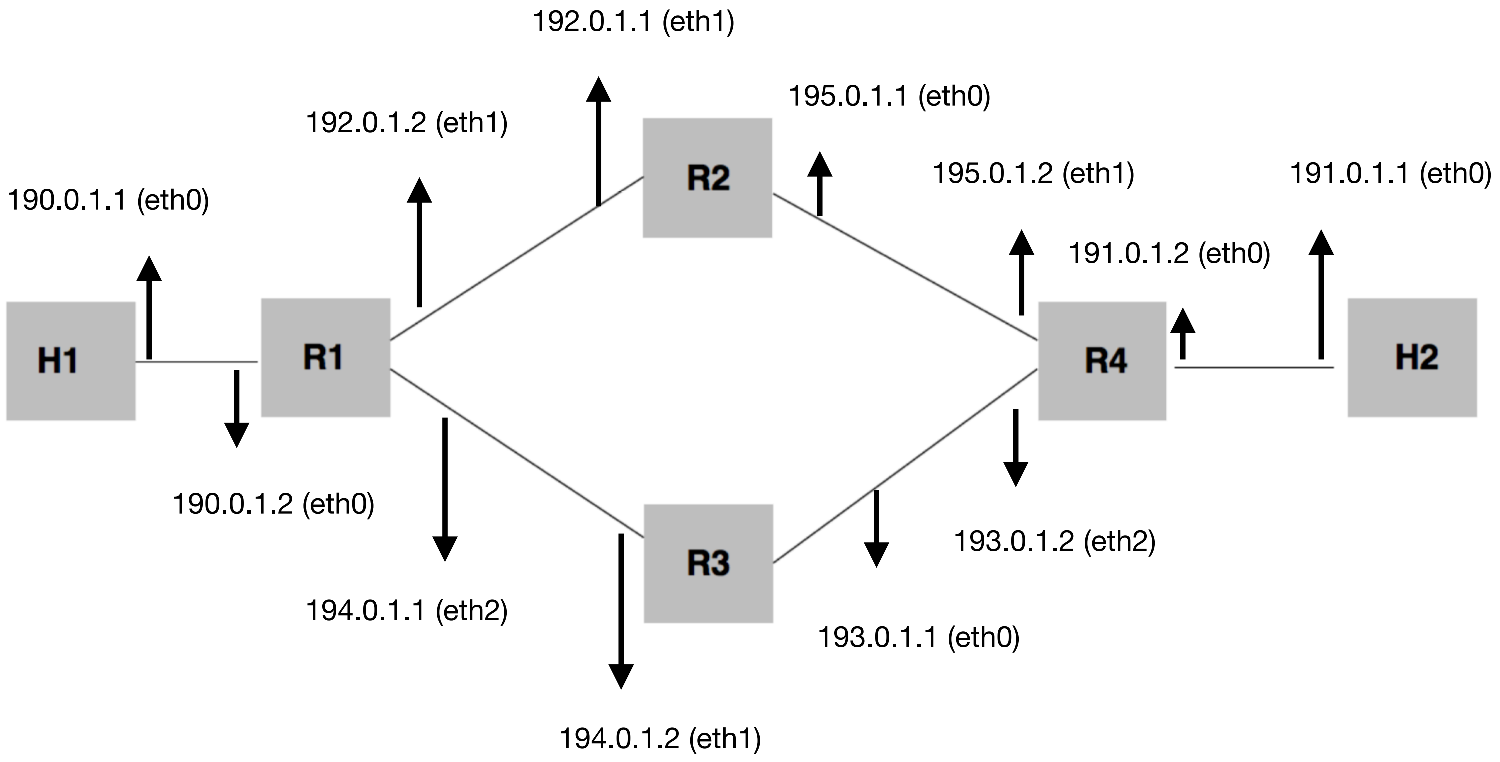
CSE 534- ASSIGNMENT 3
DIVYANSH UPRETI
112026646

Part A:

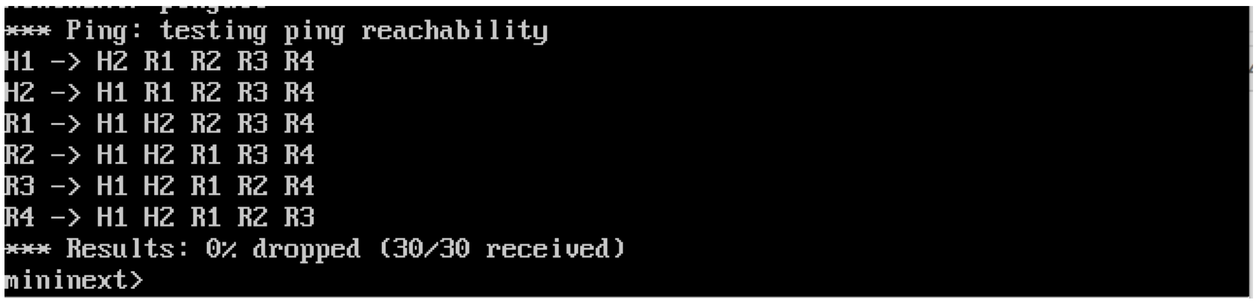
A1.

a.) topo.py is present inside the PartA folder.

b. My topology:



pingall command tests the connectivity between each component:

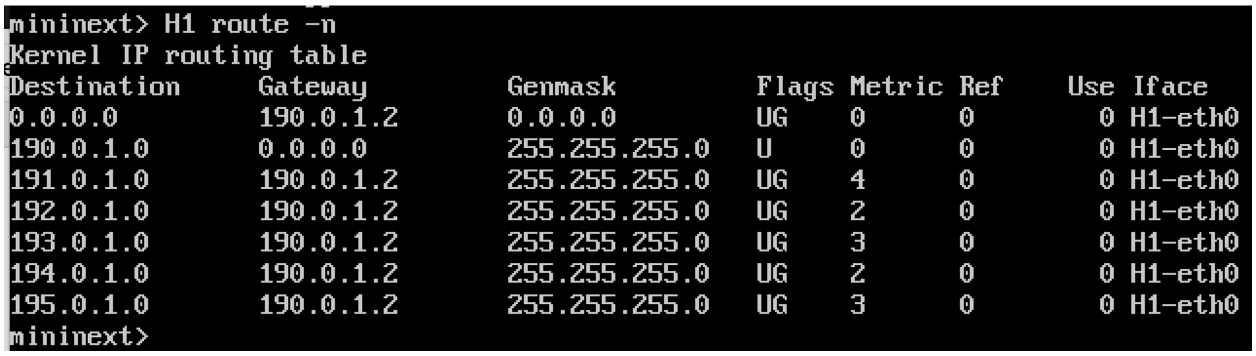


```
*** Ping: testing ping reachability
H1 -> H2 R1 R2 R3 R4
H2 -> H1 R1 R2 R3 R4
R1 -> H1 H2 R2 R3 R4
R2 -> H1 H2 R1 R3 R4
R3 -> H1 H2 R1 R2 R4
R4 -> H1 H2 R1 R2 R3
*** Results: 0% dropped (30/30 received)
mininext>
```

A2. Create static routes:

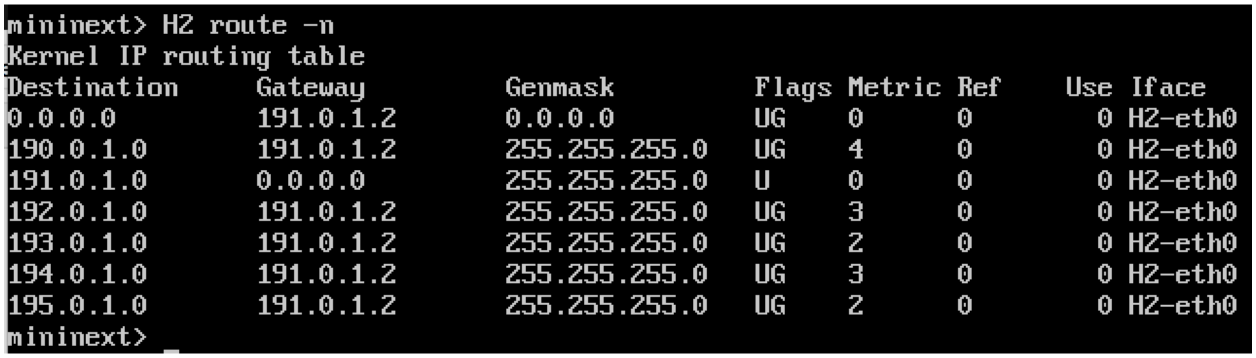
a.) Routing table at all nodes:

H1.



