CS 433 Computer Networks (2023-24)

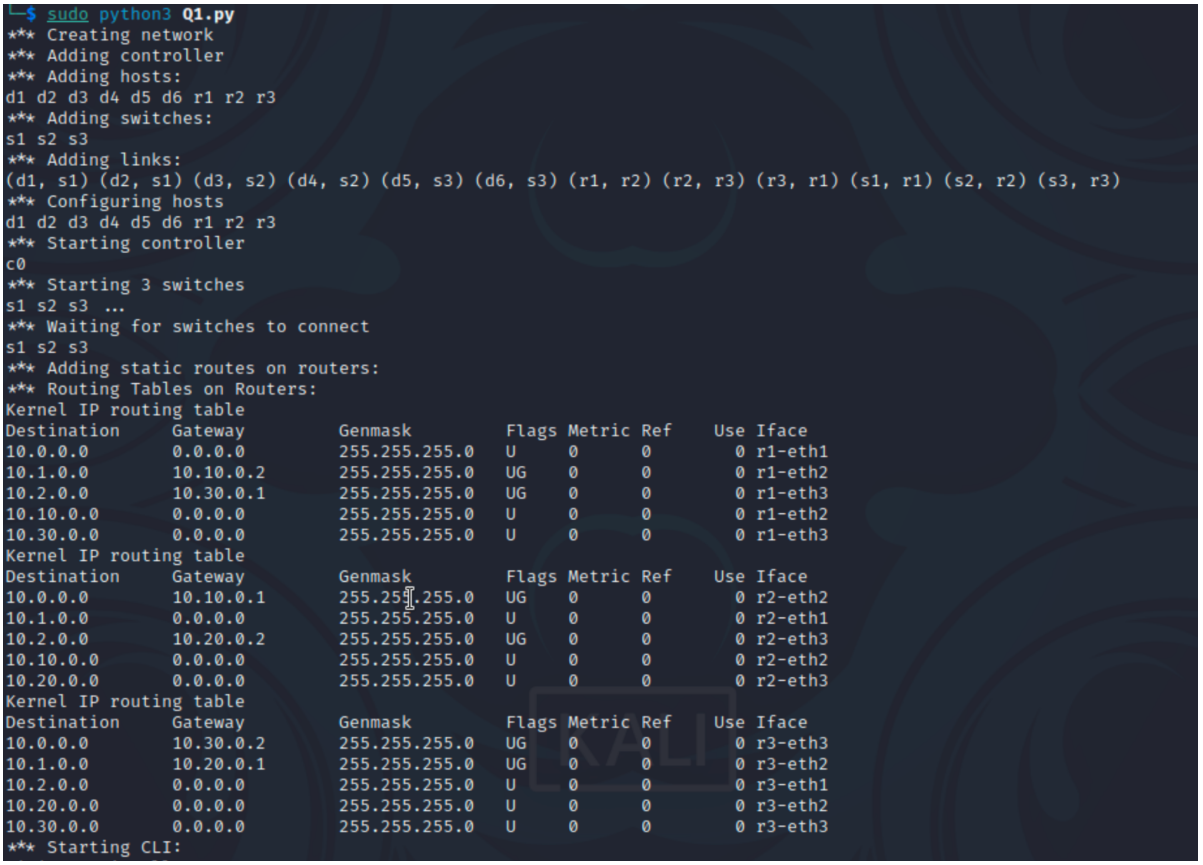
Assignment-02

Student 1 **Name**: Gali Sunny, **Roll No:** 20110067
Student 2 **Name:** Balu Karthik Ram **Roll No:** 20110036

Part I: Implement the routing functionality in mininet:

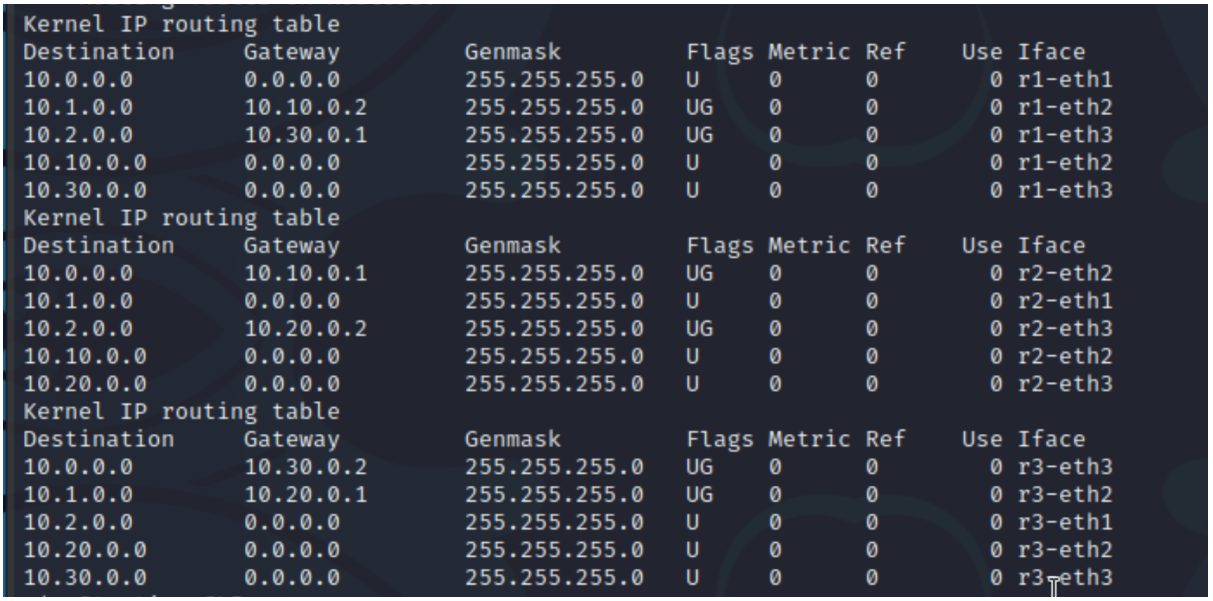
a.

