Traffic Engineering

3rd Lab Project Report

Introduction to Programmable Networks

Virtual networking using Linux namespaces

Group 8
Francisco Bento - 93581
Pedro Tracana - 93610

May 28, 2024

1 Introduction

The objective of this report is to evaluate the essential tools required for the development of programmable networks. Specifically, it focuses on using Linux namespaces to emulate network topologies and to test virtualization functionalities and modules, including namespaces, virtual interfaces and bridges.

This report offers a comprehensive configuration guide for three lab exercises, accompanied by graphical representations of the network topologies. Additionally, it will demonstrate the correct behavior and functionalities of the implemented network setups.

2 Network Topology and Configuration

2.1 Two-node network

In this initial exercise, the objective is to create a network comprising two nodes: our host Linux VM using the *default* namespace and an additional namespace named *examplns*, as illustrated in Figure 1. By the end of this exercise, we should be able to verify the connectivity between the *default* namespace and the *examplens* namespace.

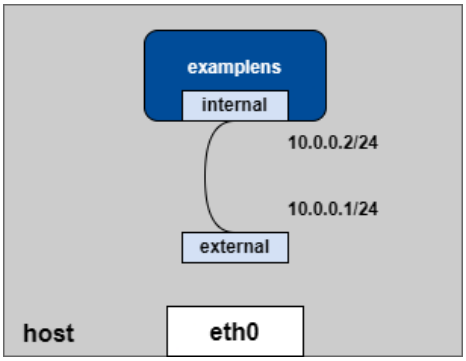

Figure 1: Two-node network topology.

The configurations related to this exercise are available in Annex A.

2.2 Three-node network

In the second exercise, the task is to chain three nodes: the host and two additional namespaces named *near* and *far*. The objective is to verify connectivity between the *default* namespace and the *far* namespace, and to ensure that the host can reach the loopback interface of the *far* namespace.

With multiple namespaces involved, it is necessary to add IP routing rules to each namespace to enable bidirectional connectivity and to enable IP forwarding for the namespaces that require routing.

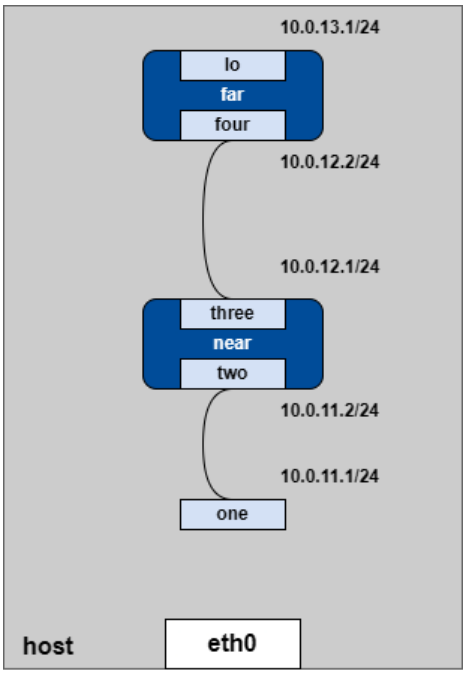

Figure 2: Three-node network topology.

The configurations related to this exercise are available in Annex B.

2.3 Full network emulation

For the third and final exercise, it is required to emulate the network topology presented in Figure 3.

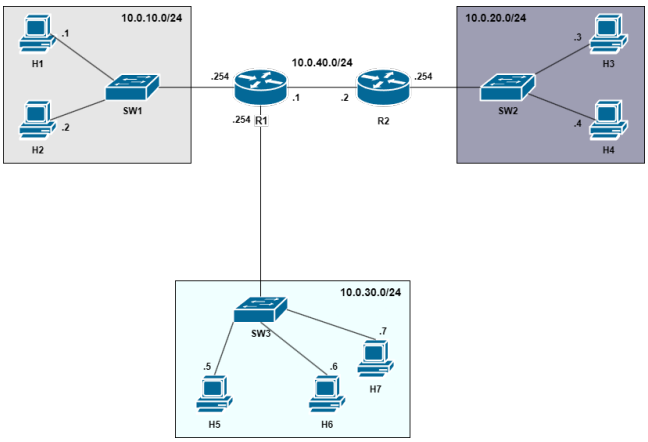

Figure 3: Full network topology.