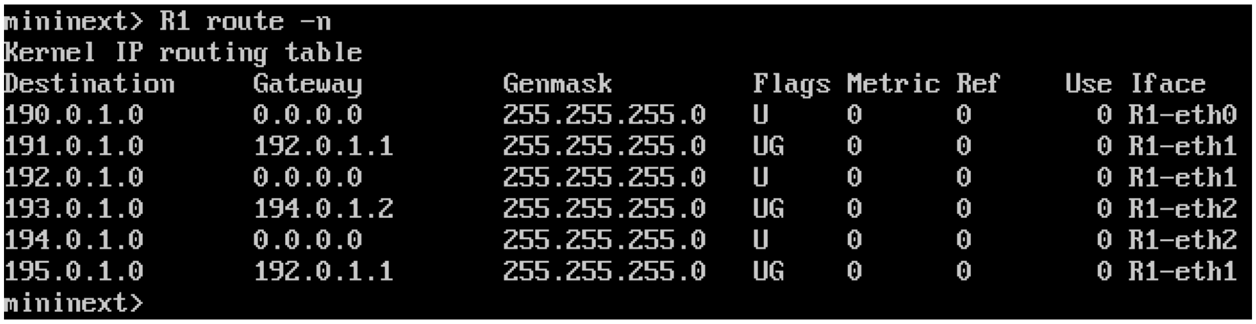
```
mininext> H1 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         190.0.1.2       0.0.0.0         UG    0      0        0 H1-eth0
190.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 H1-eth0
191.0.1.0       190.0.1.2       255.255.255.0   UG    4      0        0 H1-eth0
192.0.1.0       190.0.1.2       255.255.255.0   UG    2      0        0 H1-eth0
193.0.1.0       190.0.1.2       255.255.255.0   UG    3      0        0 H1-eth0
194.0.1.0       190.0.1.2       255.255.255.0   UG    2      0        0 H1-eth0
195.0.1.0       190.0.1.2       255.255.255.0   UG    3      0        0 H1-eth0
mininext>
```

H2



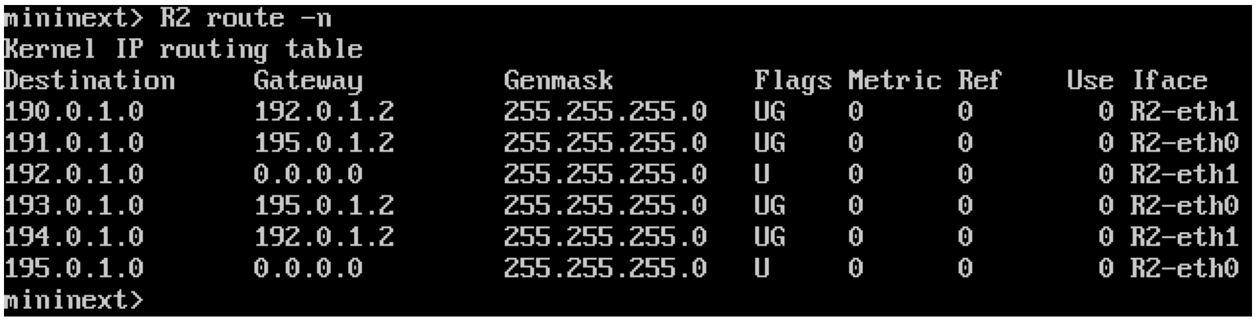
```
mininext> H2 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         191.0.1.2       0.0.0.0         UG    0      0        0 H2-eth0
190.0.1.0       191.0.1.2       255.255.255.0   UG    4      0        0 H2-eth0
191.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 H2-eth0
192.0.1.0       191.0.1.2       255.255.255.0   UG    3      0        0 H2-eth0
193.0.1.0       191.0.1.2       255.255.255.0   UG    2      0        0 H2-eth0
194.0.1.0       191.0.1.2       255.255.255.0   UG    3      0        0 H2-eth0
195.0.1.0       191.0.1.2       255.255.255.0   UG    2      0        0 H2-eth0
mininext>
```

R1



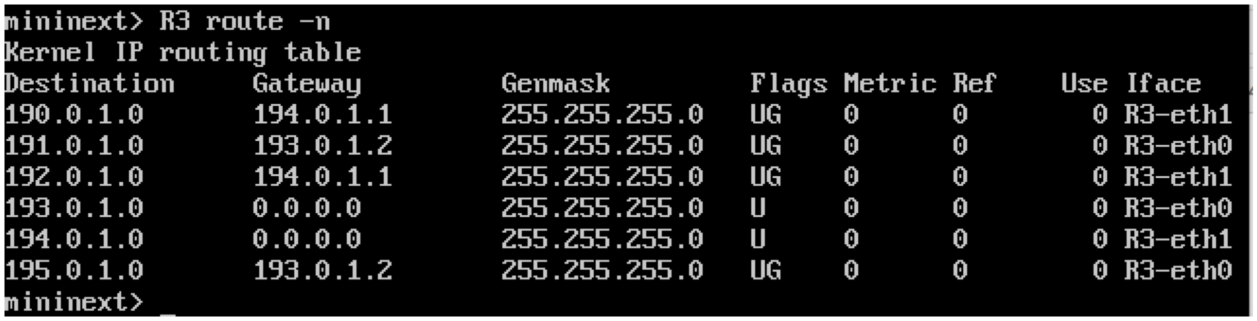
```
mininext> R1 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
190.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 R1-eth0
191.0.1.0       192.0.1.1       255.255.255.0   UG    0      0        0 R1-eth1
192.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 R1-eth1
193.0.1.0       194.0.1.2       255.255.255.0   UG    0      0        0 R1-eth2
194.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 R1-eth2
195.0.1.0       192.0.1.1       255.255.255.0   UG    0      0        0 R1-eth1
mininext>
```

R2



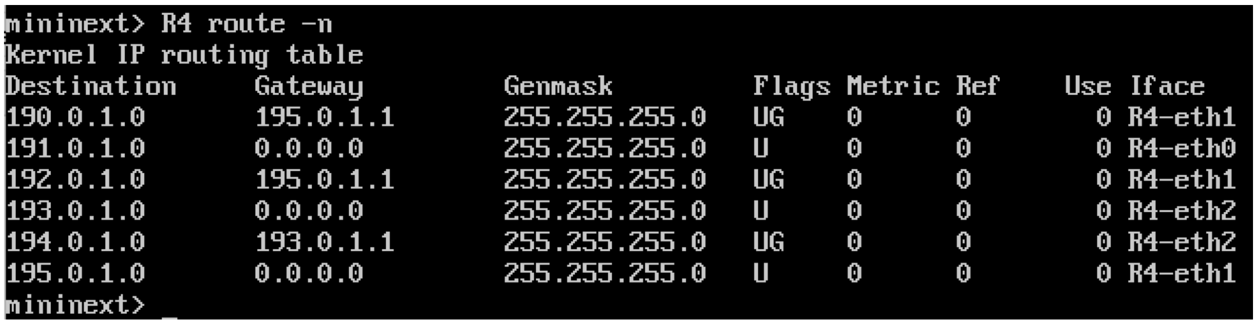
```
mininext> R2 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
190.0.1.0       192.0.1.2       255.255.255.0   UG    0      0        0 R2-eth1
191.0.1.0       195.0.1.2       255.255.255.0   UG    0      0        0 R2-eth0
192.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 R2-eth1
193.0.1.0       195.0.1.2       255.255.255.0   UG    0      0        0 R2-eth0
194.0.1.0       192.0.1.2       255.255.255.0   UG    0      0        0 R2-eth1
195.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 R2-eth0
mininext>
```

R3



```
mininext> R3 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
190.0.1.0       194.0.1.1       255.255.255.0   UG    0      0        0 R3-eth1
191.0.1.0       193.0.1.2       255.255.255.0   UG    0      0        0 R3-eth0
192.0.1.0       194.0.1.1       255.255.255.0   UG    0      0        0 R3-eth1
193.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 R3-eth0
194.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 R3-eth1
195.0.1.0       193.0.1.2       255.255.255.0   UG    0      0        0 R3-eth0
mininext> _
```

R4



```
mininext> R4 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
190.0.1.0       195.0.1.1       255.255.255.0   UG    0      0        0 R4-eth1
191.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 R4-eth0
192.0.1.0       195.0.1.1       255.255.255.0   UG    0      0        0 R4-eth1
193.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 R4-eth2
194.0.1.0       193.0.1.1       255.255.255.0   UG    0      0        0 R4-eth2
195.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 R4-eth1
mininext> _
```

How to configure nodes:

First, I removed all static routes that I had created in PartA.

