CSE534 HOMEWORK 3:

PART A

A1) We will be creating the below topology on mininet using the mininet Python API. It consists of 2 hosts, H1 and H2, connected through a network of 4 routers R1-R4. The python file that creates this topology can be found within the ***partA/ MyTopo.py*** path. H1 is connected to R1 in subnet 10.1.1.0/24, and H2 is connected to R4 in subnet 10.1.6.0/24. The result of the pingAll command can be found too, which indicates that all the nodes in the network are connected to each other as there are no message drops. Note, all experiments were conducted on an EC2 instance running Ubuntu version 20.0.

```
#                                          TOPOLOGY
#
#                                    (r2-eth0)    (r2-eth1)
#                     (r1-eth1)     10.1.2.1/24 | 10.1.4.1/24     (r4-eth1)
#                   10.1.2.2/24 +---------------R2---------------+ 10.1.4.2/24
#                              /                                  \
#  10.1.1.2/24 | 10.1.1.1/24 /                                     \ 10.1.6.1/24 | 10.1.6.2/24
#       H1------------------R1                                      R4------------------H2
#    (h1-eth0)      (r1-eth0) \                                     / (r4-eth0)      (h2-eth0)
#                              \                                   /
#                   10.1.3.2/24 +---------------R3---------------+ 10.1.5.2/24
#                    (r1-eth2)      10.1.3.1/24 | 10.1.5.1/24       (r4-eth2)
#                                    (r3-eth0)    (r3-eth1)
```

a) The network topology

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

b) Mininet pingAll command output for static routing

A2)

a) Please find the below screenshot containing the routing table information on all routers. Static routing has been enabled, and we can see an entry on each router for every subnet. We use the following command to setup static routing for every interface which is not reachable directly:

- > ip route add <destination_subnet> via <neighbor_subnet> dev <local_interface>

```
Routing Table on Router R1:
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.1.1.0        0.0.0.0         255.255.255.0   U     0      0        0 r1-eth0
10.1.2.0        0.0.0.0         255.255.255.0   U     0      0        0 r1-eth1
10.1.3.0        0.0.0.0         255.255.255.0   U     0      0        0 r1-eth2
10.1.4.0        10.1.2.1        255.255.255.0   UG    0      0        0 r1-eth1
10.1.5.0        10.1.3.1        255.255.255.0   UG    0      0        0 r1-eth2
10.1.6.0        10.1.2.1        255.255.255.0   UG    0      0        0 r1-eth1

Routing Table on Router R2:
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.1.1.0        10.1.2.2        255.255.255.0   UG    0      0        0 r2-eth0
10.1.2.0        0.0.0.0         255.255.255.0   U     0      0        0 r2-eth0
10.1.3.0        10.1.2.2        255.255.255.0   UG    0      0        0 r2-eth0
10.1.4.0        0.0.0.0         255.255.255.0   U     0      0        0 r2-eth1
10.1.5.0        10.1.4.2        255.255.255.0   UG    0      0        0 r2-eth1
10.1.6.0        10.1.4.2        255.255.255.0   UG    0      0        0 r2-eth1

Routing Table on Router R3:
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.1.1.0        10.1.3.2        255.255.255.0   UG    0      0        0 r3-eth0
10.1.2.0        10.1.3.2        255.255.255.0   UG    0      0        0 r3-eth0
10.1.3.0        0.0.0.0         255.255.255.0   U     0      0        0 r3-eth0
10.1.4.0        10.1.5.2        255.255.255.0   UG    0      0        0 r3-eth1
10.1.5.0        0.0.0.0         255.255.255.0   U     0      0        0 r3-eth1
10.1.6.0        10.1.5.2        255.255.255.0   UG    0      0        0 r3-eth1

Routing Table on Router R4:
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.1.1.0        10.1.5.1        255.255.255.0   UG    0      0        0 r4-eth2
10.1.2.0        10.1.4.1        255.255.255.0   UG    0      0        0 r4-eth1
10.1.3.0        10.1.5.1        255.255.255.0   UG    0      0        0 r4-eth2
10.1.4.0        0.0.0.0         255.255.255.0   U     0      0        0 r4-eth1
10.1.5.0        0.0.0.0         255.255.255.0   U     0      0        0 r4-eth2
10.1.6.0        0.0.0.0         255.255.255.0   U     0      0        0 r4-eth0
```

