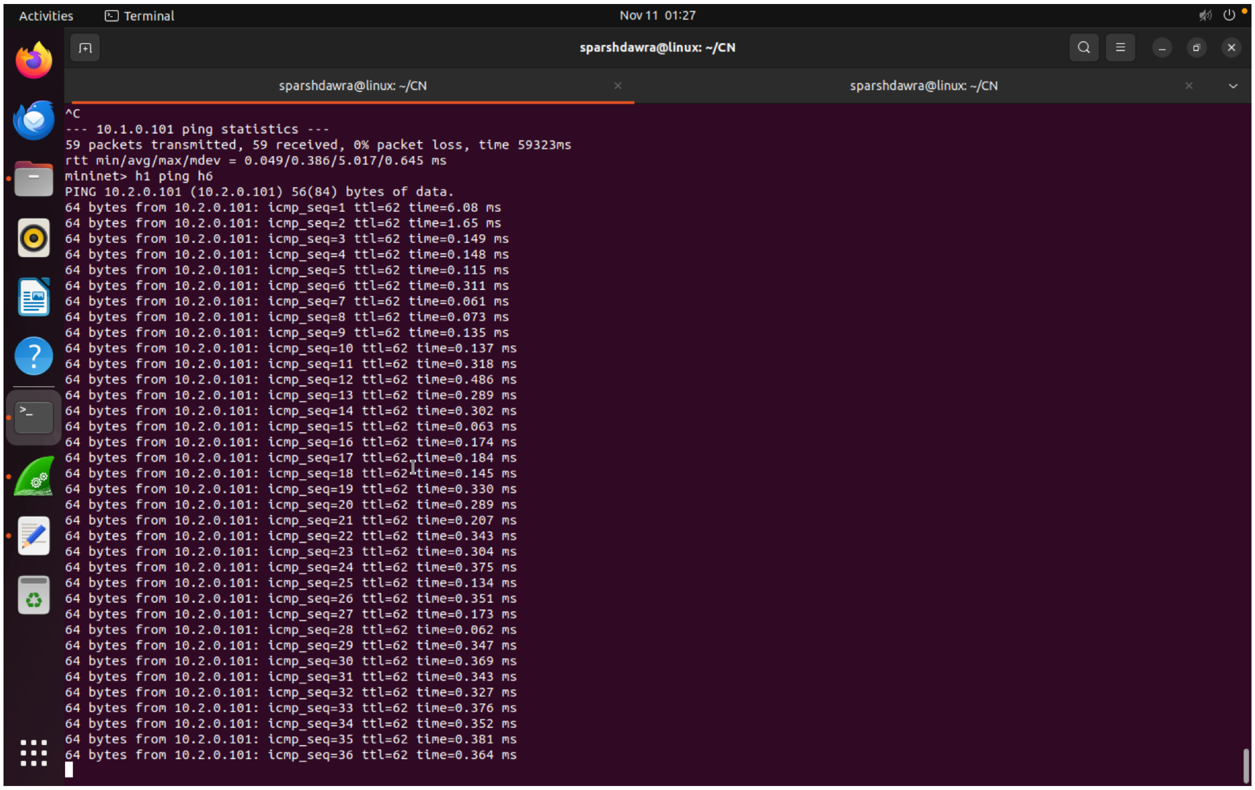
<u>CS433 Assignment-2</u>

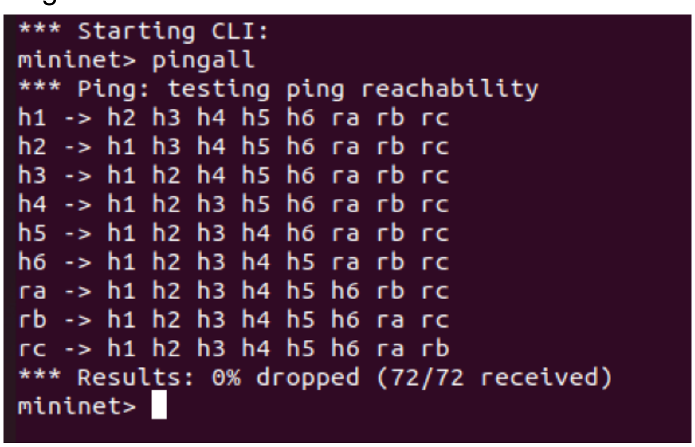
<u>Part:1</u>

A) Implementation:

