Assignment 6

Team members: Subhrajit (23210106), Pratyush (23210075)

Github Repo URL: Github_Link

To run the code please follow the instructions given in readme.md

---

Part I:

Question **a) Implementation:**

Topology setup:

We've created three routers: ra, rb, and rc; six hosts: h1, h2, h3, h4, h5, and h6; four switches: s1, s2, s3, and s4.

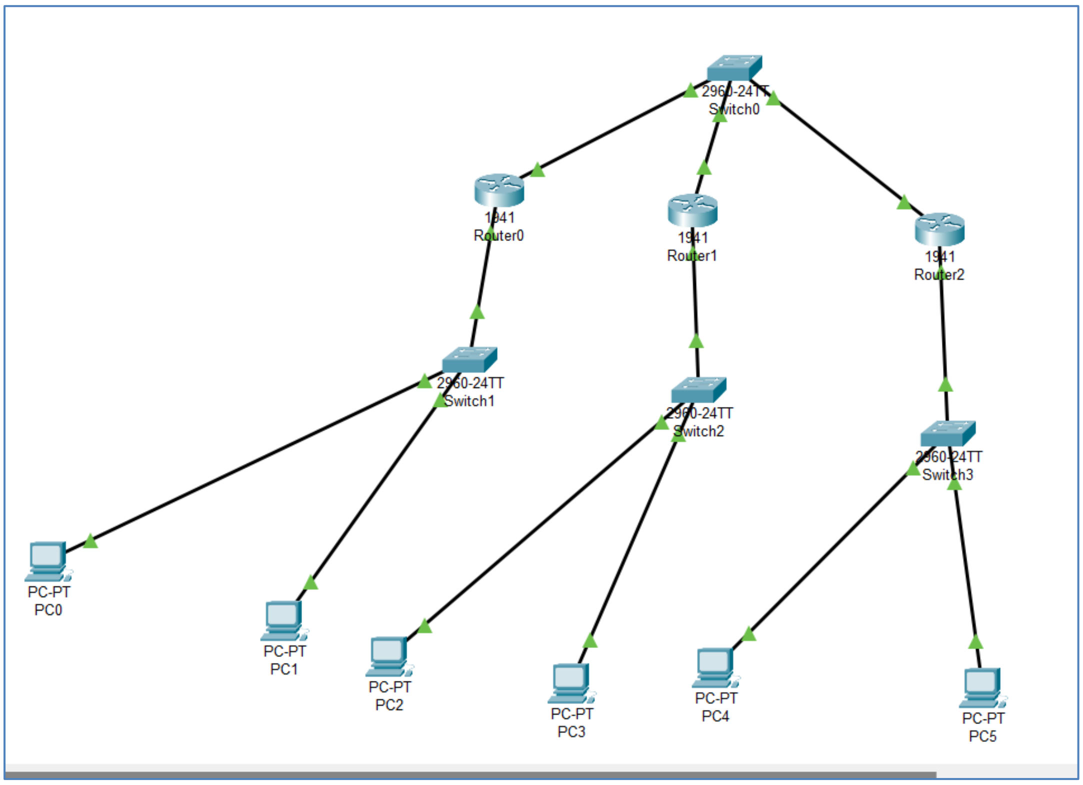
Hosts h1 and h2 are connected to ra via switch s1.

Hosts h3 and h4 are connected to rb via switch s2.

Hosts h5 and h6 are connected to rc via switch s3.

Routers ra, rb, and rc are connected to each other via switch s4.

Topology structure can be visualized like the following:



IP Configurations:

- 192.168.1.0/24 (ra-h1-h2, IP: 192.168.1.{1, 100, 101})
- 192.168.2.0/24 (rb-h3-h4, IP: 192.168.2.{1, 100, 101})
- 192.168.3.0/24 (rc-h5-h6, IP: 192.168.3.{1, 100, 101})
- 10.0.0.0/8 (ra-rb-rc, IP: 10.0.0.{1, 2, 3})

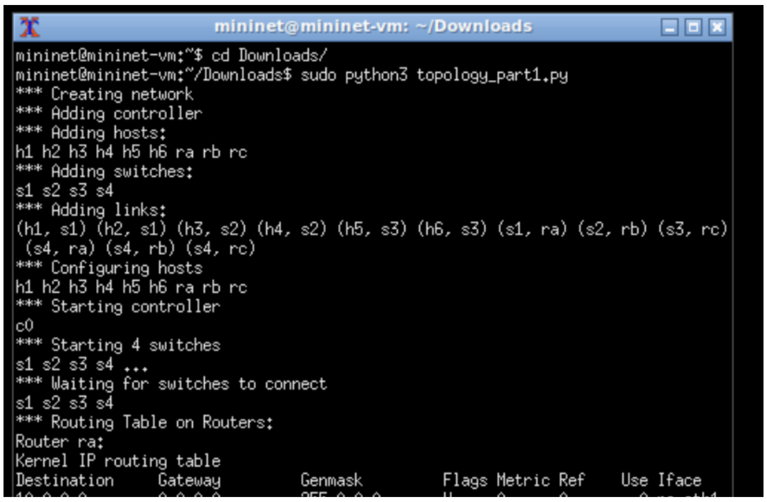
Code: Please see the code for part I implementation here: topology_part1.py

(linked files to follow: readme.md ; CLI_scripts.md)

(Note: the topology_part1.py file contains only the code to create the network topology. We have to run the CLI scripts written in the CLI_scripts.md to execute according to question a, b, and c.)

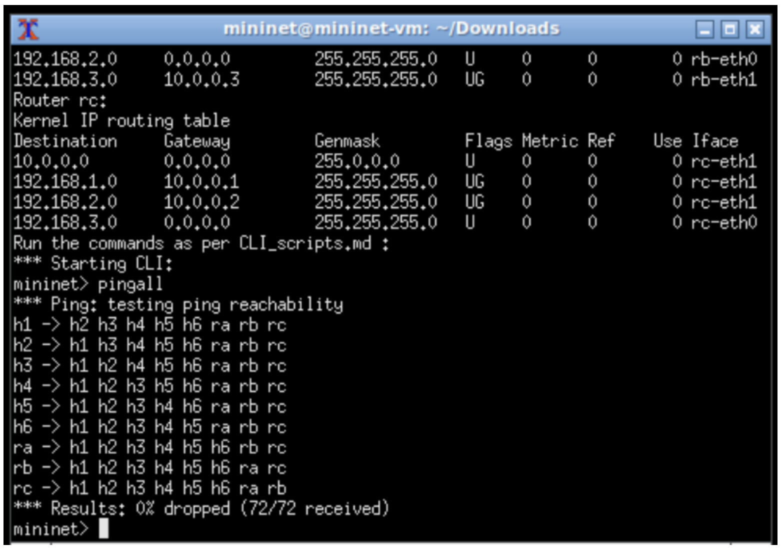
Proof of working of the network topology:

Successful establishment of routers, switches, and hosts:



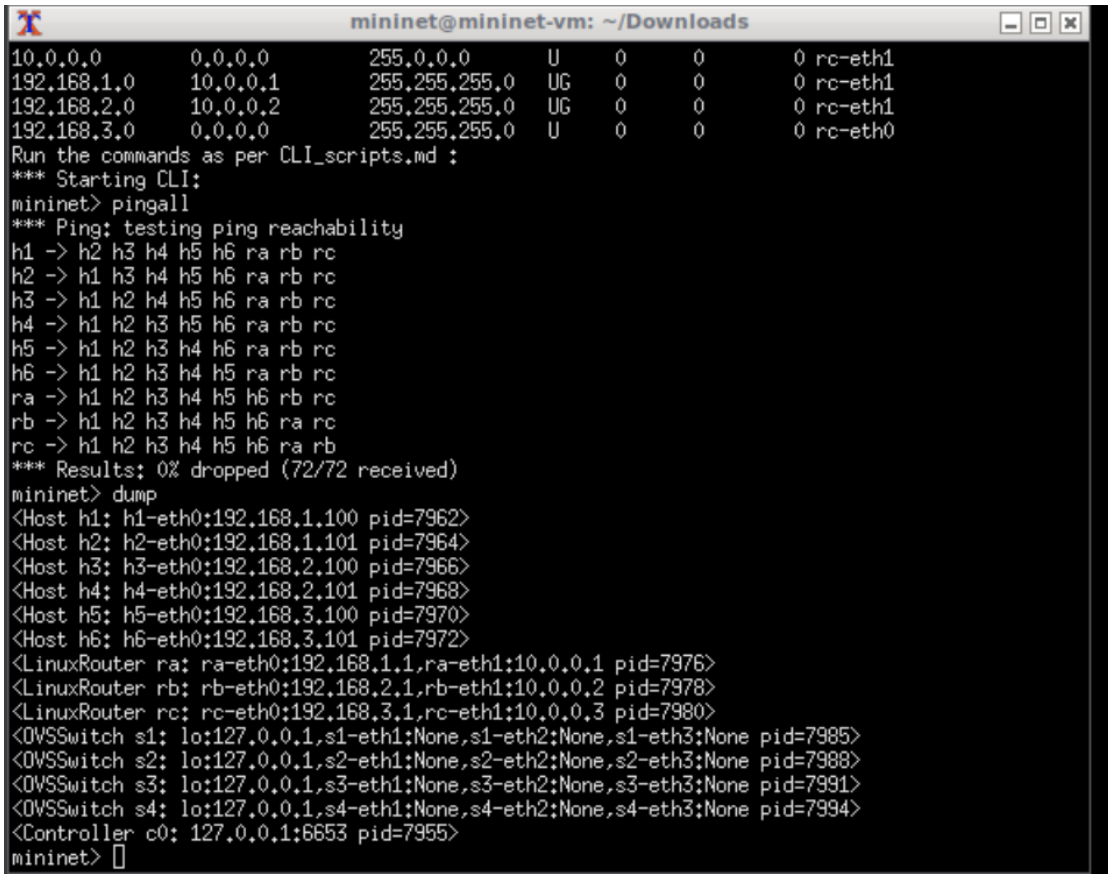
```
mininet@mininet-vm:~$ cd Downloads/
mininet@mininet-vm:~/Downloads$ sudo python3 topology_part1.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 ra rb rc
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(h1, s1) (h2, s1) (h3, s2) (h4, s2) (h5, s3) (h6, s3) (s1, ra) (s2, rb) (s3, rc)
 (s4, ra) (s4, rb) (s4, rc)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 ra rb rc
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Waiting for switches to connect
s1 s2 s3 s4
*** Routing Table on Routers:
Router ra:
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
```

Ping between all the hosts and routers to test if the network works properly:



```
192.168.2.0     0.0.0.0         255.255.255.0   U     0      0        0 rb-eth0
192.168.3.0     10.0.0.3        255.255.255.0   UG    0      0        0 rb-eth1
Router rc:
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.0.0.0       U     0      0        0 rc-eth1
192.168.1.0     10.0.0.1        255.255.255.0   UG    0      0        0 rc-eth1
192.168.2.0     10.0.0.2        255.255.255.0   UG    0      0        0 rc-eth1
192.168.3.0     0.0.0.0         255.255.255.0   U     0      0        0 rc-eth0
Run the commands as per CLI_scripts.md :
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 ra rb rc
h2 -> h1 h3 h4 h5 h6 ra rb rc
h3 -> h1 h2 h4 h5 h6 ra rb rc
h4 -> h1 h2 h3 h5 h6 ra rb rc
h5 -> h1 h2 h3 h4 h6 ra rb rc
h6 -> h1 h2 h3 h4 h5 ra rb rc
ra -> h1 h2 h3 h4 h5 h6 rb rc
rb -> h1 h2 h3 h4 h5 h6 ra rc
rc -> h1 h2 h3 h4 h5 h6 ra rb
*** Results: 0% dropped (72/72 received)
mininet> 
```