b) The trace output between H1 and H2

```
mininet> H1 traceroute H2
traceroute to 10.1.6.2 (10.1.6.2), 30 hops max, 60 byte packets
 1  10.1.1.1 (10.1.1.1)  0.032 ms  0.007 ms  0.006 ms
 2  10.1.2.1 (10.1.2.1)  0.016 ms  0.010 ms  0.009 ms
 3  10.1.4.2 (10.1.4.2)  0.019 ms  0.012 ms  0.019 ms
 4  10.1.6.2 (10.1.6.2)  0.021 ms  0.018 ms  0.015 ms
mininet> H2 traceroute H1
traceroute to 10.1.1.2 (10.1.1.2), 30 hops max, 60 byte packets
 1  10.1.6.1 (10.1.6.1)  0.033 ms  0.009 ms  0.008 ms
 2  10.1.4.1 (10.1.4.1)  0.017 ms  0.011 ms  0.010 ms
 3  10.1.2.2 (10.1.2.2)  0.020 ms  0.012 ms  0.013 ms
 4  10.1.1.2 (10.1.1.2)  0.023 ms  0.015 ms  0.015 ms
mininet>
```

PART B

B1) Please find the below screenshot containing the routing table information on all routers and hosts. We need to spawn a BIRD daemon on every router to enable dynamic routing using the RIP protocol. The main configuration file, ***bird.conf*** can be found for every router in its corresponding folder (R1/R2/R3/R4) under the main partB subdirectory in the project root directory. The python file, ***myRIP.py***, can be found there as well. Also below, the pingAll output.

```
Routing Table on Router R1:
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.1.1.0        0.0.0.0         255.255.255.0   U     0      0        0 r1-eth0
10.1.1.0        0.0.0.0         255.255.255.0   U     32     0        0 r1-eth0
10.1.2.0        0.0.0.0         255.255.255.0   U     0      0        0 r1-eth1
10.1.2.0        0.0.0.0         255.255.255.0   U     32     0        0 r1-eth1
10.1.3.0        0.0.0.0         255.255.255.0   U     0      0        0 r1-eth2
10.1.3.0        0.0.0.0         255.255.255.0   U     32     0        0 r1-eth2
10.1.4.0        10.1.2.1        255.255.255.0   UG    32     0        0 r1-eth1
10.1.5.0        10.1.3.1        255.255.255.0   UG    32     0        0 r1-eth2

Routing Table on Router R2:
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.1.1.0        10.1.2.2        255.255.255.0   UG    32     0        0 r2-eth0
10.1.2.0        0.0.0.0         255.255.255.0   U     0      0        0 r2-eth0
10.1.2.0        0.0.0.0         255.255.255.0   U     32     0        0 r2-eth0
10.1.3.0        10.1.2.2        255.255.255.0   UG    32     0        0 r2-eth0
10.1.4.0        0.0.0.0         255.255.255.0   U     0      0        0 r2-eth1
10.1.4.0        0.0.0.0         255.255.255.0   U     32     0        0 r2-eth1