The written code is added in the github repository and here are proof of its working:

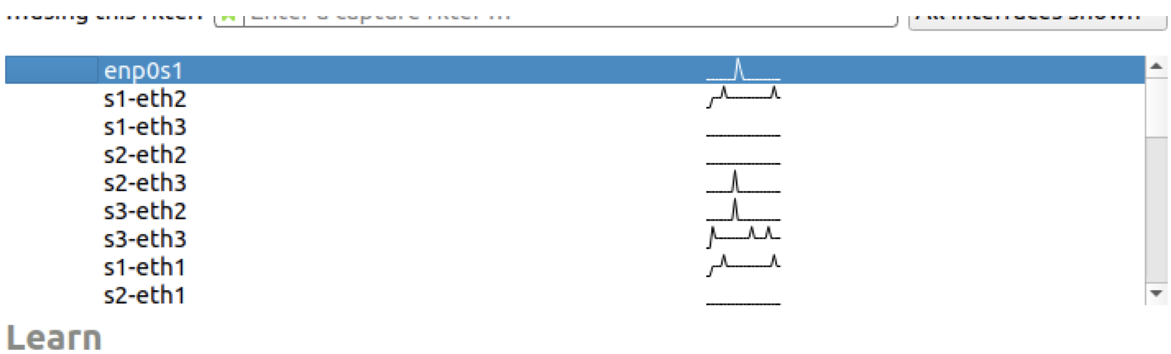


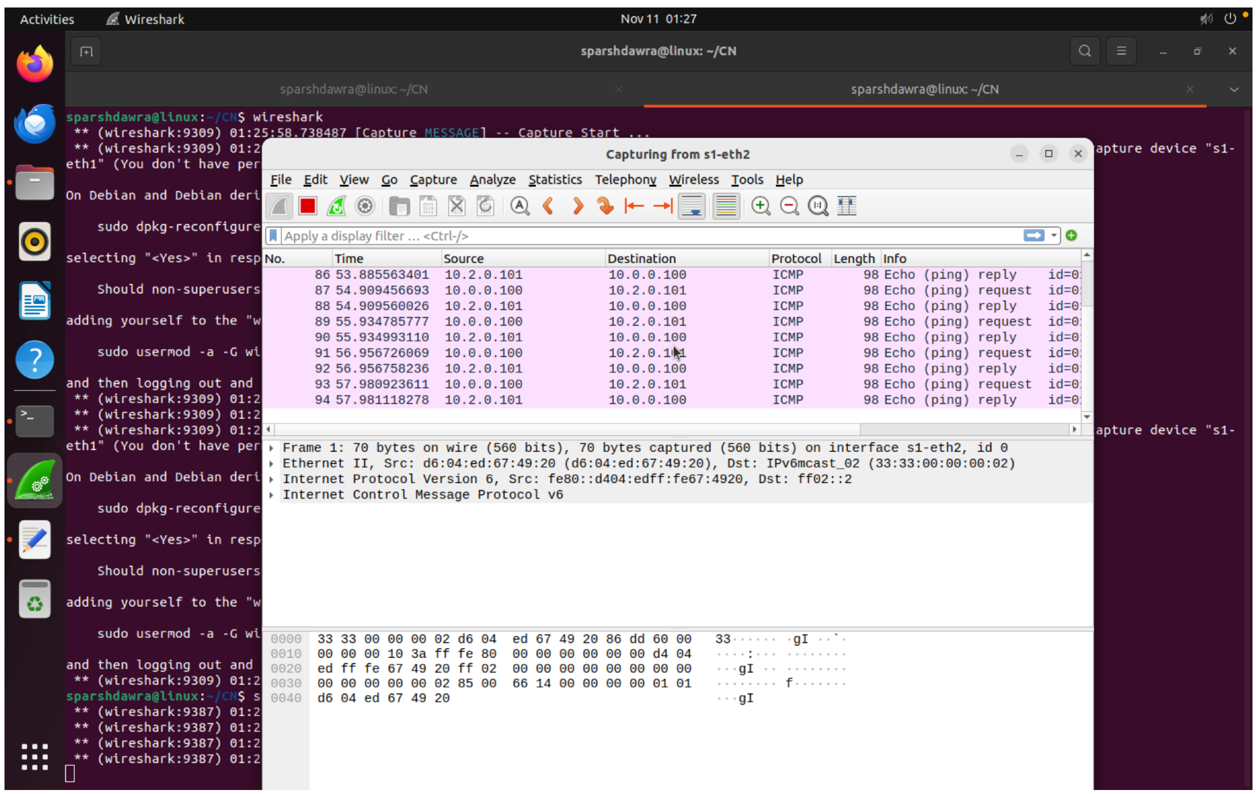
PingAll:



B) Observations:

We have pinged H1 to H6 and got the status using wireshark. I have generated the pcapng file for router ra using wireshark and here are some screenshots of it.





C) I have done the change in the net tabulation for changing the path of h1->ra->rc->h6 to h1->ra->rb->rc->h6. Here are the screenshots for ping as well as iperf.

"Node: h6"

```
root@linux:/home/sparshdawra/CN# iperf -c 10.0.0.100
------------------------------------------------------------
Client connecting to 10.0.0.100, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  1] local 10.2.0.101 port 53318 connected with 10.0.0.100 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-10.0008 sec  79.2 GBytes  68.1 Gbits/sec
root@linux:/home/sparshdawra/CN#
```

"Node: h1"

```
root@linux:/home/sparshdawra/CN# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  1] local 10.0.0.100 port 5001 connected with 10.2.0.101 port 53318
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-9.9997 sec  79.2 GBytes  68.1 Gbits/sec
```

For path h1->ra->rc->h6:

```
*** 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>
```

```
mininet> h1 ping h6
PING 10.2.0.101 (10.2.0.101) 56(84) bytes of data.
64 bytes from 10.2.0.101: icmp_seq=1 ttl=62 time=6.20 ms
64 bytes from 10.2.0.101: icmp_seq=2 ttl=62 time=1.74 ms
64 bytes from 10.2.0.101: icmp_seq=3 ttl=62 time=0.349 ms
64 bytes from 10.2.0.101: icmp_seq=4 ttl=62 time=0.096 ms
64 bytes from 10.2.0.101: icmp_seq=5 ttl=62 time=0.276 ms
64 bytes from 10.2.0.101: icmp_seq=6 ttl=62 time=0.533 ms
64 bytes from 10.2.0.101: icmp_seq=7 ttl=62 time=0.311 ms
^C
--- 10.2.0.101 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6099ms
rtt min/avg/max/mdev = 0.096/1.357/6.204/2.041 ms
```

For path h1->ra->rb->rc->h6:

```
*** 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 X X rb X
rb -> h1 h2 h3 h4 h5 h6 ra rc
rc -> X X h3 h4 h5 h6 X rb
*** Results: 8% dropped (66/72 received)
mininet>
```

```
mininet> h1 ping h6
PING 10.2.0.101 (10.2.0.101) 56(84) bytes of data.
64 bytes from 10.2.0.101: icmp_seq=1 ttl=61 time=5.49 ms
64 bytes from 10.2.0.101: icmp_seq=2 ttl=61 time=1.82 ms
64 bytes from 10.2.0.101: icmp_seq=3 ttl=61 time=0.329 ms
64 bytes from 10.2.0.101: icmp_seq=4 ttl=61 time=0.179 ms
64 bytes from 10.2.0.101: icmp_seq=5 ttl=61 time=0.156 ms
64 bytes from 10.2.0.101: icmp_seq=6 ttl=61 time=0.145 ms
64 bytes from 10.2.0.101: icmp_seq=7 ttl=61 time=0.148 ms
^C
--- 10.2.0.101 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6086ms
rtt min/avg/max/mdev = 0.145/1.180/5.487/1.846 ms
```