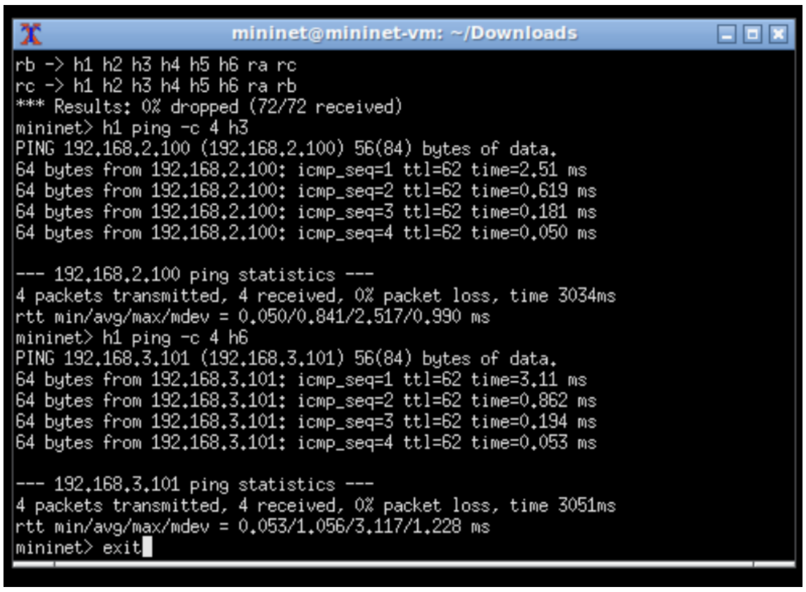
Dumping:


mininet@mininet-vm: ~/Downloads

```
10.0.0.0        0.0.0.0         255.0.0.0       U     0      0        0 rc-eth1
192.168.1.0     10.0.0.1        255.255.255.0   UG    0      0        0 rc-eth1
192.168.2.0     10.0.0.2        255.255.255.0   UG    0      0        0 rc-eth1
192.168.3.0     0.0.0.0         255.255.255.0   U     0      0        0 rc-eth0
Run the commands as per CLI_scripts.md :
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 ra rb rc
h2 -> h1 h3 h4 h5 h6 ra rb rc
h3 -> h1 h2 h4 h5 h6 ra rb rc
h4 -> h1 h2 h3 h5 h6 ra rb rc
h5 -> h1 h2 h3 h4 h6 ra rb rc
h6 -> h1 h2 h3 h4 h5 ra rb rc
ra -> h1 h2 h3 h4 h5 h6 rb rc
rb -> h1 h2 h3 h4 h5 h6 ra rc
rc -> h1 h2 h3 h4 h5 h6 ra rb
*** Results: 0% dropped (72/72 received)
mininet> dump
<Host h1: h1-eth0:192.168.1.100 pid=7962>
<Host h2: h2-eth0:192.168.1.101 pid=7964>
<Host h3: h3-eth0:192.168.2.100 pid=7966>
<Host h4: h4-eth0:192.168.2.101 pid=7968>
<Host h5: h5-eth0:192.168.3.100 pid=7970>
<Host h6: h6-eth0:192.168.3.101 pid=7972>
<LinuxRouter ra: ra-eth0:192.168.1.1,ra-eth1:10.0.0.1 pid=7976>
<LinuxRouter rb: rb-eth0:192.168.2.1,rb-eth1:10.0.0.2 pid=7978>
<LinuxRouter rc: rc-eth0:192.168.3.1,rc-eth1:10.0.0.3 pid=7980>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None pid=7985>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None pid=7988>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None,s3-eth3:None pid=7991>
<OVSSwitch s4: lo:127.0.0.1,s4-eth1:None,s4-eth2:None,s4-eth3:None pid=7994>
<Controller c0: 127.0.0.1:6653 pid=7955>
mininet> 
```

Question **b) Observations:**

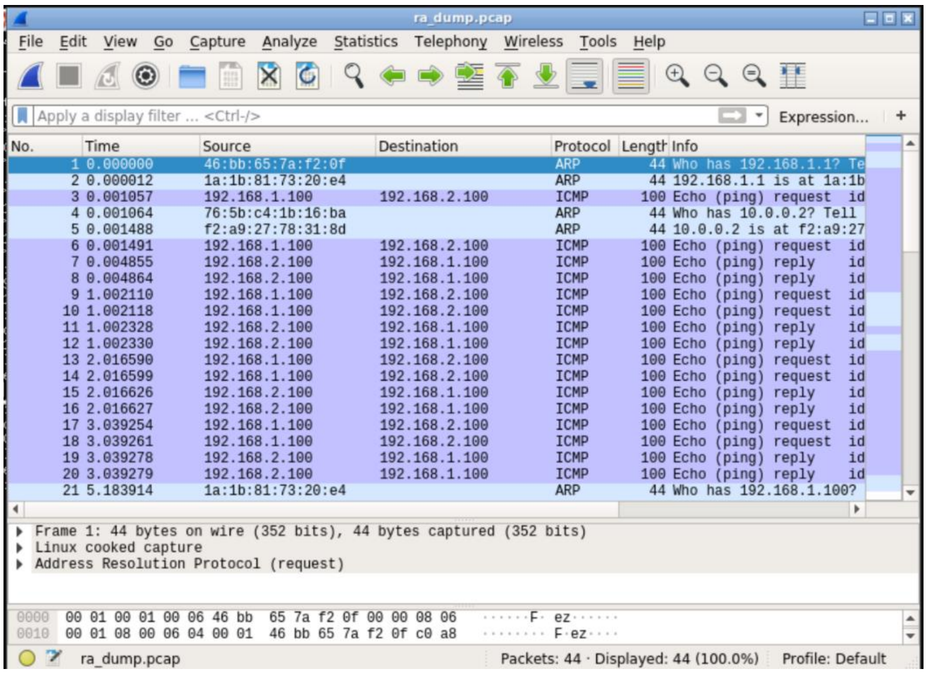
Pinging 4 packets from h1 to h3 and h1 to h6.



```
rb -> h1 h2 h3 h4 h5 h6 ra rc
rc -> h1 h2 h3 h4 h5 h6 ra rb
*** Results: 0% dropped (72/72 received)
mininet> h1 ping -c 4 h3
PING 192.168.2.100 (192.168.2.100) 56(84) bytes of data.
64 bytes from 192.168.2.100: icmp_seq=1 ttl=62 time=2.51 ms
64 bytes from 192.168.2.100: icmp_seq=2 ttl=62 time=0.619 ms
64 bytes from 192.168.2.100: icmp_seq=3 ttl=62 time=0.181 ms
64 bytes from 192.168.2.100: icmp_seq=4 ttl=62 time=0.050 ms

--- 192.168.2.100 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3034ms
rtt min/avg/max/mdev = 0.050/0.841/2.517/0.990 ms
mininet> h1 ping -c 4 h6
PING 192.168.3.101 (192.168.3.101) 56(84) bytes of data.
64 bytes from 192.168.3.101: icmp_seq=1 ttl=62 time=3.11 ms
64 bytes from 192.168.3.101: icmp_seq=2 ttl=62 time=0.862 ms
64 bytes from 192.168.3.101: icmp_seq=3 ttl=62 time=0.194 ms
64 bytes from 192.168.3.101: icmp_seq=4 ttl=62 time=0.053 ms

--- 192.168.3.101 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3051ms
rtt min/avg/max/mdev = 0.053/1.056/3.117/1.228 ms
mininet> exit
```

Opened the tcpdump through wireshark at routers ra, rb and rc to check the route setup for both pings.

ra_dump.pcap:



```
No.  Time       Source              Destination     Protocol Length Info
   1 0.000000   46:bb:65:7a:f2:0f                   ARP      44 Who has 192.168.1.1? Te
   2 0.000012   1a:1b:81:73:20:e4                   ARP      44 192.168.1.1 is at 1a:1b
   3 0.001057   192.168.1.100       192.168.2.100   ICMP    100 Echo (ping) request  id
   4 0.001064   76:5b:c4:1b:16:ba                   ARP      44 Who has 10.0.0.2? Tell
   5 0.001488   f2:a9:27:78:31:8d                   ARP      44 10.0.0.2 is at f2:a9:27
   6 0.001491   192.168.1.100       192.168.2.100   ICMP    100 Echo (ping) request  id
   7 0.004855   192.168.2.100       192.168.1.100   ICMP    100 Echo (ping) reply    id
   8 0.004864   192.168.2.100       192.168.1.100   ICMP    100 Echo (ping) reply    id
   9 1.002110   192.168.1.100       192.168.2.100   ICMP    100 Echo (ping) request  id
  10 1.002118   192.168.1.100       192.168.2.100   ICMP    100 Echo (ping) request  id
  11 1.002328   192.168.2.100       192.168.1.100   ICMP    100 Echo (ping) reply    id
  12 1.002330   192.168.2.100       192.168.1.100   ICMP    100 Echo (ping) reply    id
  13 2.016590   192.168.1.100       192.168.2.100   ICMP    100 Echo (ping) request  id
  14 2.016599   192.168.1.100       192.168.2.100   ICMP    100 Echo (ping) request  id
  15 2.016626   192.168.2.100       192.168.1.100   ICMP    100 Echo (ping) reply    id
  16 2.016627   192.168.2.100       192.168.1.100   ICMP    100 Echo (ping) reply    id
  17 3.039254   192.168.1.100       192.168.2.100   ICMP    100 Echo (ping) request  id
  18 3.039261   192.168.1.100       192.168.2.100   ICMP    100 Echo (ping) request  id
  19 3.039278   192.168.2.100       192.168.1.100   ICMP    100 Echo (ping) reply    id
  20 3.039279   192.168.2.100       192.168.1.100   ICMP    100 Echo (ping) reply    id
  21 5.183914   1a:1b:81:73:20:e4                   ARP      44 Who has 192.168.1.100?

▸ Frame 1: 44 bytes on wire (352 bits), 44 bytes captured (352 bits)
▸ Linux cooked capture
▸ Address Resolution Protocol (request)

0000  00 01 00 01 00 06 46 bb  65 7a f2 0f 00 00 08 06   ······F· ez······
0010  00 01 08 00 06 04 00 01  46 bb 65 7a f2 0f c0 a8   ········ F·ez····

ra_dump.pcap                    Packets: 44 · Displayed: 44 (100.0%)   Profile: Default
```

As we've pinged h1 to h3 first, it starts by ARP for asking 'who has 192.168.1.1?' and gets the ARP reply of MAC address. It then starts looking '10.0.0.2' through ARP. After that it starts the packet transmission from 192.168.1.100 (h1) to 192.168.2.100 (h3) while getting the replies from 192.168.2.100 to 192.168.1.100 via router ra.

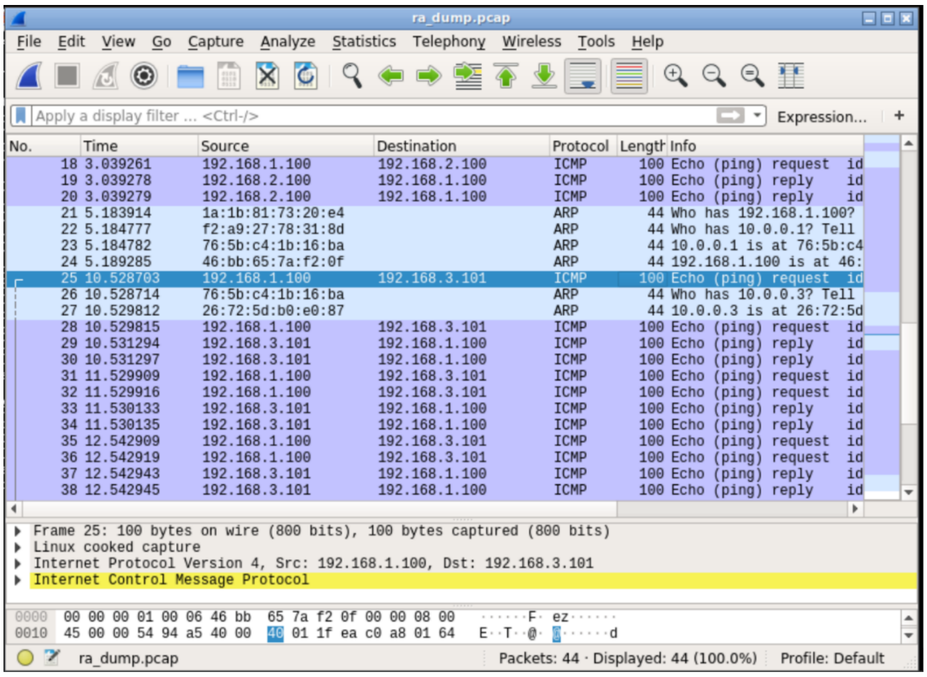
Afterwards for pinging from h1 to h6, we see that in a similar fashion at router ra:

That is,

ICMP packets transmission

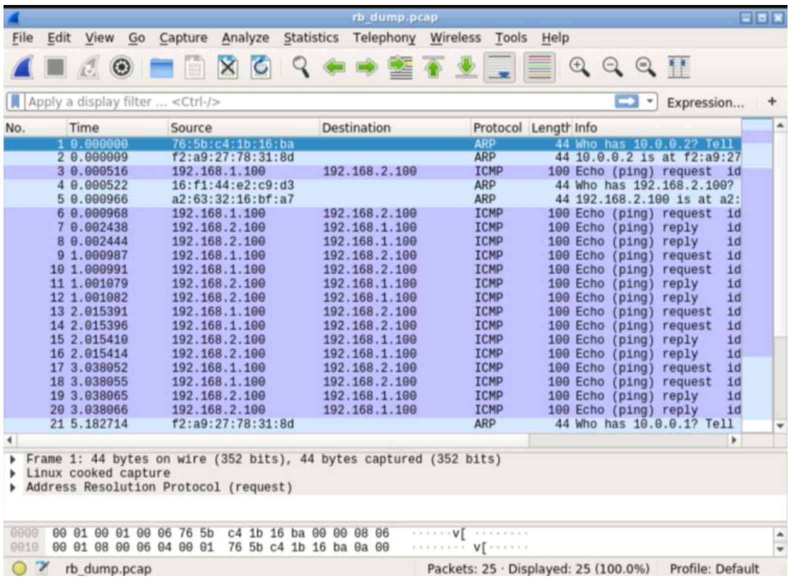
Between 192.168.1.100 and

192.168.3.101

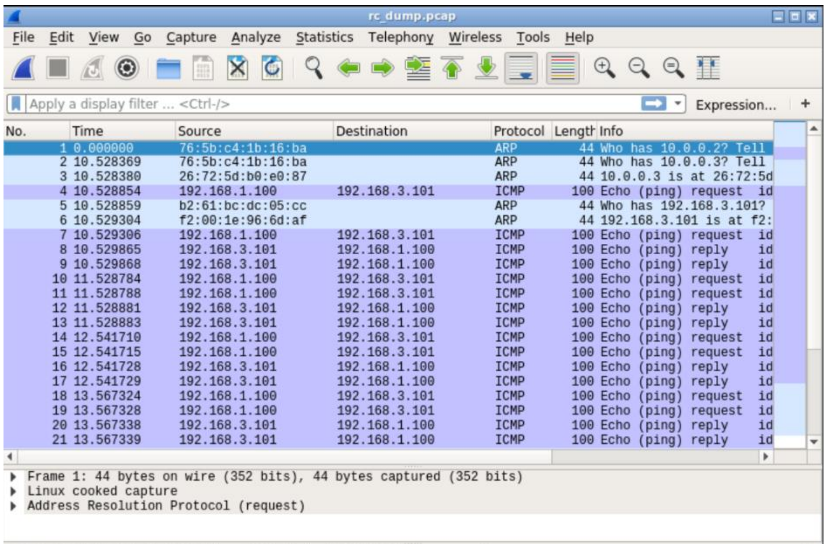


```
ra_dump.pcap
No.  Time       Source             Destination      Protocol Length Info
  18 3.039261   192.168.1.100      192.168.2.100    ICMP     100 Echo (ping) request  id
  19 3.039278   192.168.2.100      192.168.1.100    ICMP     100 Echo (ping) reply    id
  20 3.039279   192.168.2.100      192.168.1.100    ICMP     100 Echo (ping) reply    id
  21 5.183914   1a:1b:81:73:20:e4                   ARP       44 Who has 192.168.1.100?
  22 5.184777   f2:a9:27:78:31:8d                   ARP       44 Who has 10.0.0.1? Tell
  23 5.184782   76:5b:c4:1b:16:ba                   ARP       44 10.0.0.1 is at 76:5b:c4
  24 5.189285   46:bb:65:7a:f2:0f                   ARP       44 192.168.1.100 is at 46:
  25 10.528703  192.168.1.100      192.168.3.101    ICMP     100 Echo (ping) request  id
  26 10.528714  76:5b:c4:1b:16:ba                   ARP       44 Who has 10.0.0.3? Tell
  27 10.529812  26:72:5d:b0:e0:87                   ARP       44 10.0.0.3 is at 26:72:5d
  28 10.529815  192.168.1.100      192.168.3.101    ICMP     100 Echo (ping) request  id
  29 10.531294  192.168.3.101      192.168.1.100    ICMP     100 Echo (ping) reply    id
  30 10.531297  192.168.3.101      192.168.1.100    ICMP     100 Echo (ping) reply    id
  31 11.529909  192.168.1.100      192.168.3.101    ICMP     100 Echo (ping) request  id
  32 11.529916  192.168.1.100      192.168.3.101    ICMP     100 Echo (ping) request  id
  33 11.530133  192.168.3.101      192.168.1.100    ICMP     100 Echo (ping) reply    id
  34 11.530135  192.168.3.101      192.168.1.100    ICMP     100 Echo (ping) reply    id
  35 12.542909  192.168.1.100      192.168.3.101    ICMP     100 Echo (ping) request  id
  36 12.542919  192.168.1.100      192.168.3.101    ICMP     100 Echo (ping) request  id
  37 12.542943  192.168.3.101      192.168.1.100    ICMP     100 Echo (ping) reply    id
  38 12.542945  192.168.3.101      192.168.1.100    ICMP     100 Echo (ping) reply    id

▸ Frame 25: 100 bytes on wire (800 bits), 100 bytes captured (800 bits)
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 192.168.1.100, Dst: 192.168.3.101
▸ Internet Control Message Protocol

0000  00 00 00 01 00 06 46 bb  65 7a f2 0f 00 00 08 00   ······F· ez······
0010  45 00 00 54 94 a5 40 00  40 01 1f ea c0 a8 01 64   E··T··@· @······d

ra_dump.pcap                  Packets: 44 · Displayed: 44 (100.0%)   Profile: Default
```

You may look at the tcpdump at the routers rb and rc here:



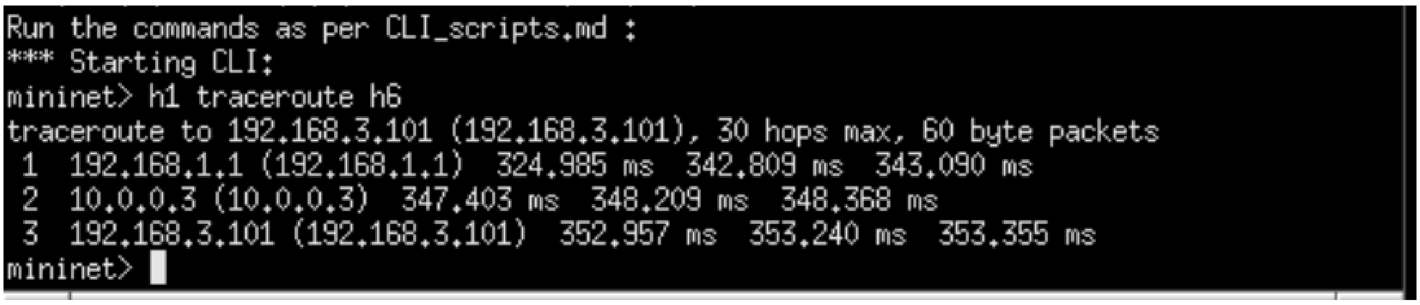
```
rb_dump.pcap
No.  Time       Source             Destination      Protocol Length Info
   1 0.000000   76:5b:c4:1b:16:ba                   ARP       44 Who has 10.0.0.2? Tell
   2 0.000009   f2:a9:27:78:31:8d                   ARP       44 10.0.0.2 is at f2:a9:27
   3 0.000516   192.168.1.100      192.168.2.100    ICMP     100 Echo (ping) request  id
   4 0.000522   16:f1:44:e2:c9:d3                   ARP       44 Who has 192.168.2.100?
   5 0.000966   a2:63:32:16:bf:a7                   ARP       44 192.168.2.100 is at a2:
   6 0.000968   192.168.1.100      192.168.2.100    ICMP     100 Echo (ping) request  id
   7 0.002438   192.168.2.100      192.168.1.100    ICMP     100 Echo (ping) reply    id
   8 0.002444   192.168.2.100      192.168.1.100    ICMP     100 Echo (ping) reply    id
   9 1.000987   192.168.1.100      192.168.2.100    ICMP     100 Echo (ping) request  id
  10 1.000991   192.168.1.100      192.168.2.100    ICMP     100 Echo (ping) request  id
  11 1.001079   192.168.2.100      192.168.1.100    ICMP     100 Echo (ping) reply    id
  12 1.001082   192.168.2.100      192.168.1.100    ICMP     100 Echo (ping) reply    id
  13 2.015391   192.168.1.100      192.168.2.100    ICMP     100 Echo (ping) request  id
  14 2.015396   192.168.1.100      192.168.2.100    ICMP     100 Echo (ping) request  id
  15 2.015410   192.168.2.100      192.168.1.100    ICMP     100 Echo (ping) reply    id
  16 2.015414   192.168.2.100      192.168.1.100    ICMP     100 Echo (ping) reply    id
  17 3.038052   192.168.1.100      192.168.2.100    ICMP     100 Echo (ping) request  id
  18 3.038055   192.168.1.100      192.168.2.100    ICMP     100 Echo (ping) request  id
  19 3.038065   192.168.2.100      192.168.1.100    ICMP     100 Echo (ping) reply    id
  20 3.038066   192.168.2.100      192.168.1.100    ICMP     100 Echo (ping) reply    id
  21 5.182714   f2:a9:27:78:31:8d                   ARP       44 Who has 10.0.0.1? Tell

▸ Frame 1: 44 bytes on wire (352 bits), 44 bytes captured (352 bits)
▸ Linux cooked capture
▸ Address Resolution Protocol (request)

0000  00 01 00 01 00 06 76 5b  c4 1b 16 ba 00 00 08 06   ······v[ ········
0010  00 01 08 00 06 04 00 01  76 5b c4 1b 16 ba 0a 00   ········ v[······

rb_dump.pcap                  Packets: 25 · Displayed: 25 (100.0%)   Profile: Default
```



```
rc_dump.pcap
No.  Time       Source             Destination      Protocol Length Info
   1 0.000000   76:5b:c4:1b:16:ba                   ARP       44 Who has 10.0.0.2? Tell
   2 10.528369  76:5b:c4:1b:16:ba                   ARP       44 Who has 10.0.0.3? Tell
   3 10.528380  26:72:5d:b0:e0:87                   ARP       44 10.0.0.3 is at 26:72:5d
   4 10.528854  192.168.1.100      192.168.3.101    ICMP     100 Echo (ping) request  id
   5 10.528859  b2:61:bc:dc:05:cc                   ARP       44 Who has 192.168.3.101?
   6 10.529304  f2:00:1e:96:6d:af                   ARP       44 192.168.3.101 is at f2:
   7 10.529306  192.168.1.100      192.168.3.101    ICMP     100 Echo (ping) request  id
   8 10.529865  192.168.3.101      192.168.1.100    ICMP     100 Echo (ping) reply    id
   9 10.529868  192.168.3.101      192.168.1.100    ICMP     100 Echo (ping) reply    id
  10 11.528784  192.168.1.100      192.168.3.101    ICMP     100 Echo (ping) request  id
  11 11.528788  192.168.1.100      192.168.3.101    ICMP     100 Echo (ping) request  id
  12 11.528881  192.168.3.101      192.168.1.100    ICMP     100 Echo (ping) reply    id
  13 11.528883  192.168.3.101      192.168.1.100    ICMP     100 Echo (ping) reply    id
  14 12.541710  192.168.1.100      192.168.3.101    ICMP     100 Echo (ping) request  id
  15 12.541715  192.168.1.100      192.168.3.101    ICMP     100 Echo (ping) request  id
  16 12.541728  192.168.3.101      192.168.1.100    ICMP     100 Echo (ping) reply    id
  17 12.541729  192.168.3.101      192.168.1.100    ICMP     100 Echo (ping) reply    id
  18 13.567324  192.168.1.100      192.168.3.101    ICMP     100 Echo (ping) request  id
  19 13.567328  192.168.1.100      192.168.3.101    ICMP     100 Echo (ping) request  id
  20 13.567338  192.168.3.101      192.168.1.100    ICMP     100 Echo (ping) reply    id
  21 13.567339  192.168.3.101      192.168.1.100    ICMP     100 Echo (ping) reply    id

▸ Frame 1: 44 bytes on wire (352 bits), 44 bytes captured (352 bits)
▸ Linux cooked capture
▸ Address Resolution Protocol (request)
```

Question **c) Comparing latency differences between default and changed route.**

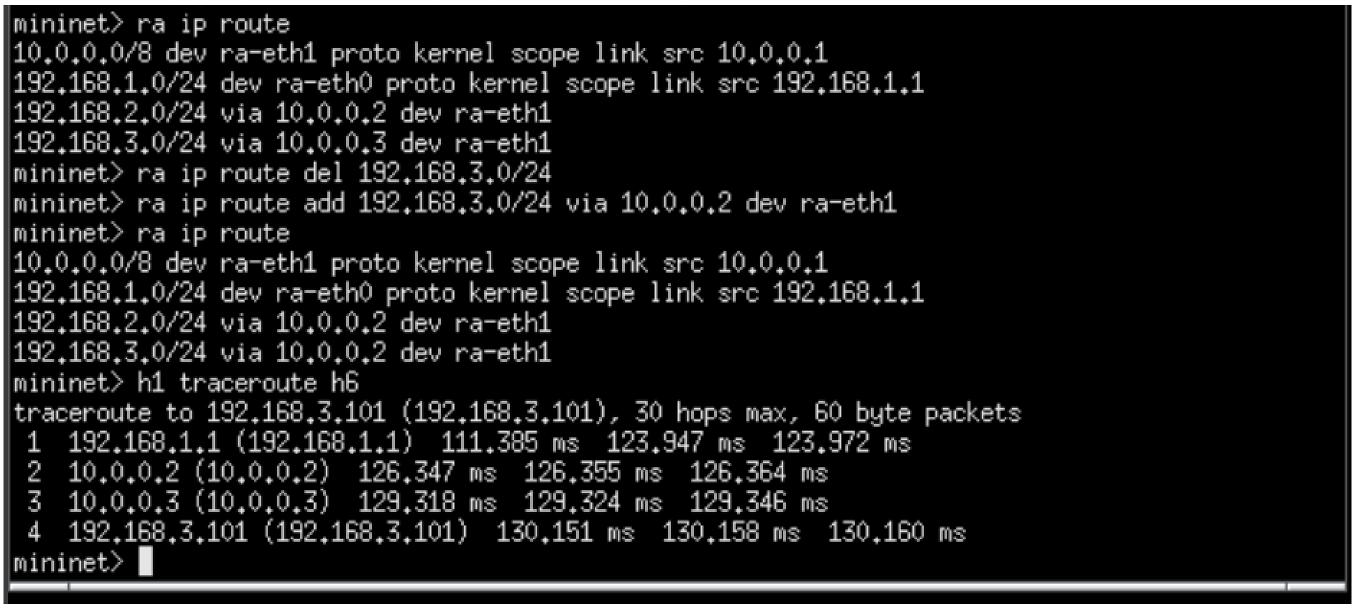
With the default route:



```
Run the commands as per CLI_scripts.md :
*** Starting CLI:
mininet> h1 traceroute h6
traceroute to 192.168.3.101 (192.168.3.101), 30 hops max, 60 byte packets
 1  192.168.1.1 (192.168.1.1)  324.985 ms  342.809 ms  343.090 ms
 2  10.0.0.3 (10.0.0.3)  347.403 ms  348.209 ms  348.368 ms
 3  192.168.3.101 (192.168.3.101)  352.957 ms  353.240 ms  353.355 ms
mininet> █
```

