

Part A

A1. Topology

(a) The code for creating the topology is in the attached files start.py and topo.py

(b)

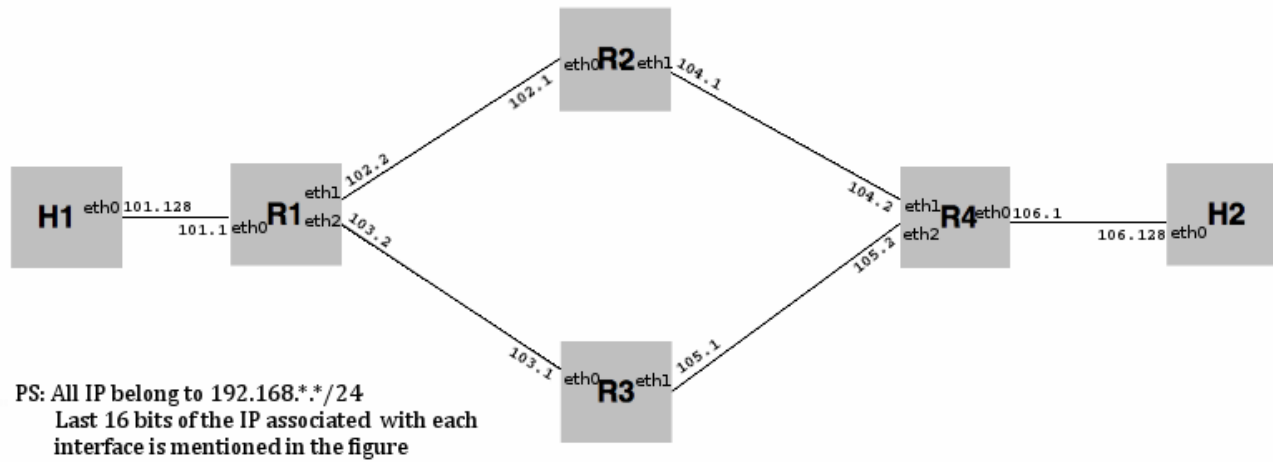


Figure 1. Network Topology

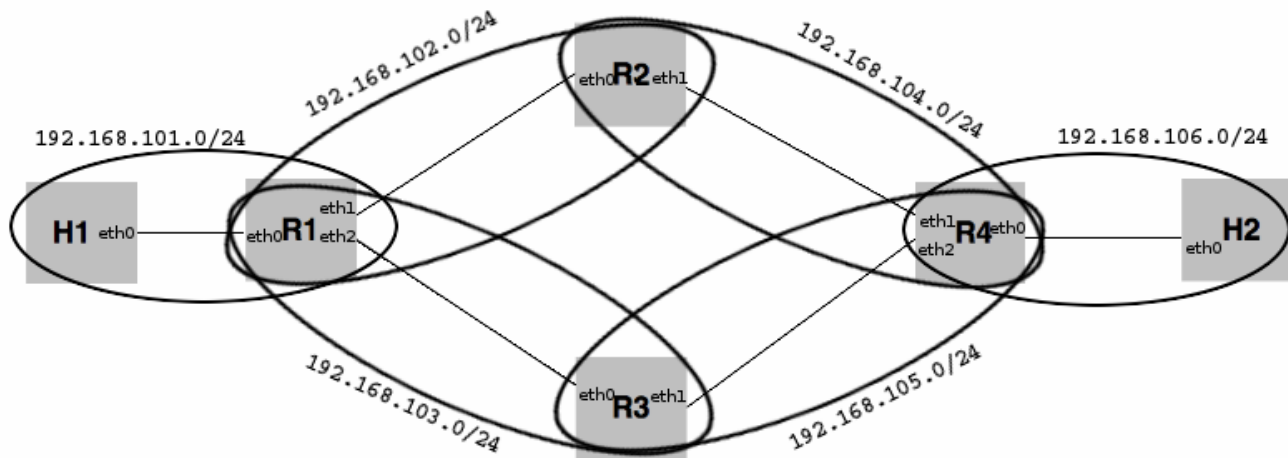


Figure 2. Subnet Figure

A2. Static Route

(a) Routing Table at all Nodes

```
root@mininet-vm: /home/mininet/Neha/PartA/A2
File Edit View Search Terminal Help
mininext> H1 route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
192.168.101.0 * 255.255.255.0 U 0 0 0 H1-eth0
192.168.102.0 192.168.101.1 255.255.255.0 UG 0 0 0 H1-eth0
192.168.103.0 192.168.101.1 255.255.255.0 UG 0 0 0 H1-eth0
192.168.104.0 192.168.101.1 255.255.255.0 UG 0 0 0 H1-eth0
192.168.105.0 192.168.101.1 255.255.255.0 UG 0 0 0 H1-eth0
192.168.106.0 192.168.101.1 255.255.255.0 UG 0 0 0 H1-eth0
mininext> H2 route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
192.168.101.0 192.168.106.1 255.255.255.0 UG 0 0 0 H2-eth0
192.168.102.0 192.168.106.1 255.255.255.0 UG 0 0 0 H2-eth0
192.168.103.0 192.168.106.1 255.255.255.0 UG 0 0 0 H2-eth0
192.168.104.0 192.168.106.1 255.255.255.0 UG 0 0 0 H2-eth0
192.168.105.0 192.168.106.1 255.255.255.0 UG 0 0 0 H2-eth0
192.168.106.0 * 255.255.255.0 U 0 0 0 H2-eth0
mininext>
```

```

root@mininet-vm: /home/mininet/Neha/PartA/A2
File Edit View Search Terminal Help
mininet> R1 route
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.101.0    *              255.255.255.0   U      0      0      0 R1-eth0
192.168.102.0    *              255.255.255.0   U      0      0      0 R1-eth1
192.168.103.0    *              255.255.255.0   U      0      0      0 R1-eth2
192.168.104.0    192.168.102.1  255.255.255.0   UG     0      0      0 R1-eth1
192.168.105.0    192.168.103.1  255.255.255.0   UG     0      0      0 R1-eth2
192.168.106.0    192.168.102.1  255.255.255.0   UG     0      0      0 R1-eth1
mininet> R2 route
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.101.0    192.168.102.2  255.255.255.0   UG     0      0      0 R2-eth0