In this scenario, a different namespace is assigned to each host, router and switch. Additionally, each switch device is configured with a bridge interface, with all other interfaces connected to it, thus achieving true switch behavior.

Moreover, in addition to adding the necessary routes for each router and enabling IP forwarding, it is also required to add a default route to each host through its respective gateway. This ensures that hosts can communicate with other networks that are implicitly defined.

The configurations related to this exercise are available in Annex C.

3 Experimental Results

3.1 Two-node network

To validate this first exercise, the environment must be inspected to ensure the correct configurations were applied. By running *sudo ip netns list*, it can be confirmed that the *examplens* namespace was created, as shown in Figure 4.

```
francisco@francisco-VirtualBox:~/ET$ sudo ip netns list
examplens (id: 0)
```

Figure 4: All network namespaces within the Host.

By executing the commands *sudo ip addr list* and *sudo ip netns exec examplens ip addr list*, the status and IP addresses of the virtual interfaces in the *default* (device 8) and *examplens* namespaces can be verified. This confirms the topology mentioned above, as shown in Figures 5 and 6.

```
francisco@francisco-VirtualBox:~/ET$ sudo ip addr list
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
    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:d4:26:f8 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.104/24 brd 192.168.56.255 scope global dynamic noprefixroute enp0s3
       valid_lft 403sec preferred_lft 403sec
    inet6 fe80::2cf4:ca05:6fc8:3dbd/64 scope link noprefixroute
       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 08:00:27:30:2a:44 brd ff:ff:ff:ff:ff:ff
    inet 10.0.3.15/24 brd 10.0.3.255 scope global dynamic noprefixroute enp0s8
       valid_lft 68351sec preferred_lft 68351sec
    inet6 fe80::b549:9e2e:827:6c56/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
4: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether 52:54:00:53:61:40 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
       valid_lft forever preferred_lft forever
5: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc fq_codel master virbr0 state DOWN group default qlen 1000
    link/ether 52:54:00:53:61:40 brd ff:ff:ff:ff:ff:ff
6: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:43:cc:7c:29 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
8: external@if7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether da:be:55:be:20:3c brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.0.1/24 scope global external
       valid_lft forever preferred_lft forever
    inet6 fe80::d8be:55ff:febe:203c/64 scope link
       valid_lft forever preferred_lft forever
```

Figure 5: Default namespace network devices.

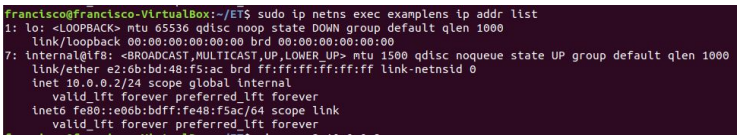

Figure 6: Examplens namespace network devices.

Finally, to test connectivity between the namespaces, the command *ping -c 3 10.0.0.1* was used to ping from the external interface (default namespace) to the internal interface (examplens), both configured under the same subnet. Figure 7 demonstrates that the namespaces are properly connected.

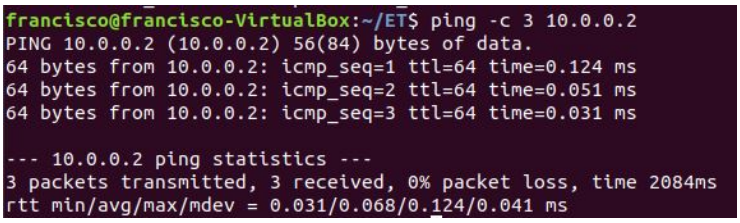

Figure 7: Ping from external to internal interface.

3.2 Three-node network

This exercise follows the same scheme as the previous one but requires chaining an additional namespace and providing routing rules to establish connectivity from one end of the chain to the other. By inspecting the list of namespaces, it can be observed that the host now detects two namespaces: *near* and *far*, as shown in Figure 8.

