

CS 433 Computer Networks

Write Up

Assignment-2

Karan Bhardwaj:20110093
Manpreet Singh:20110109

Q1

a. Implementation

Here we have implemented the given topology using three subnets each having two hosts and a router. The router has been implemented as a combination of a switch and a host. The IP addresses are as:-

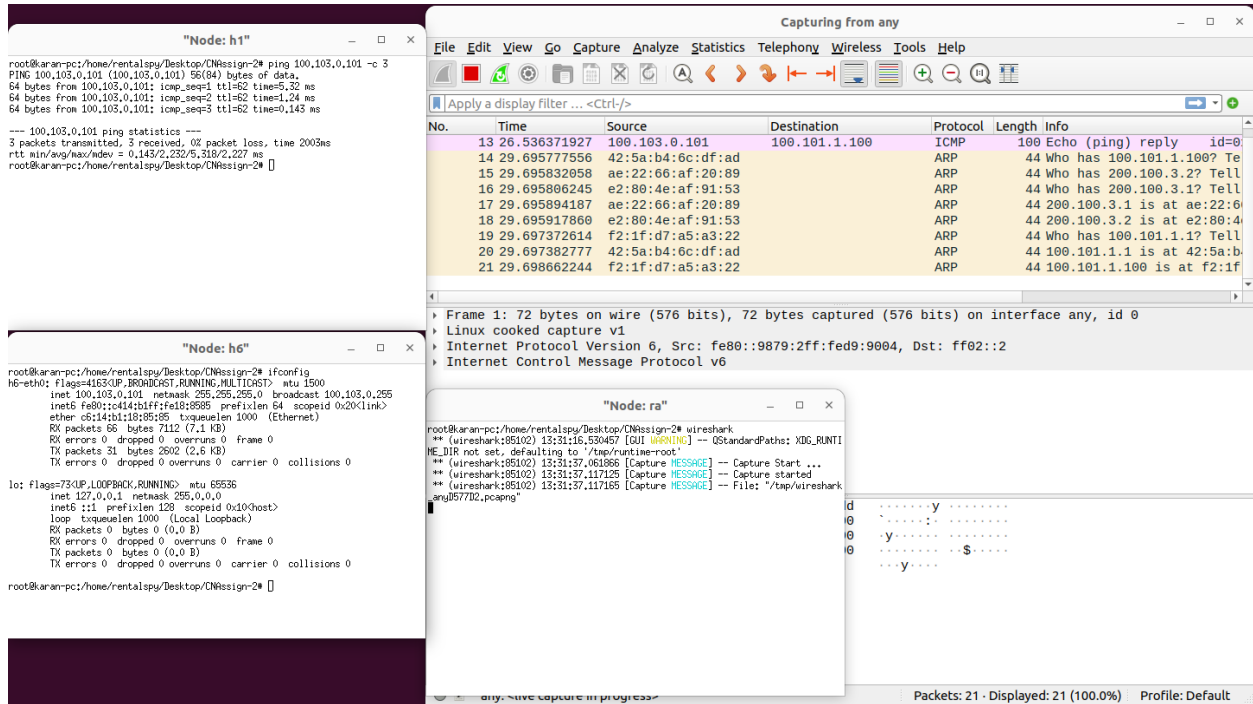
```
ra :100.101.1.1
rb:100.101.0.1
rc:100.103.0.1
h1:100.101.1.100
h2:100.101.1.101
h3:100.102.0.100
h4:100.102.0.101
h5:100.103.0.100
h6:100.103.0.101
```

Command Used: sudo mn -c; python3 Q1.py

Mininet CLI will open, with the custom topology designed.

Now, using the **pingall** command we have tested the connections between all hosts and routers and as per the results the connections are working well. We can see the topology is working well as 100 % of the packets sent are received.

```
*** 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> 
```



c. Latency

Below are the output snippets for the default route from host h1 to h6 that is **h1**->**r_a**->**r_c**->**h6**. Here we are first sending 5 packets from h1 to h6. In the image shown below the RTT corresponding to each of the sent packets is shown here.