192.168.102.0    *              255.255.255.0   U      0      0      0 R2-eth0
192.168.103.0    192.168.102.2  255.255.255.0   UG     0      0      0 R2-eth0
192.168.104.0    *              255.255.255.0   U      0      0      0 R2-eth1
192.168.105.0    192.168.104.2  255.255.255.0   UG     0      0      0 R2-eth1
192.168.106.0    192.168.104.2  255.255.255.0   UG     0      0      0 R2-eth1
mininet> R3 route
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.101.0    192.168.103.2  255.255.255.0   UG     0      0      0 R3-eth0
192.168.102.0    192.168.103.2  255.255.255.0   UG     0      0      0 R3-eth0
192.168.103.0    *              255.255.255.0   U      0      0      0 R3-eth0
192.168.104.0    192.168.105.2  255.255.255.0   UG     0      0      0 R3-eth1
192.168.105.0    *              255.255.255.0   U      0      0      0 R3-eth1
192.168.106.0    192.168.105.2  255.255.255.0   UG     0      0      0 R3-eth1
mininet> R4 route
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.101.0    192.168.104.1  255.255.255.0   UG     0      0      0 R4-eth1
192.168.102.0    192.168.104.1  255.255.255.0   UG     0      0      0 R4-eth1
192.168.103.0    192.168.105.1  255.255.255.0   UG     0      0      0 R4-eth2
192.168.104.0    *              255.255.255.0   U      0      0      0 R4-eth1
192.168.105.0    *              255.255.255.0   U      0      0      0 R4-eth2
192.168.106.0    *              255.255.255.0   U      0      0      0 R4-eth0
mininet>

```

The link between H1-R1, R1-R2, R1-R3, R2-R4, R3-R4 and R4-H2 is already created in the topology.

To configure the Routing table correctly I created six subnets: The subnet between H1 and R1 is in

subnet 192.168.101.0/24

The subnet between R1 and R2 is in subnet 192.168.102.0/24

The subnet between R1 and R3 is in subnet 192.168.103.0/24

The subnet between R2 and R4 is in subnet 192.168.104.0/24

The subnet between R3 and R4 is in subnet 192.168.105.0/24

The subnet between R4 and H2 is in subnet 192.168.106.0/24

H1 routes all its packets to R1.