Figure 8: All network namespaces within the Host.

Now, by inspecting each namespace using the command *sudo ip netns exec near/far ip addr list*, the lists of network devices and their IP configurations can be retrieved to ensure they are correctly set up according to the topology. Figure 9 shows the results of this command for both the *near* and *far* namespaces.

Additionally, it is important to check the routing tables for each namespace, including the default namespace. If everything is configured correctly, the default and near namespaces should have routes to all three networks, while the far namespace should have two entries for the subnets 10.0.11.0/24 and 10.0.12.0/24, as the other subnet is associated with its loopback interface. Figure 10 shows the routing tables for all three namespaces.

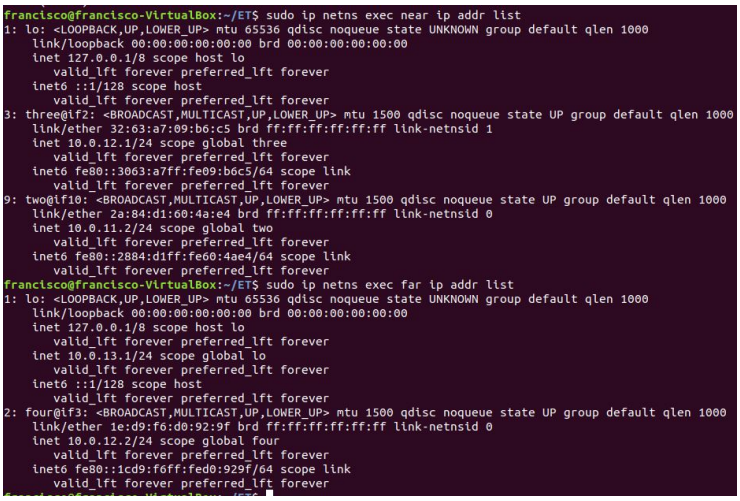

Figure 9: Network devices of Near and Far namespaces.

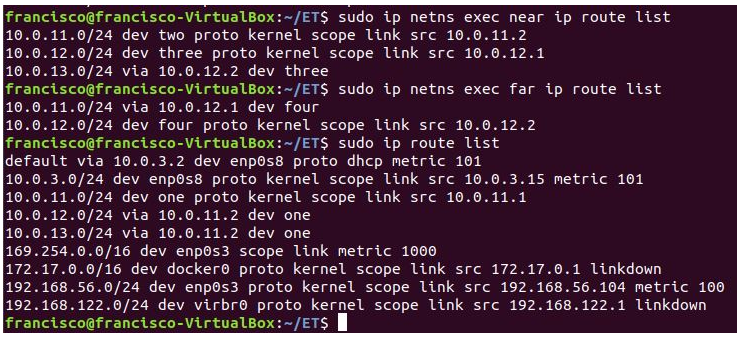

Figure 10: Routing tables of the default, near and far namespaces.

Finally, to test connectivity between the default namespace and the loopback interface of the far namespace, the command *ping -c 3 10.0.13.1* was executed from the default namespace to the loopback interface, utilizing routing rules. Figure 11 demonstrates that connectivity was successfully established.

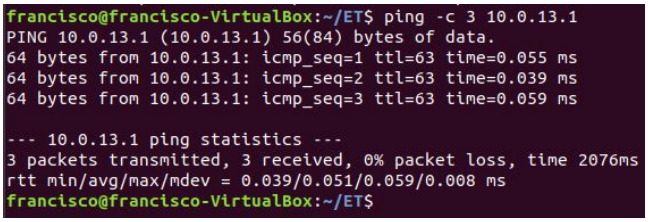

Figure 11: Pinging from interface one to far's loopback interface.