10.1.5.0        10.1.4.2        255.255.255.0   UG    32     0        0 r2-eth1
10.1.6.0        10.1.4.2        255.255.255.0   UG    32     0        0 r2-eth1

Routing Table on Router R3:
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.1.1.0        10.1.3.2        255.255.255.0   UG    32     0        0 r3-eth0
10.1.2.0        10.1.3.2        255.255.255.0   UG    32     0        0 r3-eth0
10.1.3.0        0.0.0.0         255.255.255.0   U     0      0        0 r3-eth0
10.1.3.0        0.0.0.0         255.255.255.0   U     32     0        0 r3-eth0
10.1.4.0        10.1.5.2        255.255.255.0   UG    32     0        0 r3-eth1
10.1.5.0        0.0.0.0         255.255.255.0   U     0      0        0 r3-eth1
10.1.5.0        0.0.0.0         255.255.255.0   U     32     0        0 r3-eth1
10.1.6.0        10.1.5.2        255.255.255.0   UG    32     0        0 r3-eth1

Routing Table on Router R4:
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.1.1.0        10.1.4.1        255.255.255.0   UG    32     0        0 r4-eth1
10.1.2.0        10.1.4.1        255.255.255.0   UG    32     0        0 r4-eth1
10.1.3.0        10.1.5.1        255.255.255.0   UG    32     0        0 r4-eth2
10.1.4.0        0.0.0.0         255.255.255.0   U     0      0        0 r4-eth1
10.1.4.0        0.0.0.0         255.255.255.0   U     32     0        0 r4-eth1
10.1.5.0        0.0.0.0         255.255.255.0   U     0      0        0 r4-eth2
10.1.5.0        0.0.0.0         255.255.255.0   U     32     0        0 r4-eth2
10.1.6.0        0.0.0.0         255.255.255.0   U     0      0        0 r4-eth0
10.1.6.0        0.0.0.0         255.255.255.0   U     32     0        0 r4-eth0
```

```
Pinging all nodes and routers
*** Ping: testing ping reachability
H1 -> H2 R1 R2 R3 R4
H2 -> H1 R1 R2 R3 R4
R1 -> H1 H2 R2 R3 R4
R2 -> H1 H2 R1 R3 R4
R3 -> H1 H2 R1 R2 R4
R4 -> H1 H2 R1 R2 R3
*** Results: 0% dropped (30/30 received)
0.0*** Starting CLI:
mininet> H1 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         10.1.1.1        0.0.0.0         UG    0      0        0 h1-eth0
10.1.1.0        0.0.0.0         255.255.255.0   U     0      0        0 h1-eth0
mininet> H2 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         10.1.6.1        0.0.0.0         UG    0      0        0 h2-eth0
10.1.6.0        0.0.0.0         255.255.255.0   U     0      0        0 h2-eth0
mininet>
```

