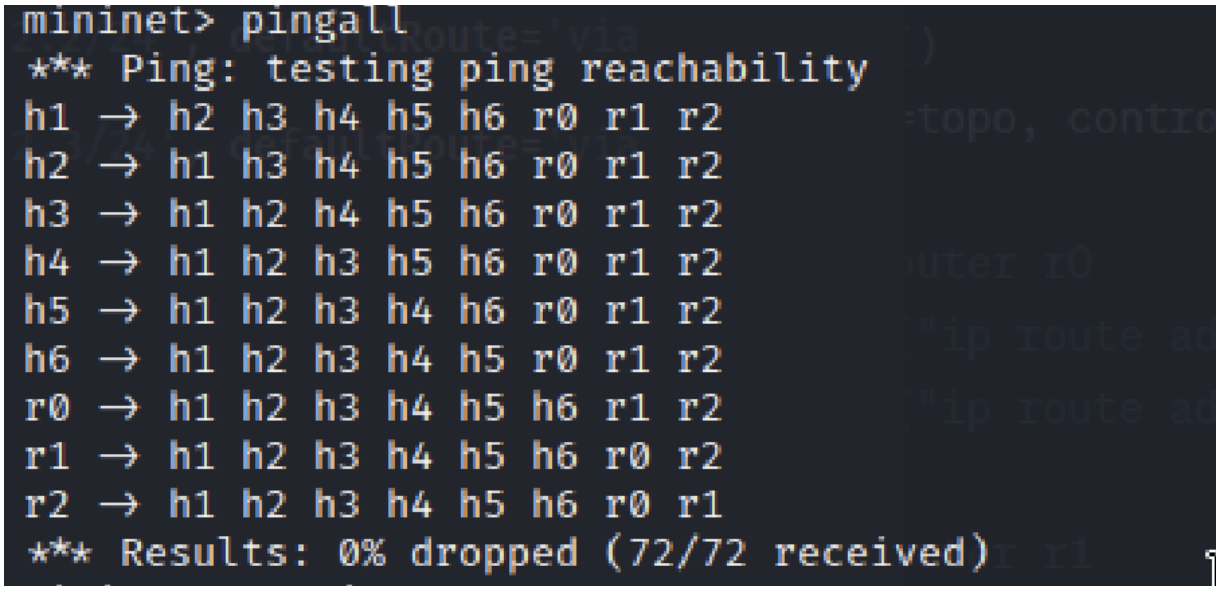
Assignment-02 **CS 433 Computer Networks (2023-24)**

Harshita Ramchandani 20110074
Ishani Chogle 20110074

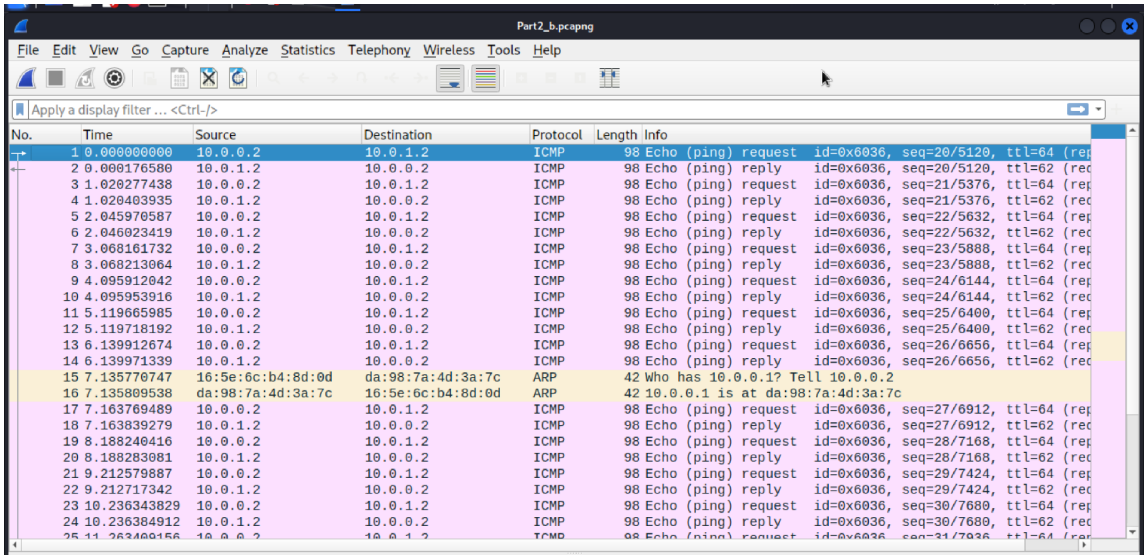
For code pls refer [here]

a. **Implementation** of the custom topology using mininet. ***(10 pts)*** (Every host should be able to send packets to every other host. You are to submit the code, as well as provide screenshots as proof of its working)