R1 route packet to subnet 192.168.104.0/24 and subnet 192.168.106.0/24 via 192.168.102.2/24, which is the interface of R2.

R1 route packet to subnet 192.168.105.0/24 via 192.168.103.1/24, which is the interface of R3

R2 route packet to subnet 192.168.101.0/24 and subnet 192.168.103.0/24 via 192.168.102.2/24, which is the interface of R1

R2 route packet to subnet 192.168.105.0/24 and subnet 192.168.106.0/24 via 192.168.104.2/24, which is the interface of R4

R3 route packet to subnet 192.168.101.0/24 and subnet 192.168.102.0/24 via 192.168.103.2/24,

which is the interface of R1

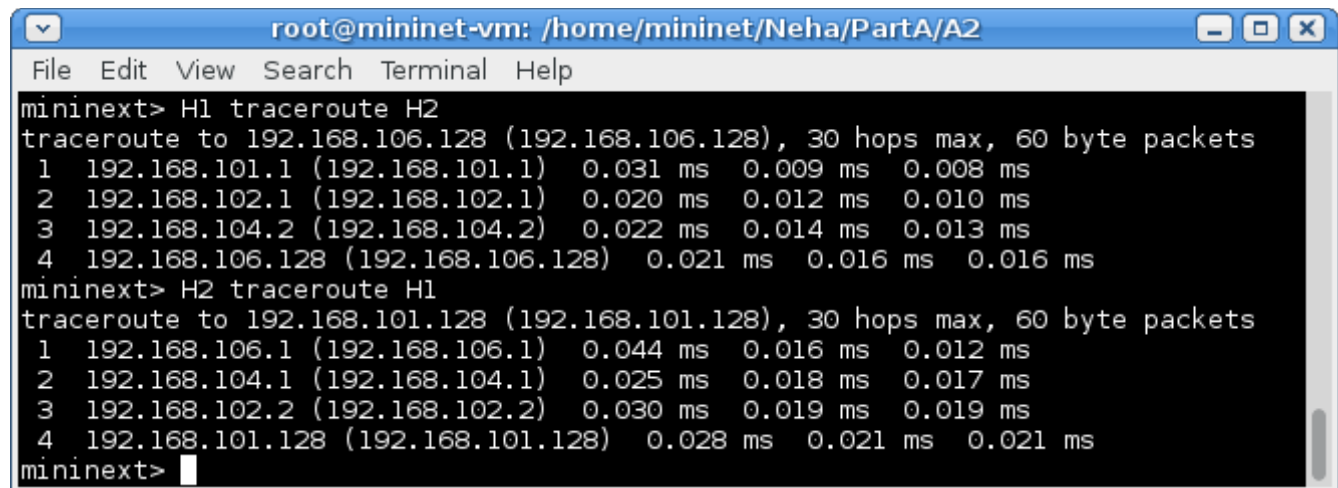
R3 route packet to subnet 192.168.104.0/24 and subnet 192.168.106.0/24 via 192.168.105.2/24, which is the interface of R4

R4 route packet to subnet 192.168.101.0/24 and subnet 192.168.102.0/24 via 192.168.104.1/24, which is the interface of R2.

R4 route packet to subnet 192.168.103.0/24 via 192.168.105.1/24, which is the interface of R3

H2 routes all its packets to R4.

(b) Traceroute



```
root@mininet-vm: /home/mininet/Neha/PartA/A2
File Edit View Search Terminal Help
mininext> H1 traceroute H2
traceroute to 192.168.106.128 (192.168.106.128), 30 hops max, 60 byte packets
 1 192.168.101.1 (192.168.101.1)  0.031 ms  0.009 ms  0.008 ms
 2 192.168.102.1 (192.168.102.1)  0.020 ms  0.012 ms  0.010 ms
 3 192.168.104.2 (192.168.104.2)  0.022 ms  0.014 ms  0.013 ms
 4 192.168.106.128 (192.168.106.128)  0.021 ms  0.016 ms  0.016 ms
mininext> H2 traceroute H1
traceroute to 192.168.101.128 (192.168.101.128), 30 hops max, 60 byte packets
 1 192.168.106.1 (192.168.106.1)  0.044 ms  0.016 ms  0.012 ms
 2 192.168.104.1 (192.168.104.1)  0.025 ms  0.018 ms  0.017 ms
 3 192.168.102.2 (192.168.102.2)  0.030 ms  0.019 ms  0.019 ms
 4 192.168.101.128 (192.168.101.128)  0.028 ms  0.021 ms  0.021 ms
mininext> 
```

Figure. TraceRoute between H1 and H2

Part B

B1. RIP Configuration

I have set the default gateway of H1(192.168.101.128/24) as R1(192.168.101.1) and H2(192.168.106.128/24) as R4(192.168.106.1). This I did in topo.py file itself.