We can also see the traceroute output between hosts H1 and H2 below, with all links connected.

```
mininet> H1 traceroute H2
traceroute to 10.1.6.2 (10.1.6.2), 30 hops max, 60 byte packets
 1  10.1.1.1 (10.1.1.1)  0.035 ms  0.009 ms  0.006 ms
 2  10.1.2.1 (10.1.2.1)  0.020 ms  0.021 ms  0.010 ms
 3  10.1.4.2 (10.1.4.2)  0.020 ms  0.013 ms  0.012 ms
 4  10.1.6.2 (10.1.6.2)  0.022 ms  0.015 ms  0.016 ms
mininet> H2 traceroute H1
traceroute to 10.1.1.2 (10.1.1.2), 30 hops max, 60 byte packets
 1  10.1.6.1 (10.1.6.1)  0.034 ms  0.008 ms  0.006 ms
 2  10.1.4.1 (10.1.4.1)  0.016 ms  0.010 ms  0.008 ms
 3  10.1.2.2 (10.1.2.2)  0.019 ms  0.011 ms  0.011 ms
 4  10.1.1.2 (10.1.1.2)  0.020 ms  0.014 ms  0.013 ms
mininet>
```

B2) We then bring down the **R1-R2** link with the following command:

- \> link R1 R2 down

  We can now see the updated traceroute output as shown below. The new route goes through the 10.1.3.x/24 and 10.1.5.x/24 subnets (previously 10.1.2.x/24 and 10.1.4.x/24).

```
mininet> link R1 R2 down
mininet> H1 traceroute H2
traceroute to 10.1.6.2 (10.1.6.2), 30 hops max, 60 byte packets
 1  10.1.1.1 (10.1.1.1)  0.038 ms  0.009 ms  0.006 ms
 2  10.1.3.1 (10.1.3.1)  0.020 ms  0.010 ms  0.009 ms
 3  10.1.5.2 (10.1.5.2)  0.022 ms  0.013 ms  0.012 ms
 4  10.1.6.2 (10.1.6.2)  0.022 ms  0.016 ms  0.014 ms
mininet> H2 traceroute H1
traceroute to 10.1.1.2 (10.1.1.2), 30 hops max, 60 byte packets
 1  10.1.6.1 (10.1.6.1)  0.030 ms  0.007 ms  0.007 ms
 2  10.1.5.1 (10.1.5.1)  0.017 ms  0.024 ms  0.009 ms
 3  10.1.3.2 (10.1.3.2)  0.019 ms  0.028 ms  0.012 ms
 4  10.1.1.2 (10.1.1.2)  0.021 ms  0.015 ms  0.014 ms
mininet>
```

PART C

In this part, we use the below 2 commands to limit router bandwidth + buffer size and introduce delays in every interface of all routers with the tc tbf and tc netem options respectively.

- \> tc qdisc add dev <interface> root handle 1: tbf rate <bandwidth> burst <burst_rate> limit <buffer_size>
- \> tc qdisc add dev <interface> parent 1:1 handle 10: netem delay <delay_in_ms>

BIRD is used to run the RIP protocol for dynamic routing. Once the routers have been reconfigured with the above commands, we then spawn xterm instances of hosts H1 and H2 to run the network performance tool, Iperf, which will simulate a TCP performance test between the two hosts, with H2 acting as the Iperf server and H1 the client. We do 3 measurements, with the router buffer size varying as 10Kb, 5Mb and 25Mb across them. Delay = 30ms and Bandwidth = 100Mbps are kept constant in all 3 runs.