Working of the Code



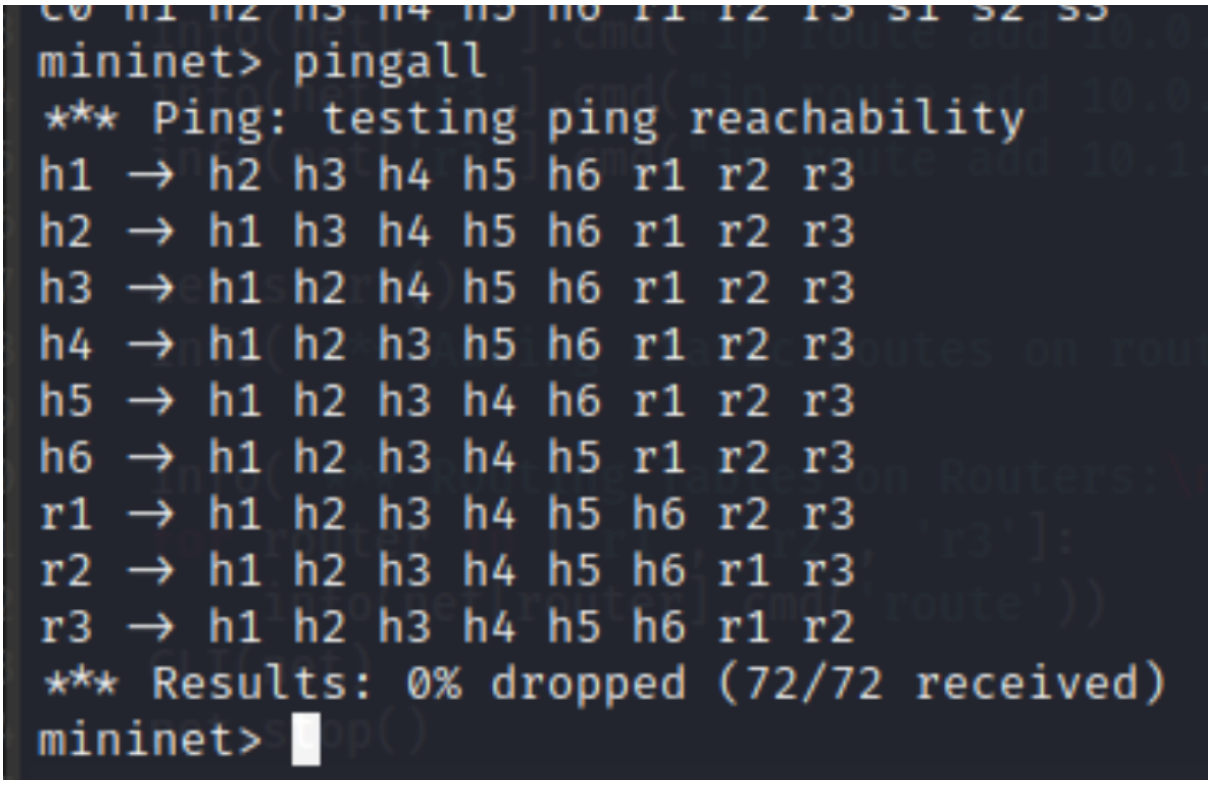
```
└─$ sudo python3 Q1.py
*** Creating network
*** Adding controller
*** Adding hosts:
d1 d2 d3 d4 d5 d6 r1 r2 r3
*** Adding switches:
s1 s2 s3
*** Adding links:
(d1, s1) (d2, s1) (d3, s2) (d4, s2) (d5, s3) (d6, s3) (r1, r2) (r2, r3) (r3, r1) (s1, r1) (s2, r2) (s3, r3)
*** Configuring hosts
d1 d2 d3 d4 d5 d6 r1 r2 r3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Waiting for switches to connect
s1 s2 s3
*** Adding static routes on routers:
*** Routing Tables on Routers:
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 r1-eth1
10.1.0.0        10.10.0.2       255.255.255.0   UG    0      0        0 r1-eth2
10.2.0.0        10.30.0.1       255.255.255.0   UG    0      0        0 r1-eth3
10.10.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r1-eth2
10.30.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r1-eth3
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        10.10.0.1       255.255.255.0   UG    0      0        0 r2-eth2
10.1.0.0        0.0.0.0         255.255.255.0   U     0      0        0 r2-eth1
10.2.0.0        10.20.0.2       255.255.255.0   UG    0      0        0 r2-eth3
10.10.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r2-eth2
10.20.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r2-eth3
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        10.30.0.2       255.255.255.0   UG    0      0        0 r3-eth3
10.1.0.0        10.20.0.1       255.255.255.0   UG    0      0        0 r3-eth2
10.2.0.0        0.0.0.0         255.255.255.0   U     0      0        0 r3-eth1
10.20.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r3-eth2
10.30.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r3-eth3
*** Starting CLI:
```

Routing Tables for r1,r2 and r3



```
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 r1-eth1
10.1.0.0        10.10.0.2       255.255.255.0   UG    0      0        0 r1-eth2
10.2.0.0        10.30.0.1       255.255.255.0   UG    0      0        0 r1-eth3
10.10.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r1-eth2
10.30.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r1-eth3
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        10.10.0.1       255.255.255.0   UG    0      0        0 r2-eth2
10.1.0.0        0.0.0.0         255.255.255.0   U     0      0        0 r2-eth1
10.2.0.0        10.20.0.2       255.255.255.0   UG    0      0        0 r2-eth3
10.10.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r2-eth2
10.20.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r2-eth3
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        10.30.0.2       255.255.255.0   UG    0      0        0 r3-eth3
10.1.0.0        10.20.0.1       255.255.255.0   UG    0      0        0 r3-eth2
10.2.0.0        0.0.0.0         255.255.255.0   U     0      0        0 r3-eth1
10.20.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r3-eth2
10.30.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r3-eth3
```

PingAll



```
mininet> pingall
*** Ping: testing ping reachability
h1 → h2 h3 h4 h5 h6 r1 r2 r3
h2 → h1 h3 h4 h5 h6 r1 r2 r3
h3 → h1 h2 h4 h5 h6 r1 r2 r3
h4 → h1 h2 h3 h5 h6 r1 r2 r3
h5 → h1 h2 h3 h4 h6 r1 r2 r3
h6 → h1 h2 h3 h4 h5 r1 r2 r3
r1 → h1 h2 h3 h4 h5 h6 r2 r3
r2 → h1 h2 h3 h4 h5 h6 r1 r3
r3 → h1 h2 h3 h4 h5 h6 r1 r2
*** Results: 0% dropped (72/72 received)
mininet>
```

b) Observations: Capture and show the wireshark/tcpdump

Capturing Packets from Router R1:
The file containing all the packets captured on router r1 is: 'r1_router_pingall.pcapng'

No.	Time	Source	Destination	Protocol	Length	Info
7	21.529095371	10.0.0.2	10.1.0.2	ICMP	98	Echo (ping) request id=0x
8	21.529897948	10.1.0.2	10.0.0.2	ICMP	98	Echo (ping) reply id=0x
9	21.532694925	10.0.0.2	10.1.0.3	ICMP	98	Echo (ping) request id=0x
10	21.533246712	10.1.0.3	10.0.0.2	ICMP	98	Echo (ping) reply id=0x
11	21.544487159	10.0.0.2	10.1.0.1	ICMP	98	Echo (ping) request id=0x
12	21.544506367	10.1.0.1	10.0.0.2	ICMP	98	Echo (ping) reply id=0x
13	21.551162228	10.0.0.3	10.1.0.2	ICMP	98	Echo (ping) request id=0x
14	21.551546142	10.1.0.2	10.0.0.3	ICMP	98	Echo (ping) reply id=0x
15	21.553406251	10.0.0.3	10.1.0.3	ICMP	98	Echo (ping) request id=0x
16	21.553835831	10.1.0.3	10.0.0.3	ICMP	98	Echo (ping) reply id=0x
17	21.562457842	10.0.0.3	10.1.0.1	ICMP	98	Echo (ping) request id=0x
18	21.562464217	10.1.0.1	10.0.0.3	ICMP	98	Echo (ping) reply id=0x
19	21.566775847	10.1.0.2	10.0.0.2	ICMP	98	Echo (ping) request id=0x
20	21.567286301	10.0.0.2	10.1.0.2	ICMP	98	Echo (ping) reply id=0x
21	21.569193077	10.1.0.2	10.0.0.3	ICMP	98	Echo (ping) request id=0x
22	21.569588116	10.0.0.3	10.1.0.2	ICMP	98	Echo (ping) reply id=0x
23	21.576961804	10.1.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x
24	21.576965054	10.0.0.1	10.1.0.2	ICMP	98	Echo (ping) reply id=0x
25	21.581955554	10.1.0.3	10.0.0.2	ICMP	98	Echo (ping) request id=0x
26	21.582237051	10.0.0.2	10.1.0.3	ICMP	98	Echo (ping) reply id=0x
27	21.583702789	10.1.0.3	10.0.0.3	ICMP	98	Echo (ping) request id=0x
28	21.583980620	10.0.0.3	10.1.0.3	ICMP	98	Echo (ping) reply id=0x
29	21.591123727	10.1.0.3	10.0.0.1	ICMP	98	Echo (ping) request id=0x
30	21.591131185	10.0.0.1	10.1.0.3	ICMP	98	Echo (ping) reply id=0x

c)

TraceRoute

For Default Route h1 -> r_a -> r_c -> h6.