h1->ra

```
root@linux:/home/sparshdawra/CN# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=3.28 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=1.23 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.090 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.189 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.186 ms
^C
--- 10.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4039ms
rtt min/avg/max/mdev = 0.090/0.994/3.275/1.214 ms
root@linux:/home/sparshdawra/CN#
```

ra->rb

```
root@linux:/home/sparshdawra/CN# ping 10.1.0.1
PING 10.1.0.1 (10.1.0.1) 56(84) bytes of data.
64 bytes from 10.1.0.1: icmp_seq=1 ttl=64 time=0.233 ms
64 bytes from 10.1.0.1: icmp_seq=2 ttl=64 time=0.163 ms
64 bytes from 10.1.0.1: icmp_seq=3 ttl=64 time=0.123 ms
64 bytes from 10.1.0.1: icmp_seq=4 ttl=64 time=0.161 ms
64 bytes from 10.1.0.1: icmp_seq=5 ttl=64 time=0.193 ms
^C
--- 10.1.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4095ms
rtt min/avg/max/mdev = 0.123/0.174/0.233/0.036 ms
root@linux:/home/sparshdawra/CN#
```

rb->rc

```
root@linux:/home/sparshdawra/CN# ping 10.2.0.1
PING 10.2.0.1 (10.2.0.1) 56(84) bytes of data.
64 bytes from 10.2.0.1: icmp_seq=1 ttl=64 time=0.334 ms
64 bytes from 10.2.0.1: icmp_seq=2 ttl=64 time=0.035 ms
64 bytes from 10.2.0.1: icmp_seq=3 ttl=64 time=0.075 ms
64 bytes from 10.2.0.1: icmp_seq=4 ttl=64 time=0.047 ms
64 bytes from 10.2.0.1: icmp_seq=5 ttl=64 time=0.135 ms
^C
--- 10.2.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4105ms
rtt min/avg/max/mdev = 0.035/0.125/0.334/0.109 ms
root@linux:/home/sparshdawra/CN#
```

rc->h6

```
root@linux:/home/sparshdawra/CN# ping 10.2.0.101
PING 10.2.0.101 (10.2.0.101) 56(84) bytes of data.
64 bytes from 10.2.0.101: icmp_seq=1 ttl=64 time=3.96 ms
64 bytes from 10.2.0.101: icmp_seq=2 ttl=64 time=1.62 ms
64 bytes from 10.2.0.101: icmp_seq=3 ttl=64 time=0.228 ms
64 bytes from 10.2.0.101: icmp_seq=4 ttl=64 time=0.219 ms
64 bytes from 10.2.0.101: icmp_seq=5 ttl=64 time=0.181 ms
^C
--- 10.2.0.101 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4040ms
rtt min/avg/max/mdev = 0.181/1.241/3.963/1.465 ms
root@linux:/home/sparshdawra/CN#
```

ra->rc

```
root@linux:/home/sparshdawra/CN# ping 10.2.0.1
PING 10.2.0.1 (10.2.0.1) 56(84) bytes of data.
64 bytes from 10.2.0.1: icmp_seq=1 ttl=64 time=0.257 ms
64 bytes from 10.2.0.1: icmp_seq=2 ttl=64 time=0.087 ms
64 bytes from 10.2.0.1: icmp_seq=3 ttl=64 time=0.201 ms
64 bytes from 10.2.0.1: icmp_seq=4 ttl=64 time=0.143 ms
64 bytes from 10.2.0.1: icmp_seq=5 ttl=64 time=0.167 ms
^C
--- 10.2.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4074ms
rtt min/avg/max/mdev = 0.087/0.171/0.257/0.056 ms
root@linux:/home/sparshdawra/CN#
```

We can calculate the hop time between h1->ra and ra->rb etc. now,(taking avg value of RTT)
Latency1(h1->ra->rb->rc->h6) = RTT(ra)+RTT(rb)+RTT(rc)+RTT(h6) = 2.534 ms
Latency2(h1->ra->rc->h6) = RTT(ra)+RTT(rc)+RTT(h6) = 2.406 ms
Here, we can see that ra is not directly going to the rc router and it is taking the rb router in its part of connecting h1 to h6.