3.3 Full network emulation

In the third and final exercise, the topology to be emulated is more complex, featuring three distinct LANs with switches inside each, all connected by two routers. The first step in verifying the configuration, detailed in Annex C, is to confirm that all namespaces were created according to the laboratory guide. Each network node (whether host, switch, or router) should be emulated with its unique namespace, as illustrated in Figure 12.

```
francisco@francisco-VirtualBox:~/ET$ sudo ip netns list
R2 (id: 11)
R1 (id: 10)
SW3 (id: 9)
SW2 (id: 8)
SW1 (id: 7)
H7 (id: 6)
H6 (id: 5)
H5 (id: 4)
H4 (id: 3)
H3 (id: 2)
H2 (id: 1)
H1 (id: 0)
```

Figure 12: All network namespaces within the Host.

Following that, we can proceed by examining the configurations of the hosts, ensuring that both IP addresses and routing tables align with the IP addresses and subnets selected for the topology outlined in Figure 3.

In particular, each host should possess only one interface in addition to the loopback, and the assigned IP address should correspond to the one designated in the topology. The outcomes of executing the command *sudo ip netns exec X ip addr*, where X represents some of the namespaces associated with the hosts, are depicted in Figure 13.

```
francisco@francisco-VirtualBox:~/ET$ sudo ip netns exec H1 ip addr
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
8: veth_H1@if7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether fa:d5:7e:9c:3d:a8 brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet 10.0.10.1/24 scope global veth_H1
       valid_lft forever preferred_lft forever
    inet6 fe80::f8d5:7eff:fe9c:3da8/64 scope link
       valid_lft forever preferred_lft forever
francisco@francisco-VirtualBox:~/ET$ sudo ip netns exec H3 ip addr
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
12: veth_H3@if11: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 9e:26:f4:71:1f:57 brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet 10.0.20.3/24 scope global veth_H3
       valid_lft forever preferred_lft forever
    inet6 fe80::9c26:f4ff:fe71:1f57/64 scope link
       valid_lft forever preferred_lft forever
francisco@francisco-VirtualBox:~/ET$ sudo ip netns exec H6 ip addr
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
18: veth_H6@if17: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 6a:95:7e:9c:30:60 brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet 10.0.30.6/24 scope global veth_H6
       valid_lft forever preferred_lft forever
    inet6 fe80::6895:7eff:fe9c:3060/64 scope link
       valid_lft forever preferred_lft forever
```

Figure 13: Network devices of some of the Hosts.

Additionally, it's valuable to inspect the routing tables of the hosts using *sudo ip netns exec X ip route list*. These routes should be uniformly configured across all hosts, featuring a default route via their corresponding gateway, as depicted in Figure 14.

```
francisco@francisco-VirtualBox:~/ET$ sudo ip netns exec H1 ip route list
default via 10.0.10.254 dev veth_H1
10.0.10.0/24 dev veth_H1 proto kernel scope link src 10.0.10.1
francisco@francisco-VirtualBox:~/ET$ sudo ip netns exec H3 ip route list
default via 10.0.20.254 dev veth_H3
10.0.20.0/24 dev veth_H3 proto kernel scope link src 10.0.20.3
francisco@francisco-VirtualBox:~/ET$ sudo ip netns exec H6 ip route list
default via 10.0.30.254 dev veth_H6
10.0.30.0/24 dev veth_H6 proto kernel scope link src 10.0.30.6
```

Figure 14: Routing tables of some of the Hosts.

Now, focusing on the switches. They are configured with a uniform structure, ensuring each switch operates as a genuine switch. To achieve this, every switch establishes a bridge interface and connects all other interfaces to it. This includes both the VETH interfaces created to pair with the hosts and the interface that links the switch to one of the two routers.

Figure 15 presents an example of the interfaces created for SW1, containing the following:

- br_SW1: bridge device created to provide network bridging.
- veth_H1_SW1: virtual ethernet device created to link the H1 namespace with the SW1 namespace.
- veth_H2_SW1: virtual ethernet device created to link the H2 namespace with the SW1 namespace.
- veth_SW1_R1: virtual ethernet device created to link the SW1 namespace with the router R1 namespace.

```
francisco@francisco-VirtualBox:~/ET$ sudo ip netns exec SW1 ip addr list
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: br_SW1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 02:ff:af:03:ed:52 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::50df:31ff:fe7f:948/64 scope link
       valid_lft forever preferred_lft forever
7: veth_H1_SW1@if8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br_SW1 state UP group default qlen 1000
    link/ether 02:ff:af:03:ed:52 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::ff:afff:fe03:ed52/64 scope link
       valid_lft forever preferred_lft forever
9: veth_H2_SW1@if10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br_SW1 state UP group default qlen 1000
    link/ether 8e:fb:5c:96:a0:40 brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet6 fe80::8cfb:5cff:fe96:a040/64 scope link
       valid_lft forever preferred_lft forever
22: veth_SW1_R1@if21: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br_SW1 state UP group default qlen 1000
    link/ether a6:d7:81:89:99:e4 brd ff:ff:ff:ff:ff:ff link-netnsid 3
    inet6 fe80::a4d7:81ff:fe89:99e4/64 scope link
       valid_lft forever preferred_lft forever
```

Figure 15: Network devices of SW1.

Now, before verifying connectivity, we need to examine the running configurations of R1 and R2. Starting with the interfaces and IP addresses, we expect R1 to have both gateways configured for subnets 10.0.10.0/24 and 10.0.30.0/24, along with the R1_R2 interface facilitating communication with R2. R2, on the

other hand, functions solely as the gateway for the network 10.0.20.0/24, with its R2_R1 interface configured to communicate with R1. Figure 16 provides an overview of these configurations.

```
francisco@francisco-VirtualBox:~/ET$ sudo ip netns exec R1 ip addr list
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
21: veth_R1_SW1@if22: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether a2:4c:14:2f:6a:3f brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.10.254/24 scope global veth_R1_SW1
       valid_lft forever preferred_lft forever
    inet6 fe80::a04c:14ff:fe2f:6a3f/64 scope link
       valid_lft forever preferred_lft forever
25: veth_R1_SW3@if26: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether ba:63:a8:31:fc:7d brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet 10.0.30.254/24 scope global veth_R1_SW3
       valid_lft forever preferred_lft forever
    inet6 fe80::b863:a8ff:fe31:fc7d/64 scope link
       valid_lft forever preferred_lft forever
28: veth_R1_R2@if27: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether de:10:53:b3:a9:cf brd ff:ff:ff:ff:ff:ff link-netnsid 3
    inet 10.0.40.1/24 scope global veth_R1_R2
       valid_lft forever preferred_lft forever
    inet6 fe80::dc10:53ff:feb3:a9cf/64 scope link
       valid_lft forever preferred_lft forever
francisco@francisco-VirtualBox:~/ET$ sudo ip netns exec R2 ip addr list
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
23: veth_R2_SW2@if24: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether c2:ec:8e:5f:1c:0b brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.20.254/24 scope global veth_R2_SW2
       valid_lft forever preferred_lft forever
    inet6 fe80::c0ec:8eff:fe5f:1c0b/64 scope link
       valid_lft forever preferred_lft forever
27: veth_R2_R1@if28: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether b6:2f:e4:7b:72:f3 brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet 10.0.40.2/24 scope global veth_R2_R1
       valid_lft forever preferred_lft forever
    inet6 fe80::b42f:e4ff:fe7b:72f3/64 scope link
       valid_lft forever preferred_lft forever
francisco@francisco-VirtualBox:~/ET$
```

Figure 16: Network devices of R1 and R2.

In addition to examining the interfaces, we can also inspect the routing tables of R1 and R2. While not identical, we anticipate that both routers indicate they can reach the same four networks (.10/.20/.30/.40), as illustrated in Figure 17. This ensures that routes are established for bidirectional traffic, and we expect connectivity to be operational.

```
francisco@francisco-VirtualBox:~/ET$ sudo ip netns exec R1 ip route list
10.0.10.0/24 dev veth_R1_SW1 proto kernel scope link src 10.0.10.254
10.0.20.0/24 via 10.0.40.2 dev veth_R1_R2
10.0.30.0/24 dev veth_R1_SW3 proto kernel scope link src 10.0.30.254
10.0.40.0/24 dev veth_R1_R2 proto kernel scope link src 10.0.40.1
francisco@francisco-VirtualBox:~/ET$ sudo ip netns exec R2 ip route list
10.0.10.0/24 via 10.0.40.1 dev veth_R2_R1
10.0.20.0/24 dev veth_R2_SW2 proto kernel scope link src 10.0.20.254
10.0.30.0/24 via 10.0.40.1 dev veth_R2_R1
10.0.40.0/24 dev veth_R2_R1 proto kernel scope link src 10.0.40.2
```

Figure 17: Routing tables of R1 and R2.