   Proof of its implementation:



b. **Observations**: Capture and show the wireshark/tcpdump (packets) for the route setup on any one of the routers.***(10 pts)***



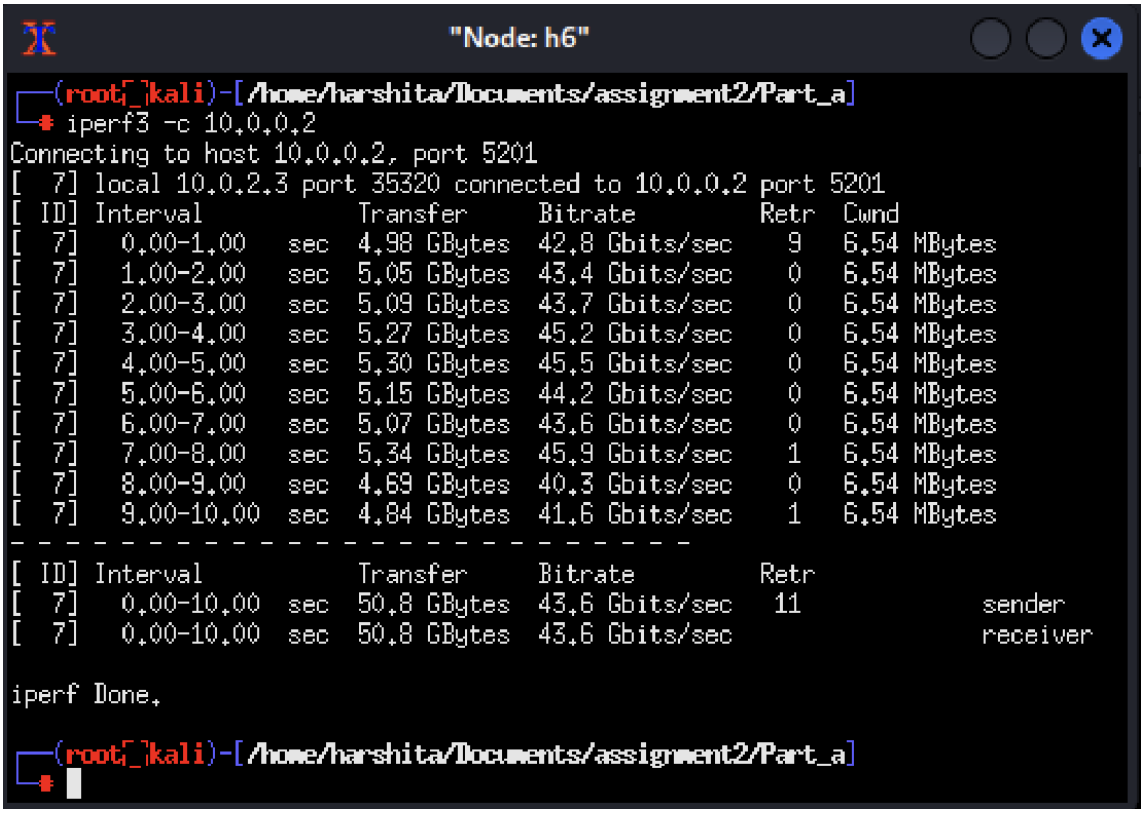
c. Vary the default routing and measure the latency difference. (i.e. the default route for a packet from h1 to h6 would be h1 -> r_a -> r_c -> h6. Try to vary the path so that it takes the path h1-> r_a -> r_b -> r_c -> h6. Provide the screenshots as proof for the latency difference, both ping as well as iperf) ***(10 pts)***

The route and result via ping:

```
mininet> h1 ping -R h6
PING 10.0.2.3 (10.0.2.3) 56(124) bytes of data.
64 bytes from 10.0.2.3: icmp_seq=1 ttl=62 time=0.387 ms
RR:     10.0.0.2
        10.0.5.1
        10.0.2.1
        10.0.2.3
        10.0.2.3
        10.0.5.2
        10.0.0.1
        10.0.0.2

64 bytes from 10.0.2.3: icmp_seq=2 ttl=62 time=0.124 ms (same route)
64 bytes from 10.0.2.3: icmp_seq=3 ttl=62 time=0.118 ms (same route)
64 bytes from 10.0.2.3: icmp_seq=4 ttl=62 time=0.153 ms (same route)
```

Via iperf:

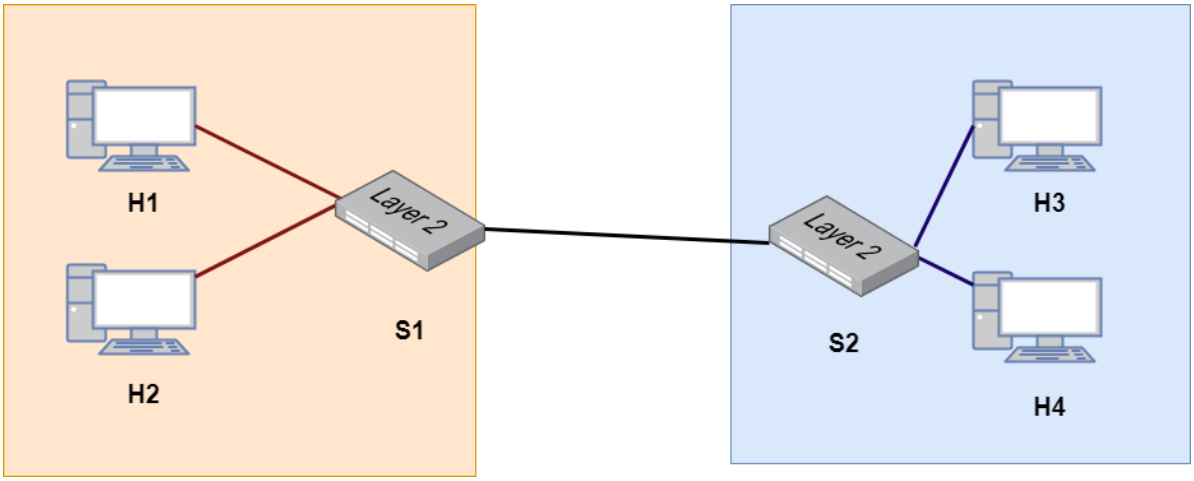


d. Dump the routing tables for all the routers for both of the above questions. ***(10 pts)***

```
mininet> r0 route -n
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 r0-eth0
10.0.1.0        10.0.3.2        255.255.255.0   UG    0      0        0 r0-eth1
10.0.2.0        10.0.5.2        255.255.255.0   UG    0      0        0 r0-eth2
10.0.3.0        0.0.0.0         255.255.255.0   U     0      0        0 r0-eth1
10.0.5.0        0.0.0.0         255.255.255.0   U     0      0        0 r0-eth2
mininet> r1 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        10.0.3.1        255.255.255.0   UG    0      0        0 r1-eth1
10.0.1.0        0.0.0.0         255.255.255.0   U     0      0        0 r1-eth0
10.0.2.0        10.0.4.2        255.255.255.0   UG    0      0        0 r1-eth2
10.0.3.0        0.0.0.0         255.255.255.0   U     0      0        0 r1-eth1
10.0.4.0        0.0.0.0         255.255.255.0   U     0      0        0 r1-eth2
mininet> r2 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        10.0.5.1        255.255.255.0   UG    0      0        0 r2-eth2
10.0.1.0        10.0.4.1        255.255.255.0   UG    0      0        0 r2-eth1
10.0.2.0        0.0.0.0         255.255.255.0   U     0      0        0 r2-eth0
10.0.4.0        0.0.0.0         255.255.255.0   U     0      0        0 r2-eth1
10.0.5.0        0.0.0.0         255.255.255.0   U     0      0        0 r2-eth2
```

Part II: Throughput for different congestion control schemes. (35 points)

Create a mininet topology as shown in the diagram below. H1 to H4 are four hosts connected to switches S1 and S2. Run a TCP Server that accepts data in the H4 system. H1, H2, H3 are the TCP clients that connect with the server (H4) and sends data packets.

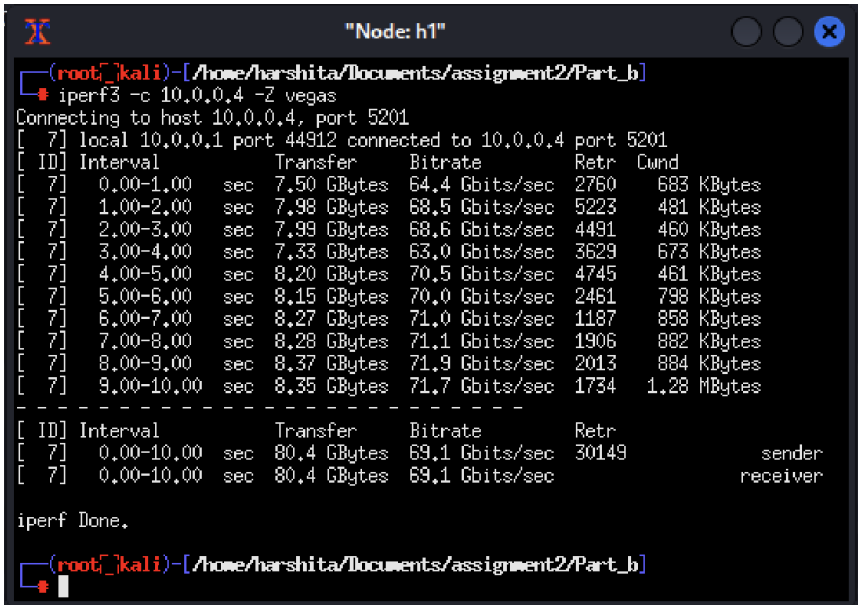


a. Implementation of the mininet topology and TCP client-server program (for packet generation).

