1, FC00:3::1, FC00:7::1,FC00:8::1,FC00:6::4}*

R6's Local SID table has the following entry:
- *FC00:6::4 -> Action: END.DX4 ipv4-next-hop: 10.6.6.1*

R2's Local SID table has the following entry:
- *FC00:2::4 -> Action: END.DX4 ipv4-next-hop: 10.2.2.1*

R6 is configured with the following static route:
- *Route 10.2.2.0/24 via SRv6 encap policy {FC00:8::1, FC00:7::1, FC00:3::1,FC00:1::1, FC00:2::4}*

PC3 and PC7



NOTE: In the sections that follow, all configuration commands and outputs are provided from the perspective of R1. Be careful to modify your own configuration to suit the router assigned to you.

Lab Exercise Part 4(a) - Routing configuration

1. Add a local SID to your localsid table with an End action. For example, in the case of R1, this entails adding a route matching FC00:1::1/128 with an End. The syntax for this command, as applicable to R1, is as follows:

   ```
   apnic@R1-ubuntu1804:~$ sudo ip -6 route add fc00:1::1 encap seg6local action E
   nd dev enp0s10 table localsid
   ```

   The purpose of this route is to apply the End action for packets with a SID (IPv6 DA) value of FC00:1::1. This action is only applicable for non-egress SRv6 nodes and results in advancing to the next SID in the SID list.

NOTE: Before proceeding to the next step, ensure that all members of the group have completed this first step.

1. Delete the static route for your paired PC that you had added in the previous. In the case of PC1, this will be PC8 and vice versa. For PC2, it will be PC6 and vice versa. For PC3, it will be PC7 and vice versa. The syntax for this command, as applicable to R1, is as follows:

   ```
   apnic@R1-ubuntu1804:~$ sudo ip route del 10.8.8.0/24
   ```

2. Add a static route for your paired PC. In the case of PC1, this will be PC8 and vice versa. For PC2, it will be PC6 and vice versa. For PC3, it will be PC7 and vice versa. The purpose of the route is to take incoming traffic from the connected PC and encapsulate it in an IPv6 packet with a Segment Routing Header (SRH). At the far-end router, it will be decapsulated and forwarded on to its connected PC. The syntax for this command, as applicable to R1, is as follows:

```
apnic@R1-ubuntu1804:~$ sudo ip route add 10.8.8.0/24 encap seg6 mode encap seg
s fc00:2::1,fc00:6::1,fc00:3::1,fc00:7::1,fc00:8::4 dev enp0s3
```

   The purpose of this route is to encapsulate packets destined for 10.8.8.0/24 in an IPv6 packet with an SRH with five segments {fc00:2::1,fc00:6::1,fc00:3::1,fc00:7::1,fc00:8::4}. The SRH is an instruction to route the packet via R2, R6, R3 and R7 before delivering it to R8.

   Verify that this route has been successfully applied using the following command:

```
apnic@R1-ubuntu1804:~$ ip route show 10.8.8.0/24
```

   An example for **R1** is given below:

```
apnic@R1-ubuntu1804:~$ ip route show 10.8.8.0/24
10.8.8.0/24  encap seg6 mode encap segs 5 [ fc00:2::1 fc00:6::1 fc00:3::1 fc00:7::1 fc00:8::4 ] dev enp0s3 scope link
apnic@R1-ubuntu1804:~$
```

NOTE: Before proceeding to the next part, ensure that your partner has completed this part.

Lab Exercise Part 4(b) - Verification

1. Telnet to your designated PC's console port as shown below:

```
telnet <IP advised by instructor> 5008 [PC1]
telnet <IP advised by instructor> 5010 [PC2]
telnet <IP advised by instructor> 5012 [PC3]
telnet <IP advised by instructor> 5014 [PC6]
telnet <IP advised by instructor> 5016 [PC7]
telnet <IP advised by instructor> 5018 [PC8]
```

2. Ping your paired PC.

   An example for **PC1** is given below:

```
PC1> ping 10.8.8.1
84 bytes from 10.8.8.1 icmp_seq=1 ttl=63 time=10.514 ms
84 bytes from 10.8.8.1 icmp_seq=2 ttl=63 time=8.907 ms
^C
PC1>
```

3. On your Ubuntu router, capture packets and analyse them using tshark.

Enable capturing of ICMP packets via tshark using the following command (replace `enp0s3` with the interface on which you want to capture packets):

```
apnic@R1-ubuntu1804:~$ sudo tshark -i enp0s3 -Y "icmp" -V > srv6_capture
```

The above command will pipe the capture output to a file named `srv6_capture` which you can then examine later.

4. Ping your paired PC from your PC in order to generate ICMP traffic.

5. On your Ubuntu router, stop the packet capture by hitting Ctrl-C. Then, examine and analyse the capture file using the following command:

```
apnic@R1-ubuntu1804:~$ more srv6_capture
```

Your instructor will provide a detailed analysis of some sample packet captures.

At the end of this step, you have ensured that your router is configured to support SRv6 processing and the attached PC is operational.

The final part of this lab is now complete.

End of lab