D) Here are the route tables:
For question a:

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.255.255.0   U     0      0        0 ra-eth1
10.0.2.0        0.0.0.0         255.255.255.0   U     0      0        0 r1
10.1.0.0        10.0.2.2        255.255.255.0   UG    0      0        0 r1
10.2.0.0        10.2.2.2        255.255.255.0   UG    0      0        0 r5
10.2.2.0        0.0.0.0         255.255.255.0   U     0      0        0 r5
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        10.0.2.1        255.255.255.0   UG    0      0        0 r2
10.0.2.0        0.0.0.0         255.255.255.0   U     0      0        0 r2
10.1.0.0        0.0.0.0         255.255.255.0   U     0      0        0 rb-eth1
10.1.2.0        0.0.0.0         255.255.255.0   U     0      0        0 r3
10.2.0.0        10.1.2.2        255.255.255.0   UG    0      0        0 r3
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        10.2.2.1        255.255.255.0   UG    0      0        0 r6
10.1.0.0        10.1.2.1        255.255.255.0   UG    0      0        0 r4
10.1.2.0        0.0.0.0         255.255.255.0   U     0      0        0 r4
10.2.0.0        0.0.0.0         255.255.255.0   U     0      0        0 rc-eth1
10.2.2.0        0.0.0.0         255.255.255.0   U     0      0        0 r6
```

For question c:

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.255.255.0   U     0      0        0 ra-eth1
10.0.2.0        0.0.0.0         255.255.255.0   U     0      0        0 r1
10.1.0.0        10.0.2.2        255.255.255.0   UG    0      0        0 r1
10.2.0.0        10.0.2.2        255.255.255.0   UG    0      0        0 r1
10.2.2.0        0.0.0.0         255.255.255.0   U     0      0        0 r5
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        10.0.2.1        255.255.255.0   UG    0      0        0 r2
10.0.2.0        0.0.0.0         255.255.255.0   U     0      0        0 r2
10.1.0.0        0.0.0.0         255.255.255.0   U     0      0        0 rb-eth1
10.1.2.0        0.0.0.0         255.255.255.0   U     0      0        0 r3
10.2.0.0        10.1.2.2        255.255.255.0   UG    0      0        0 r3
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        10.1.2.1        255.255.255.0   UG    0      0        0 r4
10.1.0.0        10.1.2.1        255.255.255.0   UG    0      0        0 r4
10.1.2.0        0.0.0.0         255.255.255.0   U     0      0        0 r4
10.2.0.0        0.0.0.0         255.255.255.0   U     0      0        0 rc-eth1
10.2.2.0        0.0.0.0         255.255.255.0   U     0      0        0 r6
```