As we can see the route for h1 (192.168.1.101) to h6 (192.168.3.101) goes directly via 10.0.0.3 (rc). That is, it takes the path from h1→ra→rc→h6

After we change the route of h1 to h6 via rb:

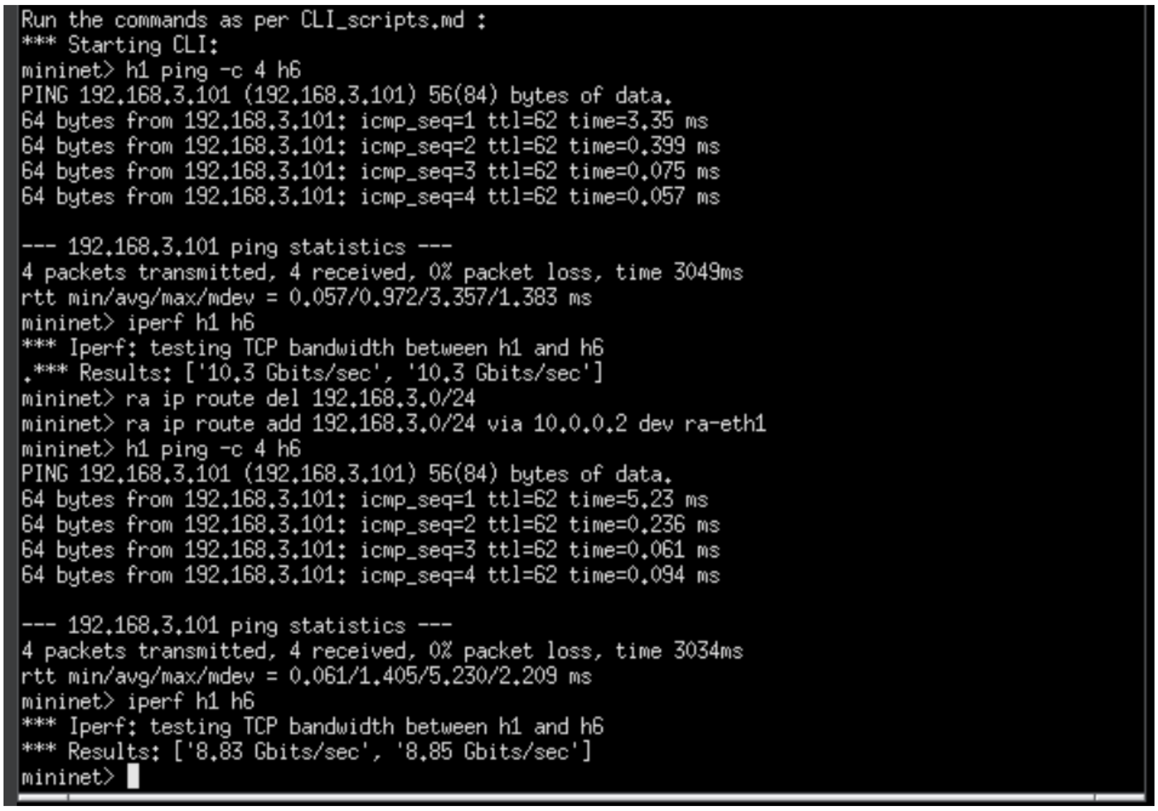


```
mininet> ra ip route
10.0.0.0/8 dev ra-eth1 proto kernel scope link src 10.0.0.1
192.168.1.0/24 dev ra-eth0 proto kernel scope link src 192.168.1.1
192.168.2.0/24 via 10.0.0.2 dev ra-eth1
192.168.3.0/24 via 10.0.0.3 dev ra-eth1
mininet> ra ip route del 192.168.3.0/24
mininet> ra ip route add 192.168.3.0/24 via 10.0.0.2 dev ra-eth1
mininet> ra ip route
10.0.0.0/8 dev ra-eth1 proto kernel scope link src 10.0.0.1
192.168.1.0/24 dev ra-eth0 proto kernel scope link src 192.168.1.1
192.168.2.0/24 via 10.0.0.2 dev ra-eth1
192.168.3.0/24 via 10.0.0.2 dev ra-eth1
mininet> h1 traceroute h6
traceroute to 192.168.3.101 (192.168.3.101), 30 hops max, 60 byte packets
 1  192.168.1.1 (192.168.1.1)  111.385 ms  123.947 ms  123.972 ms
 2  10.0.0.2 (10.0.0.2)  126.347 ms  126.355 ms  126.364 ms
 3  10.0.0.3 (10.0.0.3)  129.318 ms  129.324 ms  129.346 ms
 4  192.168.3.101 (192.168.3.101)  130.151 ms  130.158 ms  130.160 ms
mininet> █
```

As you can see router ra has the route entry to rc connected directly via 10.0.0.3.

We modified this entry so that ra has the route entry to rc via 10.0.0.2 that is through rb.

The traceroute packet also shows that path from h1 to h6 is now h1→ra→rb→rc→h6.



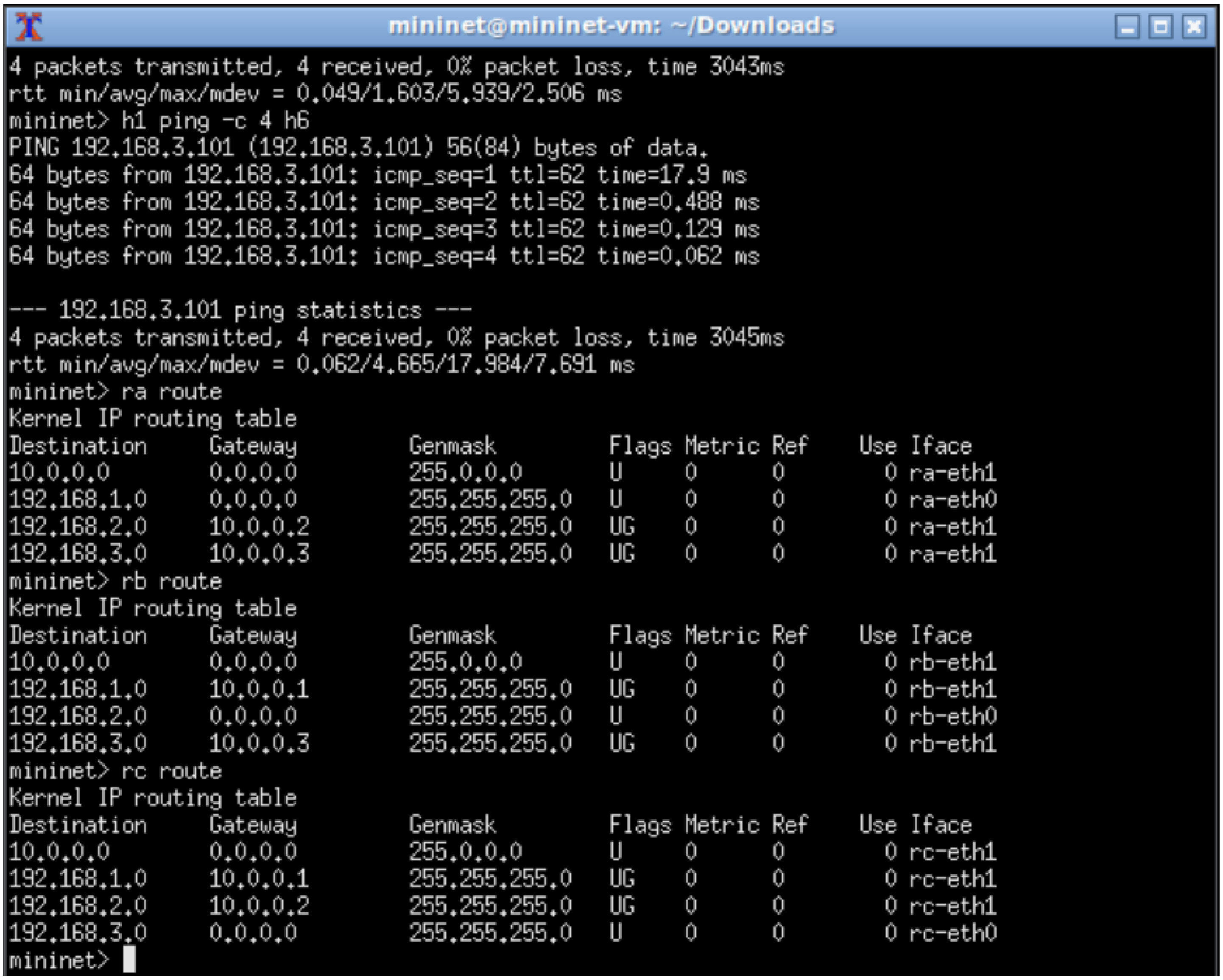
```
Run the commands as per CLI_scripts.md :
*** Starting CLI:
mininet> h1 ping -c 4 h6
PING 192.168.3.101 (192.168.3.101) 56(84) bytes of data.
64 bytes from 192.168.3.101: icmp_seq=1 ttl=62 time=3.35 ms
64 bytes from 192.168.3.101: icmp_seq=2 ttl=62 time=0.399 ms
64 bytes from 192.168.3.101: icmp_seq=3 ttl=62 time=0.075 ms
64 bytes from 192.168.3.101: icmp_seq=4 ttl=62 time=0.057 ms

--- 192.168.3.101 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3049ms
rtt min/avg/max/mdev = 0.057/0.972/3.357/1.383 ms
mininet> iperf h1 h6
*** Iperf: testing TCP bandwidth between h1 and h6
.*** Results: ['10.3 Gbits/sec', '10.3 Gbits/sec']
mininet> ra ip route del 192.168.3.0/24
mininet> ra ip route add 192.168.3.0/24 via 10.0.0.2 dev ra-eth1
mininet> h1 ping -c 4 h6
PING 192.168.3.101 (192.168.3.101) 56(84) bytes of data.
64 bytes from 192.168.3.101: icmp_seq=1 ttl=62 time=5.23 ms
64 bytes from 192.168.3.101: icmp_seq=2 ttl=62 time=0.236 ms
64 bytes from 192.168.3.101: icmp_seq=3 ttl=62 time=0.061 ms
64 bytes from 192.168.3.101: icmp_seq=4 ttl=62 time=0.094 ms

--- 192.168.3.101 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3034ms
rtt min/avg/max/mdev = 0.061/1.405/5.230/2.209 ms
mininet> iperf h1 h6
*** Iperf: testing TCP bandwidth between h1 and h6
*** Results: ['8.83 Gbits/sec', '8.85 Gbits/sec']
mininet> 
```

So, if we compare the default route and updated route:

Route	Ping (rtt min/avg/max/mdev)	Iperf (Bandwidth)
Default (h →ra→rc→h6)	0.057/0.972/3.357/1.383 ms	10.3 Gb/s, 10.3 Gb/s
Modified (h1→ra→rb→rc→h6)	0.061/1.405/5.230/2.209 ms	8.83 Gb/s, 8.85 Gb/s