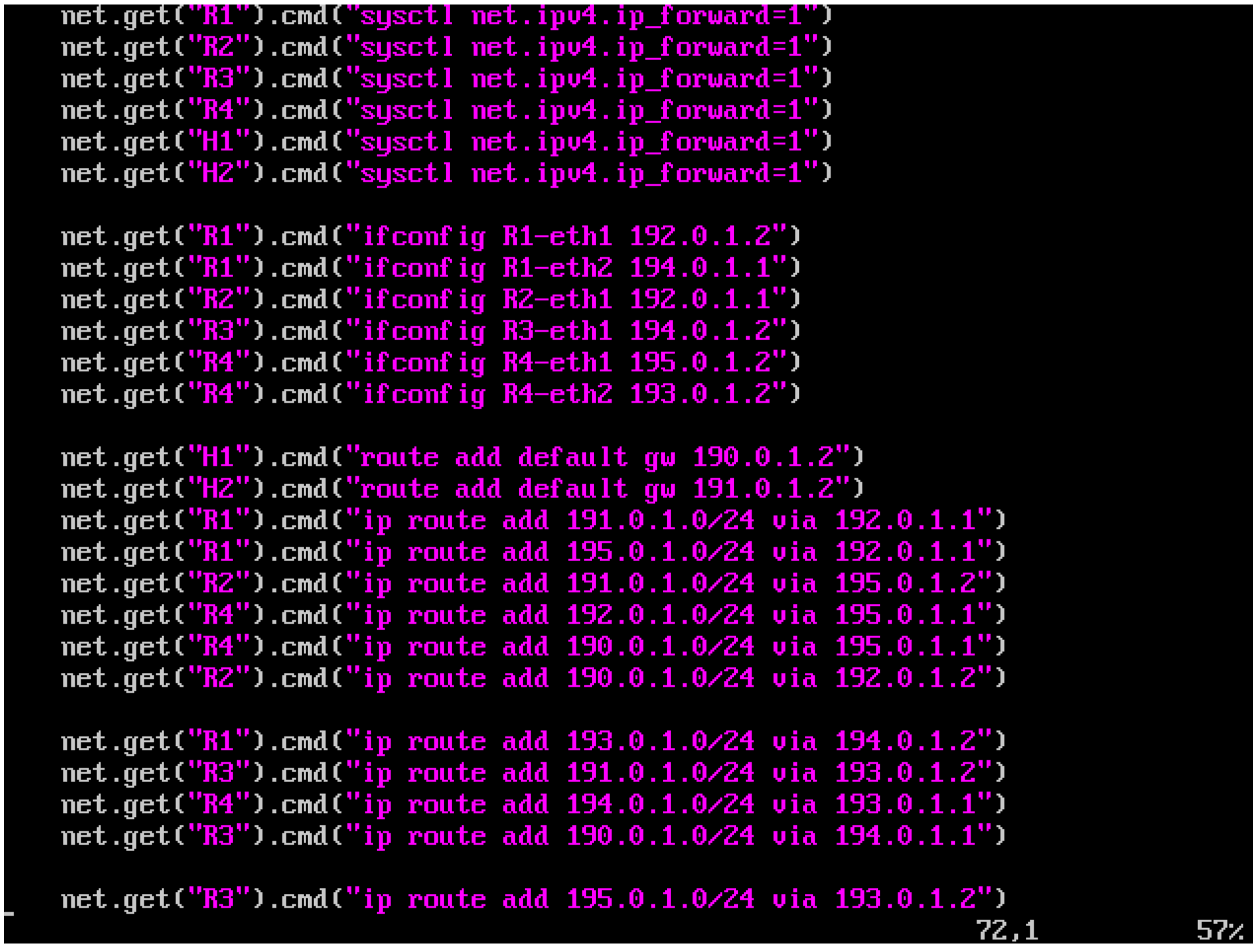
To configure each node, I enabled IP forwarding by using command ***net.get("hostname").cmd("sysctl net.ipv4.ip_forward=1")*** for each node.

Then I assigned IP addresses to to all network interfaces ***using command net.get("R1").cmd("ifconfig R1-eth1 'ip address'")***.

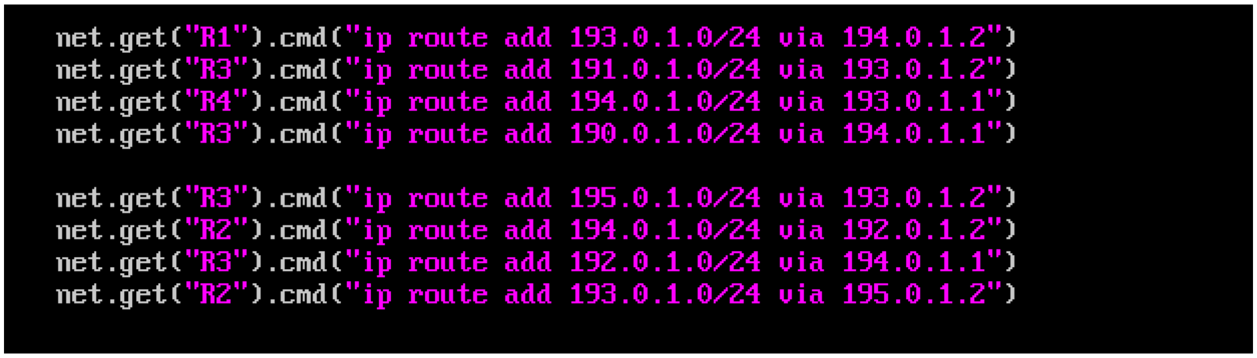
Then I set the default gateway for nodes H1 and H2 by command ***net.get("H1").cmd("route add default gw <ip-address>")***.

Finally, I added correct static routes between nodes by using command ***net.get("R1").cmd("ip route add <ip-address1>/24 via <>ip-address2")***. The first ip addresses is the destination Ip address along with its subnet information. The second ip is the address through which the packet is routed forward. All this information is present in start.py file in PartB folder.

Screenshot of commands:

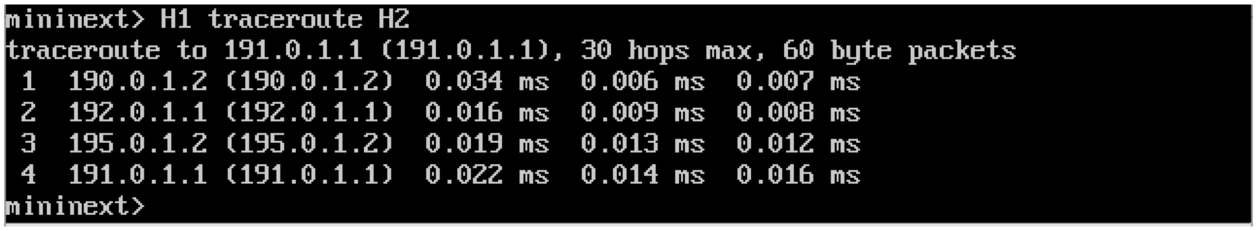


```
    net.get("R1").cmd("sysctl net.ipv4.ip_forward=1")
    net.get("R2").cmd("sysctl net.ipv4.ip_forward=1")
    net.get("R3").cmd("sysctl net.ipv4.ip_forward=1")
    net.get("R4").cmd("sysctl net.ipv4.ip_forward=1")
    net.get("H1").cmd("sysctl net.ipv4.ip_forward=1")
    net.get("H2").cmd("sysctl net.ipv4.ip_forward=1")

    net.get("R1").cmd("ifconfig R1-eth1 192.0.1.2")
    net.get("R1").cmd("ifconfig R1-eth2 194.0.1.1")
    net.get("R2").cmd("ifconfig R2-eth1 192.0.1.1")
    net.get("R3").cmd("ifconfig R3-eth1 194.0.1.2")
    net.get("R4").cmd("ifconfig R4-eth1 195.0.1.2")
    net.get("R4").cmd("ifconfig R4-eth2 193.0.1.2")

    net.get("H1").cmd("route add default gw 190.0.1.2")
    net.get("H2").cmd("route add default gw 191.0.1.2")
    net.get("R1").cmd("ip route add 191.0.1.0/24 via 192.0.1.1")
    net.get("R1").cmd("ip route add 195.0.1.0/24 via 192.0.1.1")
    net.get("R2").cmd("ip route add 191.0.1.0/24 via 195.0.1.2")
    net.get("R4").cmd("ip route add 192.0.1.0/24 via 195.0.1.1")
    net.get("R4").cmd("ip route add 190.0.1.0/24 via 195.0.1.1")
    net.get("R2").cmd("ip route add 190.0.1.0/24 via 192.0.1.2")

    net.get("R1").cmd("ip route add 193.0.1.0/24 via 194.0.1.2")
    net.get("R3").cmd("ip route add 191.0.1.0/24 via 193.0.1.2")
    net.get("R4").cmd("ip route add 194.0.1.0/24 via 193.0.1.1")
    net.get("R3").cmd("ip route add 190.0.1.0/24 via 194.0.1.1")

_   net.get("R3").cmd("ip route add 195.0.1.0/24 via 193.0.1.2")
                                                            72,1          57%
```



```
   net.get("R1").cmd("ip route add 193.0.1.0/24 via 194.0.1.2")
   net.get("R3").cmd("ip route add 191.0.1.0/24 via 193.0.1.2")
   net.get("R4").cmd("ip route add 194.0.1.0/24 via 193.0.1.1")
   net.get("R3").cmd("ip route add 190.0.1.0/24 via 194.0.1.1")

   net.get("R3").cmd("ip route add 195.0.1.0/24 via 193.0.1.2")
   net.get("R2").cmd("ip route add 194.0.1.0/24 via 192.0.1.2")
   net.get("R3").cmd("ip route add 192.0.1.0/24 via 194.0.1.1")
   net.get("R2").cmd("ip route add 193.0.1.0/24 via 195.0.1.2")
```

b.) Traceroute between nodes H1 and H2:



```
mininext> H1 traceroute H2
traceroute to 191.0.1.1 (191.0.1.1), 30 hops max, 60 byte packets
 1  190.0.1.2 (190.0.1.2)  0.034 ms  0.006 ms  0.007 ms
 2  192.0.1.1 (192.0.1.1)  0.016 ms  0.009 ms  0.008 ms
 3  195.0.1.2 (195.0.1.2)  0.019 ms  0.013 ms  0.012 ms
 4  191.0.1.1 (191.0.1.1)  0.022 ms  0.014 ms  0.016 ms
mininext>
```

Part B:

B1.

a.) and b.)

The steps to be followed are below with their explanation:

1. Copy /etc/quagga/debian.conf to each of the configs folder using command cp ‘source’ destination. Don’t change this file.

2. Copy /etc/quagga/daemons to each of the configs folder using command cp ‘source’ destination and ensure it has the following data:

```
zebra =yes
bgpd=no
ospfd=no
ospf6d=no
ripd=yes
ripngd=no
```

This ensures that the nodes are treated as rip routers and we can configure each of them and view their information using zebra config file.