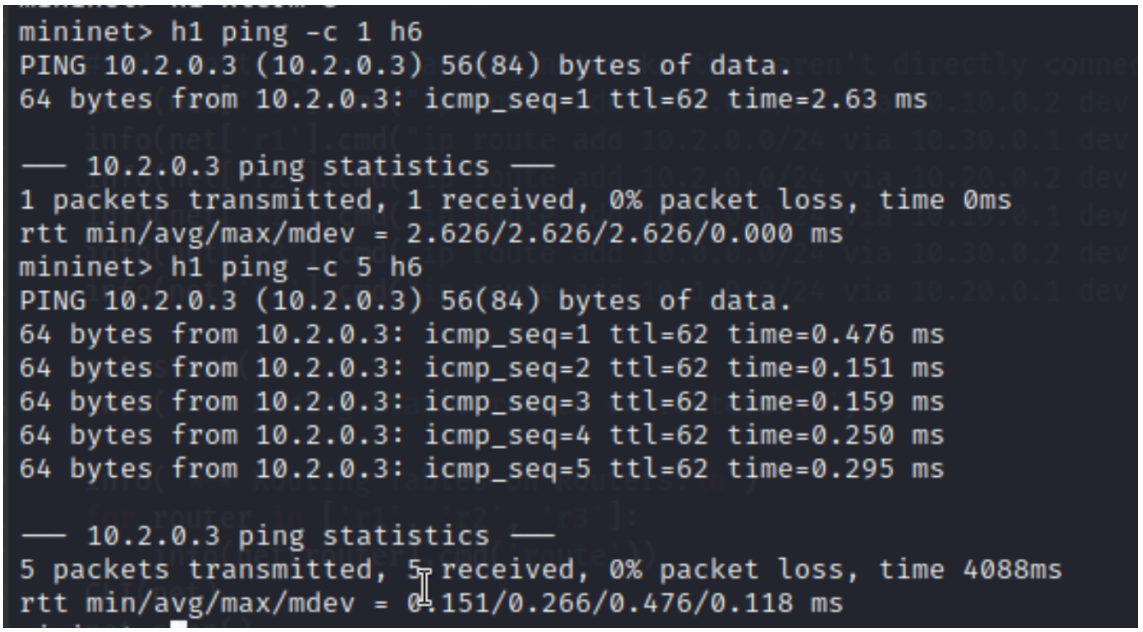
```
mininet> h1 traceroute h6
traceroute to 10.2.0.3 (10.2.0.3), 30 hops max, 60 byte packets
 1  10.0.0.1 (10.0.0.1)  3.097 ms  3.035 ms  3.041 ms
 2  10.30.0.1 (10.30.0.1)  3.048 ms  3.053 ms  3.058 ms
 3  10.2.0.3 (10.2.0.3)  4.559 ms  4.629 ms  4.638 ms
mininet>
```

For custom configuration(h1-> r_a -> r_b -> r_c -> h6)

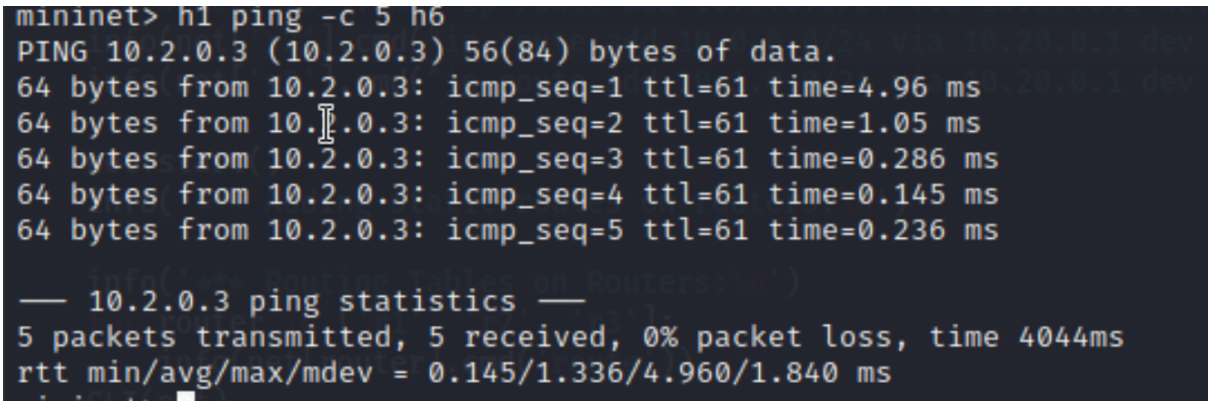
```
mininet> h1 traceroute h6
traceroute to 10.2.0.3 (10.2.0.3), 30 hops max, 60 byte packets
 1  10.0.0.1 (10.0.0.1)  5.487 ms  5.632 ms  5.962 ms
 2  10.10.0.2 (10.10.0.2)  6.293 ms  6.495 ms  6.677 ms
 3  10.20.0.2 (10.20.0.2)  7.311 ms  7.575 ms  7.818 ms
 4  10.2.0.3 (10.2.0.3)  9.794 ms  10.010 ms  10.268 ms
```

Ping

For Default Route h1 -> r_a -> r_c -> h6.



For custom configuration(h1-> r_a -> r_b -> r_c -> h6)



Measure the latency difference:

The ping results from the two paths. The latency difference is between the default path's(h1 -> ra -> rc -> h6) average ping time and the new path's (h1-> ra -> rb -> rc -> h6) average ping time.

Latency Difference = Avg ping time (h1-> ra -> rb -> rc -> h6) - Avg ping time (h1 -> ra -> rc -> h6)

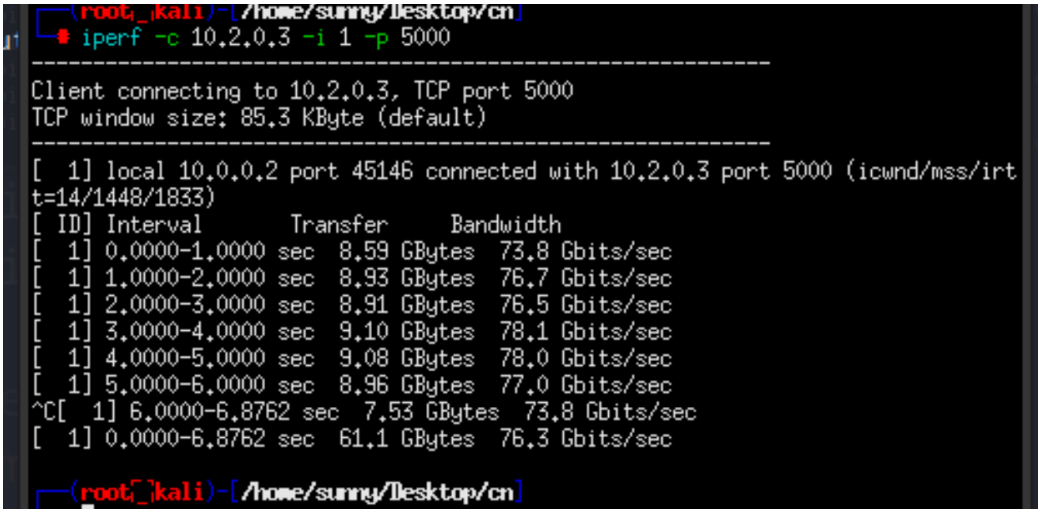
= 0.266 ms - 1.336 ms

= 1.07 ms

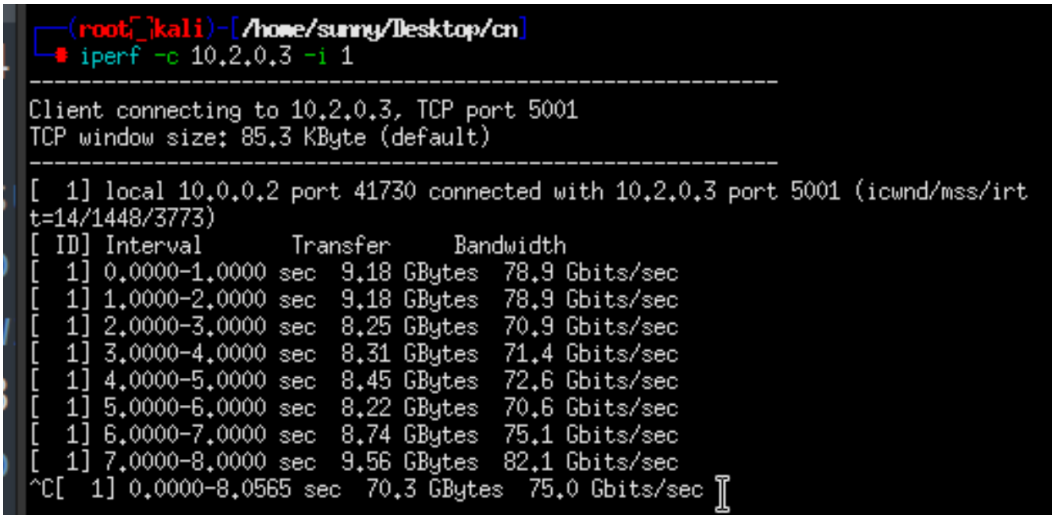
Therefore, Latency Difference = 1.07 ms.