The ultimate verification step involves confirming connectivity across various paths. For instance, if we aim to ping from H1 to H4 and H7, which represent inter-network communication, and from H1 to H2, demonstrating intra-network communication. Specifically, H4 belongs to the 10.0.20.0/24 network, while H7 belongs to the 10.0.30.0/24 network, and H2 shares the same network as H1.

The successful ping results for these hosts are shown in Figures 18 and 19, affirming operational connectivity.

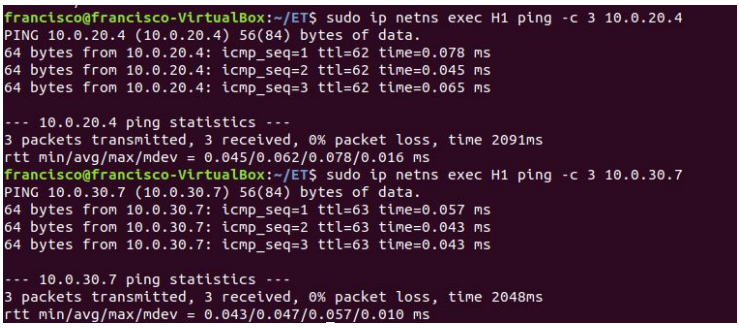

Figure 18: Pinging from H1 to H4 and H7.

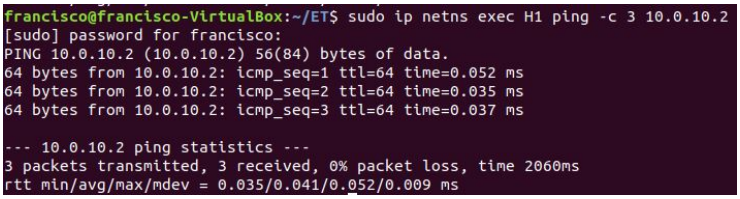

Figure 19: Pinging from H1 to H2.

Appendices

A 4.1 Two-node network

```
#!/bin/sh
sudo ip -all netns delete
#4.1
sudo ip netns add examplens

sudo ip link add external type veth peer name internal
sudo ip link set internal netns examplens

sudo ip netns exec examplens ip addr add 10.0.0.2/24 dev internal
sudo ip netns exec examplens ip link set dev internal up

sudo ip addr add 10.0.0.1/24 dev external
sudo ip link set dev external up
```

B 4.2 Three-node network

```
#!/bin/sh
sudo ip -all netns delete
#4.2
sudo ip netns add near
sudo ip netns add far

sudo ip link add one type veth peer name two
sudo ip link set two netns near

sudo ip netns exec near ip link add three type veth peer name four
sudo ip netns exec near ip link set four netns far

sudo ip addr add 10.0.11.1/24 dev one
sudo ip link set dev one up

sudo ip netns exec near ip addr add 10.0.11.2/24 dev two
sudo ip netns exec near ip link set dev two up

sudo ip netns exec near ip addr add 10.0.12.1/24 dev three
sudo ip netns exec near ip link set dev three up

sudo ip netns exec far ip addr add 10.0.12.2/24 dev four
sudo ip netns exec far ip link set dev four up

sudo ip netns exec far ip addr add 10.0.13.1/24 dev lo
sudo ip netns exec far ip link set dev lo up
sudo ip netns exec near ip link set lo up


sudo ip netns exec near sysctl -w net.ipv4.ip_forward=1
sudo ip netns exec far sysctl -w net.ipv4.ip_forward=1
```

```
sudo ip route add 10.0.12.0/24 via 10.0.11.2 dev one
sudo ip route add 10.0.13.0/24 via 10.0.11.2 dev one

sudo ip netns exec near ip route add 10.0.13.0/24 via 10.0.12.2 dev
     three

sudo ip netns exec far ip route add 10.0.11.0/24 via 10.0.12.1 dev
    four
```