I configured Quagga ripd daemon to set R1, R2, R3 and R4 as RIP router.

Step 1:

I enable zebra and ripd in the file /etc/quagga/daemons

zebra=yes

ripd=yes

Step 2:

Next I created two blank files - zebra.conf and ripd.conf in the config directory of all the routers - /config/XX, where XX is R1, R2, R3, R4

To set password for zebra I updated the zebra.conf file for every router and included the following line:
password a

Step 3:

Then I copied /etc/quagga/debian.conf to all the routers as above

Step 4:

Then I restart the quagga service:

\$ /etc/init.d/quagga restart

Step 5:

Lastly I configured every router with Quagga ripd daemons

a. Run start.py

```
$ python start.py
```

b. Login to any router/host

```
mininet> xterm R1
```

c. Launch mx on any host/router

```
$ cd /home/mininet/miniNET/util
```

```
$ ./mx R1
```

d. Find the port of running ripd

```
$ netstat -na
```

e. Remote access for localhost/ripd process

```
$ telnet localhost zebra
```

```
User Access Verification
```

```
Password:
```

-> login with password

```
ripd> enable
```

```
ripd# config term
```

-> to configure the ripd terminal

```
ripd(config)# router rip
```

-> Enables RIP as a routing protocol

```
ripd(config-router)# network X.X.X.X/24
```

-> configure the subnet where the router is connected to

```
ripd(config-router)# exit    -> come out of the terminal
ripd(config)# write          -> save the config
                             configuration saved to /configs/R1/ripd.conf
```

Finally all ripd configuration are saved in /configs/XX/ripd.conf, where XX is R1, R2, R3, R4
Once done I run the start.py file again to check. All screen shots are after this step.