Iperf

For Default Route h1 -> r_a -> r_c -> h6.



For custom configuration(h1-> r_a -> r_b -> r_c -> h6)

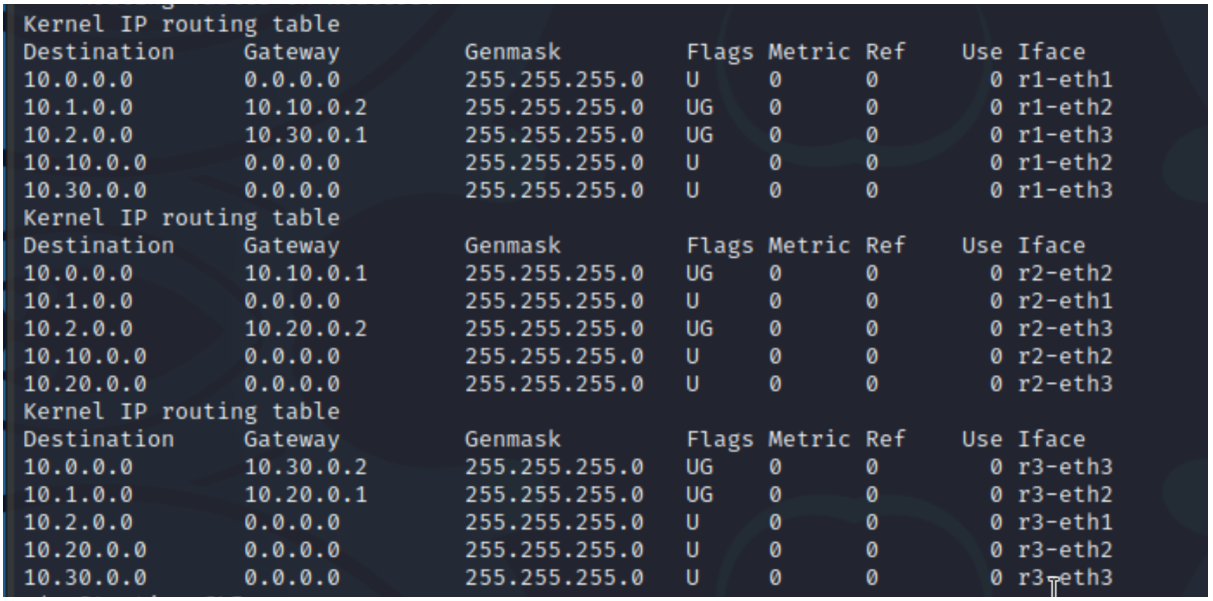


The Default route is having higher bandwidth and less initial round trip time when compared to the other route

d)

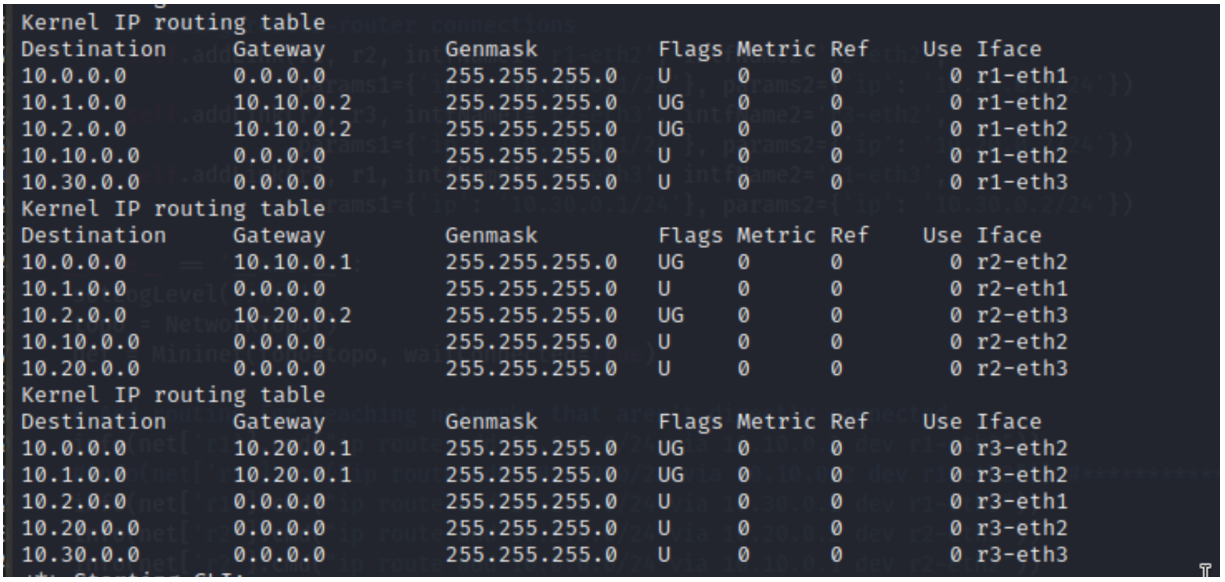
Default Path (h1 -> ra -> rc -> h6))

Routing Tables for ra,rb,rc



```
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 r1-eth1
10.1.0.0        10.10.0.2       255.255.255.0   UG    0      0        0 r1-eth2
10.2.0.0        10.30.0.1       255.255.255.0   UG    0      0        0 r1-eth3
10.10.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r1-eth2
10.30.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r1-eth3
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        10.10.0.1       255.255.255.0   UG    0      0        0 r2-eth2
10.1.0.0        0.0.0.0         255.255.255.0   U     0      0        0 r2-eth1
10.2.0.0        10.20.0.2       255.255.255.0   UG    0      0        0 r2-eth3
10.10.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r2-eth2
10.20.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r2-eth3
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        10.30.0.2       255.255.255.0   UG    0      0        0 r3-eth3
10.1.0.0        10.20.0.1       255.255.255.0   UG    0      0        0 r3-eth2
10.2.0.0        0.0.0.0         255.255.255.0   U     0      0        0 r3-eth1
10.20.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r3-eth2
10.30.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r3-eth3
```

For custom configuration(h1-> r_a -> r_b -> r_c -> h6)



```
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 r1-eth1
10.1.0.0        10.10.0.2       255.255.255.0   UG    0      0        0 r1-eth2
10.2.0.0        10.10.0.2       255.255.255.0   UG    0      0        0 r1-eth2
10.10.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r1-eth2
10.30.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r1-eth3
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        10.10.0.1       255.255.255.0   UG    0      0        0 r2-eth2
10.1.0.0        0.0.0.0         255.255.255.0   U     0      0        0 r2-eth1
10.2.0.0        10.20.0.2       255.255.255.0   UG    0      0        0 r2-eth3
10.10.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r2-eth2
10.20.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r2-eth3
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        10.20.0.1       255.255.255.0   UG    0      0        0 r3-eth2
10.1.0.0        10.20.0.1       255.255.255.0   UG    0      0        0 r3-eth2
10.2.0.0        0.0.0.0         255.255.255.0   U     0      0        0 r3-eth1
10.20.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r3-eth2
10.30.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r3-eth3
```

Part II: Throughput for different congestion control schemes:

a.