Using ping command:

Avg RTT= 0.346ms.

```
"Node: h1"
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# ping 100.103.0.101 -c 3
PING 100.103.0.101 (100.103.0.101) 56(84) bytes of data.
64 bytes from 100.103.0.101: icmp_seq=1 ttl=62 time=5.44 ms
64 bytes from 100.103.0.101: icmp_seq=2 ttl=62 time=1.17 ms
64 bytes from 100.103.0.101: icmp_seq=3 ttl=62 time=0.143 ms

--- 100.103.0.101 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 0.143/2.249/5.439/2.293 ms
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# ping 100.103.0.101 -c 5
PING 100.103.0.101 (100.103.0.101) 56(84) bytes of data.
64 bytes from 100.103.0.101: icmp_seq=1 ttl=62 time=1.20 ms
64 bytes from 100.103.0.101: icmp_seq=2 ttl=62 time=0.139 ms
64 bytes from 100.103.0.101: icmp_seq=3 ttl=62 time=0.128 ms
64 bytes from 100.103.0.101: icmp_seq=4 ttl=62 time=0.138 ms
64 bytes from 100.103.0.101: icmp_seq=5 ttl=62 time=0.131 ms

--- 100.103.0.101 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4055ms
rtt min/avg/max/mdev = 0.128/0.346/1.195/0.424 ms
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# []

"Node: h6"
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# ifconfig
h6-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 100.103.0.101 netmask 255.255.255.0 broadcast 100.103.0.255
    inet6 fe80::c414:b1ff:fe18:8585 prefixlen 64 scopeid 0x20<link>
    ether c6:14:b1:18:85:85 txqueuelen 1000 (Ethernet)
    RX packets 66 bytes 7112 (7.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 31 bytes 2602 (2.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# []
```

Using iperf command:

```
"Node: h1"
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -c 100.103.0.101 -u -b
iperf: option requires an argument -- b

Client connecting to 100.103.0.101, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size: 208 KByte (default)

[ 1] local 100.101.1.100 port 47846 connected with 100.103.0.101 port 5001
[ ID] Interval      Transfer      Bandwidth
[ 1] 0.0000-10.0156 sec  1.25 MBytes  1.05 Mbits/sec
[ 1] Sent 896 datagrams
[ 1] Server Report:
[ ID] Interval      Transfer      Bandwidth      Jitter  Lost/Total Datagrams
[ 1] 0.0000-10.0128 sec  1.25 MBytes  1.05 Mbits/sec  0.008 ms  0/895 (0%)
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# []

"Node: h6"
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -s -u -i 1

Server listening on UDP port 5001
UDP buffer size: 208 KByte (default)

[ 1] local 100.103.0.101 port 5001 connected with 100.101.1.100 port 47846
[ ID] Interval      Transfer      Bandwidth      Jitter  Lost/Total Datagrams
[ 1] 0.0000-1.0000 sec  131 KBytes  1.07 Mbits/sec  0.004 ms  0/91 (0%)
[ 1] 1.0000-2.0000 sec  128 KBytes  1.05 Mbits/sec  0.002 ms  0/89 (0%)
[ 1] 2.0000-3.0000 sec  128 KBytes  1.05 Mbits/sec  0.008 ms  0/89 (0%)
[ 1] 3.0000-4.0000 sec  128 KBytes  1.05 Mbits/sec  0.002 ms  0/89 (0%)
[ 1] 4.0000-5.0000 sec  129 KBytes  1.06 Mbits/sec  0.002 ms  0/90 (0%)
[ 1] 5.0000-6.0000 sec  128 KBytes  1.05 Mbits/sec  0.001 ms  0/89 (0%)
[ 1] 6.0000-7.0000 sec  128 KBytes  1.05 Mbits/sec  0.003 ms  0/89 (0%)
[ 1] 7.0000-8.0000 sec  128 KBytes  1.05 Mbits/sec  0.004 ms  0/89 (0%)
[ 1] 8.0000-9.0000 sec  128 KBytes  1.05 Mbits/sec  0.005 ms  0/89 (0%)
[ 1] 9.0000-10.0000 sec  128 KBytes  1.05 Mbits/sec  0.007 ms  0/89 (0%)
[ 1] 0.0000-10.0128 sec  1.25 MBytes  1.05 Mbits/sec  0.008 ms  0/895 (0%)
[]
```

Below are the output snippets for the new route from host h1 to h6 that is **h1->r_a->r_b->r_c->h6**. Here also we are sending 5 packets from h1 to h6. The image below shows the RTT corresponding to each of the packets received.

