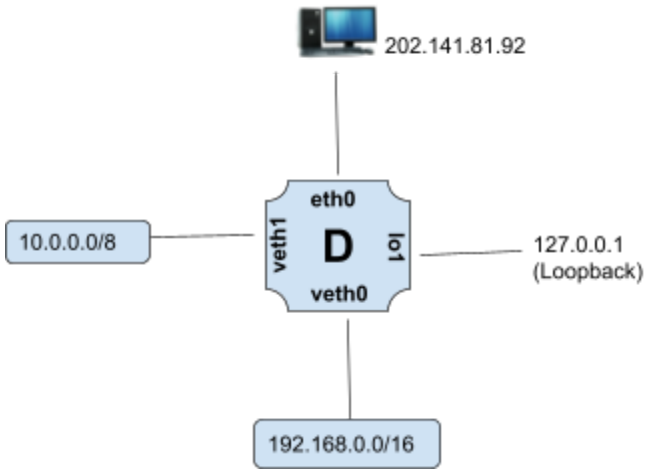
CS 39006: Networks Lab

Assignment 4: Network Namespace and Routes

Submission Deadline: February 5, 2020, 2:00 PM

We learnt about the concept of network namespaces in the last assignment, and also how to use Ethernet bridge. The aim of this assignment is to construct more complex network topologies using network namespaces, virtual ethernet devices (veth), and ip routes.

The individual L3 devices in a network (both the end-hosts as well as routers) maintain a table called a routing table to dictate the routing behavior in the Internet. For example, when a L3 device receives a packet with a destination IP, say 192.168.10.12, and this destination IP is different from the IP of the device itself (so, the device is not the final destination of the packet), then the device consults its routing table to check through which interface it will forward the packet next. Remember that multiple network interfaces (physical as well as virtual) can be connected with a device, through which multiple different networks can be reached. For example, see the following figure.



In the above example, the device D has four interfaces -- one physical interface (eth0), one loopback interface (lo1), and two virtual interfaces (veth0 and veth1). Four different networks/hosts are connected to these four interfaces. Depending on the destination IP of a received packet (the received packet can be from any of these four interfaces), the device needs to decide to which interface it should forward the packet.

The network layer of a router typically runs a routing protocol to construct the routing table. However, we'll learn about the routing protocols later. For the time being, we'll

work with a static routing table where the routing entries will be entered statically using the `ip route` command.

In a routing table, the most important entries are the destination network IP and the interface through which the destination network can be reached. Consider the following diagram.

Ideally, a router has multiple physical interfaces as it interconnects multiple networks together. Every network in the Internet is a group of computers/devices having IP addresses assigned from a pool. This pool of IP addresses can be grouped together and have a common network IP (we'll learn all these details later, for now, just get the basic idea). In the above example, 10.12.9.0/24 is the network IP. So the individual devices of that network can have the IP addresses like 10.12.9.1, 10.12.9.2, etc. All these devices can be reached through the router interface eth0. Therefore, the routing table within the router will contain an entry having 10.12.9.0/24 as the destination network address, and eth0 as the interface through which that destination network can be reached. Therefore, once the router receives a packet having a destination IP of 10.12.9.1 or 10.12.9.2 or anything from the group of IPs given by 10.12.9.0/24, it forwards the packet through the interface eth0.

`ip route` command is used to manage routing tables in Linux. This routing table is then used to forward packets between multiple routers. Each network namespace will have its own routing table. An example output of `ip route`:

```
~ ip route
default via 192.168.0.1 dev wlp1s0 proto static metric 600
169.254.0.0/16 dev wlp1s0 scope link metric 1000
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
172.18.0.0/16 dev br-557d108b9759 proto kernel scope link src 172.18.0.1 linkdown
172.19.0.0/16 dev docker_gwbridge proto kernel scope link src 172.19.0.1
172.21.0.0/16 dev br-7f896f0a28c6 proto kernel scope link src 172.21.0.1 linkdown
192.168.0.0/24 dev wlp1s0 proto kernel scope link src 192.168.0.4 metric 600
```

In the above example, 169.254.0.0/16 is the network address and wlp1s0 is the interface name through which that network can be reached. Similarly, 172.17.0.0/16 is another network address and docker0 is the interface name through which that network can be reached. So, a packet for an IP 169.254.128.19 will be forwarded to wlp1s0, whereas a packet for the IP 172.17.19.20 will be forwarded to the interface docker0.