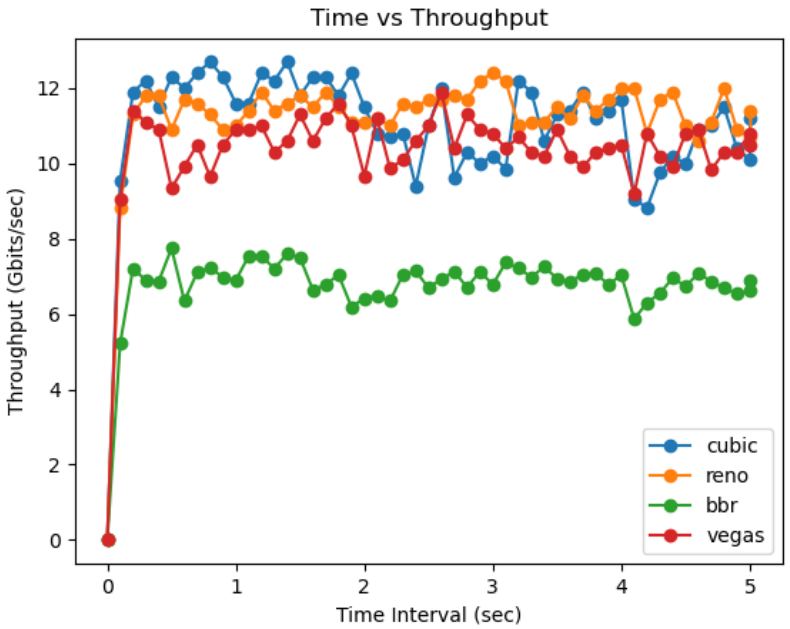
Working of Code:

```
┌──(sunny㉿kali)-[~/Desktop/cn]
└─$ sudo python3 Q2.py
None None None
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s1) (h3, s2) (h4, s2) (s1, s2)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
********* reno *********
iperf -s -t 5 -i 0.1 -p 5001 -f G -Z reno > ./a/text/h4/a_h4_server_reno.txt
iperf -c 10.0.0.4 -t 5 -i 0.1 -p 5001 -f G -Z reno > ./a/text/h2/a_h2_client_reno.txt
iperf -c 10.0.0.4 -t 5 -i 0.1 -p 5001 -f G -Z reno > ./a/text/h1/a_h1_client_reno.txt
iperf -c 10.0.0.4 -t 5 -i 0.1 -p 5001 -f G -Z reno > ./a/text/h3/a_h3_client_reno.txt
********* cubic *********
iperf -s -t 5 -i 0.1 -p 5001 -f G -Z cubic > ./a/text/h4/a_h4_server_cubic.txt
iperf -c 10.0.0.4 -t 5 -i 0.1 -p 5001 -f G -Z cubic > ./a/text/h2/a_h2_client_cubic.txt
iperf -c 10.0.0.4 -t 5 -i 0.1 -p 5001 -f G -Z cubic > ./a/text/h1/a_h1_client_cubic.txt
iperf -c 10.0.0.4 -t 5 -i 0.1 -p 5001 -f G -Z cubic > ./a/text/h3/a_h3_client_cubic.txt
********* bbr *********
iperf -s -t 5 -i 0.1 -p 5001 -f G -Z bbr > ./a/text/h4/a_h4_server_bbr.txt
iperf -c 10.0.0.4 -t 5 -i 0.1 -p 5001 -f G -Z bbr > ./a/text/h2/a_h2_client_bbr.txt
iperf -c 10.0.0.4 -t 5 -i 0.1 -p 5001 -f G -Z bbr > ./a/text/h1/a_h1_client_bbr.txt
iperf -c 10.0.0.4 -t 5 -i 0.1 -p 5001 -f G -Z bbr > ./a/text/h3/a_h3_client_bbr.txt
********* vegas *********
```

b.

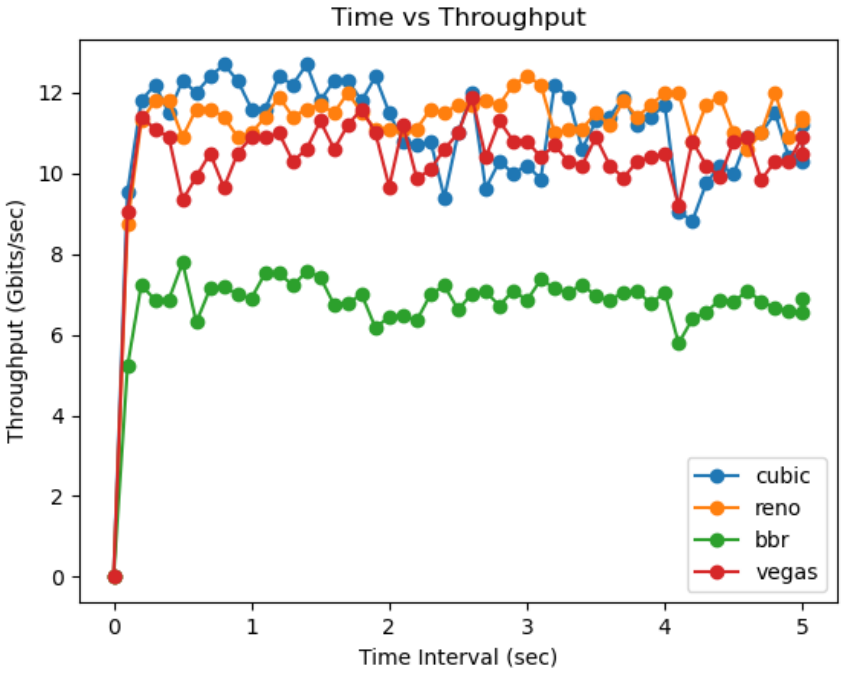
(i) H1 - client, for all congestion Control Schemes (Vegas, Reno, Cubic, BBR):

Plots will be stored in ./b/plots



H1-client

(ii) H4 - Server, for all Congestion Control (Vegas, Reno, Cubic, BBR):



H4-Server

Observations

BBR congestion control has less throughput than cubic, reno, and vegas. While Cubic in the first phase had higher throughput, it decreased later. Cubic is aggressive at increasing the congestion window and less aggressive at decreasing it.

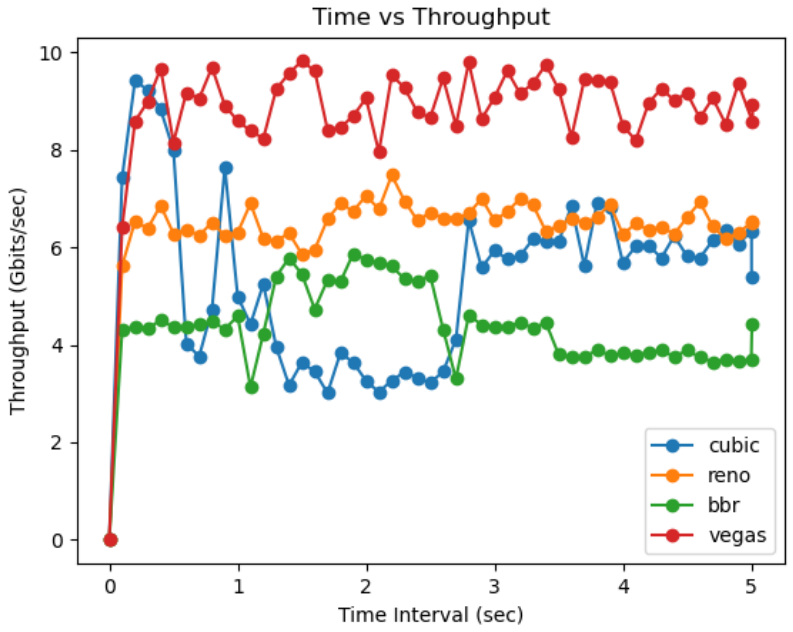
Reno has a slow start. It starts with a small congestion window and increases exponentially until it detects congestion.

Vegas is also similar to Reno.

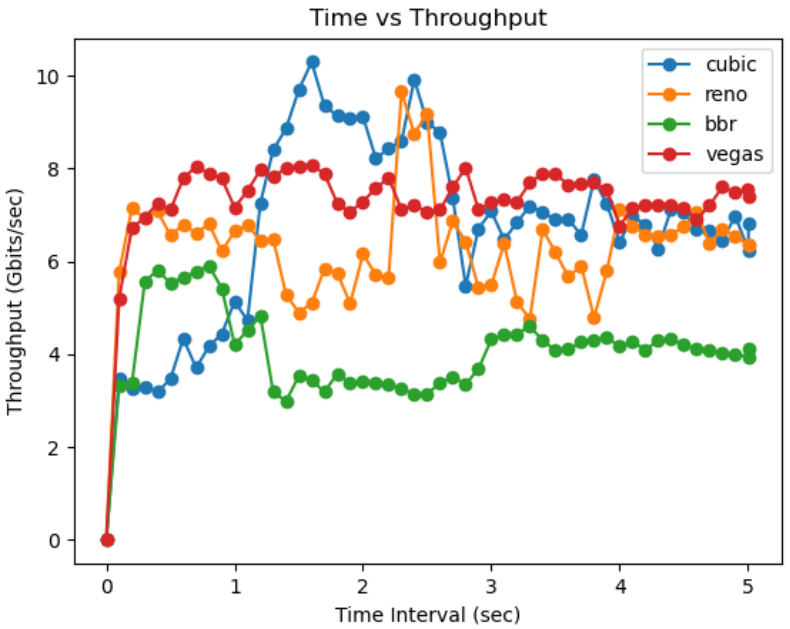
C.