Question **d) Dumping routing tables:**

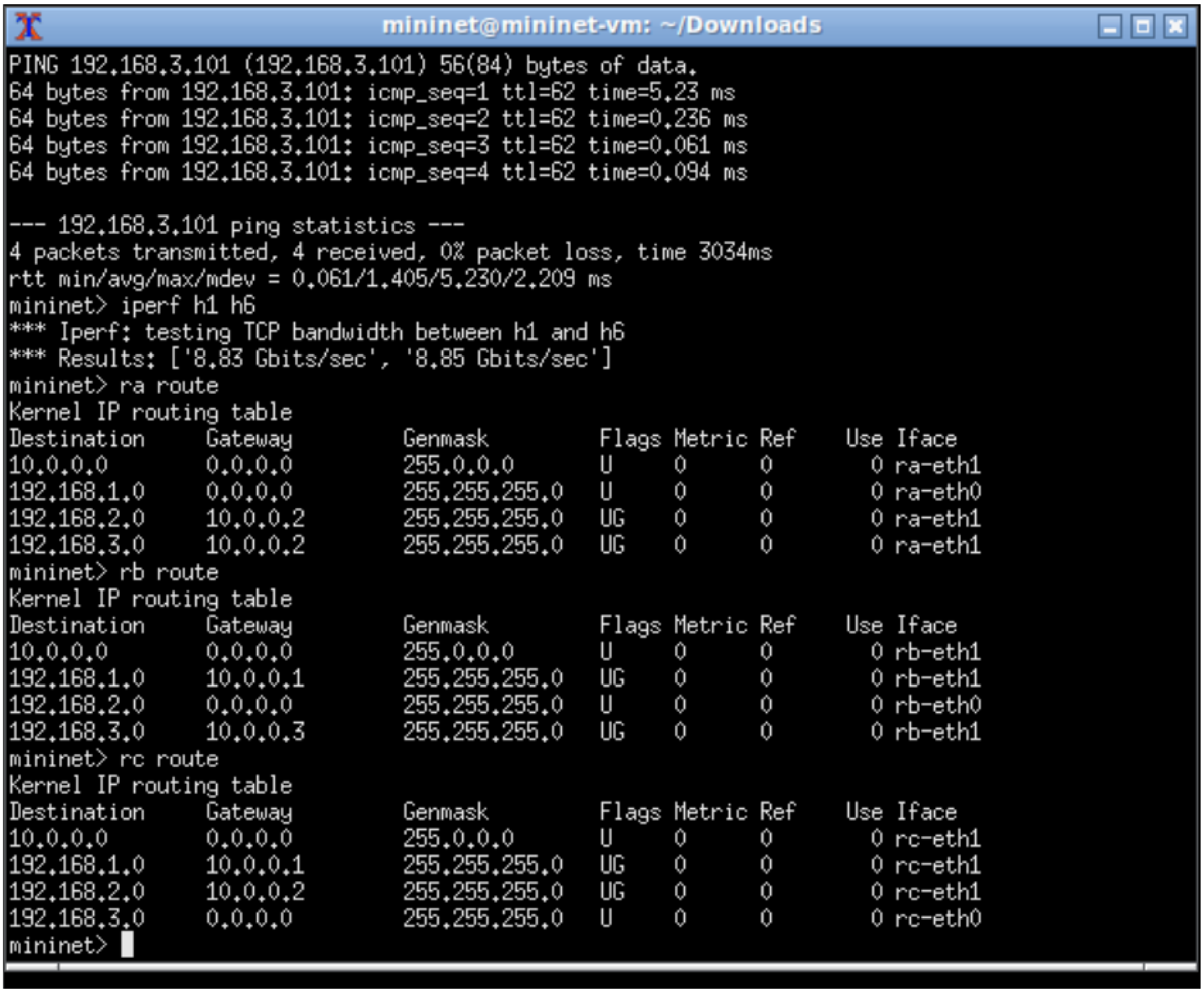
Routing tables in context of question b)



mininet@mininet-vm: ~/Downloads

```
4 packets transmitted, 4 received, 0% packet loss, time 3043ms
rtt min/avg/max/mdev = 0.049/1.603/5.939/2.506 ms
mininet> h1 ping -c 4 h6
PING 192.168.3.101 (192.168.3.101) 56(84) bytes of data.
64 bytes from 192.168.3.101: icmp_seq=1 ttl=62 time=17.9 ms
64 bytes from 192.168.3.101: icmp_seq=2 ttl=62 time=0.488 ms
64 bytes from 192.168.3.101: icmp_seq=3 ttl=62 time=0.129 ms
64 bytes from 192.168.3.101: icmp_seq=4 ttl=62 time=0.062 ms

--- 192.168.3.101 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3045ms
rtt min/avg/max/mdev = 0.062/4.665/17.984/7.691 ms
mininet> ra route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.0.0.0       U     0      0        0 ra-eth1
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 ra-eth0
192.168.2.0     10.0.0.2        255.255.255.0   UG    0      0        0 ra-eth1
192.168.3.0     10.0.0.3        255.255.255.0   UG    0      0        0 ra-eth1
mininet> rb route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.0.0.0       U     0      0        0 rb-eth1
192.168.1.0     10.0.0.1        255.255.255.0   UG    0      0        0 rb-eth1
192.168.2.0     0.0.0.0         255.255.255.0   U     0      0        0 rb-eth0
192.168.3.0     10.0.0.3        255.255.255.0   UG    0      0        0 rb-eth1
mininet> rc route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.0.0.0       U     0      0        0 rc-eth1
192.168.1.0     10.0.0.1        255.255.255.0   UG    0      0        0 rc-eth1
192.168.2.0     10.0.0.2        255.255.255.0   UG    0      0        0 rc-eth1
192.168.3.0     0.0.0.0         255.255.255.0   U     0      0        0 rc-eth0
mininet> 
```

Routing tables in context of question c)



```
PING 192.168.3.101 (192.168.3.101) 56(84) bytes of data.
64 bytes from 192.168.3.101: icmp_seq=1 ttl=62 time=5.23 ms
64 bytes from 192.168.3.101: icmp_seq=2 ttl=62 time=0.236 ms
64 bytes from 192.168.3.101: icmp_seq=3 ttl=62 time=0.061 ms
64 bytes from 192.168.3.101: icmp_seq=4 ttl=62 time=0.094 ms

--- 192.168.3.101 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3034ms
rtt min/avg/max/mdev = 0.061/1.405/5.230/2.209 ms
mininet> iperf h1 h6
*** Iperf: testing TCP bandwidth between h1 and h6
*** Results: ['8.83 Gbits/sec', '8.85 Gbits/sec']
mininet> ra route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.0.0.0       U     0      0        0 ra-eth1
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 ra-eth0
192.168.2.0     10.0.0.2        255.255.255.0   UG    0      0        0 ra-eth1
192.168.3.0     10.0.0.2        255.255.255.0   UG    0      0        0 ra-eth1
mininet> rb route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.0.0.0       U     0      0        0 rb-eth1
192.168.1.0     10.0.0.1        255.255.255.0   UG    0      0        0 rb-eth1
192.168.2.0     0.0.0.0         255.255.255.0   U     0      0        0 rb-eth0
192.168.3.0     10.0.0.3        255.255.255.0   UG    0      0        0 rb-eth1
mininet> rc route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.0.0.0       U     0      0        0 rc-eth1
192.168.1.0     10.0.0.1        255.255.255.0   UG    0      0        0 rc-eth1
192.168.2.0     10.0.0.2        255.255.255.0   UG    0      0        0 rc-eth1
192.168.3.0     0.0.0.0         255.255.255.0   U     0      0        0 rc-eth0
mininet> 
```

Part II

Question **a)** Implementation of mininet topology and TCP-Server

Created four hosts: h1, h2, h3 and h4; and two switches: s1 and s2.

Hosts h1 and h2 are connected to s1; and h3 and h4 are connected to s2.

s1 and s2 are connected to each other.

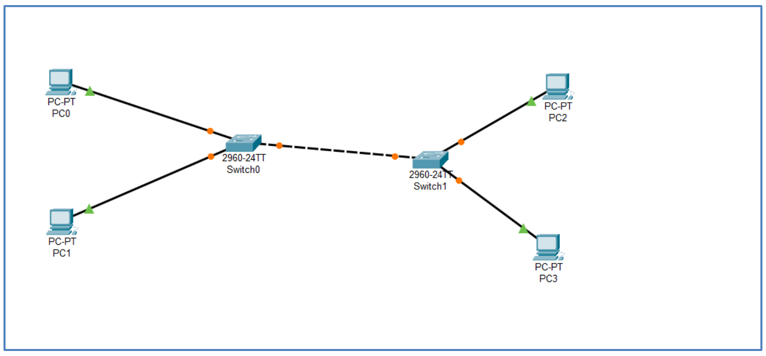
IP of the hosts: h1: 192.168.1.1/24

h2: 192.168.1.2/24

h3: 192.168.1.3/24

` h4: 192.168.1.4/24

Visualization:

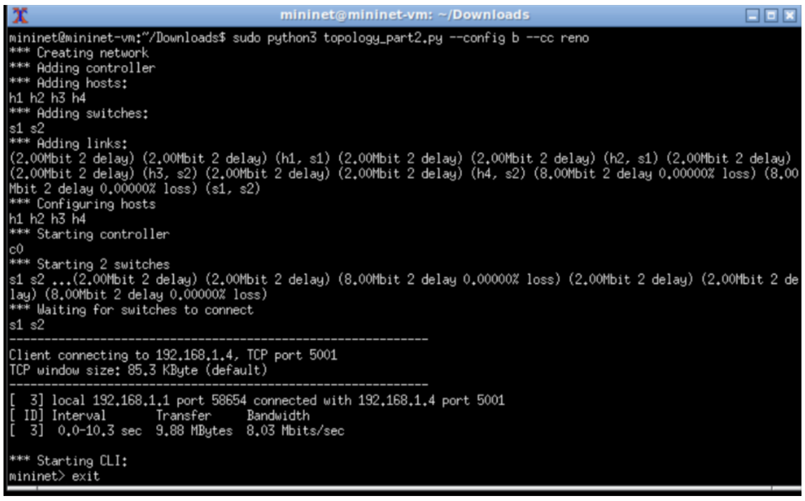


Code: Please see the code for part II implementation here: topology_part2.py

(please follow the instructions given here: readme.md for part II)

Proof of working of the network:

(executed with -- config b, no loss and congestion control scheme reno)

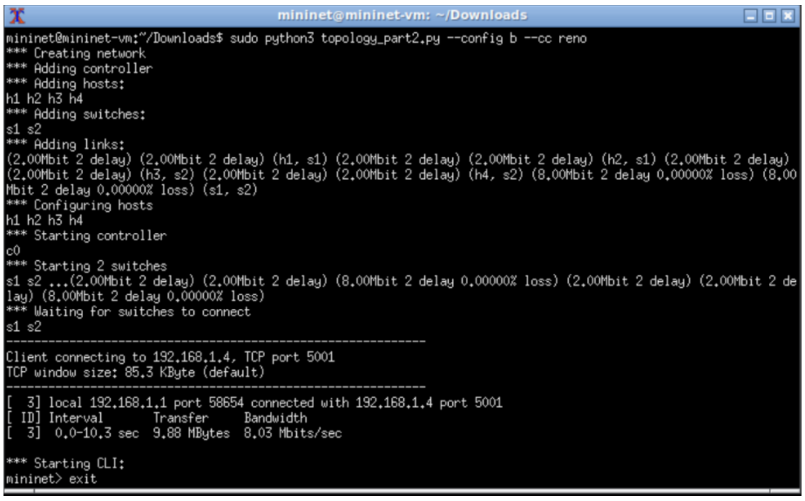


```
mininet@mininet-vm:~/Downloads$ sudo python3 topology_part2.py --config b --cc reno
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(2.00Mbit 2 delay) (2.00Mbit 2 delay) (h1, s1) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h2, s1) (2.00Mbit 2 delay)
(2.00Mbit 2 delay) (h3, s2) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h4, s2) (8.00Mbit 2 delay 0.00000% loss) (8.00
Mbit 2 delay 0.00000% loss) (s1, s2)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...(2.00Mbit 2 delay) (2.00Mbit 2 delay) (8.00Mbit 2 delay 0.00000% loss) (2.00Mbit 2 delay) (2.00Mbit 2 de
lay) (8.00Mbit 2 delay 0.00000% loss)
*** Waiting for switches to connect
s1 s2
------------------------------------------------------------
Client connecting to 192.168.1.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.1 port 58654 connected with 192.168.1.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.3 sec  9.88 MBytes  8.03 Mbits/sec

*** Starting CLI:
mininet> exit
```

Question **b)**

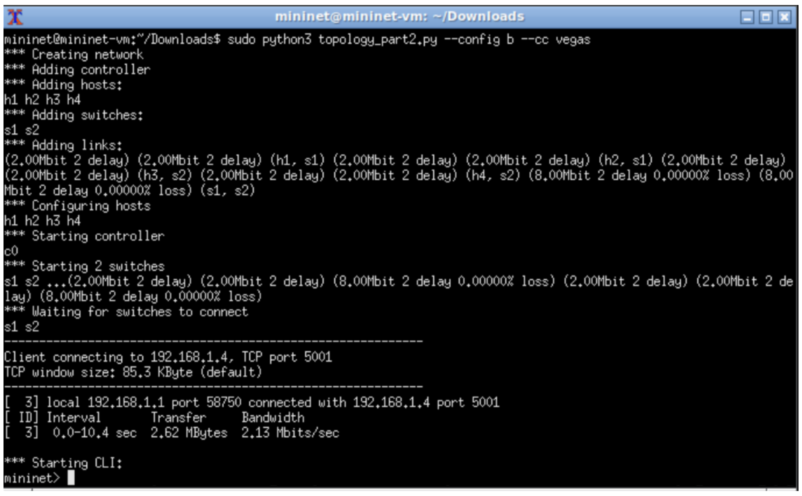
Running client on h1 and server on h4:

Scheme: reno



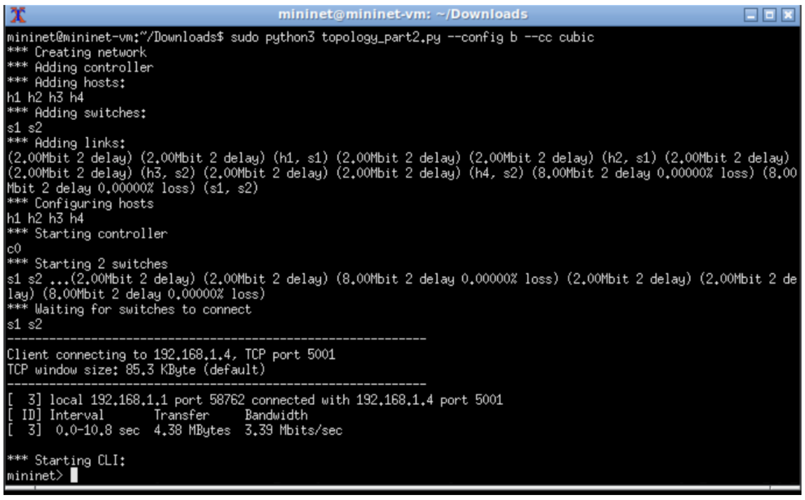
```
mininet@mininet-vm:~/Downloads$ sudo python3 topology_part2.py --config b --cc reno
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(2.00Mbit 2 delay) (2.00Mbit 2 delay) (h1, s1) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h2, s1) (2.00Mbit 2 delay)
(2.00Mbit 2 delay) (h3, s2) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h4, s2) (8.00Mbit 2 delay 0.00000% loss) (8.00
Mbit 2 delay 0.00000% loss) (s1, s2)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...(2.00Mbit 2 delay) (2.00Mbit 2 delay) (8.00Mbit 2 delay 0.00000% loss) (2.00Mbit 2 delay) (2.00Mbit 2 de
lay) (8.00Mbit 2 delay 0.00000% loss)
*** Waiting for switches to connect
s1 s2
------------------------------------------------------------
Client connecting to 192.168.1.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.1 port 58654 connected with 192.168.1.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.3 sec  9.88 MBytes  8.03 Mbits/sec

*** Starting CLI:
mininet> exit
```

Scheme: vegas



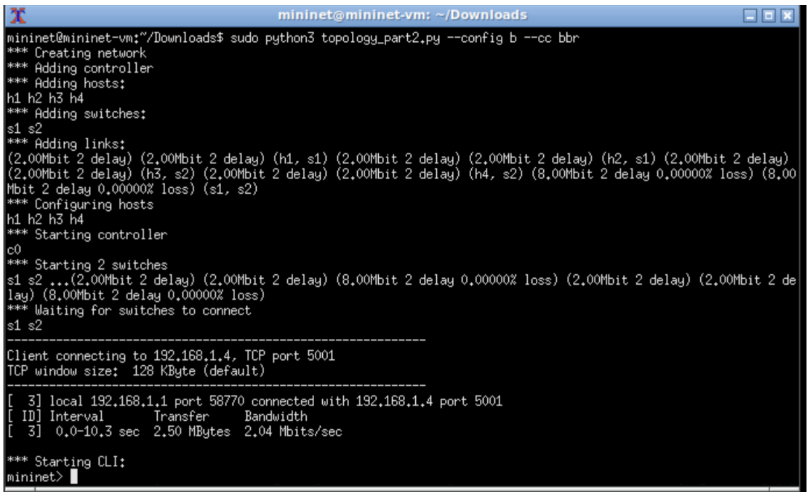
```
mininet@mininet-vm:~/Downloads$ sudo python3 topology_part2.py --config b --cc vegas
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(2.00Mbit 2 delay) (2.00Mbit 2 delay) (h1, s1) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h2, s1) (2.00Mbit 2 delay)
(2.00Mbit 2 delay) (h3, s2) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h4, s2) (8.00Mbit 2 delay 0.00000% loss) (8.00
Mbit 2 delay 0.00000% loss) (s1, s2)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...(2.00Mbit 2 delay) (2.00Mbit 2 delay) (8.00Mbit 2 delay 0.00000% loss) (2.00Mbit 2 delay) (2.00Mbit 2 de
lay) (8.00Mbit 2 delay 0.00000% loss)
*** Waiting for switches to connect
s1 s2
------------------------------------------------------------
Client connecting to 192.168.1.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.1 port 58750 connected with 192.168.1.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.4 sec  2.62 MBytes  2.13 Mbits/sec

*** Starting CLI:
mininet> 
```

Scheme: cubic



```
mininet@mininet-vm:~/Downloads$ sudo python3 topology_part2.py --config b --cc cubic
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(2.00Mbit 2 delay) (2.00Mbit 2 delay) (h1, s1) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h2, s1) (2.00Mbit 2 delay)
(2.00Mbit 2 delay) (h3, s2) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h4, s2) (8.00Mbit 2 delay 0.00000% loss) (8.00
Mbit 2 delay 0.00000% loss) (s1, s2)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...(2.00Mbit 2 delay) (2.00Mbit 2 delay) (8.00Mbit 2 delay 0.00000% loss) (2.00Mbit 2 delay) (2.00Mbit 2 de
lay) (8.00Mbit 2 delay 0.00000% loss)
*** Waiting for switches to connect
s1 s2
------------------------------------------------------------
Client connecting to 192.168.1.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.1 port 58762 connected with 192.168.1.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.8 sec  4.38 MBytes  3.39 Mbits/sec

*** Starting CLI:
mininet> 
```

Scheme: bbr



```
mininet@mininet-vm:~/Downloads$ sudo python3 topology_part2.py --config b --cc bbr
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(2.00Mbit 2 delay) (2.00Mbit 2 delay) (h1, s1) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h2, s1) (2.00Mbit 2 delay)
(2.00Mbit 2 delay) (h3, s2) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h4, s2) (8.00Mbit 2 delay 0.00000% loss) (8.00
Mbit 2 delay 0.00000% loss) (s1, s2)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...(2.00Mbit 2 delay) (2.00Mbit 2 delay) (8.00Mbit 2 delay 0.00000% loss) (2.00Mbit 2 delay) (2.00Mbit 2 de
lay) (8.00Mbit 2 delay 0.00000% loss)
*** Waiting for switches to connect
s1 s2
------------------------------------------------------------
Client connecting to 192.168.1.4, TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.1 port 58770 connected with 192.168.1.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.3 sec  2.50 MBytes  2.04 Mbits/sec

*** Starting CLI:
mininet> 
```

Summary of the throughput: (h1→h4)

Scheme	Throughput	Transferred Data
RENO	8.03 Mbits/sec	9.88 MB
VEGAS	2.13 Mbits/sec	2.62 MB
CUBIC	3.39 Mbits/sec	4.38 MB
BBR	2.04 Mbits/sec	2.50 MB

We're seeing the maximum throughput in reno scheme, whereas the least in bbr scheme.

Reasoning of the throughput differences:

The observed throughput differences between the Congestion Control Algorithms (CCAs) can be attributed to their distinct mechanisms for dealing with network congestion. Each CCA employs a different approach to adjusting transmission rates in response to network conditions, leading to varying throughput performance.

Here's a breakdown of the CCAs and their impact on throughput:

Reno: The default CCA for TCP, Reno is a loss-based algorithm that slows down the transmission rate upon packet loss, assuming network congestion. While simple and easy to implement, Reno's conservative rate reduction can limit throughput in congested networks. But the iperf test methodology might have inadvertently favored Reno leading to the max throughput. Also, it got the highest size of transferred data, that also may be reason of getting higher throughput.

Vegas: Vegas utilizes both packet loss and delay information to estimate network congestion. This allows for more accurate rate adjustments compared to Reno, potentially leading to higher throughput in some scenarios. But in this case, the iperf test methodology might have not favoured reno, leading to the lower throughput.

Cubic: Cubic employs a combination of loss, delay, and packet arrival times to estimate congestion, enabling quicker rate adjustments compared to Vegas. That may be the reason of it getting higher throughput than vegas.

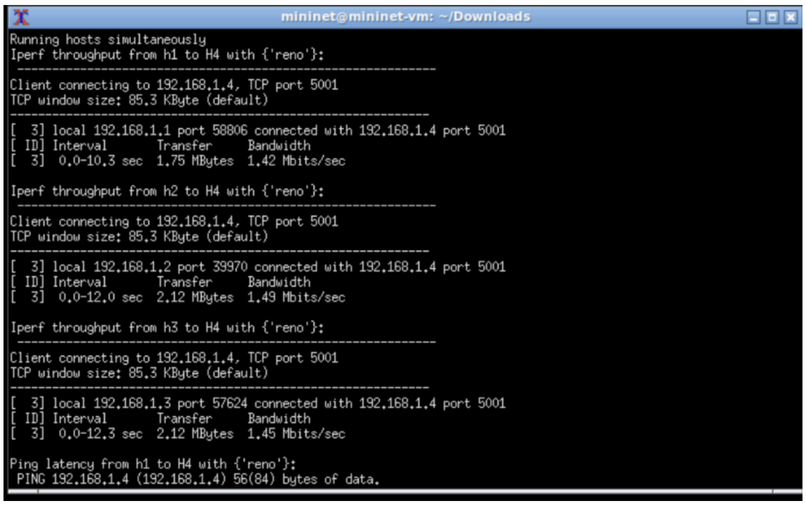
BBR: Google's BBR CCA relies on a delay-based approach, using a network model to estimate the optimal sending rate. BBR can achieve high throughput in networks with high latency or packet loss, but it may be more complex to implement than Reno, Vegas, or Cubic. And iperf testing methodology may have negatively impacted the throughput.

Question **c)**

To simultaneously run the three clients h1, h2, and h3 on h4 server, we did the following:

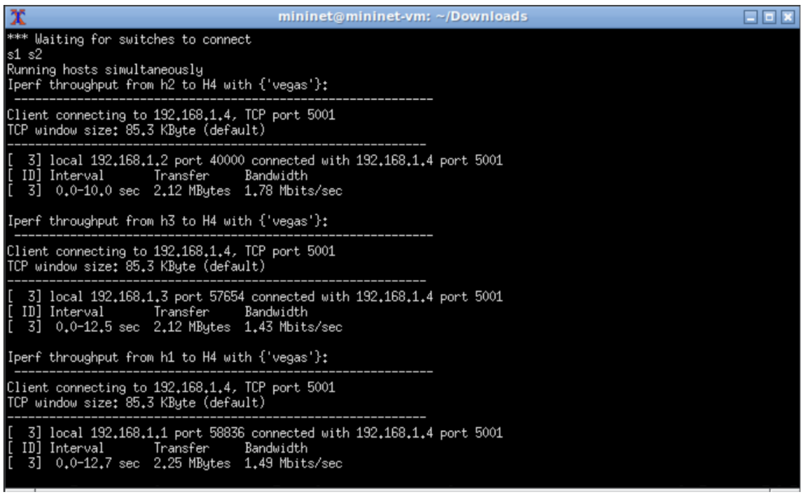
We first started iperf server on h4. Then we created three threads for h1, h2, and h3 as clients to be served by h4. Then we simultaneously ran the three threads to be served by h4. And all threads are defined on the run() function during the net active period.