B2. Run RIP Daemon

(a) Routing Table at Each Node (both the kernel and the Quagga routing table)

```
root@mininet-vm: /home/mininet/Neha/PartB/B1
File Edit View Search Terminal Help
mininet> H1 route
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
default          192.168.101.1   0.0.0.0          UG    0      0      0 H1-eth0
192.168.101.0    *               255.255.255.0    U      0      0      0 H1-eth0
mininet> H2 route
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
default          192.168.106.1   0.0.0.0          UG    0      0      0 H2-eth0
192.168.106.0    *               255.255.255.0    U      0      0      0 H2-eth0
mininet>
```

```
root@mininet-vm: /home/mininet/Neha/PartB/B1
File Edit View Search Terminal Help
mininet> R1 route
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
192.168.101.0    *               255.255.255.0    U      0      0      0 R1-eth0
192.168.102.0    *               255.255.255.0    U      0      0      0 R1-eth1
192.168.103.0    *               255.255.255.0    U      0      0      0 R1-eth2
192.168.104.0    192.168.102.1   255.255.255.0    UG     2      0      0 R1-eth1
192.168.105.0    192.168.103.1   255.255.255.0    UG     2      0      0 R1-eth2
192.168.106.0    192.168.102.1   255.255.255.0    UG     3      0      0 R1-eth1
mininet>
```

```
root@mininet-vm: /home/mininet/Neha/PartB/B1
File Edit View Search Terminal Help
mininet> R1 route
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
192.168.101.0    *                255.255.255.0    U        0      0        0 R1-eth0
192.168.102.0    *                255.255.255.0    U        0      0        0 R1-eth1
192.168.103.0    *                255.255.255.0    U        0      0        0 R1-eth2
192.168.104.0    192.168.102.1    255.255.255.0    UG       2      0        0 R1-eth1
192.168.105.0    192.168.103.1    255.255.255.0    UG       2      0        0 R1-eth2
192.168.106.0    192.168.102.1    255.255.255.0    UG       3      0        0 R1-eth1
mininet>
```

```
root@mininet-vm: /home/mininet/Neha/PartB/B1
File Edit View Search Terminal Help
mininet> R1 ip route show
192.168.101.0/24 dev R1-eth0 proto kernel scope link src 192.168.101.1
192.168.102.0/24 dev R1-eth1 proto kernel scope link src 192.168.102.2
192.168.103.0/24 dev R1-eth2 proto kernel scope link src 192.168.103.2
192.168.104.0/24 via 192.168.102.1 dev R1-eth1 proto zebra metric 2
192.168.105.0/24 via 192.168.103.1 dev R1-eth2 proto zebra metric 2
192.168.106.0/24 via 192.168.102.1 dev R1-eth1 proto zebra metric 3
mininet>
```

```
root@mininet-vm: /home/mininet/Neha/PartB/B1
File Edit View Search Terminal Help
mininet> R2 route
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
192.168.101.0    192.168.102.2    255.255.255.0    UG       2      0        0 R2-eth0
192.168.102.0    *                255.255.255.0    U        0      0        0 R2-eth0
192.168.103.0    192.168.102.2    255.255.255.0    UG       2      0        0 R2-eth0
192.168.104.0    *                255.255.255.0    U        0      0        0 R2-eth1
192.168.105.0    192.168.104.2    255.255.255.0    UG       2      0        0 R2-eth1
192.168.106.0    192.168.104.2    255.255.255.0    UG       2      0        0 R2-eth1
mininet>
```



```
root@mininet-vm: /home/mininet/Neha/PartB/B1
File Edit View Search Terminal Help
mininet> R2 ip route show

root@mininet-vm: /home/mininet/Neha/PartB/B1
File Edit View Search Terminal Help
mininet> R3 route
Kernel IP routing table

root@mininet-vm: /home/mininet/Neha/PartB/B1
File Edit View Search Terminal Help
mininet> R3 ip route show

root@mininet-vm: /home/mininet/Neha/PartB/B1
File Edit View Search Terminal Help
mininet> R4 route
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.101.0    192.168.104.1   255.255.255.0   UG      3      0      0 R4-eth1
192.168.102.0    192.168.104.1   255.255.255.0   UG      2      0      0 R4-eth1
192.168.103.0    192.168.105.1   255.255.255.0   UG      2      0      0 R4-eth2
192.168.104.0    *               255.255.255.0   U       0      0      0 R4-eth1
192.168.105.0    *               255.255.255.0   U       0      0      0 R4-eth2
192.168.106.0    *               255.255.255.0   U       0      0      0 R4-eth0
mininet>
```

1
1

```
root@mininet-vm: /home/mininet/Neha/PartB/B1
File Edit View Search Terminal Help
mininet> R4 ip route show
192.168.101.0/24 via 192.168.104.1 dev R4-eth1 proto zebra metric 3
192.168.102.0/24 via 192.168.104.1 dev R4-eth1 proto zebra metric 2
192.168.103.0/24 via 192.168.105.1 dev R4-eth2 proto zebra metric 2
192.168.104.0/24 dev R4-eth1 proto kernel scope link src 192.168.104.2
192.168.105.0/24 dev R4-eth2 proto kernel scope link src 192.168.105.2
192.168.106.0/24 dev R4-eth0 proto kernel scope link src 192.168.106.1
mininet>
```


(b) Traceroute between H1 and H2

```
root@mininet-vm: /home/mininet/Neha/PartB/B1
File Edit View Search Terminal Help
** Dumping host connections
H1 H1-eth0:R1-eth0
H2 H2-eth0:R4-eth0
R1 R1-eth0:H1-eth0 R1-eth1:R2-eth0 R1-eth2:R3-eth0
R2 R2-eth0:R1-eth1 R2-eth1:R4-eth1
R3 R3-eth0:R1-eth2 R3-eth1:R4-eth2
R4 R4-eth0:H2-eth0 R4-eth1:R2-eth1 R4-eth2:R3-eth1
** Testing network connectivity
*** H1 : ('ping -c4 192.168.106.128',)
PING 192.168.106.128 (192.168.106.128) 56(84) bytes of data.
From 192.168.101.1 icmp_seq=1 Destination Net Unreachable
From 192.168.101.1 icmp_seq=2 Destination Net Unreachable
64 bytes from 192.168.101.1: icmp_seq=3 ttl=61 time=0.084 ms
64 bytes from 192.168.101.1: icmp_seq=4 ttl=61 time=0.078 ms

--- 192.168.106.128 ping statistics ---
4 packets transmitted, 2 received, +2 errors, 50% packet loss, time 2997ms
rtt min/avg/max/mdev = 0.078/0.081/0.084/0.003 ms
** Setup Time - 3.05957198143 s
H1 -> H2 R1 R2 R3 R4
H2 -> H1 R1 R2 R3 R4
R1 -> H1 H2 R2 R3 R4
R2 -> H1 H2 R1 R3 R4
R3 -> H1 H2 R1 R2 R4
R4 -> H1 H2 R1 R2 R3
*** Results: 0% dropped (30/30 received)
** Running CLI
*** Starting CLI:
mininext>
```