C 4.3 Full network emulation

```
#!/bin/sh
sudo ip -all netns delete
#4.3
# Create namespaces

sudo ip netns add H1
sudo ip netns add H2
sudo ip netns add H3
sudo ip netns add H4
sudo ip netns add H5
sudo ip netns add H6
sudo ip netns add H7

sudo ip netns add SW1
sudo ip netns add SW2
sudo ip netns add SW3

sudo ip netns add R1
sudo ip netns add R2

# Create VETH pairs and assign them to the respective namespaces

sudo ip link add veth_H1 type veth peer name veth_H1_SW1
sudo ip link add veth_H2 type veth peer name veth_H2_SW1
sudo ip link add veth_H3 type veth peer name veth_H3_SW2
sudo ip link add veth_H4 type veth peer name veth_H4_SW2
sudo ip link add veth_H5 type veth peer name veth_H5_SW3
sudo ip link add veth_H6 type veth peer name veth_H6_SW3
sudo ip link add veth_H7 type veth peer name veth_H7_SW3

sudo ip link add veth_SW1_R1 type veth peer name veth_R1_SW1
sudo ip link add veth_SW2_R2 type veth peer name veth_R2_SW2
sudo ip link add veth_SW3_R1 type veth peer name veth_R1_SW3

sudo ip link add veth_R1_R2 type veth peer name veth_R2_R1

sudo ip link set veth_H1 netns H1
sudo ip link set veth_H2 netns H2
sudo ip link set veth_H3 netns H3
sudo ip link set veth_H4 netns H4
sudo ip link set veth_H5 netns H5
sudo ip link set veth_H6 netns H6
sudo ip link set veth_H7 netns H7
```

```
sudo ip link set veth_H1_SW1 netns SW1
sudo ip link set veth_H2_SW1 netns SW1
sudo ip link set veth_SW1_R1 netns SW1

sudo ip link set veth_H3_SW2 netns SW2
sudo ip link set veth_H4_SW2 netns SW2
sudo ip link set veth_SW2_R2 netns SW2

sudo ip link set veth_H5_SW3 netns SW3
sudo ip link set veth_H6_SW3 netns SW3
sudo ip link set veth_H7_SW3 netns SW3
sudo ip link set veth_SW3_R1 netns SW3

sudo ip link set veth_R1_SW1 netns R1
sudo ip link set veth_R2_SW2 netns R2
sudo ip link set veth_R1_SW3 netns R1
sudo ip link set veth_R1_R2 netns R1
sudo ip link set veth_R2_R1 netns R2

# Create and configure bridges for the switch devices

sudo ip netns exec SW1 ip link add name br_SW1 type bridge
sudo ip netns exec SW1 ip link set br_SW1 up
sudo ip netns exec SW1 ip link set dev veth_H1_SW1 master br_SW1
sudo ip netns exec SW1 ip link set dev veth_H2_SW1 master br_SW1
sudo ip netns exec SW1 ip link set dev veth_SW1_R1 master br_SW1
sudo ip netns exec SW1 ip link set dev veth_H1_SW1 up
sudo ip netns exec SW1 ip link set dev veth_H2_SW1 up
sudo ip netns exec SW1 ip link set dev veth_SW1_R1 up

sudo ip netns exec SW2 ip link add name br_SW2 type bridge
sudo ip netns exec SW2 ip link set br_SW2 up
sudo ip netns exec SW2 ip link set dev veth_H3_SW2 master br_SW2
sudo ip netns exec SW2 ip link set dev veth_H4_SW2 master br_SW2
sudo ip netns exec SW2 ip link set dev veth_SW2_R2 master br_SW2
sudo ip netns exec SW2 ip link set dev veth_H3_SW2 up
sudo ip netns exec SW2 ip link set dev veth_H4_SW2 up