3. Copy /usr/share/doc/quagga/examples/zebra.conf.sample and /usr/share/doc/quagga/examples/ripd.conf.sample to /etc/quagga as well as each of the configs folder of nodes using command ***cp ‘source’ destination***. Do not change zebra.conf.sample file . Add the following data in ripd.conf.sample for each H1,H2,R1,R2,R3,R4:

```
router rip
network a.b.c.d/24
…
…
```

where a.b.c.d is one network interface for the node. We add all network interfaces for each node in ripd.conf.sample (later to be renamed to ripd.conf). I have attached the these files for each node in the configs folder in PartB folder for review.

Rename zebra.conf.sample to zebra.conf and ripd.conf.sample to ripd.conf respectively.

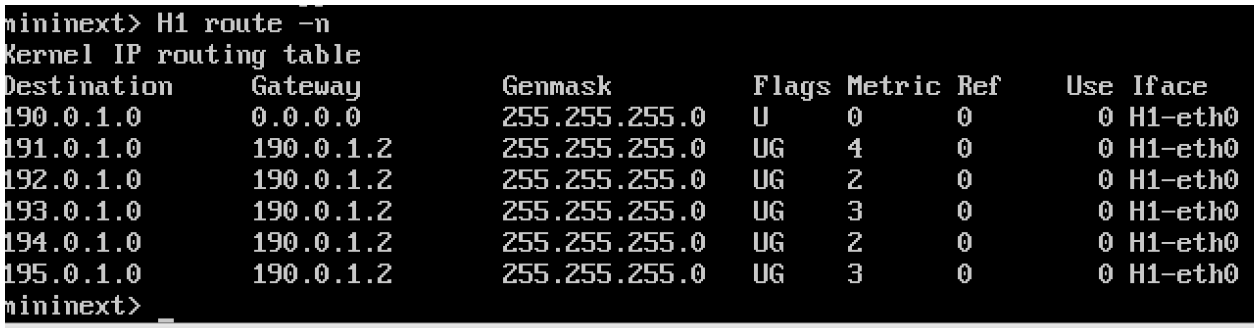
We add these lines to ensure that every interface is available to be viewed when dynamic routing finds path between each node till it finally converges.

B2.

a.) ROUTING TABLES:

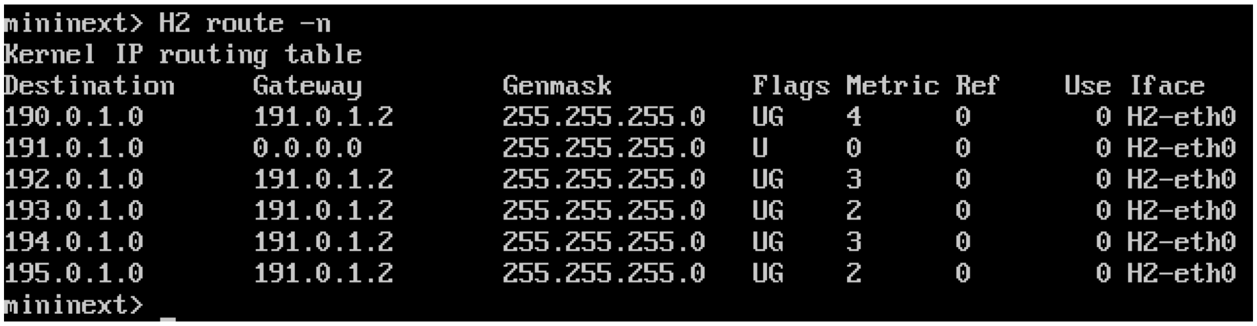
i.) Kernel routing table at each node:

H1.



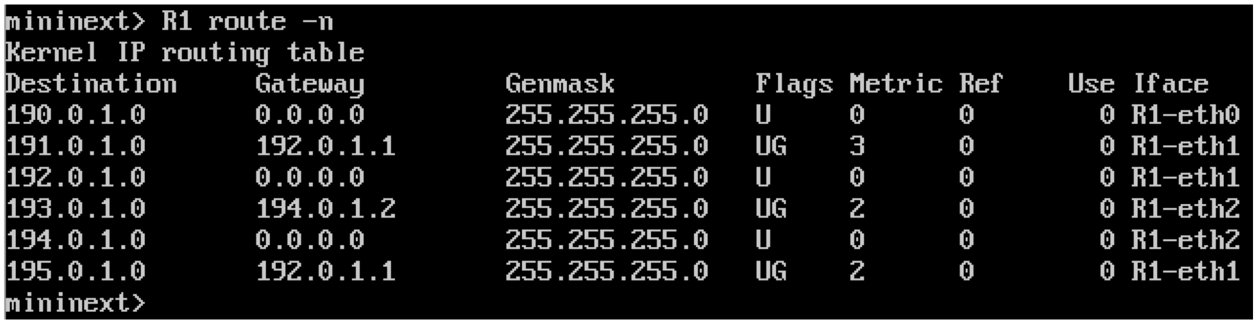
```
mininext> H1 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
190.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 H1-eth0
191.0.1.0       190.0.1.2       255.255.255.0   UG    4      0        0 H1-eth0
192.0.1.0       190.0.1.2       255.255.255.0   UG    2      0        0 H1-eth0
193.0.1.0       190.0.1.2       255.255.255.0   UG    3      0        0 H1-eth0
194.0.1.0       190.0.1.2       255.255.255.0   UG    2      0        0 H1-eth0
195.0.1.0       190.0.1.2       255.255.255.0   UG    3      0        0 H1-eth0
mininext> _
```

H2.



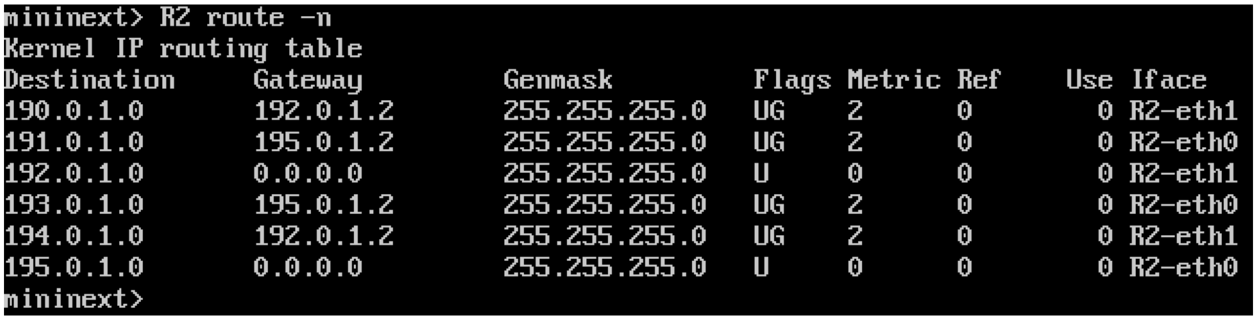
```
mininext> H2 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
190.0.1.0       191.0.1.2       255.255.255.0   UG    4      0        0 H2-eth0
191.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 H2-eth0
192.0.1.0       191.0.1.2       255.255.255.0   UG    3      0        0 H2-eth0
193.0.1.0       191.0.1.2       255.255.255.0   UG    2      0        0 H2-eth0
194.0.1.0       191.0.1.2       255.255.255.0   UG    3      0        0 H2-eth0
195.0.1.0       191.0.1.2       255.255.255.0   UG    2      0        0 H2-eth0
mininext> _
```

R1.



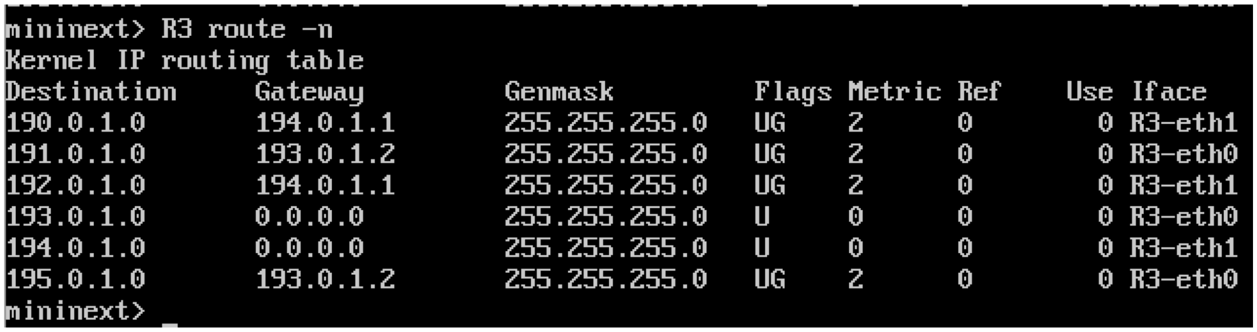
```
mininext> R1 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
190.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 R1-eth0
191.0.1.0       192.0.1.1       255.255.255.0   UG    3      0        0 R1-eth1
192.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 R1-eth1
193.0.1.0       194.0.1.2       255.255.255.0   UG    2      0        0 R1-eth2
194.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 R1-eth2
195.0.1.0       192.0.1.1       255.255.255.0   UG    2      0        0 R1-eth1
mininext>
```

R2.