(c) Time taken for ping = 3.06s

(d) Convergence Time.

As it can be seen from the output, initially it takes around 3 secs to converge.

To determine convergence time I'm using the ping command. By implementation ping sends packets every 1 sec on unsuccessful ping. So as soon as the setup is done I'm sending some ping command from H1 to H2. This gives the number of seconds it takes for the algorithm to converge.

These 3.05957198143s includes initial setup and convergence time.

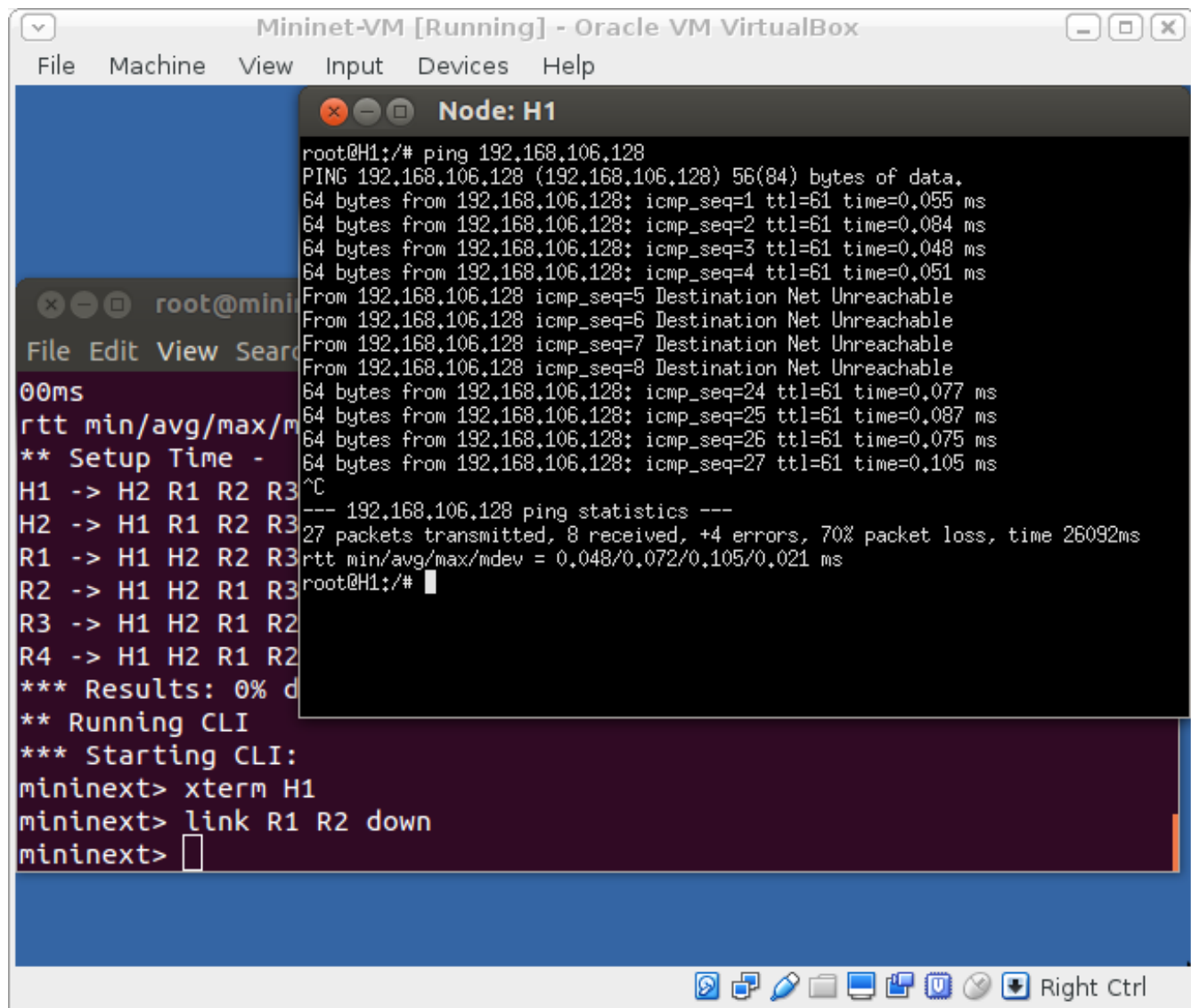
B3. Connectivity Re-establish

(a) To bring a link down I am using the command:

```
$ link R1 R2 down
```

(b) Time for connectivity to be established: 10s (approx).

First I log into H1 (xterm H1) and start ping to H2. Then I break down the link from R1 and R2. This results in the ping to fail, since the route between H1 and H2 is broken. After 10 consecutive destination host unreachable H1 connects to H2 again. Since ping waits for 1 sec by default before sending another packet, we can assume that it took 10 secs to establish connection.



```
Mininet-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

Node: H1
root@H1:~# ping 192.168.106.128
PING 192.168.106.128 (192.168.106.128) 56(84) bytes of data:
64 bytes from 192.168.106.128: icmp_seq=1 ttl=61 time=0.055 ms
64 bytes from 192.168.106.128: icmp_seq=2 ttl=61 time=0.084 ms
64 bytes from 192.168.106.128: icmp_seq=3 ttl=61 time=0.048 ms
64 bytes from 192.168.106.128: icmp_seq=4 ttl=61 time=0.051 ms
From 192.168.106.128 icmp_seq=5 Destination Net Unreachable
From 192.168.106.128 icmp_seq=6 Destination Net Unreachable