Using ping command:

Avg RTT = 1.941ms

The image shows two terminal windows side-by-side. The left window, titled "Node: h1", shows the output of a ping command from h1 to 100.103.0.101. It displays five successful pings with times ranging from 0.144 ms to 0.149 ms. Below the pings, it shows the ping statistics: 5 packets transmitted, 5 received, 0% packet loss, and an average RTT of 1.941 ms. The right window, titled "Node: h6", shows the output of the ifconfig command for h6. It displays the configuration for the eth0 interface, including IP address, netmask, broadcast address, and various statistics showing zero errors and collisions.

Using iperf command:

The image shows two terminal windows side-by-side. The left window, titled "Node: h1", shows the output of the iperf client command. It displays the connection to the server at 100.103.0.101 on port 5001 and the transfer of 1470 byte datagrams. The right window, titled "Node: h6", shows the output of the iperf server command. It displays the server listening on UDP port 5001 and the transfer of 1470 byte datagrams. Both windows show detailed statistics including interval, transfer, bandwidth, jitter, and lost/total datagrams.

Observation: The latency is higher for the longer path **h1->r_a->r_b->r_c->h6**.

d. Routing table

Topology with route **h1->r_a->r_c->h6**.

```

*** Adding links:
(h1, s1) (h2, s1) (h3, s2) (h4, s2) (h5, s3) (h6, s3) (ra, rb) (ra, rc) (rb, rc) (
*** Configuring hosts
h1 h2 h3 h4 h5 h6 ra rb rc
*** 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
100.101.1.0      0.0.0.0        255.255.255.0   U        0      0      0 ra-eth1
100.102.0.0      200.100.1.2    255.255.255.0   UG        0      0      0 l
100.103.0.0      200.100.3.2    255.255.255.0   UG        0      0      0 p
200.100.1.0      0.0.0.0        255.255.255.0   U        0      0      0 l
200.100.3.0      0.0.0.0        255.255.255.0   U        0      0      0 p
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
100.101.1.0      200.100.1.1    255.255.255.0   UG        0      0      0 m
100.102.0.0      0.0.0.0        255.255.255.0   U        0      0      0 rb-eth1
100.103.0.0      200.100.2.2    255.255.255.0   UG        0      0      0 n
200.100.1.0      0.0.0.0        255.255.255.0   U        0      0      0 m
200.100.2.0      0.0.0.0        255.255.255.0   U        0      0      0 n
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
100.101.1.0      200.100.3.1    255.255.255.0   UG        0      0      0 q
100.102.0.0      200.100.2.1    255.255.255.0   UG        0      0      0 o
100.103.0.0      0.0.0.0        255.255.255.0   U        0      0      0 rc-eth1
200.100.2.0      0.0.0.0        255.255.255.0   U        0      0      0 o
200.100.3.0      0.0.0.0        255.255.255.0   U        0      0      0 q
*** Starting CLI:
mininet> pingall

```

Now, applying **pingall** with changes made

```

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 -> h1 h2 h3 h4 h5 h6 ra rb
*** Results: 4% dropped (69/72 received)
mininet> 

```

There was a 4% drop.

Topology with route **h1->r_a->r_b->r_c->h6**.

```
*** 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
100.101.1.0      0.0.0.0          255.255.255.0    U        0      0      0 ra-eth1
100.102.0.0      200.100.1.2      255.255.255.0    UG        0      0      0 l
100.103.0.0      200.100.1.2      255.255.255.0    UG        0      0      0 l
200.100.1.0      0.0.0.0          255.255.255.0    U        0      0      0 l
200.100.3.0      0.0.0.0          255.255.255.0    U        0      0      0 p
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
100.101.1.0      200.100.1.1      255.255.255.0    UG        0      0      0 m
100.102.0.0      0.0.0.0          255.255.255.0    U        0      0      0 rb-eth1
100.103.0.0      200.100.2.2      255.255.255.0    UG        0      0      0 n
200.100.1.0      0.0.0.0          255.255.255.0    U        0      0      0 m
200.100.2.0      0.0.0.0          255.255.255.0    U        0      0      0 n
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
100.101.1.0      200.100.3.1      255.255.255.0    UG        0      0      0 q
100.102.0.0      200.100.2.1      255.255.255.0    UG        0      0      0 o
100.103.0.0      0.0.0.0          255.255.255.0    U        0      0      0 rc-eth1
200.100.2.0      0.0.0.0          255.255.255.0    U        0      0      0 o
200.100.3.0      0.0.0.0          255.255.255.0    U        0      0      0 q
*** Starting CLI:
mininet> 
```