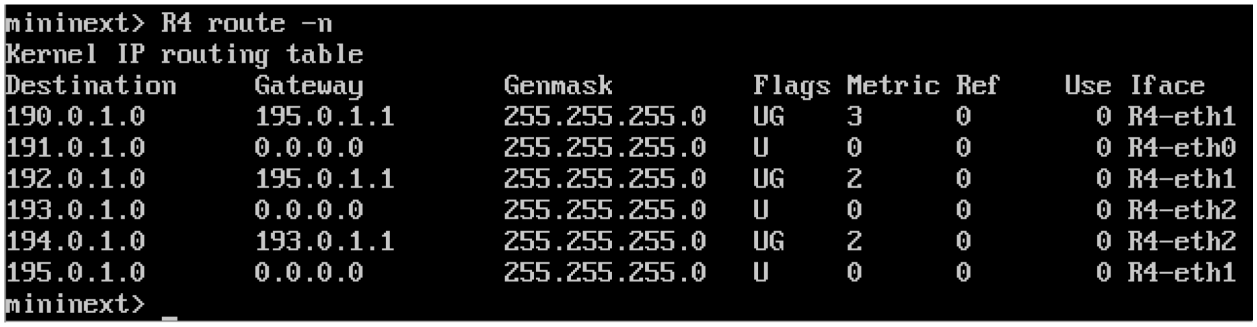
```
mininext> R2 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
190.0.1.0       192.0.1.2       255.255.255.0   UG    2      0        0 R2-eth1
191.0.1.0       195.0.1.2       255.255.255.0   UG    2      0        0 R2-eth0
192.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 R2-eth1
193.0.1.0       195.0.1.2       255.255.255.0   UG    2      0        0 R2-eth0
194.0.1.0       192.0.1.2       255.255.255.0   UG    2      0        0 R2-eth1
195.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 R2-eth0
mininext>
```

R3.



```
mininext> R3 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
190.0.1.0       194.0.1.1       255.255.255.0   UG    2      0        0 R3-eth1
191.0.1.0       193.0.1.2       255.255.255.0   UG    2      0        0 R3-eth0
192.0.1.0       194.0.1.1       255.255.255.0   UG    2      0        0 R3-eth1
193.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 R3-eth0
194.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 R3-eth1
195.0.1.0       193.0.1.2       255.255.255.0   UG    2      0        0 R3-eth0
mininext> _
```

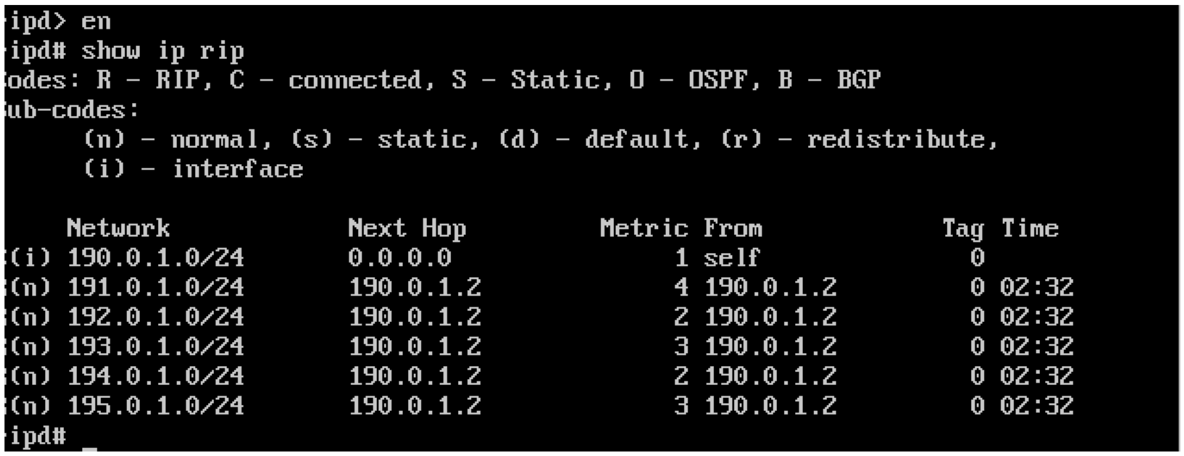
R4.



```
mininext> R4 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
190.0.1.0       195.0.1.1       255.255.255.0   UG    3      0        0 R4-eth1
191.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 R4-eth0
192.0.1.0       195.0.1.1       255.255.255.0   UG    2      0        0 R4-eth1
193.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 R4-eth2
194.0.1.0       193.0.1.1       255.255.255.0   UG    2      0        0 R4-eth2
195.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 R4-eth1
mininext> _
```

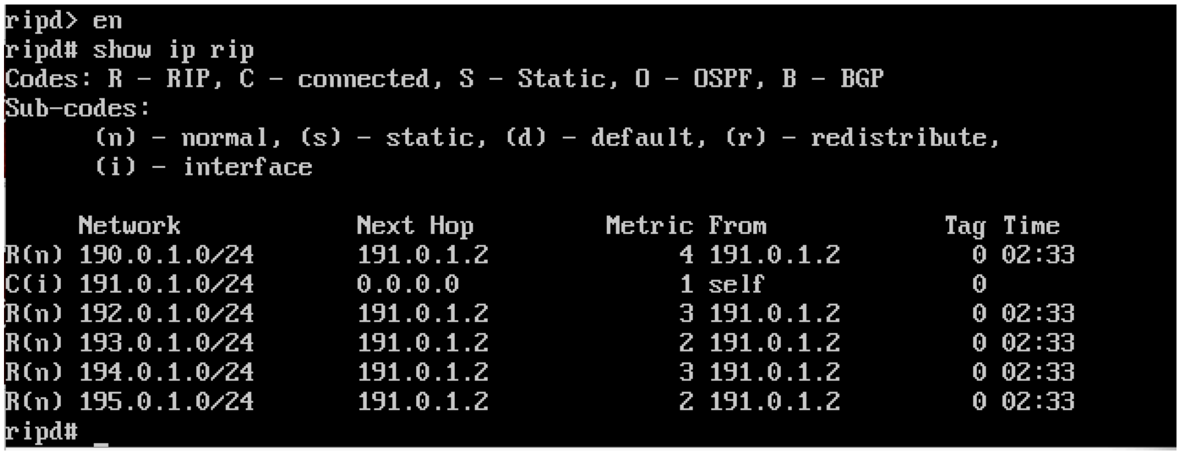
ii. Quagga routing table at each node:

H1.



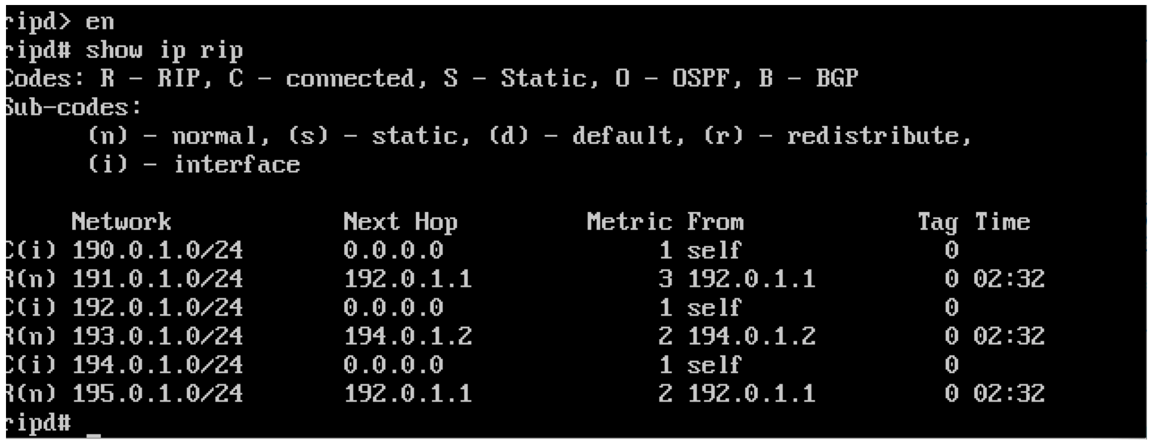
```
ipd> en
ipd# show ip rip
odes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
ub-codes:
     (n) - normal, (s) - static, (d) - default, (r) - redistribute,
     (i) - interface

    Network            Next Hop         Metric From            Tag Time
(i) 190.0.1.0/24       0.0.0.0               1 self              0
(n) 191.0.1.0/24       190.0.1.2             4 190.0.1.2         0 02:32
(n) 192.0.1.0/24       190.0.1.2             2 190.0.1.2         0 02:32
(n) 193.0.1.0/24       190.0.1.2             3 190.0.1.2         0 02:32
(n) 194.0.1.0/24       190.0.1.2             2 190.0.1.2         0 02:32
(n) 195.0.1.0/24       190.0.1.2             3 190.0.1.2         0 02:32
ipd# _
```

H2.