<u>Part:2</u>

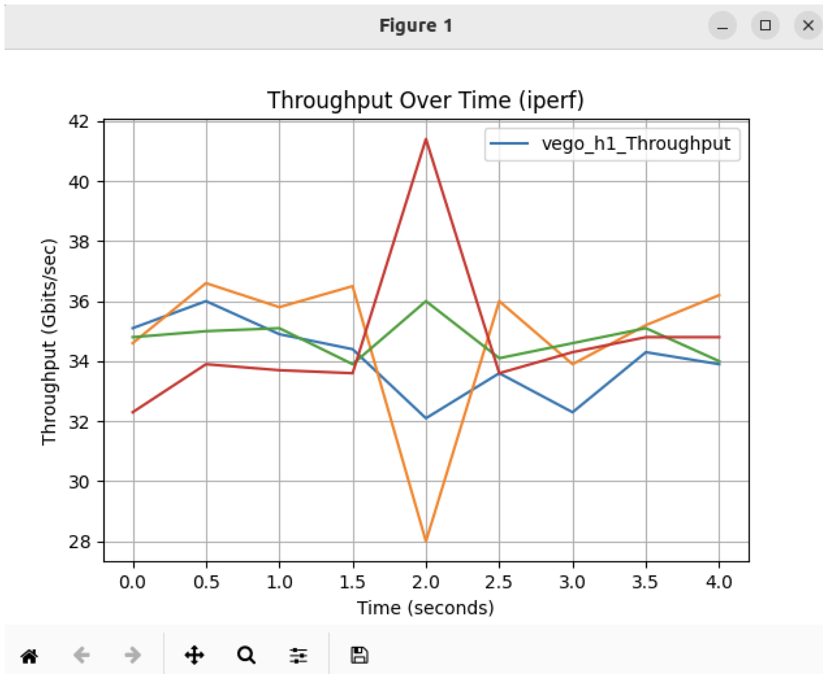
A)

```
sparshdawra@linux:~/CN$ sudo python3 Q2.py
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s2-eth1
h4 h4-eth0:s2-eth2
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet>
```

```
sparshdawra@linux:~/CN$ sudo python3 Q2.py --config=b
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s2-eth1
h4 h4-eth0:s2-eth2
mininet> h1 ping h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=4.14 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=1.73 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.119 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=0.199 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=0.247 ms
64 bytes from 10.0.0.4: icmp_seq=6 ttl=64 time=0.340 ms
64 bytes from 10.0.0.4: icmp_seq=7 ttl=64 time=0.262 ms
64 bytes from 10.0.0.4: icmp_seq=8 ttl=64 time=0.183 ms
64 bytes from 10.0.0.4: icmp_seq=9 ttl=64 time=0.219 ms
64 bytes from 10.0.0.4: icmp_seq=10 ttl=64 time=0.211 ms
64 bytes from 10.0.0.4: icmp_seq=11 ttl=64 time=0.103 ms
64 bytes from 10.0.0.4: icmp_seq=12 ttl=64 time=0.246 ms
64 bytes from 10.0.0.4: icmp_seq=13 ttl=64 time=0.274 ms
64 bytes from 10.0.0.4: icmp_seq=14 ttl=64 time=0.237 ms
64 bytes from 10.0.0.4: icmp_seq=15 ttl=64 time=0.200 ms
64 bytes from 10.0.0.4: icmp_seq=16 ttl=64 time=0.399 ms
64 bytes from 10.0.0.4: icmp_seq=17 ttl=64 time=0.246 ms
64 bytes from 10.0.0.4: icmp_seq=18 ttl=64 time=0.269 ms
64 bytes from 10.0.0.4: icmp_seq=19 ttl=64 time=0.270 ms
64 bytes from 10.0.0.4: icmp_seq=20 ttl=64 time=0.115 ms
64 bytes from 10.0.0.4: icmp_seq=21 ttl=64 time=0.119 ms
64 bytes from 10.0.0.4: icmp_seq=22 ttl=64 time=0.257 ms
64 bytes from 10.0.0.4: icmp_seq=23 ttl=64 time=0.268 ms
64 bytes from 10.0.0.4: icmp_seq=24 ttl=64 time=0.057 ms
64 bytes from 10.0.0.4: icmp_seq=25 ttl=64 time=0.215 ms
64 bytes from 10.0.0.4: icmp_seq=26 ttl=64 time=0.213 ms
64 bytes from 10.0.0.4: icmp_seq=27 ttl=64 time=0.261 ms
```