To run H2 as the Iperf server, we use the below command on its xterm terminal

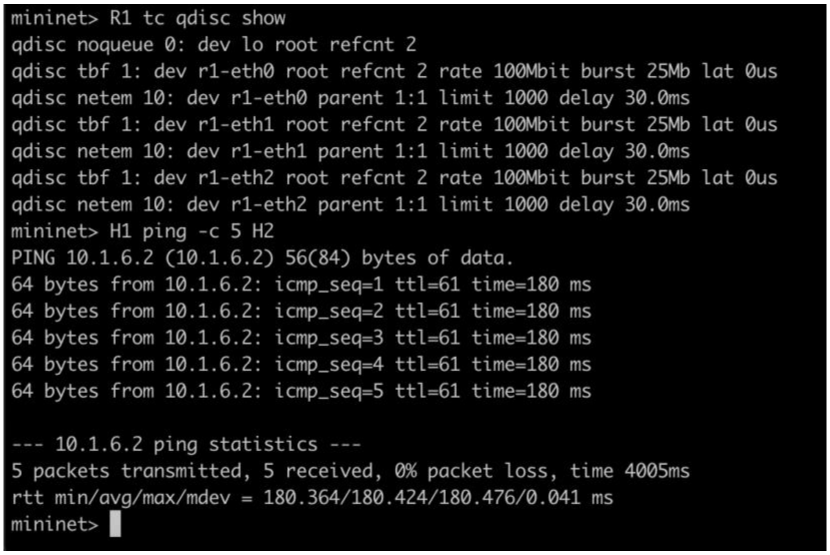
- \> iperf3 -s

To run H1 as the Iperf client, we use the below command on its xterm terminal

- \> iperf3 -c <H2_IP>

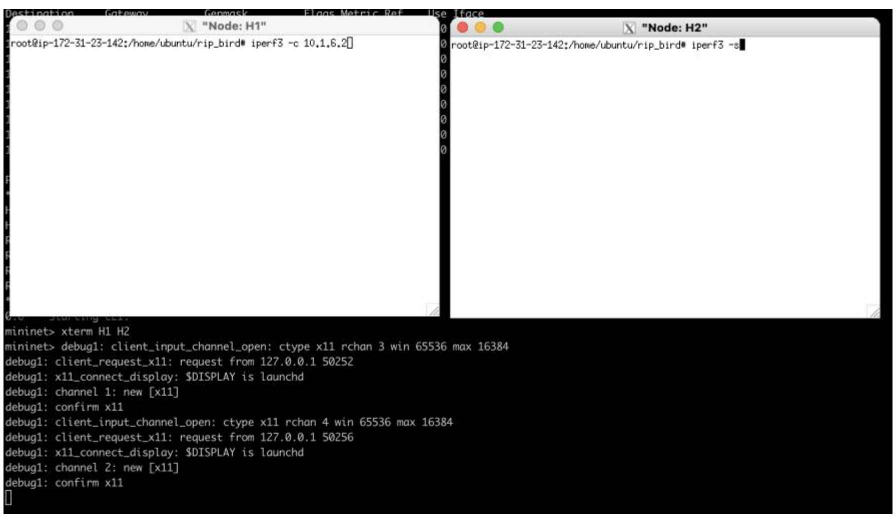
We can also add the -J option and pipe the result of both server and client to create output json files.

We first run the ***MyIperf.py*** file in the partC directory that automates the configuration of the routers with the ***tc*** commands. We can then verify that the routers have been configured correctly using the below 2 commands:



From the above image, we can see that all interfaces of router R1 have been correctly configured by the tc command to update its bandwidth, buffer_size and delay parameters. Similarly, we can verify the same for the other routers as well. Also from the ping command, we see that the RTT delay is now 180ms. Every packet flows through 6 interfaces from H1 to H2 (refer topology diagram in Part A, the interfaces are H1 -> r1-eth0 -> r1-eth1 -> r2-eth0 -> r2-eth1 -> r4-eth1 -> r4-eth0 -> H2), and hence total delay is 6 * 30 = 180ms per packet.

Now that we have setup the routers, we will run iperf between H1 and H2 through their xterms, as shown below. We will then describe the results for buffer_size = 10Kb, 5Mb and 25Mb.



We also capture the client/server result json files for each buffer_size configuration and store them in the ***partC/iperf_results*** directory.

We can then estimate the actual bandwidth, number of retransmissions and the round-trip times of the entire flow from the '*bits_per_second*', '*retransmits*' and '*mean_rtt*' attributes (in *$.end.streams[0].sender.** JSONPath) in the ***client_*.json*** files, as shown below. The **Bandwidth Delay Product** (BDP) can then be calculated as **BDP = bits_per_second * mean_rtt**


```
"end":  {
        "streams":      [{
                "sender":       {
                        "socket":       7,
                        "start":        0,
                        "end":  10.000179,
                        "seconds":      10.000179,
                        "bytes":        112317088,
                        "bits_per_second":      89852062.04808934,
                        "retransmits":  5,
                        "max_snd_cwnd": 2829392,
                        "max_rtt":      271434,
                        "min_rtt":      180893,
                        "mean_rtt":     215799,
                        "sender":       true
                },
                "receiver":     {
                        "socket":       7,
                        "start":        0,
                        "end":  10.234367,
                        "seconds":      10.000179,
                        "bytes":        103481320,
                        "bits_per_second":      80889278.252382383,
                        "sender":       true
                }
        }],
```


Test 1 – Using buffer_size = 10Kb (Refer to client_10k.json file):