From 192.168.106.128 icmp_seq=7 Destination Net Unreachable
From 192.168.106.128 icmp_seq=8 Destination Net Unreachable
64 bytes from 192.168.106.128: icmp_seq=24 ttl=61 time=0.077 ms
64 bytes from 192.168.106.128: icmp_seq=25 ttl=61 time=0.087 ms
64 bytes from 192.168.106.128: icmp_seq=26 ttl=61 time=0.075 ms
64 bytes from 192.168.106.128: icmp_seq=27 ttl=61 time=0.105 ms
^C
--- 192.168.106.128 ping statistics ---
27 packets transmitted, 8 received, 70% packet loss, time 26092ms
rtt min/avg/max/mdev = 0.048/0.072/0.105/0.021 ms
root@H1:~#

root@mininet:~#
File Edit View Search
00ms
rtt min/avg/max/mdev = 0.048/0.072/0.105/0.021 ms
** Setup Time -
H1 -> H2 R1 R2 R3
H2 -> H1 R1 R2 R3
R1 -> H1 H2 R2 R3
R2 -> H1 H2 R1 R3
R3 -> H1 H2 R1 R2
R4 -> H1 H2 R1 R2
*** Results: 0% d
** Running CLI
*** Starting CLI:
mininet> xterm H1
mininet> link R1 R2 down
mininet>
```

(c) Provide the Traceroute Output

```
root@mininet-vm: /home/mininet/Neha/PartB/B1
File Edit View Search Terminal Help
mininet> H1 traceroute H2
traceroute to 192.168.106.128 (192.168.106.128), 30 hops max, 60 byte packets
 1 192.168.101.1 (192.168.101.1)  0.039 ms  0.011 ms  0.010 ms
 2 192.168.103.1 (192.168.103.1)  0.023 ms  0.014 ms  0.014 ms
 3 192.168.105.2 (192.168.105.2)  0.025 ms  0.018 ms  0.017 ms
 4 192.168.106.128 (192.168.106.128)  0.027 ms  0.020 ms  0.020 ms
mininet> 
```

Part C

C1. Running Custom RIP Protocol

(a) Code – the following code are submitted:

- start.py – Starting and creating the topology
- topo.py – Topology description
- server.py – The server code to run on each node
- weight.txt – The weight of each edge in the topology