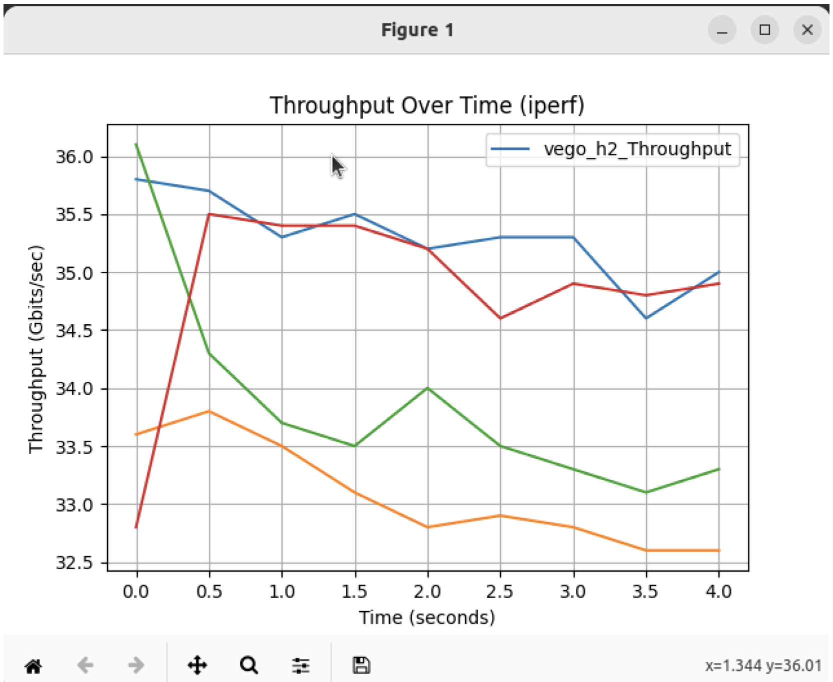
B) In all the graph,

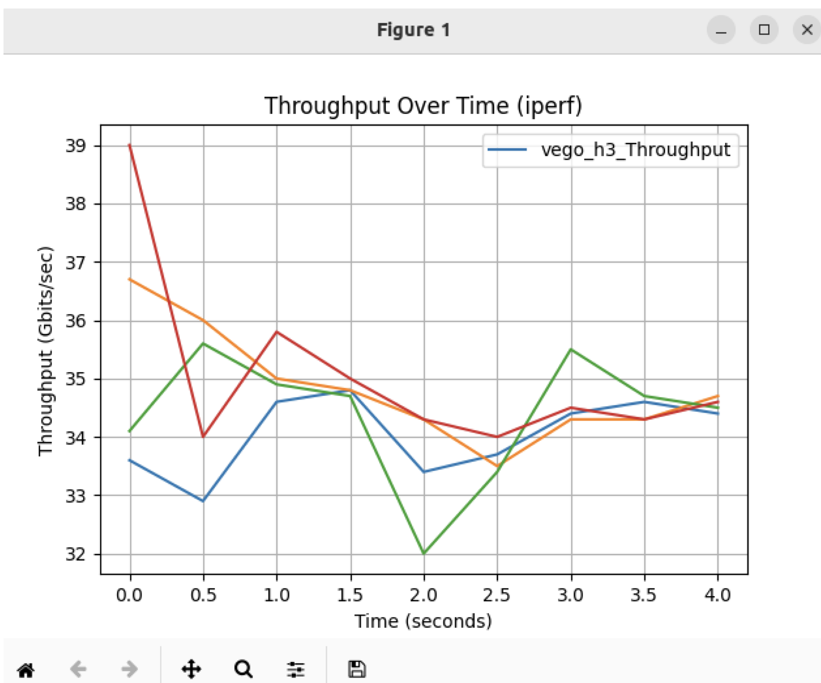
- Blue->Vegas
- Orange->Reno
- Red->Cubic
- Green->BBR



Vegas prioritizes low latency, reacting to round-trip time changes; Reno reacts traditionally to packet loss; Cubic optimizes high-speed network performance with scalable dynamics; BBR dynamically adjusts based on bottleneck bandwidth and round-trip time. Differences arise from varied congestion response algorithms, packet loss interpretation, adaptability to network conditions, and feedback mechanisms. The observed variations in throughput reflect each scheme's unique design for congestion control, demonstrating their distinct approaches and performances in specific network scenarios.