"Node: H2"

```
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 10.1.1.2, port 54730
[  7] local 10.1.6.2 port 5201 connected to 10.1.1.2 port 54732
[ ID] Interval           Transfer     Bitrate
[  7]   0.00-1.00   sec   181 KBytes  1.48 Mbits/sec
[  7]   1.00-2.00   sec  5.10 MBytes  42.7 Mbits/sec
[  7]   2.00-3.00   sec  10.9 MBytes  91.8 Mbits/sec
[  7]   3.00-4.00   sec  11.4 MBytes  95.4 Mbits/sec
[  7]   4.00-5.00   sec  11.4 MBytes  95.6 Mbits/sec
[  7]   5.00-6.00   sec  11.4 MBytes  95.6 Mbits/sec
[  7]   6.00-7.00   sec  11.4 MBytes  95.6 Mbits/sec
[  7]   7.00-8.00   sec  11.4 MBytes  95.6 Mbits/sec
[  7]   8.00-9.00   sec  11.4 MBytes  95.6 Mbits/sec
[  7]   9.00-10.00  sec  11.4 MBytes  95.6 Mbits/sec
[  7]  10.00-10.24  sec  2.68 MBytes  95.6 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate
[  7]   0.00-10.24  sec  98.7 MBytes  80.9 Mbits/sec                  receiver
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

"Node: H1"

```
root@ip-172-31-23-142:/home/ubuntu/rip_bird# iperf3 -c 10.1.6.2
Connecting to host 10.1.6.2, port 5201
[  7] local 10.1.1.2 port 54732 connected to 10.1.6.2 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  7]   0.00-1.00   sec   872 KBytes  7.15 Mbits/sec    0    195 KBytes
[  7]   1.00-2.00   sec  15.9 MBytes   133 Mbits/sec    0   2.70 MBytes
[  7]   2.00-3.00   sec  10.0 MBytes  83.9 Mbits/sec    7   2.05 MBytes
[  7]   3.00-4.00   sec  11.2 MBytes  94.4 Mbits/sec    0   2.22 MBytes
[  7]   4.00-5.00   sec  12.5 MBytes   105 Mbits/sec    0   2.35 MBytes
[  7]   5.00-6.00   sec  11.2 MBytes  94.4 Mbits/sec    0   2.46 MBytes
[  7]   6.00-7.00   sec  11.2 MBytes  94.4 Mbits/sec    0   2.55 MBytes
[  7]   7.00-8.00   sec  11.2 MBytes  94.4 Mbits/sec    0   2.61 MBytes
[  7]   8.00-9.00   sec  11.2 MBytes  94.4 Mbits/sec    0   2.66 MBytes
[  7]   9.00-10.00  sec  11.2 MBytes  94.4 Mbits/sec    0   2.69 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  7]   0.00-10.00  sec   107 MBytes  89.5 Mbits/sec    7             sender
[  7]   0.00-10.24  sec  98.7 MBytes  80.9 Mbits/sec                  receiver

iperf Done.
root@ip-172-31-23-142:/home/ubuntu/rip_bird#
```

a) BDP = 89852062.04808934 * 215799 * 10^{-6} = 2.42 Mbytes (comparable to avg cwnd in H1 above)

Test 2 – Using buffer_size = 5Mb (Refer to client_5m.json file):

"Node: H2"

```
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 10.1.1.2, port 54776
[  7] local 10.1.6.2 port 5201 connected to 10.1.1.2 port 54778
[ ID] Interval           Transfer     Bitrate
[  7]   0.00-1.00   sec   167 KBytes  1.37 Mbits/sec
[  7]   1.00-2.00   sec  9.33 MBytes  78.2 Mbits/sec
[  7]   2.00-3.00   sec  11.6 MBytes  97.6 Mbits/sec
[  7]   3.00-4.00   sec  11.4 MBytes  95.9 Mbits/sec
[  7]   4.00-5.00   sec  11.4 MBytes  95.6 Mbits/sec
[  7]   5.00-6.00   sec  11.4 MBytes  95.7 Mbits/sec
[  7]   6.00-7.00   sec  11.4 MBytes  95.6 Mbits/sec
[  7]   7.00-8.00   sec  11.4 MBytes  95.7 Mbits/sec
[  7]   8.00-9.00   sec  11.4 MBytes  95.6 Mbits/sec
[  7]   9.00-10.00  sec  11.4 MBytes  95.5 Mbits/sec
[  7]  10.00-10.70  sec  7.97 MBytes  95.5 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate
[  7]   0.00-10.70  sec   109 MBytes  85.4 Mbits/sec                  receiver
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