Plots For Each Host

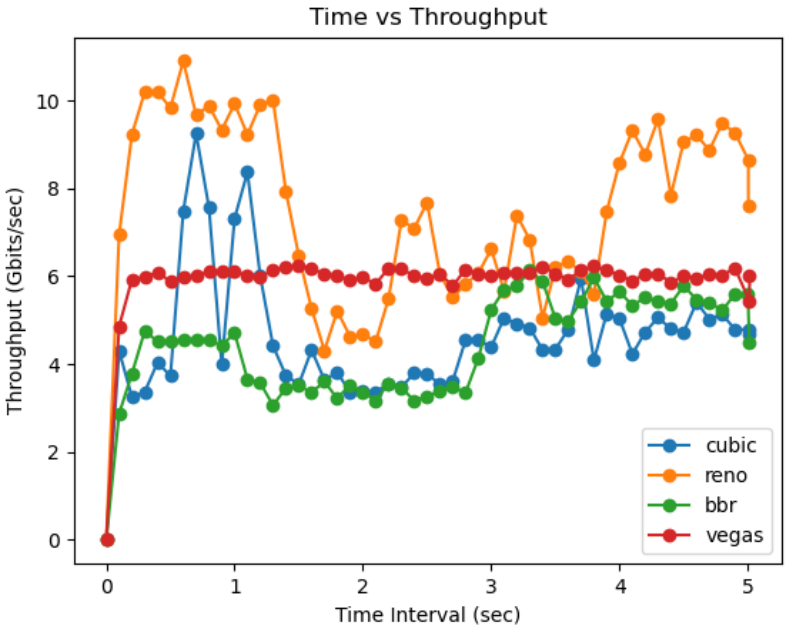
(i) H1 - client, for all congestion Control (Vegas, Reno, Cubic, BBR):



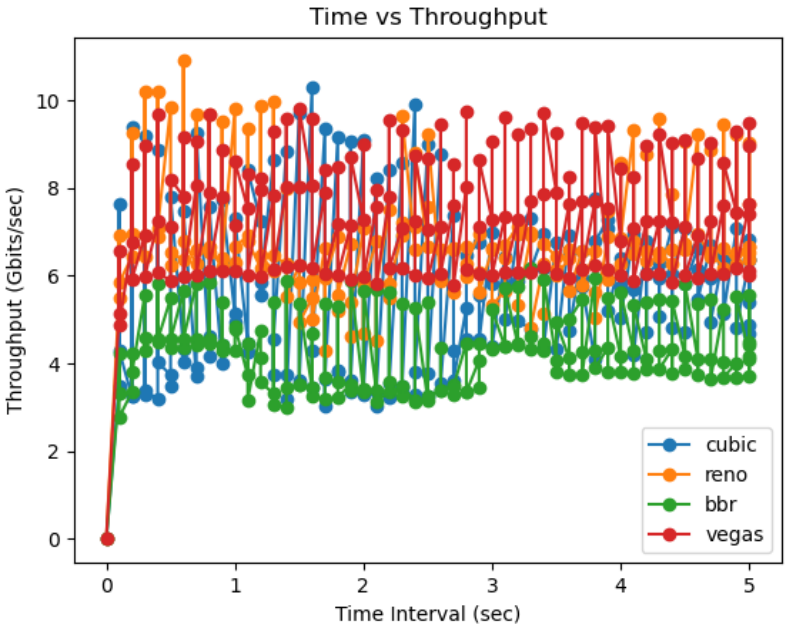
(ii) H2 - client, for all congestion Control (Vegas, Reno, Cubic, BBR):



(iii) H3 - client, for all congestion Control (Vegas, Reno, Cubic, BBR):

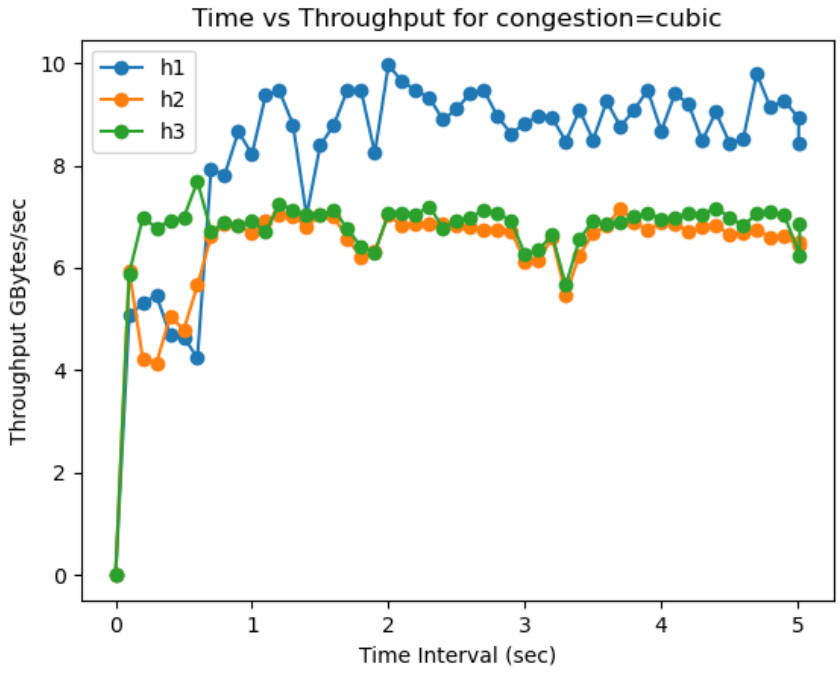


(iv) H4 - server, for all congestion Control (Vegas, Reno, Cubic, BBR):