For negative cycle Bellman Ford algorithm does not converge, rather it exits. In distributed case, there are many ways of handling a negative weight edge. What I am doing is convert the negative weight to 0. This might not always give the optimal path, but does gives a proper path and the algorithm converges.

```
Mininet-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

Node:R2
R1 R1 10.0
R2 R2 0
R3 R1 16.0
R4 R4 4
R2 Routing Table; Convergence Time - 8.72570705414
Destination via Weight
H1 R1 12.0
R1 R1 10.0
R2 R2 0
R3 R4 9.0
R4 R4 4.0
H2 R4 6.0

Node:R4
R2 R2 4.0
R3 R3 5.0
R4 R4 0
H2 H2 2.0
R4 Routing Table; Convergence Time - 12.7707469463
Destination via Weight
H1 R3 13.0
R1 R3 11.0
R2 R2 4.0
R3 R3 5.0
R4 R4 0
H2 H2 2.0

Node:R1
R2 R2 10.0
R3 R3 6.0
R4 R3 11.0
H2 R2 16.0
R1 Routing Table; Convergence Time - 13.8968241215
Destination via Weight
H1 H1 2.0
R1 R1 0
R2 R2 10.0
R3 R3 6.0
R4 R3 11.0
H2 R3 13.0

mininet> xterm h2
mininet> xterm r1

Node:H2
R3 R4 7.0
R4 R4 2.0
H2 H2 0
H2 Routing Table; Convergence Time - 14.6770019531
Destination via Weight
H1 R4 15.0
R1 R4 13.0
R2 R4 6.0
R3 R4 7.0
R4 R4 2.0
H2 H2 0

Node:R3
R2 R1 16.0
R3 R3 0
R4 R4 5
R3 Routing Table; Convergence Time - 7.49285411835
Destination via Weight
H1 R1 8.0
R1 R1 6.0
R2 R4 9.0
R3 R3 0
R4 R4 5.0
H2 R4 7.0

Node:H1
R2 R1 12.0
R3 R1 8.0
R4 R1 13.0
H1 Routing Table; Convergence Time - 15.0286939144
Destination via Weight
H1 H1 0
R1 R1 2.0
R2 R1 12.0
R3 R1 8.0
R4 R1 13.0
H2 R1 15.0

Right Ctrl
```

My solution works fine for positive weights. The output is for the following case:

R1—R2 10 R1—R3 6 R2—R4 4
R3—R4 5 H1—R1 2 R4—H2 2

(b) Time taken to find shortest Path – 15.028 sec

(c) Application Layer Routing Table at each node

C2. Weight Change

(a) Time Taken to Converge – 28.030 sec

(b) Application Layer Routing Table at each node

The screenshot displays a Mininet-VM environment with five terminal windows showing routing tables for different nodes. The terminal also shows the execution of `xterm h2` and `xterm r1` commands.

Node: R2

Destination	via	Weight
H1	R1	12.0
R1	R1	10.0
R2	R2	0
R3	R4	9.0
R4	R4	4.0
H2	R4	6.0

R2 Routing Table; Convergence Time - 15.1030180454

Node: R4

Destination	via	Weight
H1	R3	8.0
R1	R3	6.0
R2	R2	4.0
R3	R3	5.0
R4	R4	0
H2	H2	2.0

R4 Routing Table; Convergence Time - 28.0306560993

Node: R1

Destination	via	Weight
H1	H1	2.0
R1	R1	0
R2	R2	10.0
R3	R3	1.0
R4	R3	6.0
H2	R3	8.0

R1 Routing Table; Convergence Time - 23.4532020092

Node: H2

Destination	via	Weight
H1	R4	10.0
R1	R4	8.0
R2	R4	6.0
R3	R4	7.0
R4	R4	2.0
H2	H2	0

H2 Routing Table; Convergence Time - 23.377466917

Node: R3

Destination	via	Weight
H1	R1	3.0
R1	R1	1.0
R2	R4	9.0
R3	R3	0
R4	R4	5.0
H2	R4	7.0

R3 Routing Table; Convergence Time - 20.2702821032

Node: H1

Destination	via	Weight
H1	H1	0
R1	R1	2.0
R2	R1	12.0
R3	R1	3.0
R4	R1	8.0
H2	R1	10.0

H1 Routing Table; Convergence Time - 27.8705821037

mininet> xterm h2
mininet> xterm r1