Scheme: reno



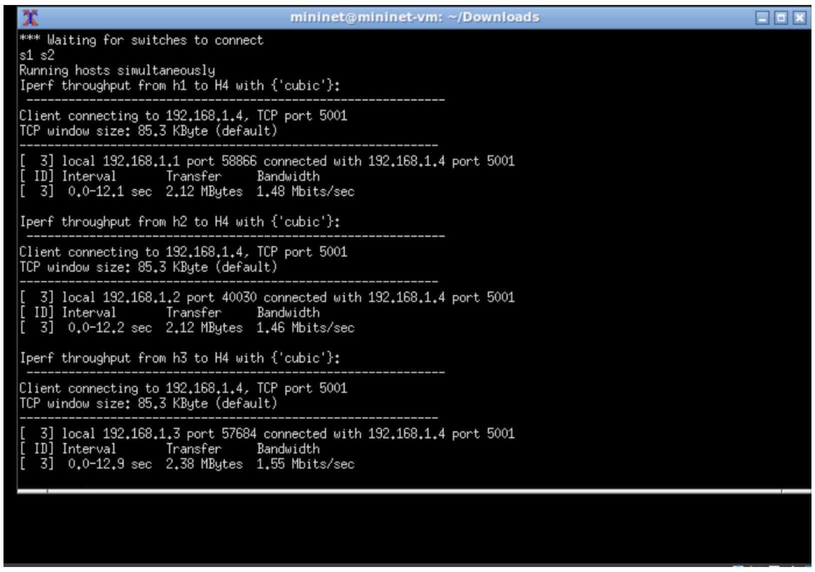
```
Running hosts simultaneously
Iperf throughput from h1 to H4 with {'reno'}:
------------------------------------------------------------
Client connecting to 192.168.1.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.1 port 58806 connected with 192.168.1.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.3 sec  1.75 MBytes  1.42 Mbits/sec

Iperf throughput from h2 to H4 with {'reno'}:
------------------------------------------------------------
Client connecting to 192.168.1.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.2 port 39970 connected with 192.168.1.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-12.0 sec  2.12 MBytes  1.49 Mbits/sec

Iperf throughput from h3 to H4 with {'reno'}:
------------------------------------------------------------
Client connecting to 192.168.1.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.3 port 57624 connected with 192.168.1.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-12.3 sec  2.12 MBytes  1.45 Mbits/sec

Ping latency from h1 to H4 with {'reno'}:
 PING 192.168.1.4 (192.168.1.4) 56(84) bytes of data.
```

Scheme: vegas


mininet@mininet-vm: ~/Downloads

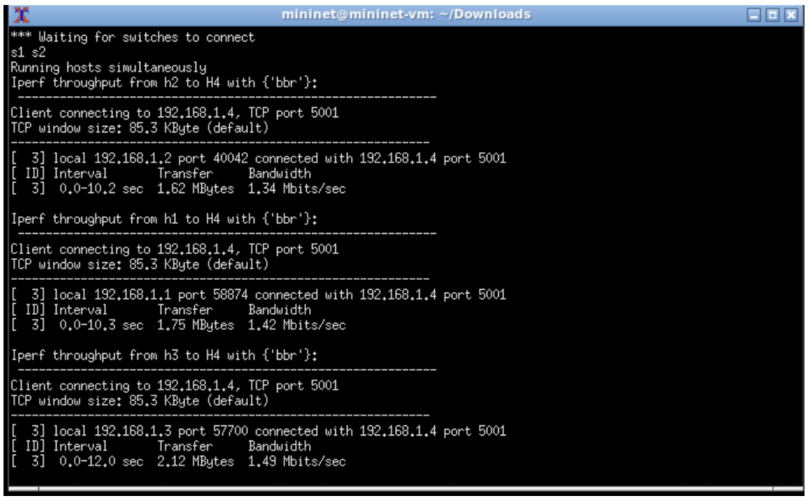
```
*** Waiting for switches to connect
s1 s2
Running hosts simultaneously
Iperf throughput from h2 to H4 with {'vegas'}:
------------------------------------------------------------
Client connecting to 192.168.1.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.2 port 40000 connected with 192.168.1.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  2.12 MBytes  1.78 Mbits/sec

Iperf throughput from h3 to H4 with {'vegas'}:
------------------------------------------------------------
Client connecting to 192.168.1.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.3 port 57654 connected with 192.168.1.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-12.5 sec  2.12 MBytes  1.43 Mbits/sec

Iperf throughput from h1 to H4 with {'vegas'}:
------------------------------------------------------------
Client connecting to 192.168.1.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.1 port 58836 connected with 192.168.1.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-12.7 sec  2.25 MBytes  1.49 Mbits/sec
```

Scheme: cubic


mininet@mininet-vm: ~/Downloads

```
*** Waiting for switches to connect
s1 s2
Running hosts simultaneously
Iperf throughput from h1 to H4 with {'cubic'}:
------------------------------------------------------------
Client connecting to 192.168.1.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.1 port 58866 connected with 192.168.1.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-12.1 sec  2.12 MBytes  1.48 Mbits/sec

Iperf throughput from h2 to H4 with {'cubic'}:
------------------------------------------------------------
Client connecting to 192.168.1.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.2 port 40030 connected with 192.168.1.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-12.2 sec  2.12 MBytes  1.46 Mbits/sec

Iperf throughput from h3 to H4 with {'cubic'}:
------------------------------------------------------------
Client connecting to 192.168.1.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.3 port 57684 connected with 192.168.1.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-12.9 sec  2.38 MBytes  1.55 Mbits/sec
```

Scheme: bbr


mininet@mininet-vm: ~/Downloads

```
*** Waiting for switches to connect
s1 s2
Running hosts simultaneously
Iperf throughput from h2 to H4 with {'bbr'}:
------------------------------------------------------------
Client connecting to 192.168.1.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.2 port 40042 connected with 192.168.1.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.2 sec  1.62 MBytes  1.34 Mbits/sec

Iperf throughput from h1 to H4 with {'bbr'}:
------------------------------------------------------------
Client connecting to 192.168.1.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.1 port 58874 connected with 192.168.1.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.3 sec  1.75 MBytes  1.42 Mbits/sec

Iperf throughput from h3 to H4 with {'bbr'}:
------------------------------------------------------------
Client connecting to 192.168.1.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.3 port 57700 connected with 192.168.1.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-12.0 sec  2.12 MBytes  1.49 Mbits/sec
```

Summary of the schemes:

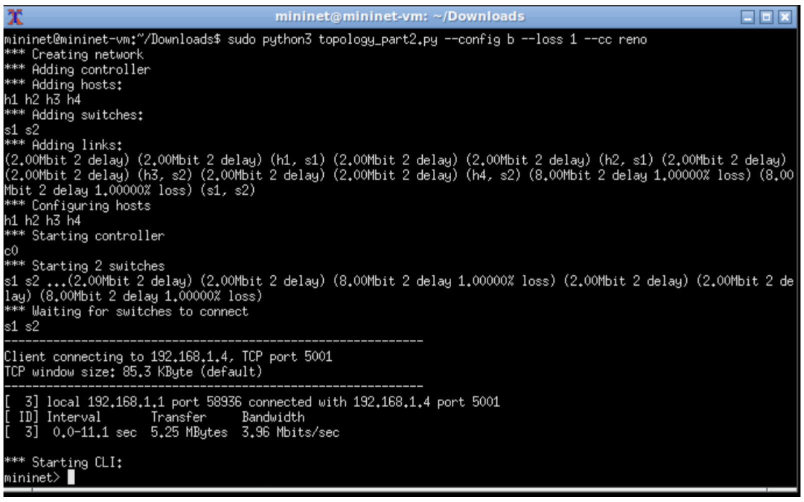
Scheme	Source	Throughput	File Transferred
reno	h1→h4	1.4 Mb/s	1.7 MB
reno	h2→h4	1.49 Mb/s	2.12 MB
reno	h3→h4	1.45 Mb/s	2.12 MB
vegas	h1→h4	1.49 Mb/s	2.25 MB
vegas	h2→h4	1.78 Mb/s	2.12 MB
vegas	h3→h4	1.43 Mb/s	2.12 MB
cubic	h1→h4	1.48 Mb/s	2.12 MB
cubic	h2→h4	1.46 Mb/s	2.12 MB
cubic	h3→h4	1.55 Mb/s	2.38 MB
bbr	h1→h4	1.42 Mb/s	1.75 MB
bbr	h2→h4	1.34 Mb/s	1.62 MB
bbr	h3→h4	1.49 Mb/s	2.12 MB

We got similar results across all the schemes. Although the CCAs are different but we still got the similar results may be due to the methodology that is implemented internally by iperf.

Question **d)**

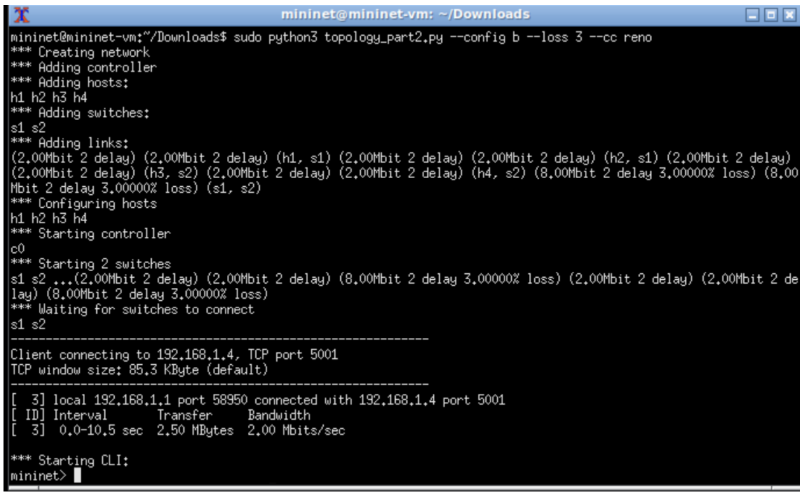
Running the –config b with all the schemes:

Scheme: reno with 1% loss



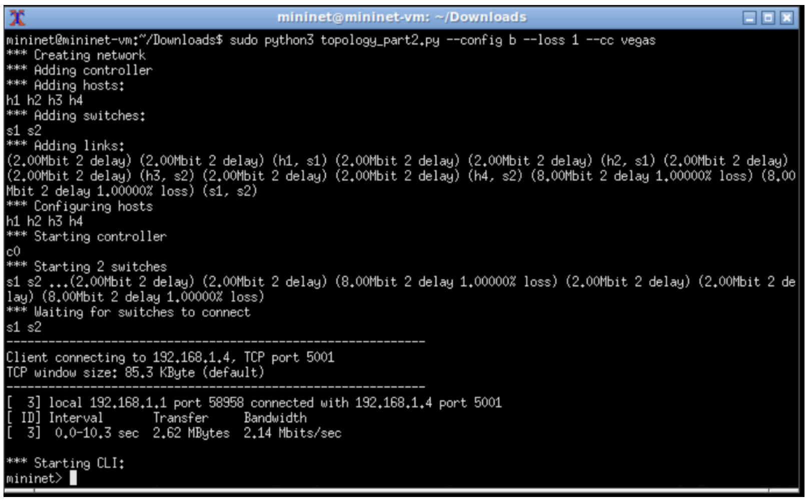
```
mininet@mininet-vm:~/Downloads$ sudo python3 topology_part2.py --config b --loss 1 --cc reno
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(2.00Mbit 2 delay) (2.00Mbit 2 delay) (h1, s1) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h2, s1) (2.00Mbit 2 delay)
(2.00Mbit 2 delay) (h3, s2) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h4, s2) (8.00Mbit 2 delay 1.00000% loss) (8.00
Mbit 2 delay 1.00000% loss) (s1, s2)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...(2.00Mbit 2 delay) (2.00Mbit 2 delay) (8.00Mbit 2 delay 1.00000% loss) (2.00Mbit 2 delay) (2.00Mbit 2 de
lay) (8.00Mbit 2 delay 1.00000% loss)
*** Waiting for switches to connect
s1 s2
------------------------------------------------------------
Client connecting to 192.168.1.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.1 port 58936 connected with 192.168.1.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-11.1 sec  5.25 MBytes  3.96 Mbits/sec

*** Starting CLI:
mininet> 
```

Scheme: reno with 3% loss



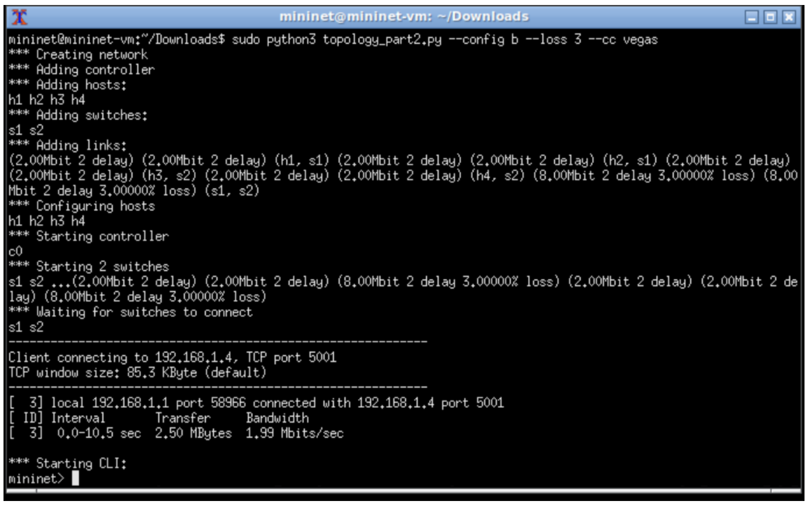
```
mininet@mininet-vm:~/Downloads$ sudo python3 topology_part2.py --config b --loss 3 --cc reno
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(2.00Mbit 2 delay) (2.00Mbit 2 delay) (h1, s1) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h2, s1) (2.00Mbit 2 delay)
(2.00Mbit 2 delay) (h3, s2) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h4, s2) (8.00Mbit 2 delay 3.00000% loss) (8.00
Mbit 2 delay 3.00000% loss) (s1, s2)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...(2.00Mbit 2 delay) (2.00Mbit 2 delay) (8.00Mbit 2 delay 3.00000% loss) (2.00Mbit 2 delay) (2.00Mbit 2 de
lay) (8.00Mbit 2 delay 3.00000% loss)
*** Waiting for switches to connect
s1 s2
------------------------------------------------------------
Client connecting to 192.168.1.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.1 port 58950 connected with 192.168.1.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.5 sec  2.50 MBytes  2.00 Mbits/sec

*** Starting CLI:
mininet> 
```