Q2

The IP addresses as:

h1:10.0.0.1
h2:10.0.0.2
h3:10.0.0.3
h4:10.0.0.4

```
class Mytopo(Topo):
    def build(self, **_opts):
        s1 = self.addSwitch('s1')
        s2 = self.addSwitch('s2')
        h1 = self.addHost('h1')
        h2 = self.addHost('h2')
        h3 = self.addHost('h3')
        h4 = self.addHost('h4')
        self.addLink(h1, s1)
        self.addLink(h2, s1)
        self.addLink(h3, s2)
        self.addLink(h4, s2)
        self.addLink(s1, s2)
```

a. TCP Client-Server Program

Created two files server.py and client.py. The host h1, h2 and h3 will run client.py and will send a message to the server h4 by creating a TCP socket. The server h4 will be running server.py. It will receive the message. All the sent and received messages are stored in a txt.file. This is done to check whether the client-server connection is working properly or not

Command Used: sudo mn -c; python3 Q2.py

Client.py

```
import socket

client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(('10.0.0.4', 8888)) # Assuming server IP is 10.0.0.4
message = 'Hello from the client'
client_socket.send(message.encode())
client_socket.close()

# Save the message to a file
with open('messages.txt', 'a') as file:
    file.write(f'Client: {message}\n')
```

Server.py

```
import socket

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(('0.0.0.0', 8888))
server_socket.listen(1)
print("Server listening on port 8888")

while True:
    connection, address = server_socket.accept()
    data = connection.recv(1024).decode()
    print(f'Received data from {address}: {data}')

    # Save the message to a file
    with open('messages.txt', 'a') as file:
        file.write(f'Server message received from ({address}): {data}\n')

    connection.close()
```

```
Client: Hello from the client
Client: Hello from the client
Server message received from (('10.0.0.1', 53858)): Hello from the client
Server message received from (('10.0.0.2', 42112)): Hello from the client
Client: Hello from the client
Server message received from (('10.0.0.3', 44278)): Hello from the client
```

b. Client- h1, Server- h4

Running the iperf server on host h4 and iperf client on host h1 simultaneously for 30 seconds.

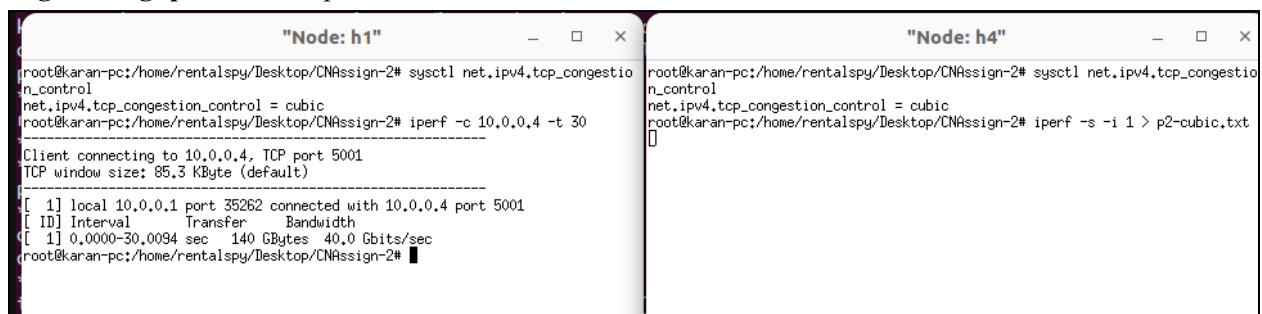
Note: Output of xterm terminals are provided as a confirmation that congestion has been applied. These outputs are generated manually, although the code is automated.

The command on execution generates a txt file containing the Throughput information along with bandwidth.