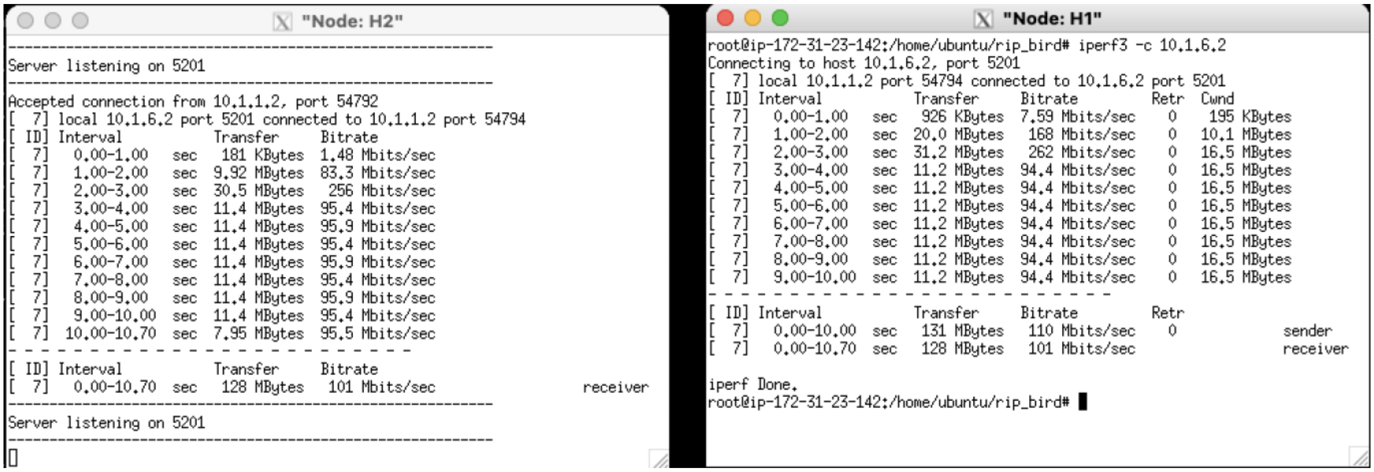
"Node: H1"

```
root@ip-172-31-23-142:/home/ubuntu/rip_bird# iperf3 -c 10.1.6.2
Connecting to host 10.1.6.2, port 5201
[  7] local 10.1.1.2 port 54778 connected to 10.1.6.2 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  7]   0.00-1.00   sec   820 KBytes  6.72 Mbits/sec    0    177 KBytes
[  7]   1.00-2.00   sec  19.7 MBytes   166 Mbits/sec    0   9.50 MBytes
[  7]   2.00-3.00   sec  11.2 MBytes  94.4 Mbits/sec    0   16.0 MBytes
[  7]   3.00-4.00   sec  11.2 MBytes  94.4 Mbits/sec    0   16.0 MBytes
[  7]   4.00-5.00   sec  11.2 MBytes  94.4 Mbits/sec    0   16.0 MBytes
[  7]   5.00-6.00   sec  11.2 MBytes  94.4 Mbits/sec    0   16.0 MBytes
[  7]   6.00-7.00   sec  12.5 MBytes   105 Mbits/sec    0   16.0 MBytes
[  7]   7.00-8.00   sec  11.2 MBytes  94.4 Mbits/sec    0   16.0 MBytes
[  7]   8.00-9.00   sec  11.2 MBytes  94.4 Mbits/sec    0   16.0 MBytes
[  7]   9.00-10.00  sec  11.2 MBytes  94.4 Mbits/sec    0   16.0 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  7]   0.00-10.00  sec   112 MBytes  93.8 Mbits/sec    0             sender
[  7]   0.00-10.70  sec   109 MBytes  85.4 Mbits/sec                  receiver

iperf Done.
root@ip-172-31-23-142:/home/ubuntu/rip_bird#
```