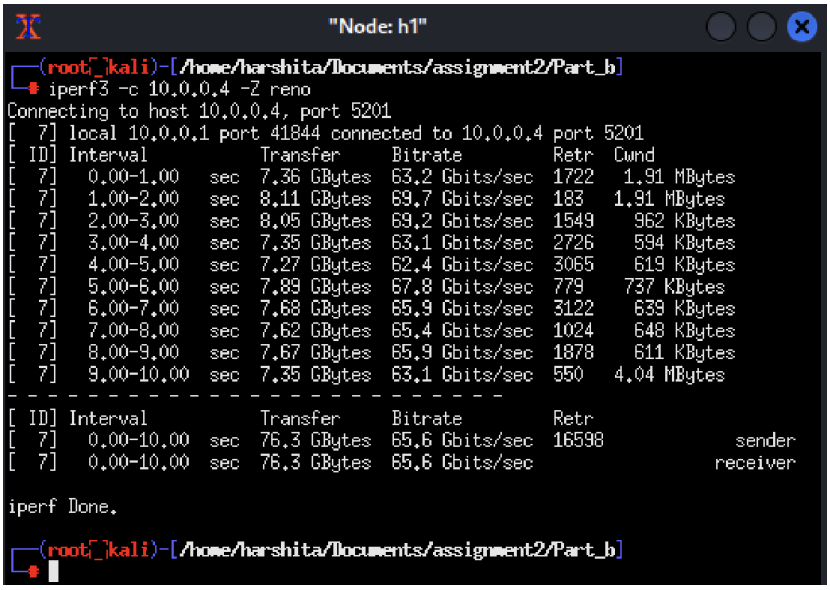
```
mininet> pingall
*** Ping: testing ping reachability
h1 → h2 h3 h4
h2 → h1 h3 h4
h3 → h1 h2 h4
h4 → h1 h2 h3
*** Results: 0% dropped (12/12 received)
```

The TCP client-server program should run with the required configuration for question (b) when `--config=b` and with required configuration for question (c) when `--config=c` Other configurable parameters like congestion control scheme, link loss should also be taken as command line arguments to the program. **(5 points)**

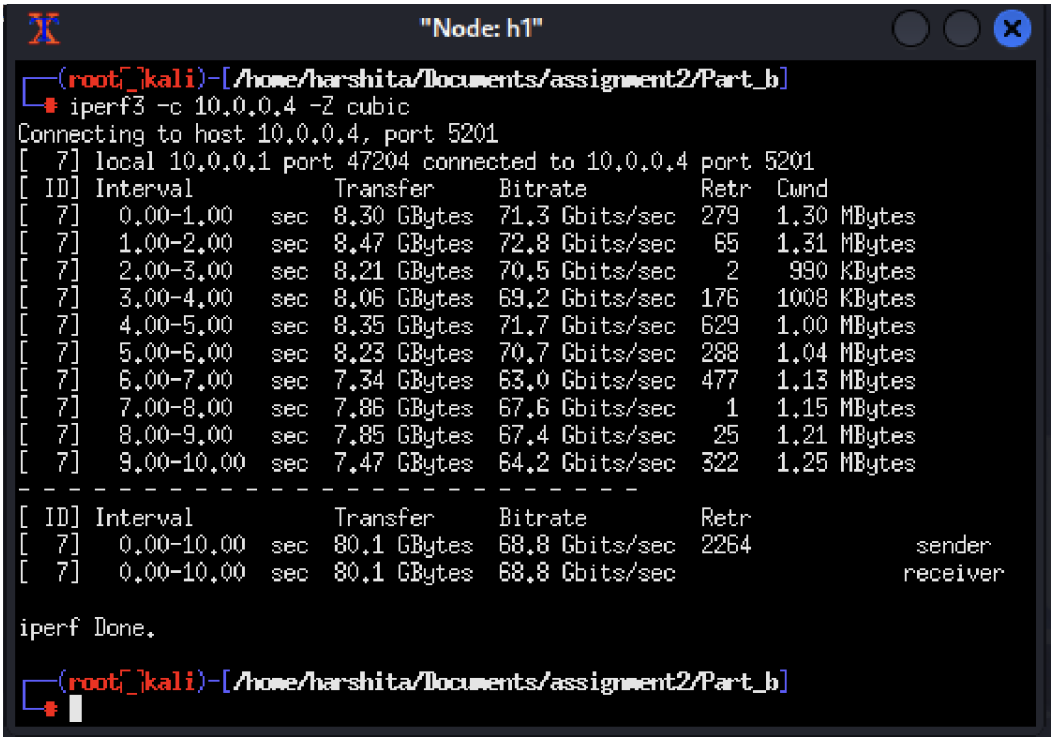
b. Run the client on H1 and the server on H4. Report and reason the throughput over time for each congestion control scheme. **(10 points)**
For vegas:



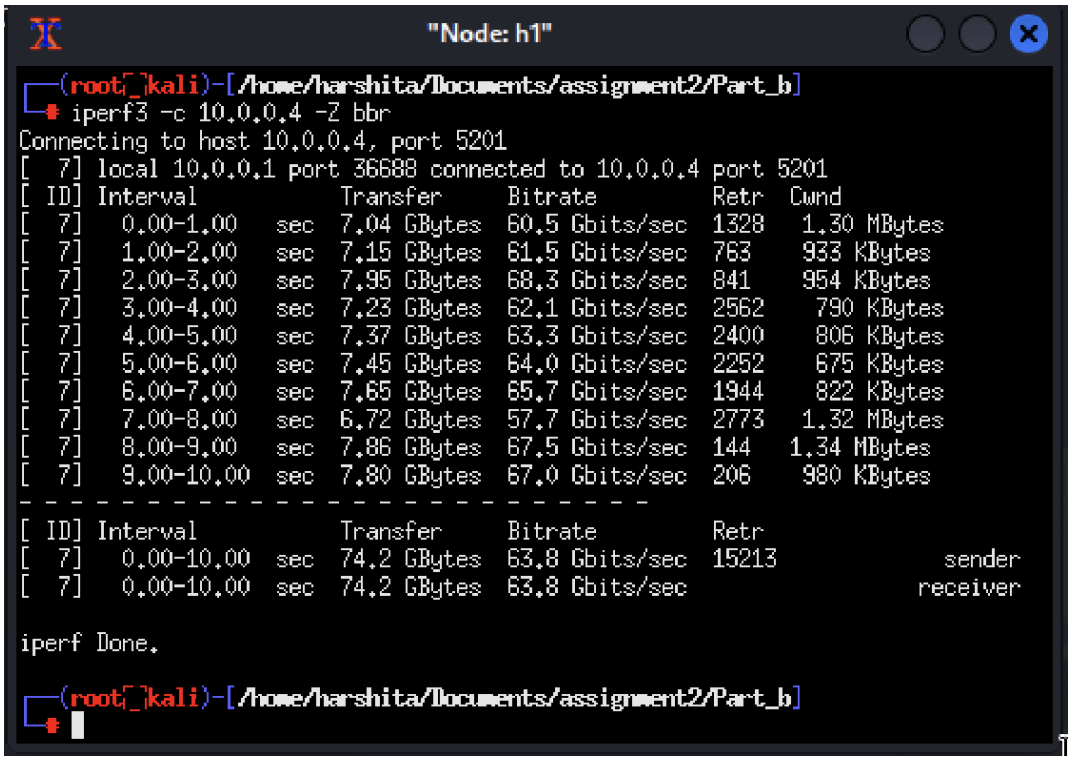
For reno:



For cubic:


"Node: h1"
(root㉿kali)-[/home/harshita/Documents/assignment2/Part_b]
iperf3 -c 10.0.0.4 -Z cubic
Connecting to host 10.0.0.4, port 5201
[7] local 10.0.0.1 port 47204 connected to 10.0.0.4 port 5201
[ID] Interval Transfer Bitrate Retr Cwnd
[7] 0.00-1.00 sec 8.30 GBytes 71.3 Gbits/sec 279 1.30 MBytes
[7] 1.00-2.00 sec 8.47 GBytes 72.8 Gbits/sec 65 1.31 MBytes
[7] 2.00-3.00 sec 8.21 GBytes 70.5 Gbits/sec 2 990 KBytes
[7] 3.00-4.00 sec 8.06 GBytes 69.2 Gbits/sec 176 1008 KBytes
[7] 4.00-5.00 sec 8.35 GBytes 71.7 Gbits/sec 629 1.00 MBytes
[7] 5.00-6.00 sec 8.23 GBytes 70.7 Gbits/sec 288 1.04 MBytes
[7] 6.00-7.00 sec 7.34 GBytes 63.0 Gbits/sec 477 1.13 MBytes
[7] 7.00-8.00 sec 7.86 GBytes 67.6 Gbits/sec 1 1.15 MBytes
[7] 8.00-9.00 sec 7.85 GBytes 67.4 Gbits/sec 25 1.21 MBytes
[7] 9.00-10.00 sec 7.47 GBytes 64.2 Gbits/sec 322 1.25 MBytes
[ID] Interval Transfer Bitrate Retr
[7] 0.00-10.00 sec 80.1 GBytes 68.8 Gbits/sec 2264 sender
[7] 0.00-10.00 sec 80.1 GBytes 68.8 Gbits/sec receiver
iperf Done.
(root㉿kali)-[/home/harshita/Documents/assignment2/Part_b]
#