Congestion control mechanism = cubic

Command: sudo mn -c; python3 Q2.py --config=b --congestion=cubic

Avg.Throughput: 40.0 Gbps



The image shows two terminal windows side-by-side. The left window is titled "Node: h1" and the right window is titled "Node: h4". Both windows show the execution of commands to set the congestion control mechanism to 'cubic' and run iperf. The output in the h1 window shows a client connecting to 10.0.0.4 on port 5001 and a table of results showing a bandwidth of 40.0 Gbits/sec. The h4 window shows the server listening on port 5001 and the iperf command being executed.

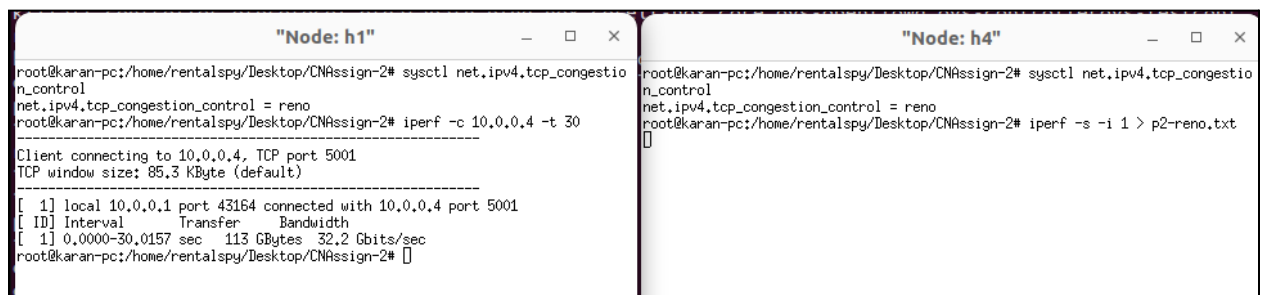
```
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sysctl net.ipv4.tcp_congestion_control
net.ipv4.tcp_congestion_control = cubic
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -c 10.0.0.4 -t 30
-----
Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 85.3 KByte (default)
-----
[ 1] local 10.0.0.1 port 35262 connected with 10.0.0.4 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 1] 0.0000-30.0094 sec  140 GBytes  40.0 Gbits/sec
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2#
```

```
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sysctl net.ipv4.tcp_congestion_control
net.ipv4.tcp_congestion_control = cubic
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -s -i 1 > p2-cubic.txt
```

Congestion control mechanism = reno

Command: sudo mn -c; python3 Q2.py --config=b --congestion=reno

Avg.Throughput: 32.2 Gbps



The image shows two terminal windows side-by-side. The left window is titled "Node: h1" and the right window is titled "Node: h4". Both windows show the execution of commands to set the congestion control mechanism to 'reno' and run iperf. The output in the h1 window shows a client connecting to 10.0.0.4 on port 5001 and a table of results showing a bandwidth of 32.2 Gbits/sec. The h4 window shows the server listening on port 5001 and the iperf command being executed.

```
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sysctl net.ipv4.tcp_congestion_control
net.ipv4.tcp_congestion_control = reno
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -c 10.0.0.4 -t 30
-----
Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 85.3 KByte (default)
-----
[ 1] local 10.0.0.1 port 43164 connected with 10.0.0.4 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 1] 0.0000-30.0157 sec  113 GBytes  32.2 Gbits/sec
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2#
```

```
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sysctl net.ipv4.tcp_congestion_control
net.ipv4.tcp_congestion_control = reno
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -s -i 1 > p2-reno.txt
```

Congestion control mechanism = vegas

Command: sudo mn -c; python3 Q2.py --config=b --congestion=vegas

Avg.Throughput:38.3 Gbps


```

root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sudo mn -c; python3 Q2.py --config=b --congestion=vegas
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes

"Node: h1"
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sysctl net.ipv4.tcp_congestion_control=vegas
net.ipv4.tcp_congestion_control = vegas
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -c 10.0.0.4 -t 30
Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 85.3 KByte (default)
[ 1] local 10.0.0.1 port 53666 connected with 10.0.0.4 port 5001

"Node: h4"
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sysctl net.ipv4.tcp_congestion_control=vegas
net.ipv4.tcp_congestion_control = vegas
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -s -i 1 > p2-vegas.txt