Scheme: vegas with 1% loss



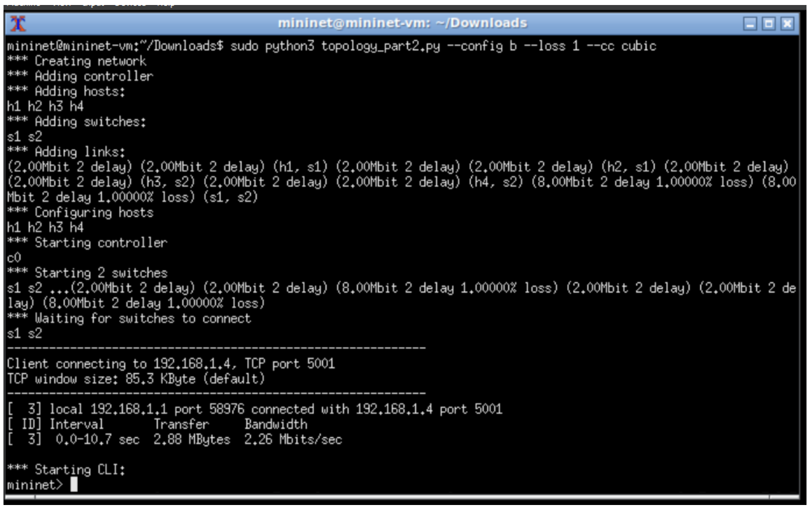
```
mininet@mininet-vm:~/Downloads$ sudo python3 topology_part2.py --config b --loss 1 --cc vegas
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(2.00Mbit 2 delay) (2.00Mbit 2 delay) (h1, s1) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h2, s1) (2.00Mbit 2 delay)
(2.00Mbit 2 delay) (h3, s2) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h4, s2) (8.00Mbit 2 delay 1.00000% loss) (8.00
Mbit 2 delay 1.00000% loss) (s1, s2)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...(2.00Mbit 2 delay) (2.00Mbit 2 delay) (8.00Mbit 2 delay 1.00000% loss) (2.00Mbit 2 delay) (2.00Mbit 2 de
lay) (8.00Mbit 2 delay 1.00000% loss)
*** Waiting for switches to connect
s1 s2
------------------------------------------------------------
Client connecting to 192.168.1.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.1 port 58958 connected with 192.168.1.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.3 sec  2.62 MBytes  2.14 Mbits/sec

*** Starting CLI:
mininet> 
```

Scheme: vegas with 3% loss



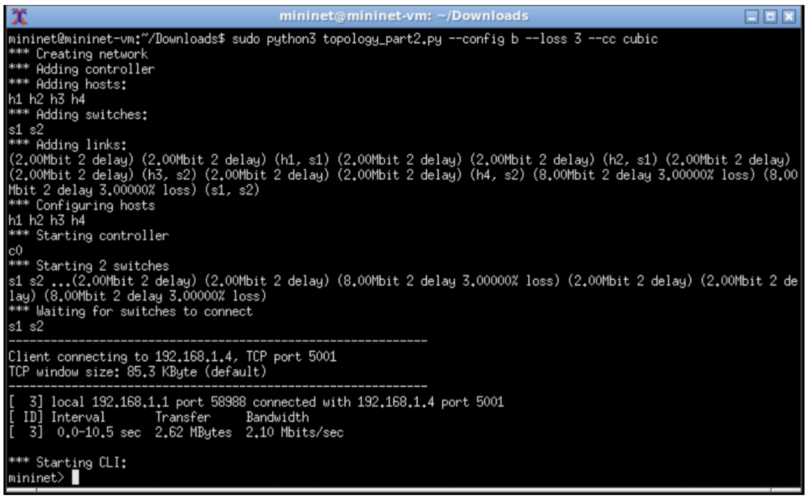
```
mininet@mininet-vm:~/Downloads$ sudo python3 topology_part2.py --config b --loss 3 --cc vegas
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(2.00Mbit 2 delay) (2.00Mbit 2 delay) (h1, s1) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h2, s1) (2.00Mbit 2 delay)
(2.00Mbit 2 delay) (h3, s2) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h4, s2) (8.00Mbit 2 delay 3.00000% loss) (8.00
Mbit 2 delay 3.00000% loss) (s1, s2)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...(2.00Mbit 2 delay) (2.00Mbit 2 delay) (8.00Mbit 2 delay 3.00000% loss) (2.00Mbit 2 delay) (2.00Mbit 2 de
lay) (8.00Mbit 2 delay 3.00000% loss)
*** Waiting for switches to connect
s1 s2
------------------------------------------------------------
Client connecting to 192.168.1.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.1 port 58966 connected with 192.168.1.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.5 sec  2.50 MBytes  1.99 Mbits/sec

*** Starting CLI:
mininet> 
```

Scheme: cubic with 1% loss



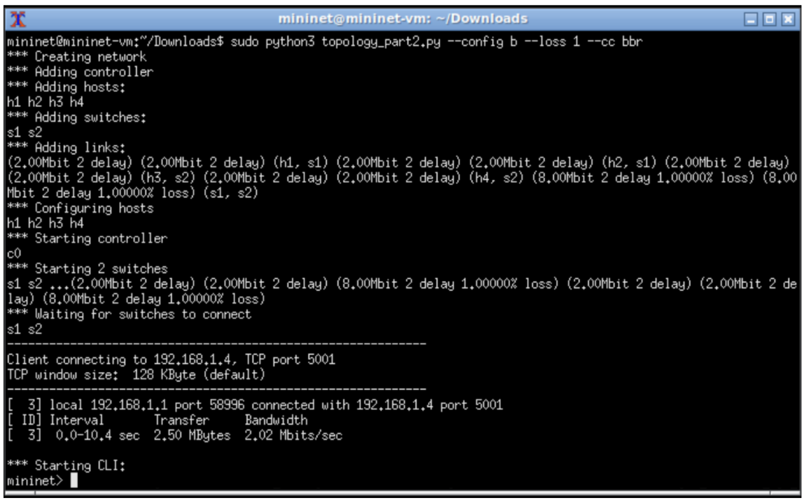
```
mininet@mininet-vm:~/Downloads$ sudo python3 topology_part2.py --config b --loss 1 --cc cubic
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(2.00Mbit 2 delay) (2.00Mbit 2 delay) (h1, s1) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h2, s1) (2.00Mbit 2 delay)
(2.00Mbit 2 delay) (h3, s2) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h4, s2) (8.00Mbit 2 delay 1.00000% loss) (8.00
Mbit 2 delay 1.00000% loss) (s1, s2)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...(2.00Mbit 2 delay) (2.00Mbit 2 delay) (8.00Mbit 2 delay 1.00000% loss) (2.00Mbit 2 delay) (2.00Mbit 2 de
lay) (8.00Mbit 2 delay 1.00000% loss)
*** Waiting for switches to connect
s1 s2
------------------------------------------------------------
Client connecting to 192.168.1.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.1 port 58976 connected with 192.168.1.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.7 sec  2.88 MBytes  2.26 Mbits/sec

*** Starting CLI:
mininet> █
```

Scheme: cubic with 3% loss



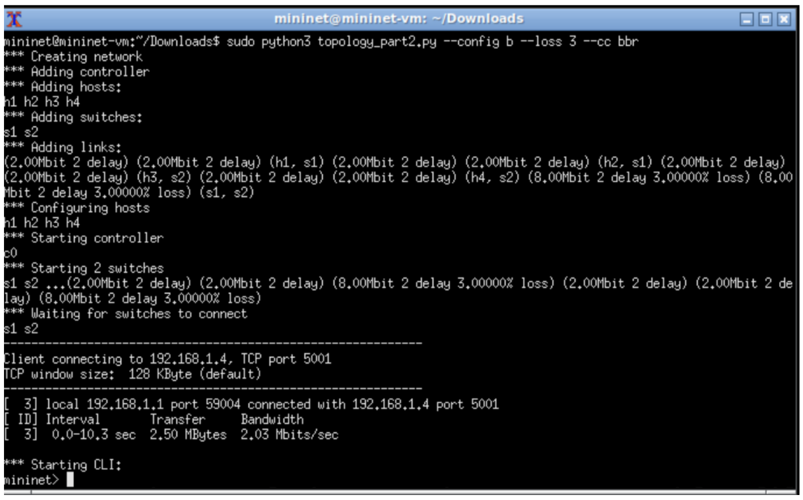
```
mininet@mininet-vm:~/Downloads$ sudo python3 topology_part2.py --config b --loss 3 --cc cubic
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(2.00Mbit 2 delay) (2.00Mbit 2 delay) (h1, s1) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h2, s1) (2.00Mbit 2 delay)
(2.00Mbit 2 delay) (h3, s2) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h4, s2) (8.00Mbit 2 delay 3.00000% loss) (8.00
Mbit 2 delay 3.00000% loss) (s1, s2)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...(2.00Mbit 2 delay) (2.00Mbit 2 delay) (8.00Mbit 2 delay 3.00000% loss) (2.00Mbit 2 delay) (2.00Mbit 2 de
lay) (8.00Mbit 2 delay 3.00000% loss)
*** Waiting for switches to connect
s1 s2
------------------------------------------------------------
Client connecting to 192.168.1.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.1 port 58988 connected with 192.168.1.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.5 sec  2.62 MBytes  2.10 Mbits/sec

*** Starting CLI:
mininet> █
```

Scheme: bbr with 1% loss



```
mininet@mininet-vm:~/Downloads$ sudo python3 topology_part2.py --config b --loss 1 --cc bbr
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(2.00Mbit 2 delay) (2.00Mbit 2 delay) (h1, s1) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h2, s1) (2.00Mbit 2 delay)
(2.00Mbit 2 delay) (h3, s2) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h4, s2) (8.00Mbit 2 delay 1.00000% loss) (8.00
Mbit 2 delay 1.00000% loss) (s1, s2)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...(2.00Mbit 2 delay) (2.00Mbit 2 delay) (8.00Mbit 2 delay 1.00000% loss) (2.00Mbit 2 delay) (2.00Mbit 2 de
lay) (8.00Mbit 2 delay 1.00000% loss)
*** Waiting for switches to connect
s1 s2
------------------------------------------------------------
Client connecting to 192.168.1.4, TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.1 port 58996 connected with 192.168.1.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.4 sec  2.50 MBytes  2.02 Mbits/sec

*** Starting CLI:
mininet> █
```

Scheme: bbr with 3% loss


mininet@mininet-vm: ~/Downloads

```
mininet@mininet-vm:~/Downloads$ sudo python3 topology_part2.py --config b --loss 3 --cc bbr
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(2.00Mbit 2 delay) (2.00Mbit 2 delay) (h1, s1) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h2, s1) (2.00Mbit 2 delay)
(2.00Mbit 2 delay) (h3, s2) (2.00Mbit 2 delay) (2.00Mbit 2 delay) (h4, s2) (8.00Mbit 2 delay 3.00000% loss) (8.00
Mbit 2 delay 3.00000% loss) (s1, s2)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...(2.00Mbit 2 delay) (2.00Mbit 2 delay) (8.00Mbit 2 delay 3.00000% loss) (2.00Mbit 2 delay) (2.00Mbit 2 de
lay) (8.00Mbit 2 delay 3.00000% loss)
*** Waiting for switches to connect
s1 s2
------------------------------------------------------------
Client connecting to 192.168.1.4, TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.1 port 59004 connected with 192.168.1.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.3 sec  2.50 MBytes  2.03 Mbits/sec

*** Starting CLI:
mininet> 
```

Summary of the results: (h1→h4)

Scheme	Throughput	Transferred Data
RENO (with no loss)	8.03 Mbits/sec	9.88 MB
RENO (with 1% loss)	3.96 Mbits/sec	5.25 MB
RENO (with 3% loss)	2 Mbits/sec	2.50 MB
VEGAS (with no loss)	2.13 Mbits/sec	2.62 MB
VEGAS (with 1% loss)	2.14 Mbits/sec	2.62 MB
VEGAS (with 3% loss)	1.99 Mbits/sec	2.50 MB
CUBIC (with no loss)	3.39 Mbits/sec	4.38 MB
CUBIC (with 1% loss)	2.26 Mbits/sec	2.88 MB
CUBIC (with 3% loss)	2.10 Mbits/sec	2.62 MB
BBR (with no loss)	2.04 Mbits/sec	2.50 MB
BBR (with 1% loss)	2.02 Mbits/sec	2.50 MB
BBR (with 3% loss)	2.03 Mbits/sec	2.50 MB

For reno, we can see that due to 1% and 3% loss on the link, the throughput has decreased significantly.

For vegas, we see that due to 3% loss on the link only, throughput has decreased slightly. For 1% loss on the link, throughput is similar.

For cubic, we see that due to 1% and 3% loss on the link, throughput has decreased.

For bbr, we see that 1% and 3% loss on the link plays no significant role.

The reason for the throughputs on the CCAs are similar to what have mentioned for question b.