Check this page to know more about how the routing table in Linux looks like: http://linux-ip.net/html/tools-ip-route.html

Some common use case examples are as follows:

`ip route`

Show all route entries in the kernel.

`ip route add 192.168.2.0/24 via 192.168.1.1 dev eth0`

Adds a route for the network address 192.168.2.0/24 via the gateway 192.168.1.1 that can be reached on device eth0.

`ip route del 192.168.2.0/24`

Read more about ip route here: https://man7.org/linux/man-pages/man8/ip-route.8.html

In this assignment, you have to construct three network topologies (using shell scripts) and perform a set of experiments. Part A also contains a brief tutorial on how to construct the routes. The details follow.

PART - A:

Create four network namespaces and connect them with virtual ethernet (veth) interfaces as shown in Fig 1.

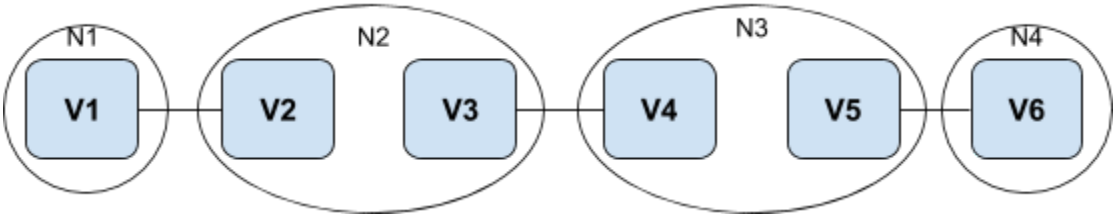


Fig 1.

Assign ip address to the interfaces as follows:

v1: 10.0.10.X
v2: 10.0.10.(X+1)
v3: 10.0.20.X
v4: 10.0.20.(X+1)
v5: 10.0.30.X
v6: 10.0.30.(X+1)

Where X is the last two digits of your roll number.

Make sure all the interfaces are up.

At this point, you will find some routes already added in each of the namespaces. For example:
In N1 you will find: (considering X = 1)

```
10.0.10.0/24 dev v1 proto kernel scope link src 10.0.10.1
```

In N2 you will find:

```
  ip route
10.0.10.0/24 dev v2 proto kernel scope link src 10.0.10.2
10.0.20.0/24 dev v3 proto kernel scope link src 10.0.20.1
```

Therefore, **from N2**, 10.0.10.1 (of N1) will be reachable, as well as 10.0.20.2 (of N3) will be reachable. This is because, from the routes of N2, we see that there are correct rules to forward the packets in v2 and v3, for networks 10.0.10.0/24 and 10.0.20.0/23 respectively.
But, we cannot reach 10.0.30.1 from N2, because there are no valid routes for the same.

```
s3   ping 10.0.10.1
PING 10.0.10.1 (10.0.10.1) 56(84) bytes of data.
64 bytes from 10.0.10.1: icmp_seq=1 ttl=64 time=0.061 ms
```

```
  ping 10.0.20.2
PING 10.0.20.2 (10.0.20.2) 56(84) bytes of data.
64 bytes from 10.0.20.2: icmp_seq=1 ttl=64 time=0.071 ms
```

```
  ping 10.0.30.1
connect: Network is unreachable
```

To add a route for 10.0.30.0/24 in N2, we have to forward it to 10.0.20.2 (i.e. next hop) through the interface v3.