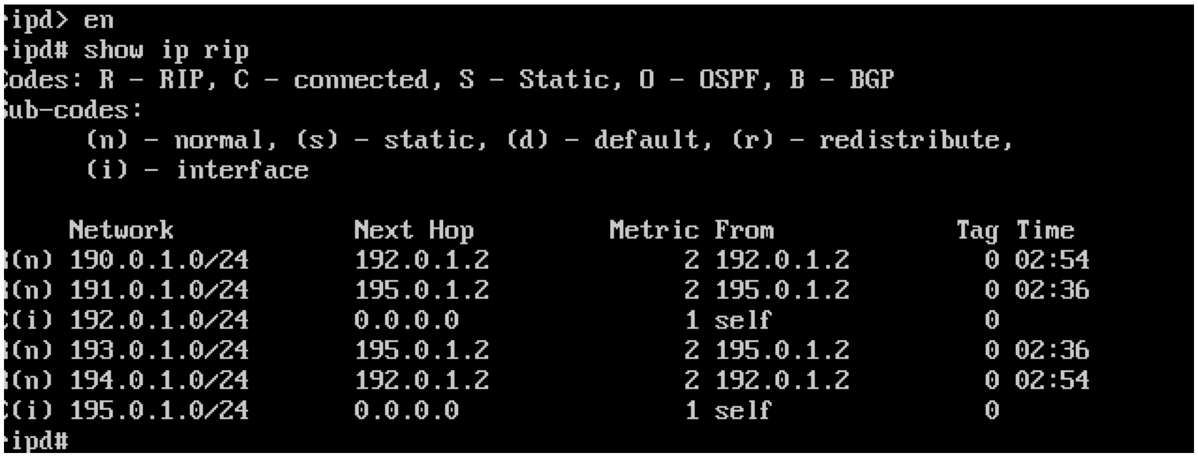
```
ripd> en
ripd# show ip rip
Codes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
Sub-codes:
      (n) - normal, (s) - static, (d) - default, (r) - redistribute,
      (i) - interface

     Network            Next Hop         Metric From            Tag Time
R(n) 190.0.1.0/24       191.0.1.2             4 191.0.1.2         0 02:33
C(i) 191.0.1.0/24       0.0.0.0               1 self              0
R(n) 192.0.1.0/24       191.0.1.2             3 191.0.1.2         0 02:33
R(n) 193.0.1.0/24       191.0.1.2             2 191.0.1.2         0 02:33
R(n) 194.0.1.0/24       191.0.1.2             3 191.0.1.2         0 02:33
R(n) 195.0.1.0/24       191.0.1.2             2 191.0.1.2         0 02:33
ripd# _
```

R1.



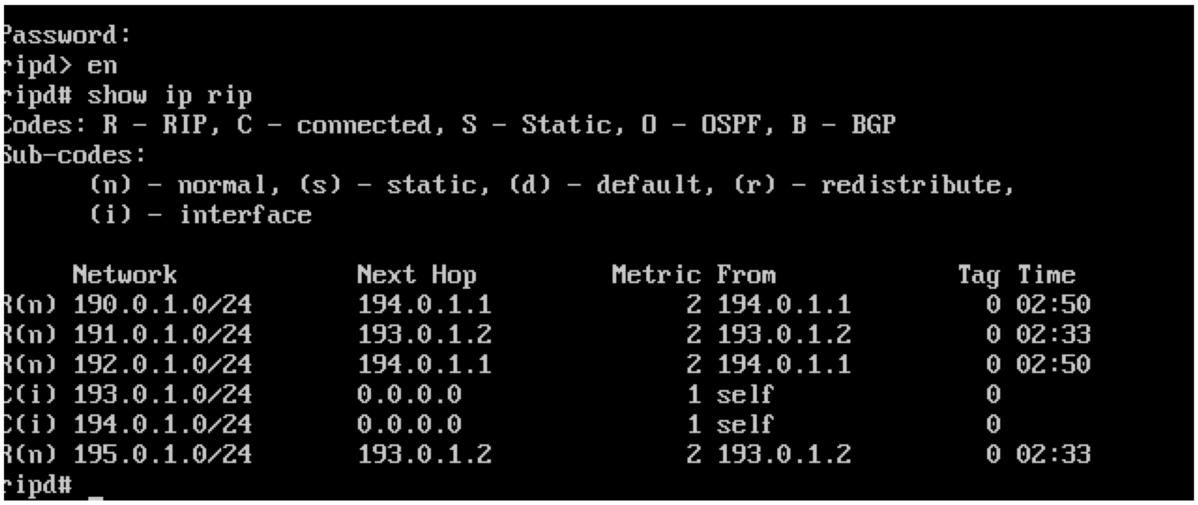
```
ipd> en
ipd# show ip rip
Codes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
Sub-codes:
      (n) - normal, (s) - static, (d) - default, (r) - redistribute,
      (i) - interface

     Network            Next Hop         Metric From            Tag Time
C(i) 190.0.1.0/24       0.0.0.0               1 self              0
R(n) 191.0.1.0/24       192.0.1.1             3 192.0.1.1         0 02:32
C(i) 192.0.1.0/24       0.0.0.0               1 self              0
R(n) 193.0.1.0/24       194.0.1.2             2 194.0.1.2         0 02:32
C(i) 194.0.1.0/24       0.0.0.0               1 self              0
R(n) 195.0.1.0/24       192.0.1.1             2 192.0.1.1         0 02:32
ipd# _
```

R2.



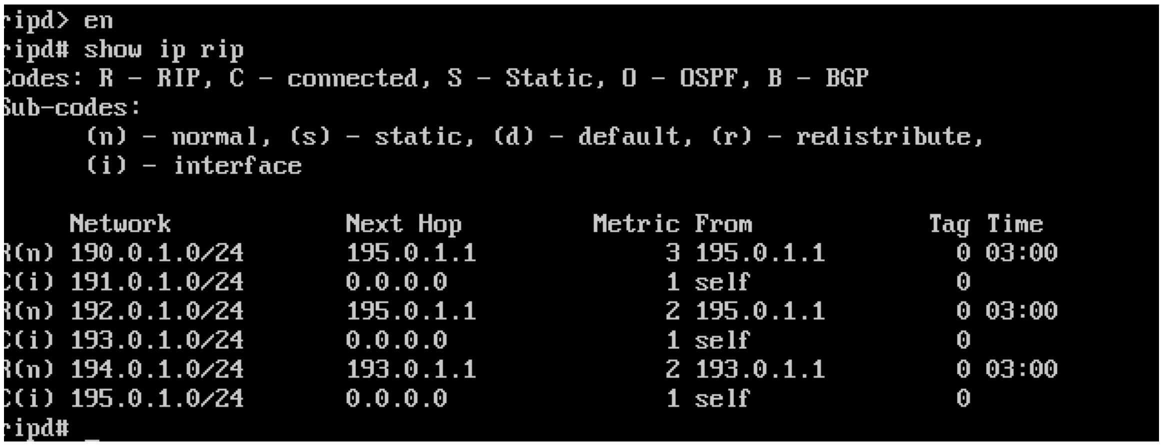
```
ipd> en
ipd# show ip rip
odes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
ub-codes:
      (n) - normal, (s) - static, (d) - default, (r) - redistribute,
      (i) - interface

     Network            Next Hop         Metric From            Tag Time
(n) 190.0.1.0/24       192.0.1.2             2 192.0.1.2         0 02:54
(n) 191.0.1.0/24       195.0.1.2             2 195.0.1.2         0 02:36
(i) 192.0.1.0/24       0.0.0.0               1 self              0
(n) 193.0.1.0/24       195.0.1.2             2 195.0.1.2         0 02:36
(n) 194.0.1.0/24       192.0.1.2             2 192.0.1.2         0 02:54
(i) 195.0.1.0/24       0.0.0.0               1 self              0
ipd#
```

R3.



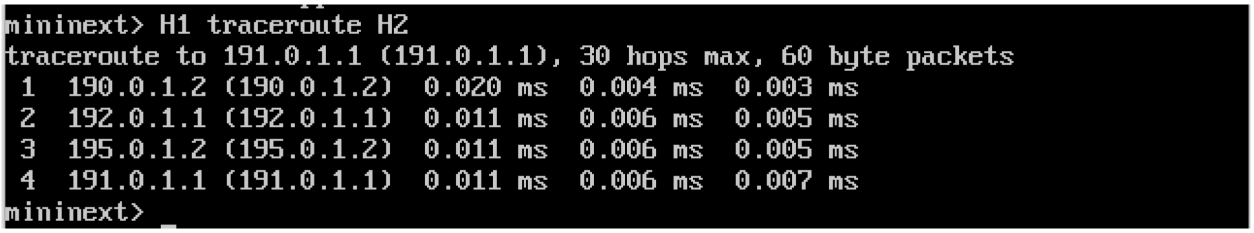
```
assword:
ripd> en
ripd# show ip rip
odes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
ub-codes:
      (n) - normal, (s) - static, (d) - default, (r) - redistribute,
      (i) - interface

     Network            Next Hop         Metric From            Tag Time
R(n) 190.0.1.0/24       194.0.1.1            2 194.0.1.1         0 02:50
R(n) 191.0.1.0/24       193.0.1.2            2 193.0.1.2         0 02:33
R(n) 192.0.1.0/24       194.0.1.1            2 194.0.1.1         0 02:50
C(i) 193.0.1.0/24       0.0.0.0              1 self              0
C(i) 194.0.1.0/24       0.0.0.0              1 self              0
R(n) 195.0.1.0/24       193.0.1.2            2 193.0.1.2         0 02:33
ripd# _
```

R4.