```

Congestion control mechanism = bbr

Command: sudo mn -c; python3 Q2.py --config=b --congestion=bbr

Avg.Throughput: 33.99 Gbps

```

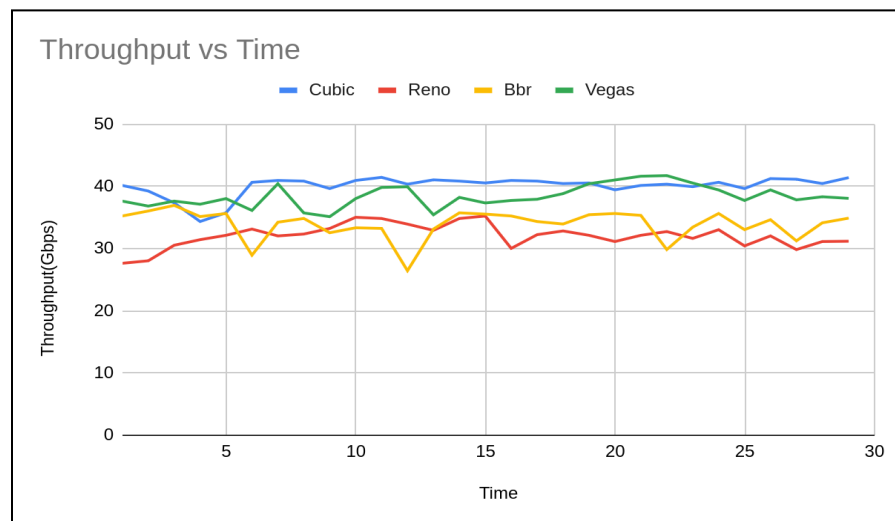
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sudo mn -c; python3 Q2.py --config=b --congestion=bbr
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes

"Node: h1"
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sysctl net.ipv4.tcp_congestion_control=bbr
net.ipv4.tcp_congestion_control = bbr
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -c 10.0.0.4 -t 30
Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 128 KByte (default)
[ 1] local 10.0.0.1 port 60880 connected with 10.0.0.4 port 5001

"Node: h4"
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sysctl net.ipv4.tcp_congestion_control=bbr
net.ipv4.tcp_congestion_control = bbr
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -s -i 1 > p2-bbr.txt

```

Graph



Avg. Throughput Order: Cubic > Vegas > Bbr > Reno

Since, there is only connection h1->h4, there is no competition for available bandwidth with any connection. So, we can easily see the effect of different congestion schemes.