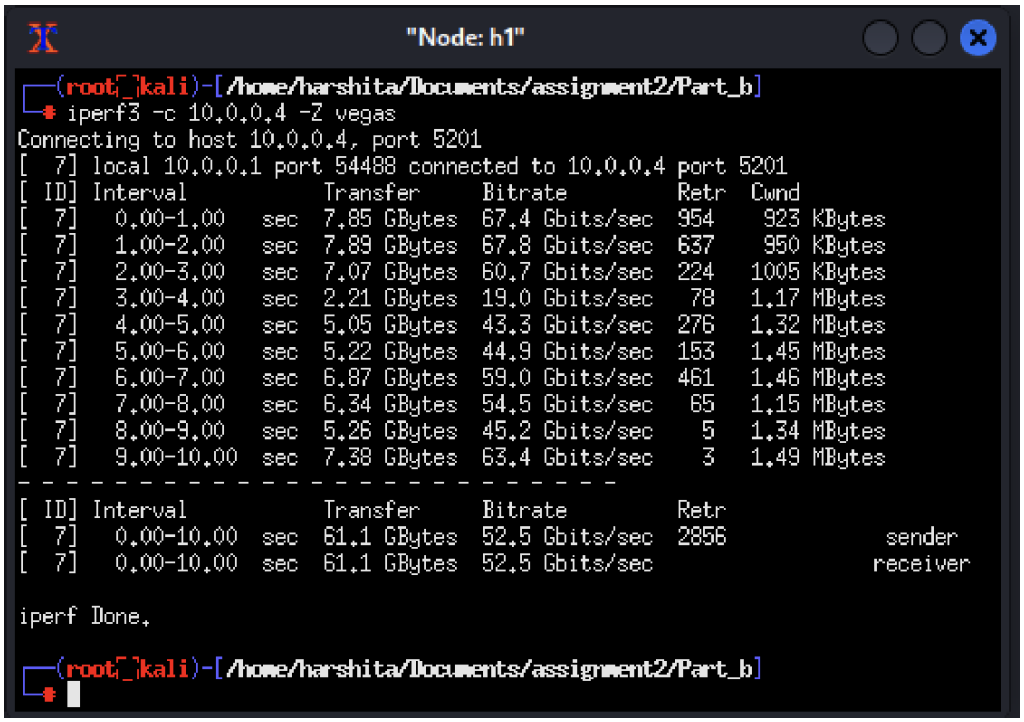
For BBR:


"Node: h1"
(root㉿kali)-[/home/harshita/Documents/assignment2/Part_b]
iperf3 -c 10.0.0.4 -Z bbr
Connecting to host 10.0.0.4, port 5201
[7] local 10.0.0.1 port 36688 connected to 10.0.0.4 port 5201
[ID] Interval Transfer Bitrate Retr Cwnd
[7] 0.00-1.00 sec 7.04 GBytes 60.5 Gbits/sec 1328 1.30 MBytes
[7] 1.00-2.00 sec 7.15 GBytes 61.5 Gbits/sec 763 933 KBytes
[7] 2.00-3.00 sec 7.95 GBytes 68.3 Gbits/sec 841 954 KBytes
[7] 3.00-4.00 sec 7.23 GBytes 62.1 Gbits/sec 2562 790 KBytes
[7] 4.00-5.00 sec 7.37 GBytes 63.3 Gbits/sec 2400 806 KBytes
[7] 5.00-6.00 sec 7.45 GBytes 64.0 Gbits/sec 2252 675 KBytes
[7] 6.00-7.00 sec 7.65 GBytes 65.7 Gbits/sec 1944 822 KBytes
[7] 7.00-8.00 sec 6.72 GBytes 57.7 Gbits/sec 2773 1.32 MBytes
[7] 8.00-9.00 sec 7.86 GBytes 67.5 Gbits/sec 144 1.34 MBytes
[7] 9.00-10.00 sec 7.80 GBytes 67.0 Gbits/sec 206 980 KBytes
[ID] Interval Transfer Bitrate Retr
[7] 0.00-10.00 sec 74.2 GBytes 63.8 Gbits/sec 15213 sender
[7] 0.00-10.00 sec 74.2 GBytes 63.8 Gbits/sec receiver
iperf Done.
(root㉿kali)-[/home/harshita/Documents/assignment2/Part_b]
#