sudo ip netns exec SW2 ip link set dev veth_SW2_R2 up

sudo ip netns exec SW3 ip link add name br_SW3 type bridge
sudo ip netns exec SW3 ip link set br_SW3 up
sudo ip netns exec SW3 ip link set dev veth_H5_SW3 master br_SW3
sudo ip netns exec SW3 ip link set dev veth_H6_SW3 master br_SW3
sudo ip netns exec SW3 ip link set dev veth_H7_SW3 master br_SW3
sudo ip netns exec SW3 ip link set dev veth_SW3_R1 master br_SW3
sudo ip netns exec SW3 ip link set dev veth_H5_SW3 up
sudo ip netns exec SW3 ip link set dev veth_H6_SW3 up
sudo ip netns exec SW3 ip link set dev veth_H7_SW3 up
sudo ip netns exec SW3 ip link set dev veth_SW3_R1 up

# Bring up and Assign IP addresses to Hosts and Routers

sudo ip netns exec H1 ip addr add 10.0.10.1/24 dev veth_H1
sudo ip netns exec H1 ip link set dev veth_H1 up
sudo ip netns exec H2 ip addr add 10.0.10.2/24 dev veth_H2
sudo ip netns exec H2 ip link set dev veth_H2 up
```

```
sudo ip netns exec H3 ip addr add 10.0.20.3/24 dev veth_H3
sudo ip netns exec H3 ip link set dev veth_H3 up
sudo ip netns exec H4 ip addr add 10.0.20.4/24 dev veth_H4
sudo ip netns exec H4 ip link set dev veth_H4 up
sudo ip netns exec H5 ip addr add 10.0.30.5/24 dev veth_H5
sudo ip netns exec H5 ip link set dev veth_H5 up
sudo ip netns exec H6 ip addr add 10.0.30.6/24 dev veth_H6
sudo ip netns exec H6 ip link set dev veth_H6 up
sudo ip netns exec H7 ip addr add 10.0.30.7/24 dev veth_H7
sudo ip netns exec H7 ip link set dev veth_H7 up

sudo ip netns exec R1 ip addr add 10.0.10.254/24 dev veth_R1_SW1
sudo ip netns exec R1 ip addr add 10.0.30.254/24 dev veth_R1_SW3
sudo ip netns exec R1 ip addr add 10.0.40.1/24 dev veth_R1_R2
sudo ip netns exec R1 ip link set dev veth_R1_SW1 up
sudo ip netns exec R1 ip link set dev veth_R1_SW3 up
sudo ip netns exec R1 ip link set dev veth_R1_R2 up

sudo ip netns exec R2 ip addr add 10.0.20.254/24 dev veth_R2_SW2
sudo ip netns exec R2 ip addr add 10.0.40.2/24 dev veth_R2_R1
sudo ip netns exec R2 ip link set dev veth_R2_SW2 up
sudo ip netns exec R2 ip link set dev veth_R2_R1 up

#Enable IP forwarding on Routers and Configure Routing tables

sudo ip netns exec R1 sysctl -w net.ipv4.ip_forward=1
sudo ip netns exec R2 sysctl -w net.ipv4.ip_forward=1

sudo ip netns exec H1 ip route add default via 10.0.10.254
sudo ip netns exec H2 ip route add default via 10.0.10.254
sudo ip netns exec H3 ip route add default via 10.0.20.254
sudo ip netns exec H4 ip route add default via 10.0.20.254
sudo ip netns exec H5 ip route add default via 10.0.30.254
sudo ip netns exec H6 ip route add default via 10.0.30.254
sudo ip netns exec H7 ip route add default via 10.0.30.254

sudo ip netns exec R1 ip route add 10.0.20.0/24 via 10.0.40.2
sudo ip netns exec R2 ip route add 10.0.10.0/24 via 10.0.40.1
sudo ip netns exec R2 ip route add 10.0.30.0/24 via 10.0.40.1
```