c. Client - h1, h2, h3 , Server- h4

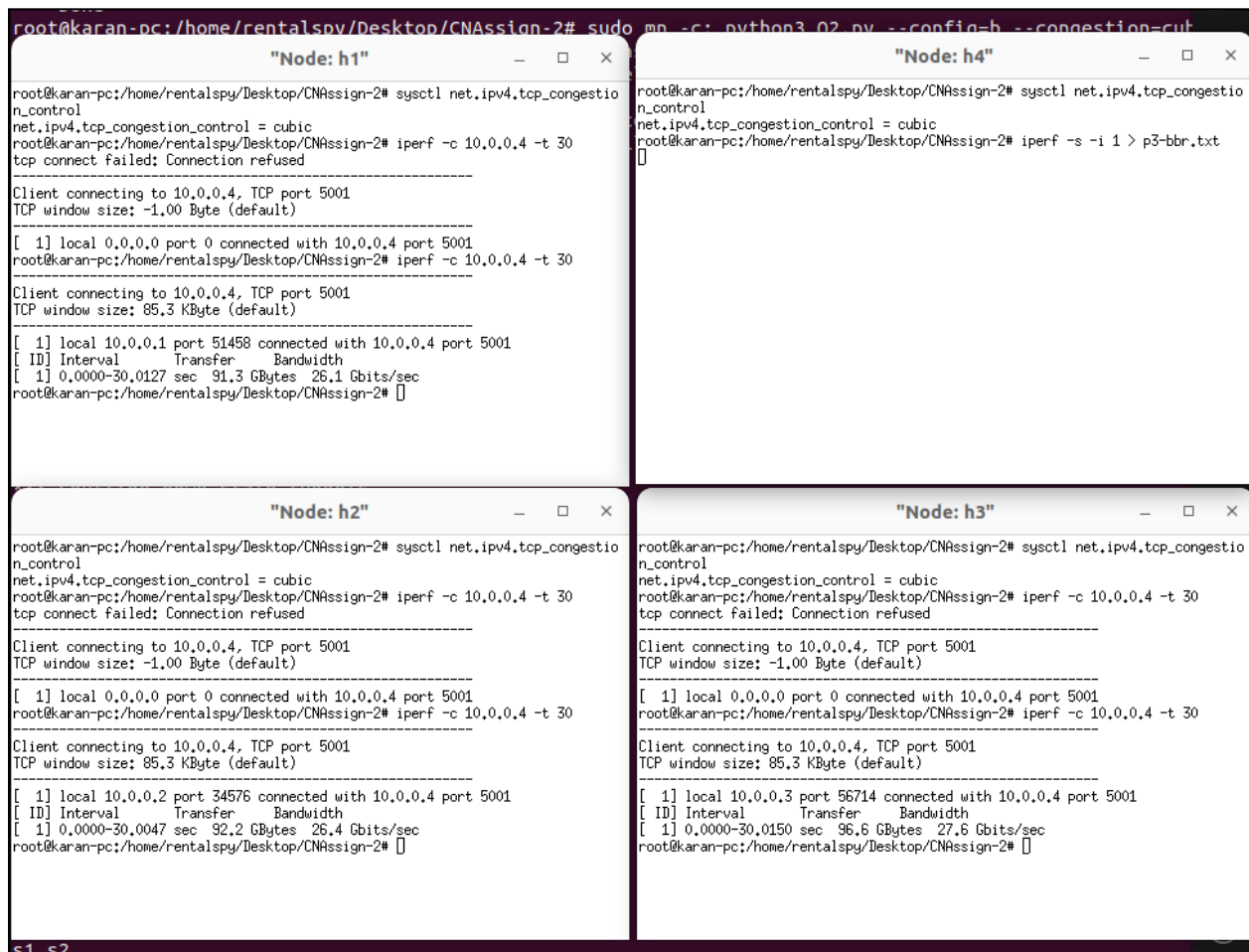
Running the iperf server on host h4 and iperf client on host h1, h2 and h3 simultaneously for 30 seconds.

Note: Output of xterm terminals are provided as a confirmation that congestion has been applied. These outputs are generated manually, although the code is automated.

The command on execution generates a txt file containing the Throughput information along with bandwidth.

Congestion control mechanism = cubic

Command: sudo mn -c; python3 Q2.py --config=c --congestion=cubic



The image displays four terminal windows arranged in a 2x2 grid, each showing the execution of network configuration and iperf commands on different nodes. The top-left window is titled "Node: h1", the top-right "Node: h4", the bottom-left "Node: h2", and the bottom-right "Node: h3". All windows show the same initial commands: `sysctl net.ipv4.tcp_congestion_control = cubic` and `iperf -c 10.0.0.4 -t 30`. The output for h1, h2, and h3 shows a successful connection to 10.0.0.4 port 5001 with a TCP window size of 85.3 KByte. The iperf results for h1 show a bandwidth of 26.1 Gbits/sec, while h2 and h3 show 26.4 Gbits/sec and 27.6 Gbits/sec respectively. The top-right window (h4) shows a different iperf command: `iperf -s -i 1 > p3-bbr.txt`, indicating it is acting as a server.

```
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sudo mn -c; python3 Q2.py --config=c --congestion=cubic

"Node: h1"
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sysctl net.ipv4.tcp_congestion_control = cubic
net.ipv4.tcp_congestion_control = cubic
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -c 10.0.0.4 -t 30
tcp connect failed: Connection refused

Client connecting to 10.0.0.4, TCP port 5001
TCP window size: -1.00 Byte (default)

[ 1] local 0.0.0.0 port 0 connected with 10.0.0.4 port 5001
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -c 10.0.0.4 -t 30

Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 85.3 KByte (default)

[ 1] local 10.0.0.1 port 51458 connected with 10.0.0.4 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 1] 0.0000-30.0127 sec  91.3 GBytes 26.1 Gbits/sec
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2#

"Node: h4"
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sysctl net.ipv4.tcp_congestion_control = cubic
net.ipv4.tcp_congestion_control = cubic
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -s -i 1 > p3-bbr.txt

"Node: h2"
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sysctl net.ipv4.tcp_congestion_control = cubic
net.ipv4.tcp_congestion_control = cubic
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -c 10.0.0.4 -t 30
tcp connect failed: Connection refused

Client connecting to 10.0.0.4, TCP port 5001
TCP window size: -1.00 Byte (default)

[ 1] local 0.0.0.0 port 0 connected with 10.0.0.4 port 5001
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -c 10.0.0.4 -t 30

Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 85.3 KByte (default)

[ 1] local 10.0.0.2 port 34576 connected with 10.0.0.4 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 1] 0.0000-30.0047 sec  92.2 GBytes 26.4 Gbits/sec
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2#

"Node: h3"
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sysctl net.ipv4.tcp_congestion_control = cubic
net.ipv4.tcp_congestion_control = cubic
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -c 10.0.0.4 -t 30
tcp connect failed: Connection refused

Client connecting to 10.0.0.4, TCP port 5001
TCP window size: -1.00 Byte (default)

[ 1] local 0.0.0.0 port 0 connected with 10.0.0.4 port 5001
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -c 10.0.0.4 -t 30

Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 85.3 KByte (default)

[ 1] local 10.0.0.3 port 56714 connected with 10.0.0.4 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 1] 0.0000-30.0150 sec  96.6 GBytes 27.6 Gbits/sec
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2#
```