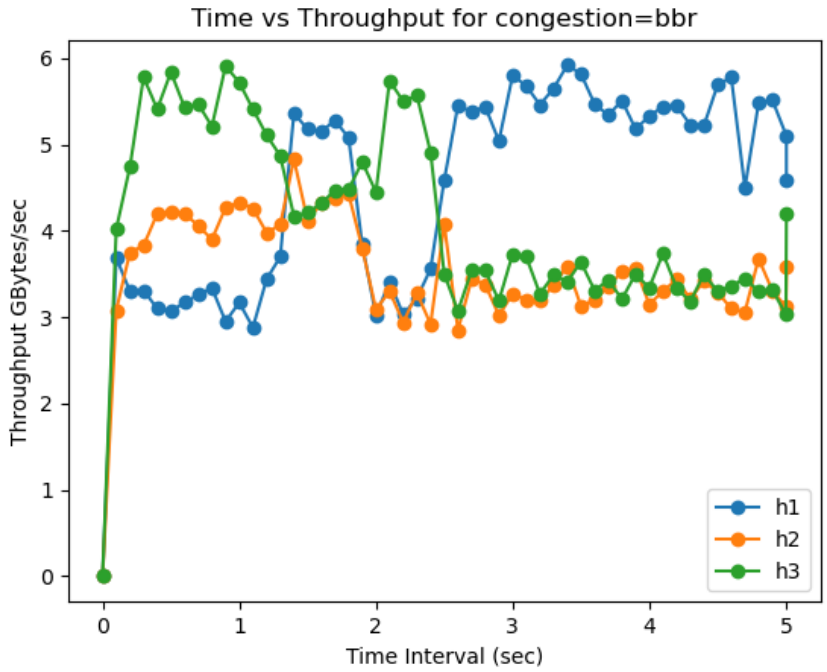


Plots For Each Congestion

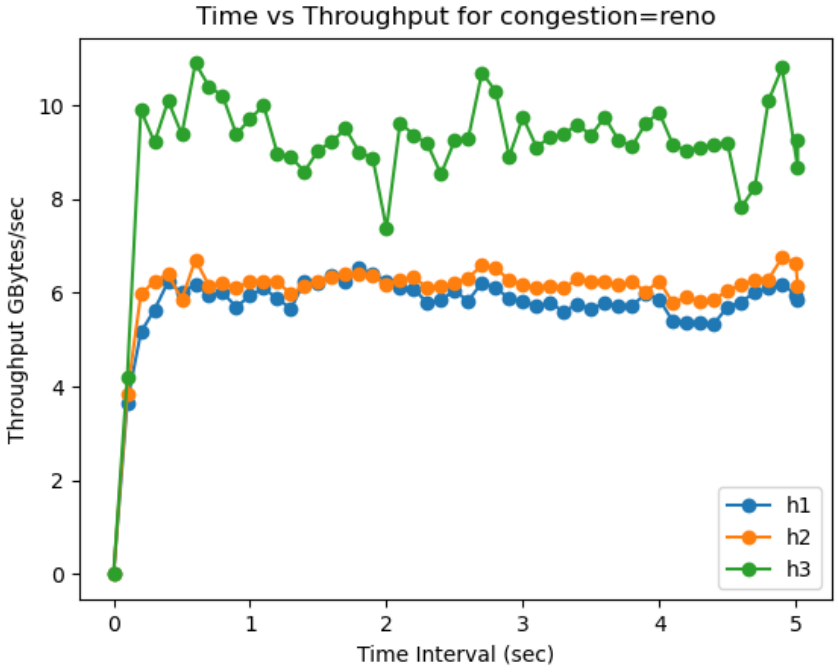
(I) Cubic



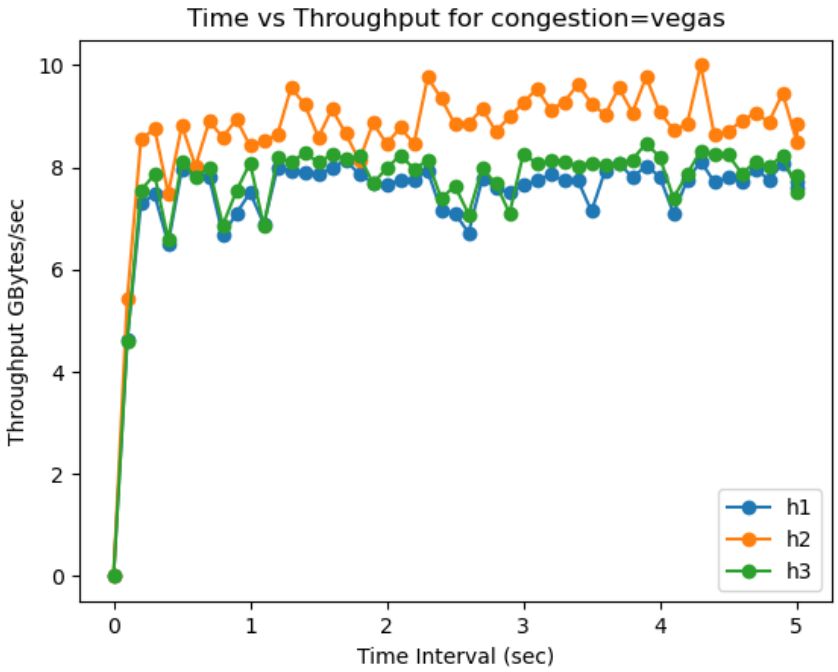
(II) BBR



(III) Reno



(IV) Vegas



Observations

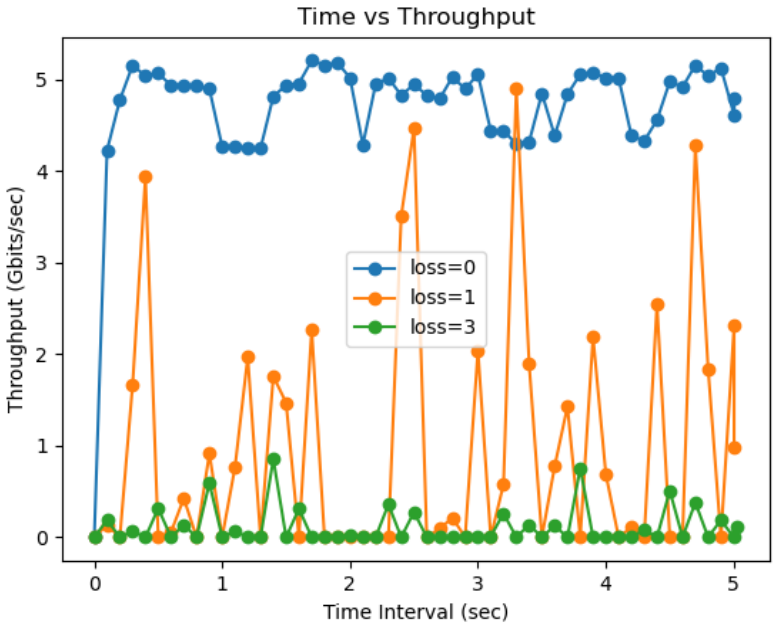
In Vegas, the Throughput is fairly distributed between hosts and the throughput hasn't changed abruptly with the changing network conditions. This might be because Vegas use RTT to control congestion

In BBR, the utilisation of the network links is less compared to the other congestion control algorithms

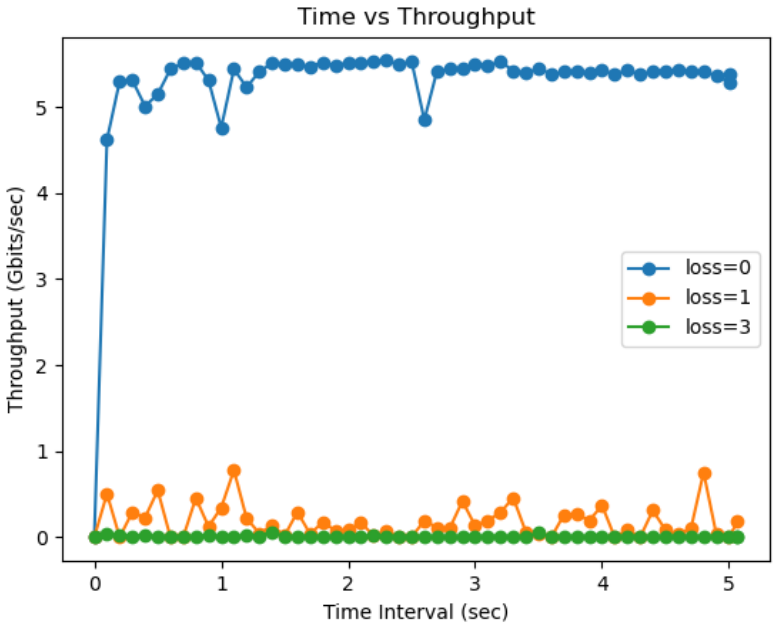
d)