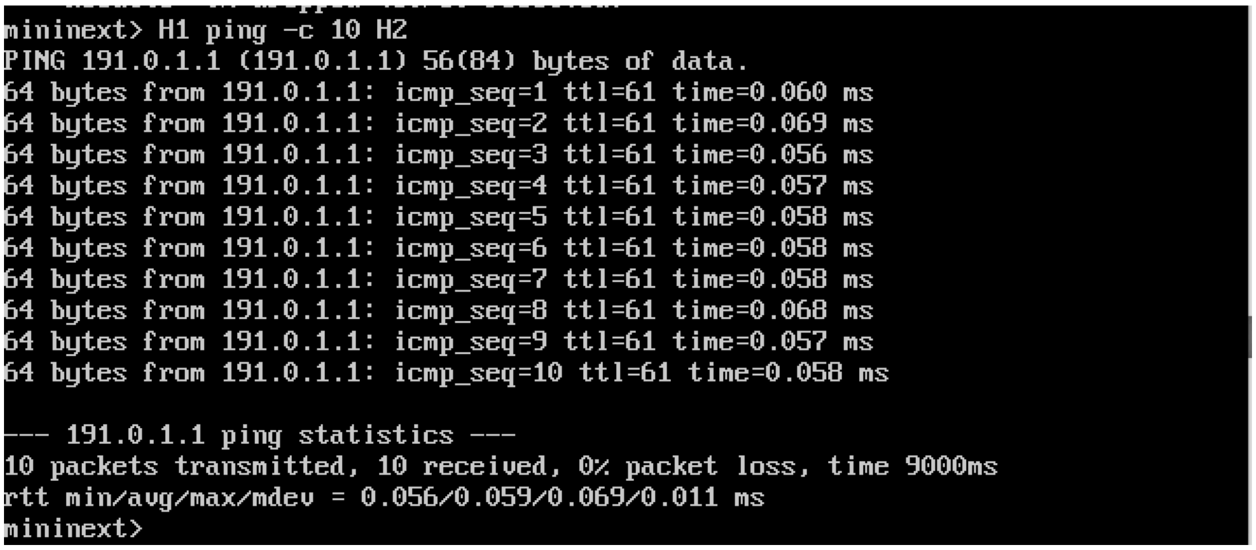
```
ipd> en
ipd# show ip rip
odes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
ub-codes:
      (n) - normal, (s) - static, (d) - default, (r) - redistribute,
      (i) - interface

     Network            Next Hop         Metric From            Tag Time
(n) 190.0.1.0/24       195.0.1.1             3 195.0.1.1         0 03:00
(i) 191.0.1.0/24       0.0.0.0               1 self              0
(n) 192.0.1.0/24       195.0.1.1             2 195.0.1.1         0 03:00
(i) 193.0.1.0/24       0.0.0.0               1 self              0
(n) 194.0.1.0/24       193.0.1.1             2 193.0.1.1         0 03:00
(i) 195.0.1.0/24       0.0.0.0               1 self              0
ipd# _
```

b.) The traceroute output that gives the path between nodes H1 and H2:



```
mininext> H1 traceroute H2
traceroute to 191.0.1.1 (191.0.1.1), 30 hops max, 60 byte packets
 1  190.0.1.2 (190.0.1.2)  0.020 ms  0.004 ms  0.003 ms
 2  192.0.1.1 (192.0.1.1)  0.011 ms  0.006 ms  0.005 ms
 3  195.0.1.2 (195.0.1.2)  0.011 ms  0.006 ms  0.005 ms
 4  191.0.1.1 (191.0.1.1)  0.011 ms  0.006 ms  0.007 ms
mininext> _
```

c.) Average time taken for ping was 0.059 ms.



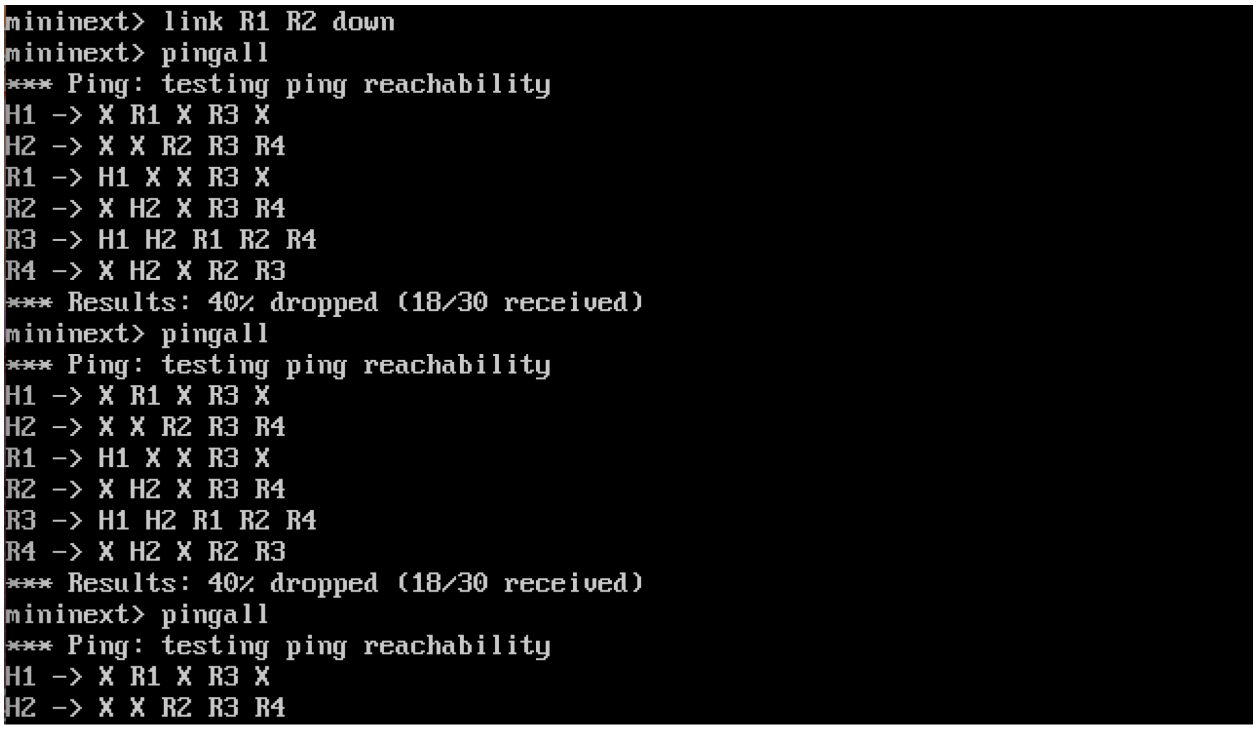
```
mininext> H1 ping -c 10 H2
PING 191.0.1.1 (191.0.1.1) 56(84) bytes of data.
64 bytes from 191.0.1.1: icmp_seq=1 ttl=61 time=0.060 ms
64 bytes from 191.0.1.1: icmp_seq=2 ttl=61 time=0.069 ms
64 bytes from 191.0.1.1: icmp_seq=3 ttl=61 time=0.056 ms
64 bytes from 191.0.1.1: icmp_seq=4 ttl=61 time=0.057 ms
64 bytes from 191.0.1.1: icmp_seq=5 ttl=61 time=0.058 ms
64 bytes from 191.0.1.1: icmp_seq=6 ttl=61 time=0.058 ms
64 bytes from 191.0.1.1: icmp_seq=7 ttl=61 time=0.058 ms
64 bytes from 191.0.1.1: icmp_seq=8 ttl=61 time=0.068 ms
64 bytes from 191.0.1.1: icmp_seq=9 ttl=61 time=0.057 ms
64 bytes from 191.0.1.1: icmp_seq=10 ttl=61 time=0.058 ms

--- 191.0.1.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9000ms
rtt min/avg/max/mdev = 0.056/0.059/0.069/0.011 ms
mininext>
```

d.) Convergence time as measured from stopwatch was 48 seconds. It was the time since the topology first established till the time all nodes were connected with optimal route between each of them.

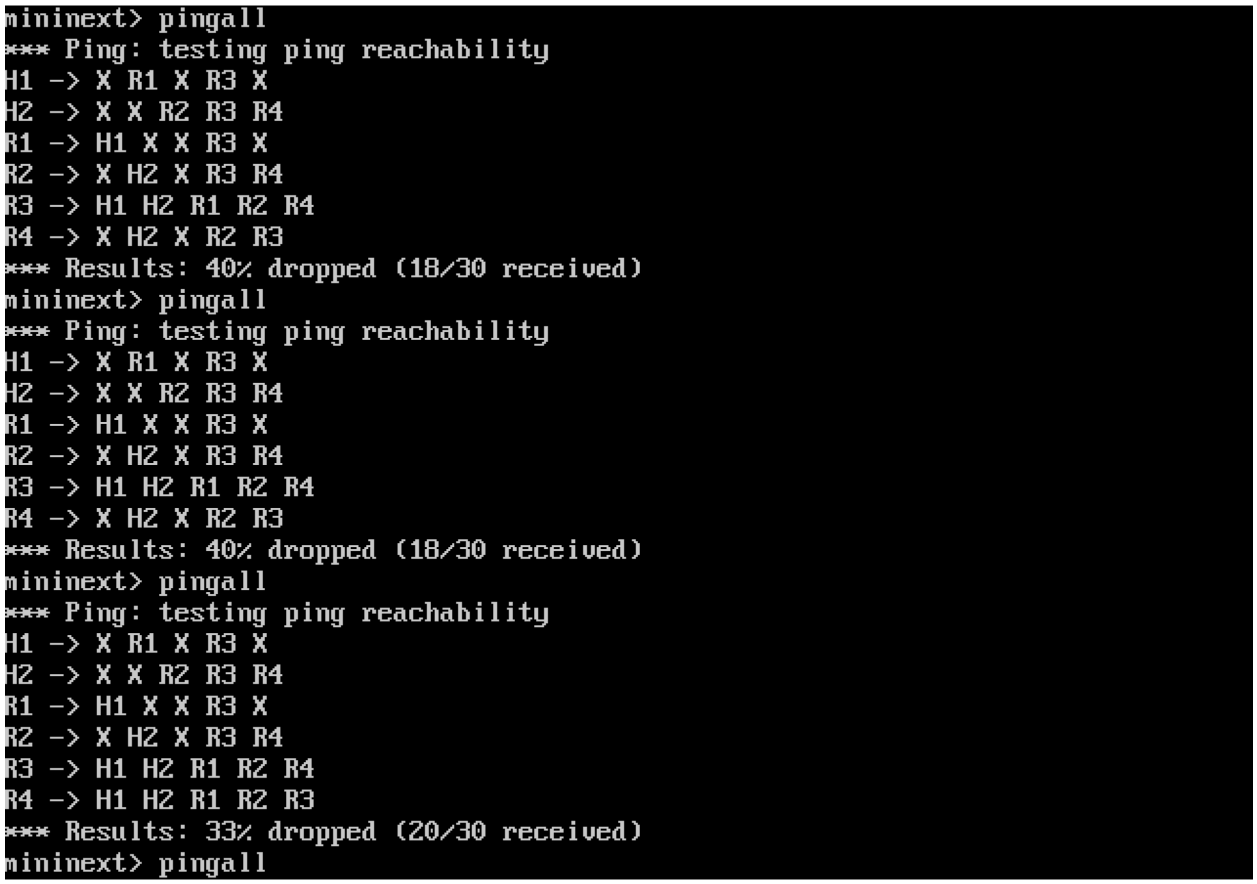
B3.) Making link down: ***(R1-R2) (Since current path contains R1-R2 link)***

a.) The link was taken down by command ***link R1 R2 down***.