`ip route add 10.0.30.0/24 via 10.0.20.2 dev v3`

Now, the routing table of N2 knows where to send packets going to 10.0.30.0/24.

```
  ip route
10.0.10.0/24 dev v2 proto kernel scope link src 10.0.10.2
10.0.20.0/24 dev v3 proto kernel scope link src 10.0.20.1
10.0.30.0/24 via 10.0.20.2 dev v3
```

```
  ping 10.0.30.1
PING 10.0.30.1 (10.0.30.1) 56(84) bytes of data.
64 bytes from 10.0.30.1: icmp_seq=1 ttl=64 time=0.031 ms
```

But, ping 10.0.30.2 (N4) will now work since N4 does not have the routes to send the echo reply back to N2.

To enable that, you have to run the following in N4:

`ip route add 10.0.20.0/24 via 10.0.30.1 dev v6`

Similarly, configure all the required routes so that all interfaces (irrespective of network namespace) can be pinged from every namespace.

Include the ping commands in the script. Use,
ping -c 3 <IPADDR>
so that it only repeats 3 times.

You need to submit this shell script with name `assignment4_partA.sh`.

PART - B:

Write a shell script to create a virtual network topology as in Fig 2.

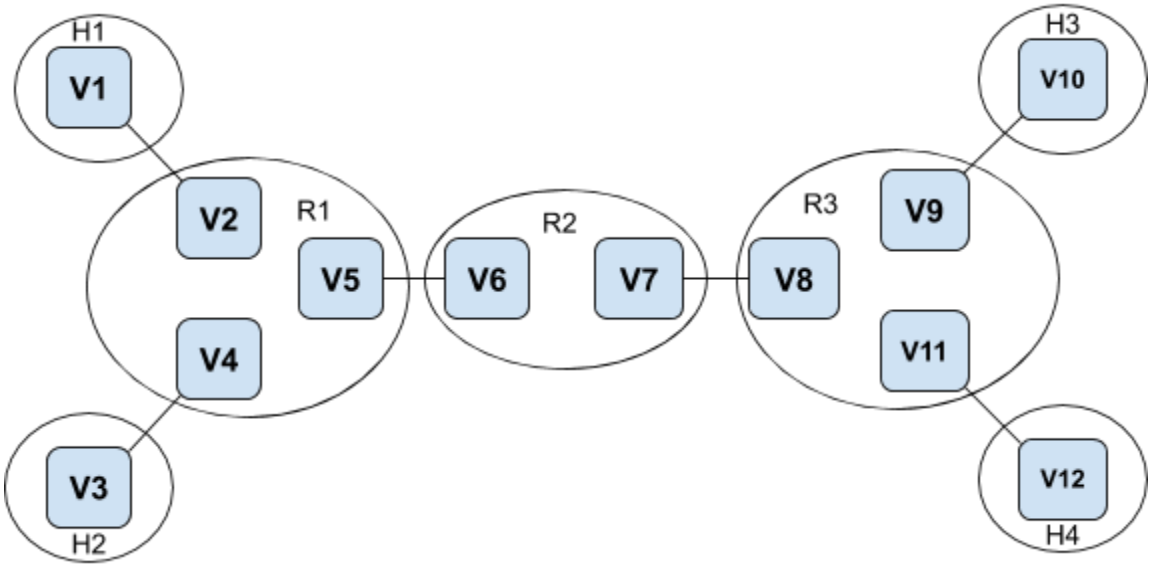


Fig 2.

Here H1, H2,… , R1, R2,…, are network namespaces and they are connected using virtual ethernet interfaces v1 …. v12:

Assign ip address to the interfaces as follows:

v1: 10.0.10.X
v2: 10.0.10.X+1
v3: 10.0.20.X
v4: 10.0.20.X+1
v5: 10.0.30.X
v6: 10.0.30.X+1
v7: 10.0.40.X
v8: 10.0.40.X+1
v9: 10.0.50.X
v10: 10.0.50.X+1
v11: 10.0.60.X
v12: 10.0.60.X+1

Where X is the last two digits of your roll number.

Configure all the required routes so that all interfaces (irrespective of network namespace) can be pinged from every namespace.