(i) For H1 - Client

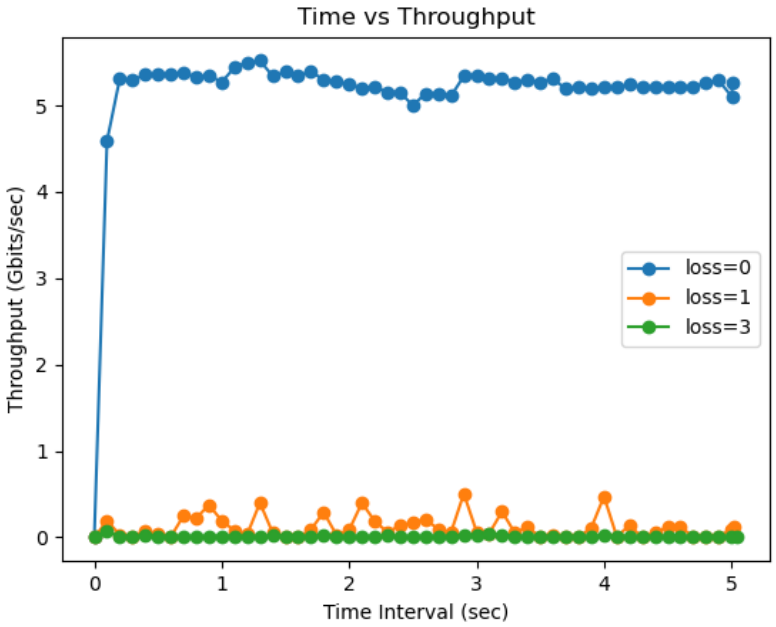
a. Congestion - BBR



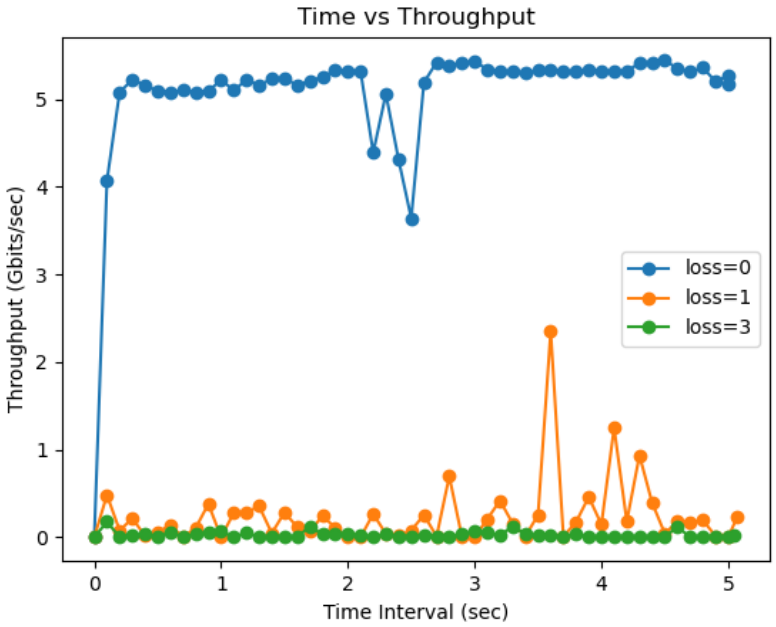
b. Congestion - Reno



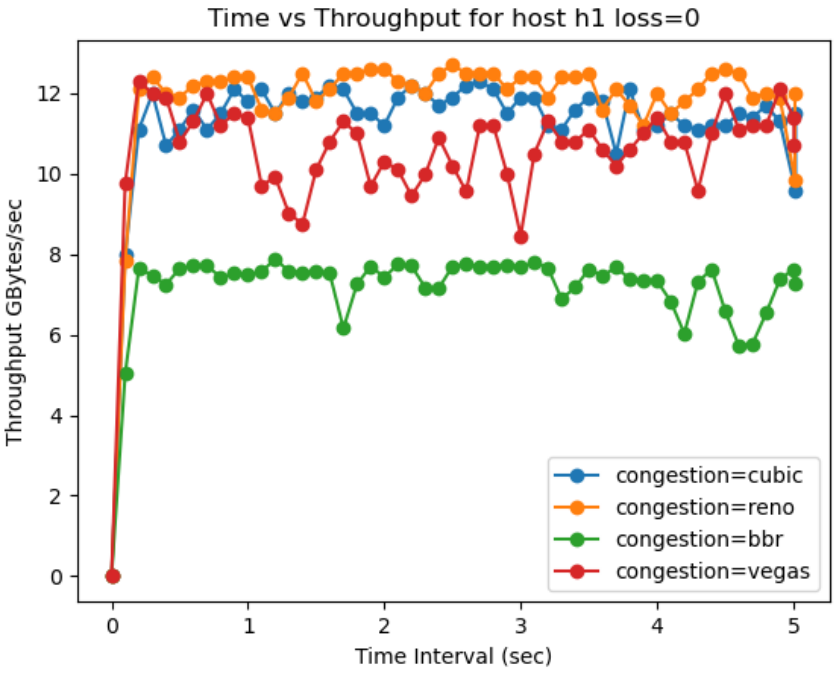
c. Congestion - Vegas

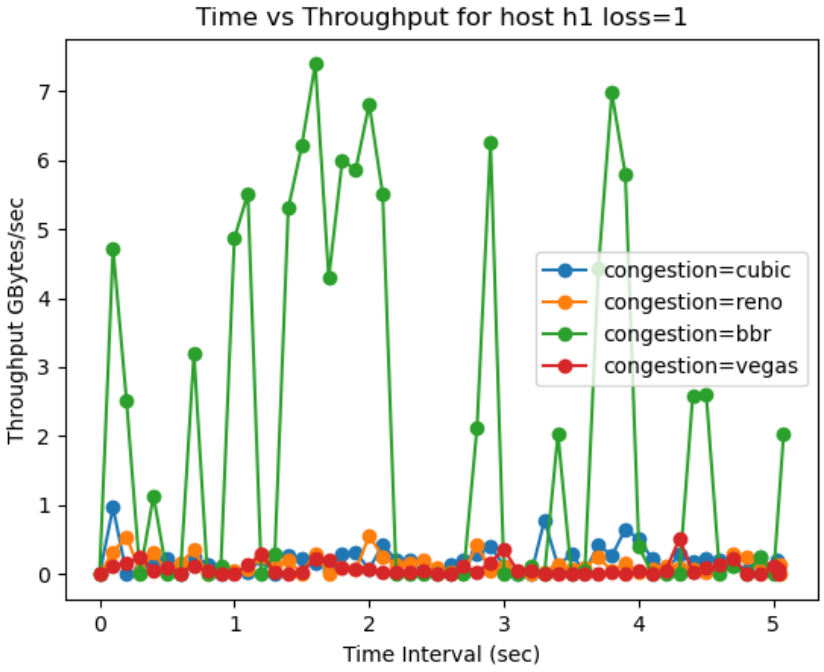


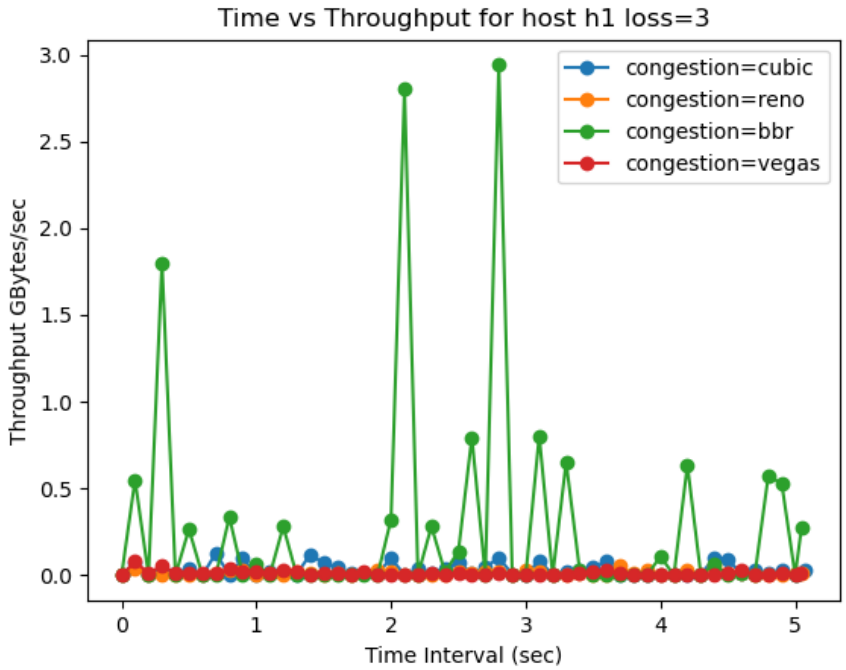
d. Congestion - Cubic



(II) For each Loss Percentage Throughput analysis on Host h1







Observations:

As you can see, the throughput for all four congestion control schemes is slightly lower with 3% link loss than with 1% link loss. This is because link loss can cause packets to be dropped, which reduces the amount of data that can be transmitted over the network.

BBR congestion control scheme that achieves the highest throughput, with 1%, 3% link loss.

Vegas having the least fluctuations than others, as this is a delay-based congestion control algorithm

Vegas and Reno are the least aggressive congestion control schemes, and they achieve the lowest throughput compared to others. Vegas and Reno are also the most conservative congestion control schemes, and they are less likely to cause congestion on the network.

References:

https://stackoverflow.com/questions/51160836/can-i-dynamically-modify-link-characteristics-during-a-mininet-emulation

https://linuxthrill.blogspot.com/2016/04/iperf-test-network-throughput-delay.html

https://stackoverflow.com/questions/46595423/mininet-how-to-create-a-topology-with-two-routers-and-their-respective-hosts