a) BDP = 93934247.55341047 * 596035 * 10^{-6} = 7 Mbytes

Test 3 – Using buffer_size = 25Mb (Refer to client_25m.json file):


```
"Node: H2"
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 10.1.1.2, port 54792
[  7] local 10.1.6.2 port 5201 connected to 10.1.1.2 port 54794
[ ID] Interval           Transfer     Bitrate
[  7]   0.00-1.00   sec   181 KBytes  1.48 Mbits/sec
[  7]   1.00-2.00   sec  9.92 MBytes  83.3 Mbits/sec
[  7]   2.00-3.00   sec  30.5 MBytes   256 Mbits/sec
[  7]   3.00-4.00   sec  11.4 MBytes  95.4 Mbits/sec
[  7]   4.00-5.00   sec  11.4 MBytes  95.9 Mbits/sec
[  7]   5.00-6.00   sec  11.4 MBytes  95.4 Mbits/sec
[  7]   6.00-7.00   sec  11.4 MBytes  95.9 Mbits/sec
[  7]   7.00-8.00   sec  11.4 MBytes  95.4 Mbits/sec
[  7]   8.00-9.00   sec  11.4 MBytes  95.9 Mbits/sec
[  7]   9.00-10.00  sec  11.4 MBytes  95.4 Mbits/sec
[  7]  10.00-10.70  sec  7.95 MBytes  95.5 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate
[  7]   0.00-10.70  sec   128 MBytes   101 Mbits/sec                  receiver
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------

"Node: H1"
root@ip-172-31-23-142:/home/ubuntu/rip_bird# iperf3 -c 10.1.6.2
Connecting to host 10.1.6.2, port 5201
[  7] local 10.1.1.2 port 54794 connected to 10.1.6.2 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  7]   0.00-1.00   sec   926 KBytes  7.59 Mbits/sec    0    195 KBytes
[  7]   1.00-2.00   sec  20.0 MBytes   168 Mbits/sec    0   10.1 MBytes
[  7]   2.00-3.00   sec  31.2 MBytes   262 Mbits/sec    0   16.5 MBytes
[  7]   3.00-4.00   sec  11.2 MBytes  94.4 Mbits/sec    0   16.5 MBytes
[  7]   4.00-5.00   sec  11.2 MBytes  94.4 Mbits/sec    0   16.5 MBytes
[  7]   5.00-6.00   sec  11.2 MBytes  94.4 Mbits/sec    0   16.5 MBytes
[  7]   6.00-7.00   sec  11.2 MBytes  94.4 Mbits/sec    0   16.5 MBytes
[  7]   7.00-8.00   sec  11.2 MBytes  94.4 Mbits/sec    0   16.5 MBytes
[  7]   8.00-9.00   sec  11.2 MBytes  94.4 Mbits/sec    0   16.5 MBytes
[  7]   9.00-10.00  sec  11.2 MBytes  94.4 Mbits/sec    0   16.5 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  7]   0.00-10.00  sec   131 MBytes   110 Mbits/sec    0             sender
[  7]   0.00-10.70  sec   128 MBytes   101 Mbits/sec                  receiver

iperf Done.
root@ip-172-31-23-142:/home/ubuntu/rip_bird#
```

a) BDP = 109744553.52140266 * 565538 * 10^{-6} = 7.76 Mbytes

OBERVATIONS OF THE ABOVE MEASUREMENTS:

- We see some retransmissions in the first measurement with 10Kb buffer size. This may be due to packets being dropped from one of the router interfaces due to the small-size buffer queue filling up. For the other 2 measurements, we see that there are no retransmissions after the router buffer size was increased.
- The BDP is increasing with increase in buffer_size. This is because of increase in the actual average bandwidth (refer Bitrate in H1) with increase in buffer_size.
- We see a major multiplicative increase of the cwnd in the first 2 seconds, followed by a sharp drop in the bandwidth (bitrate) which then leads to the cwnd stabilizing across successive intervals.
- Since retransmission does not occur for buffer_sizes 5Mb and 25Mb, the retransmission time is 0. For 10Kb, the retransmission time may be around 0.24s (10.24s – 10s).