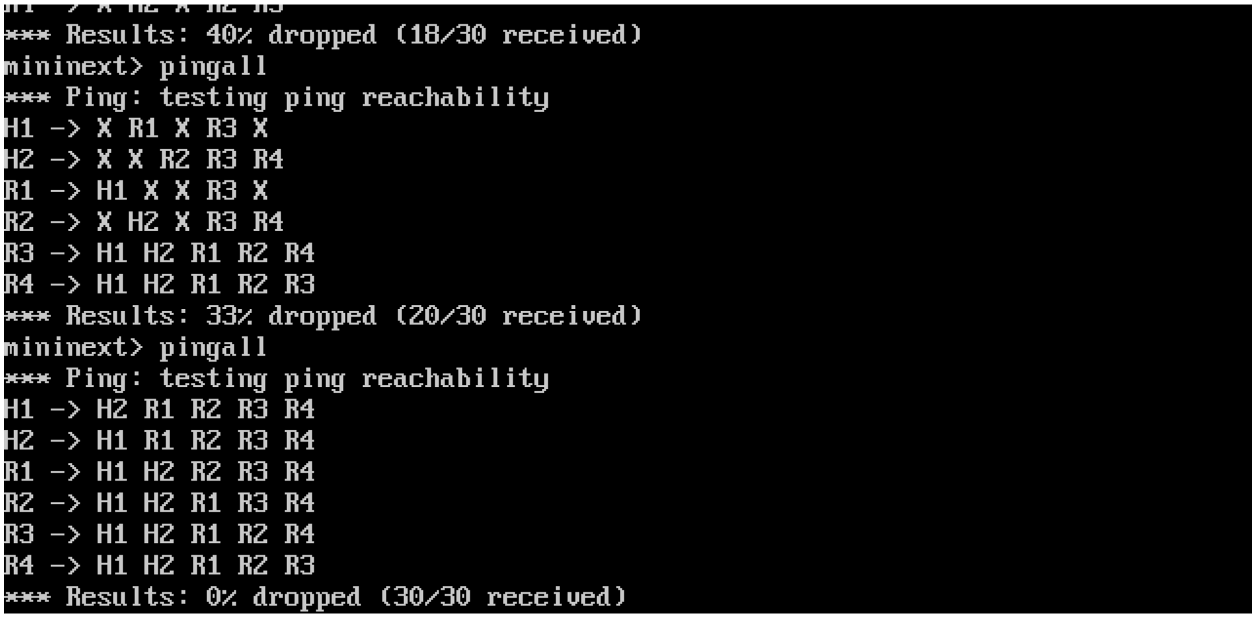
b.) Time taken for the connectivity to reestablish was 6 seconds as measured from timer. Following are the screenshots since the taking down of link till the connectivity was reestablished:



```
mininext> link R1 R2 down
mininext> pingall
*** Ping: testing ping reachability
H1 -> X R1 X R3 X
H2 -> X X R2 R3 R4
R1 -> H1 X X R3 X
R2 -> X H2 X R3 R4
R3 -> H1 H2 R1 R2 R4
R4 -> X H2 X R2 R3
*** Results: 40% dropped (18/30 received)
mininext> pingall
*** Ping: testing ping reachability
H1 -> X R1 X R3 X
H2 -> X X R2 R3 R4
R1 -> H1 X X R3 X
R2 -> X H2 X R3 R4
R3 -> H1 H2 R1 R2 R4
R4 -> X H2 X R2 R3
*** Results: 40% dropped (18/30 received)
mininext> pingall
*** Ping: testing ping reachability
H1 -> X R1 X R3 X
H2 -> X X R2 R3 R4
```



```
mininext> pingall
*** Ping: testing ping reachability
H1 -> X R1 X R3 X
H2 -> X X R2 R3 R4
R1 -> H1 X X R3 X
R2 -> X H2 X R3 R4
R3 -> H1 H2 R1 R2 R4
R4 -> X H2 X R2 R3
*** Results: 40% dropped (18/30 received)
mininext> pingall
*** Ping: testing ping reachability
H1 -> X R1 X R3 X
H2 -> X X R2 R3 R4
R1 -> H1 X X R3 X
R2 -> X H2 X R3 R4
R3 -> H1 H2 R1 R2 R4
R4 -> X H2 X R2 R3
*** Results: 40% dropped (18/30 received)
mininext> pingall
*** Ping: testing ping reachability
H1 -> X R1 X R3 X
H2 -> X X R2 R3 R4
R1 -> H1 X X R3 X
R2 -> X H2 X R3 R4
R3 -> H1 H2 R1 R2 R4
R4 -> H1 H2 R1 R2 R3
*** Results: 33% dropped (20/30 received)
mininext> pingall
```

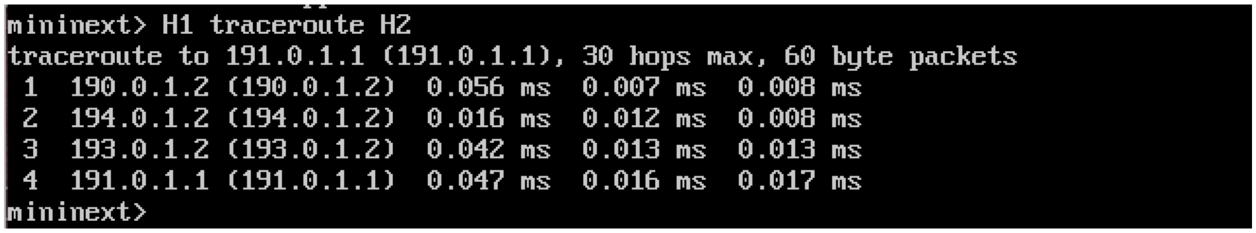


```
*** Results: 40% dropped (18/30 received)
mininext> pingall
*** Ping: testing ping reachability
H1 -> X R1 X R3 X
H2 -> X X R2 R3 R4
R1 -> H1 X X R3 X
R2 -> X H2 X R3 R4
R3 -> H1 H2 R1 R2 R4
R4 -> H1 H2 R1 R2 R3
*** Results: 33% dropped (20/30 received)
mininext> pingall
*** Ping: testing ping reachability
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)
```

In all, it took 6 seconds for the connectivity to re-establish (30/30 packets received)

b.) Traceroute output between nodes H1 and H2:

As we can see, the new path between H1 and H2 goes via R1-R3 link because R1-R2 link is down.



```
mininext> H1 traceroute H2
traceroute to 191.0.1.1 (191.0.1.1), 30 hops max, 60 byte packets
 1  190.0.1.2 (190.0.1.2)  0.056 ms  0.007 ms  0.008 ms
 2  194.0.1.2 (194.0.1.2)  0.016 ms  0.012 ms  0.008 ms
 3  193.0.1.2 (193.0.1.2)  0.042 ms  0.013 ms  0.013 ms
 4  191.0.1.1 (191.0.1.1)  0.047 ms  0.016 ms  0.017 ms
mininext>
```

Part C:

A client server model has been implemented. First, servers per node get created which then interact with the clients at neighboring nodes. The client class at each node accepts the distance vector sent by server and runs Bellman ford algorithm on it to get new values. The Server class continuously checks this file and when there is an update, sends the distance vector to its neighbors.

To run:

Go inside folder PartC using terminal and type:

```
sudo python start.py
```

to create topology.

Then : type following:

```
mininext> source code.sh
```

C1.

a. The routing protocol code has been provided in folder.

b. The time taken to converge is 8 seconds (using stop watch).

c. The routing table at each node can be viewed in the Part C folder.

C2.

When the weight of link R1-R3 changed from 6 to 1,

a.) the network size reduced and time taken to converge reduced to 3 seconds. (Timer to calculate time.)
b.) The routing table had a similar structure as the case when R1-R3's weight was 6. but times were slightly reduced.

C3.

Our topology has a cycle and so the negative weight will mean a negative cycle in the topology and hence Bellman Ford algorithm will not work in this case.

We can avoid this problem by :

a. checking the distances of nodes initially and if the weight comes out as negative, we can report link failure and come out of the program.
b. We can ignore this negative value by making it 0 or giving it some other value based on the positive weights of nodes.

All the above cases can be implemented in code as check conditions to help in case of negative values.

- END