Include the ping commands in the script. Use,
ping -c 3 <IPADDR>
so that it only repeats 3 times.

Use traceroute to show the hops from **H1 to H4**, **H3 to H4** , and **H4 to H2**

You need to submit this shell script with name `assignment4_partB.sh`.

PART - C:

Write a shell script to create a virtual network topology as in Fig 2.

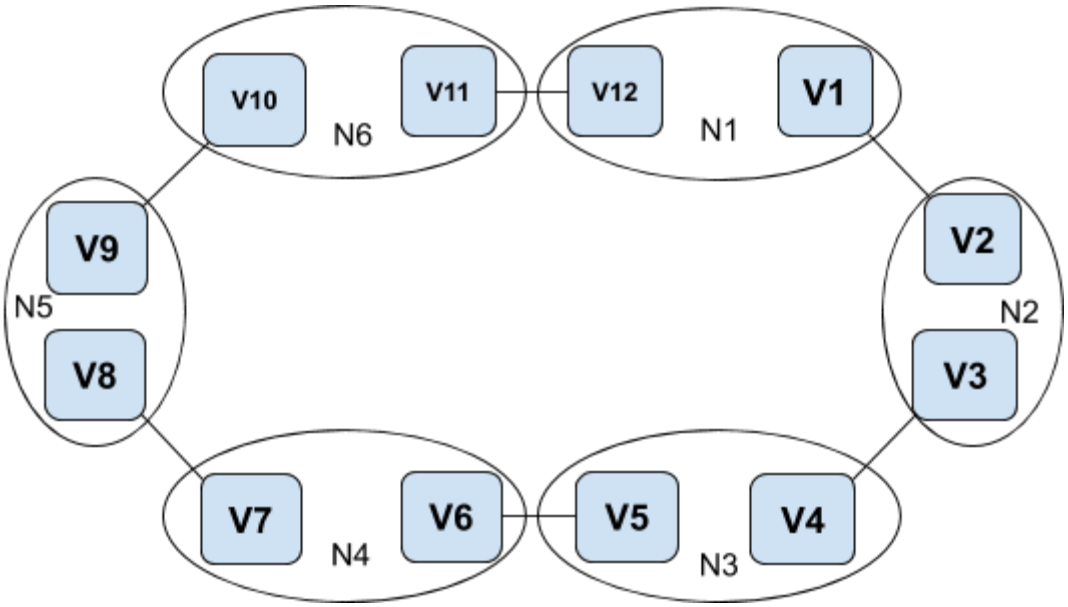


Fig 3.

Assign ip address to the interfaces as follows:

v1: 10.0.10.X
v2: 10.0.10.X+1
v3: 10.0.20.X
v4: 10.0.20.X+1

v5: 10.0.30.X
v6: 10.0.30.X+1
v7: 10.0.40.X
v8: 10.0.40.X+1
v9: 10.0.50.X
v10: 10.0.50.X+1
v11: 10.0.60.X
v12: 10.0.60.X+1

Where X is the last two digits of your roll number.

Configure the routes such that the packet moves only in clockwise direction in the ring. That is, N1->N2->N3->N4->N5->N6->N1->N2 ….
(You can ignore the adjacent namespaces which are directly connected via an virtual ethernet interface)

Therefore, a packet from N1 may move to N6 through v12->v11, but a packet from N1 to N5 will move as N1->N2->N3->N4->N5

Use **traceroute** to show the hops from **N1 to N5**, **N3 to H5** , and **N3 to N1**

You need to submit this shell script with name `assignment4_partC.sh`.

Submission Instruction:

You need to submit the three shell scripts, enclosed in a zip file named `assignment4_<roll_number>.zip` via Moodle by the deadline. Your shell scripts should have sufficient comments explaining the steps you have used in the code.

Tips:

Use the sysctl command to set `net.ipv4.ip_forward=1`

Enable loopback interface **lo** to check if you can ping a namespace's own interfaces (sanity check).