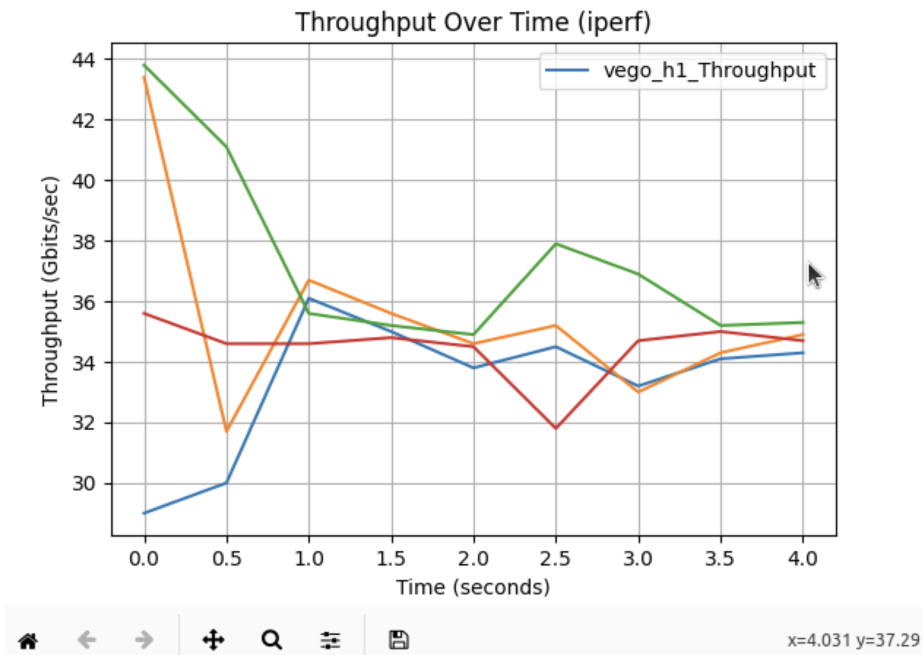
C) H1 client has been shown above. And H2 and H3 as clients are shown below.



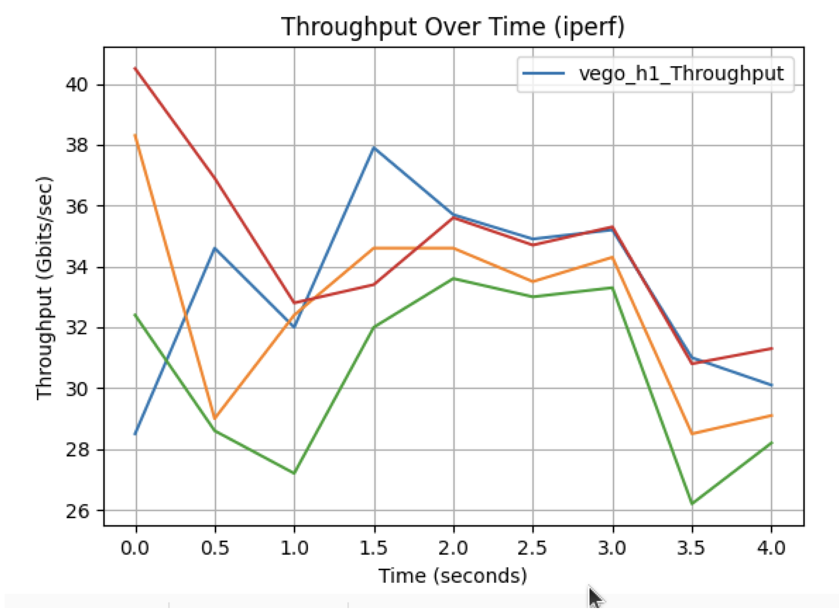


According to our Observations, the differences in average throughput among h1, h2, and h3 can be attributed to a combination of path characteristics, initial conditions, randomness in network events, competition for resources, and the dynamic nature of TCP congestion control algorithms. Analyzing the specific conditions of your network scenario and the behavior of each host's TCP connection can provide more insights into the observed throughput variations.

D) Link-Loss->1%



Link-Loss->3%



According to our observation, higher throughput observed in the case of 1% link loss compared to 3% link loss is likely due to the more moderate impact of packet loss on the TCP congestion control mechanisms. The TCP algorithms respond less aggressively to occasional and less severe packet loss, allowing for faster recovery and maintaining higher average throughput.