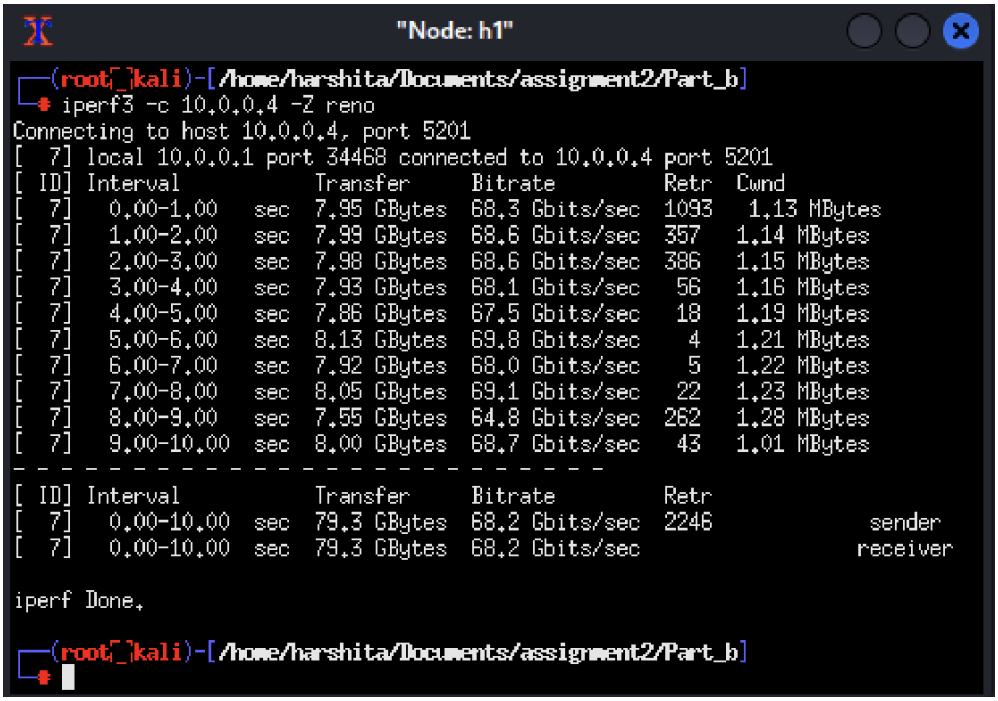
c. Run the client on H1, H2, H3 simultaneously and the server on H4. Report and reason the throughput over time for each host, each congestion control scheme. **(10 points)**

d. Configure the link loss parameter of the middle link (s1 - s2) to 1% and 3% and run the experiment in (b). Report and compare the throughput over time for each congestion control scheme. **(10 points)**

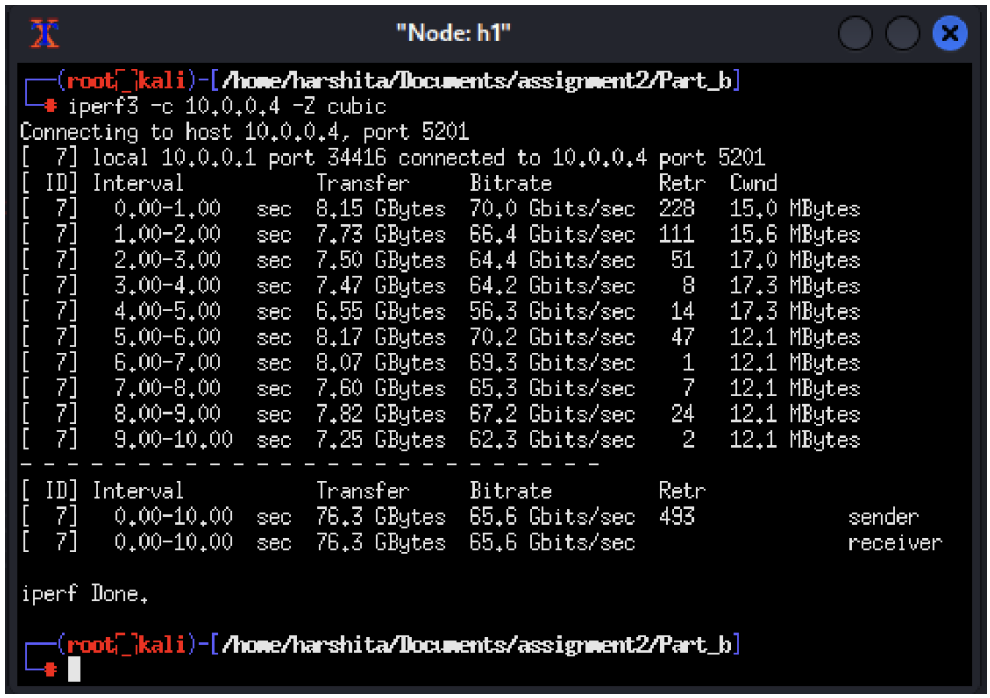
For vegas:



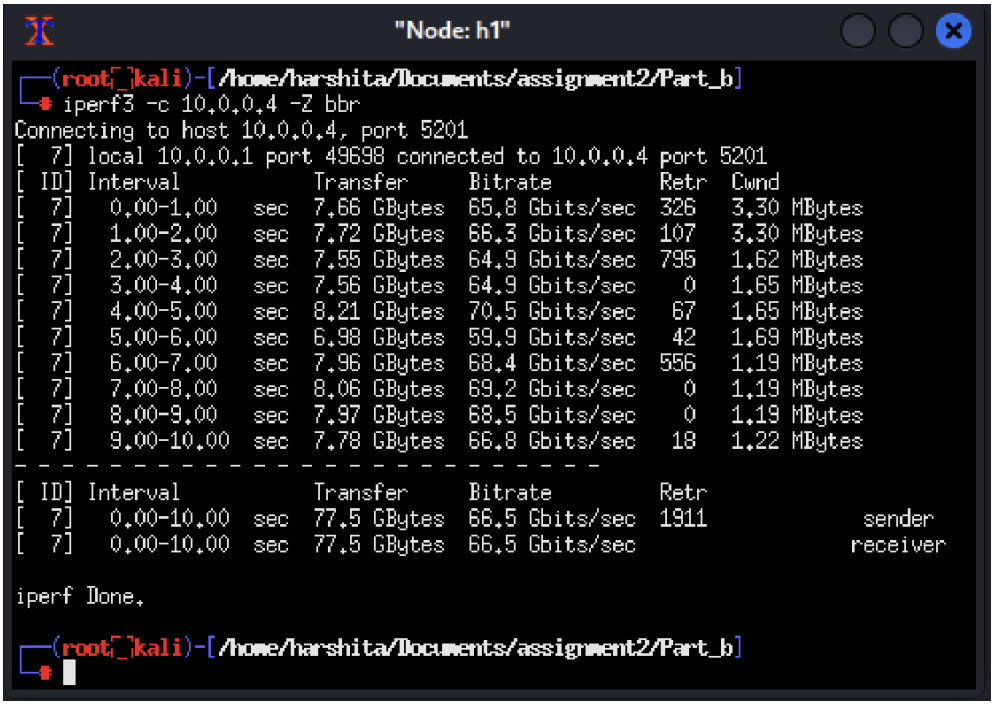
For reno:



For cubic:



For BBR:


"Node: h1"
┌──(root㉿kali)-[/home/harshita/Documents/assignment2/Part_b]
└─# iperf3 -c 10.0.0.4 -Z bbr
Connecting to host 10.0.0.4, port 5201
[7] local 10.0.0.1 port 49698 connected to 10.0.0.4 port 5201
[ID] Interval Transfer Bitrate Retr Cwnd
[7] 0.00-1.00 sec 7.66 GBytes 65.8 Gbits/sec 326 3.30 MBytes
[7] 1.00-2.00 sec 7.72 GBytes 66.3 Gbits/sec 107 3.30 MBytes
[7] 2.00-3.00 sec 7.55 GBytes 64.9 Gbits/sec 795 1.62 MBytes
[7] 3.00-4.00 sec 7.56 GBytes 64.9 Gbits/sec 0 1.65 MBytes
[7] 4.00-5.00 sec 8.21 GBytes 70.5 Gbits/sec 67 1.65 MBytes
[7] 5.00-6.00 sec 6.98 GBytes 59.9 Gbits/sec 42 1.69 MBytes
[7] 6.00-7.00 sec 7.96 GBytes 68.4 Gbits/sec 556 1.19 MBytes
[7] 7.00-8.00 sec 8.06 GBytes 69.2 Gbits/sec 0 1.19 MBytes
[7] 8.00-9.00 sec 7.97 GBytes 68.5 Gbits/sec 0 1.19 MBytes
[7] 9.00-10.00 sec 7.78 GBytes 66.8 Gbits/sec 18 1.22 MBytes
- -
[ID] Interval Transfer Bitrate Retr
[7] 0.00-10.00 sec 77.5 GBytes 66.5 Gbits/sec 1911 sender
[7] 0.00-10.00 sec 77.5 GBytes 66.5 Gbits/sec receiver
iperf Done.
┌──(root㉿kali)-[/home/harshita/Documents/assignment2/Part_b]
└─#