Congestion control mechanism = reno

Command: sudo mn -c; python3 Q2.py --config=c --congestion=reno

```

"Node: h3"
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sysctl net.ipv4.tcp_congestion_control
net.ipv4.tcp_congestion_control = reno
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -c 10.0.0.4 -t 30

Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 85.3 KByte (default)

[ 1] local 10.0.0.3 port 53542 connected with 10.0.0.4 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 1] 0.0000-30.0210 sec 87.4 GBytes 25.0 Gbits/sec
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2#

"Node: h2"
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sysctl net.ipv4.tcp_congestion_control
net.ipv4.tcp_congestion_control = reno
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -c 10.0.0.4 -t 30

Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 85.3 KByte (default)

[ 1] local 10.0.0.2 port 41892 connected with 10.0.0.4 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 1] 0.0000-30.0003 sec 80.2 GBytes 22.9 Gbits/sec
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2#

"Node: h4"
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sysctl net.ipv4.tcp_congestion_control
net.ipv4.tcp_congestion_control = reno
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -s -i 1 > p3-reno.txt

"Node: h1"
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sysctl net.ipv4.tcp_congestion_control
net.ipv4.tcp_congestion_control = reno
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -c 10.0.0.4 -t 30

Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 85.3 KByte (default)

[ 1] local 10.0.0.1 port 46006 connected with 10.0.0.4 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 1] 0.0000-30.0074 sec 84.3 GBytes 24.1 Gbits/sec
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2#

```

Congestion control mechanism = vegas

Command: sudo mn -c; python3 Q2.py --config=c --congestion=vegas

```

"Node: h2"
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sysctl net.ipv4.tcp_congestion_control
net.ipv4.tcp_congestion_control = vegas
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -c 10.0.0.4 -t 30

Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 85.3 KByte (default)

[ 1] local 10.0.0.2 port 49948 connected with 10.0.0.4 port 5001

"Node: h1"
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sysctl net.ipv4.tcp_congestion_control
net.ipv4.tcp_congestion_control = vegas
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -c 10.0.0.4 -t 30

Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 85.3 KByte (default)

[ 1] local 10.0.0.1 port 42530 connected with 10.0.0.4 port 5001

"Node: h4"
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sysctl net.ipv4.tcp_congestion_control
net.ipv4.tcp_congestion_control = vegas
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -s -i 1 > p3-vegas.txt

"Node: h3"
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sysctl net.ipv4.tcp_congestion_control
net.ipv4.tcp_congestion_control = vegas
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -c 10.0.0.4 -t 30

Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 85.3 KByte (default)

[ 1] local 10.0.0.3 port 49446 connected with 10.0.0.4 port 5001

```

Congestion control mechanism = bbr

Command: sudo mn -c; python3 Q2.py --config=c --congestion=bbr

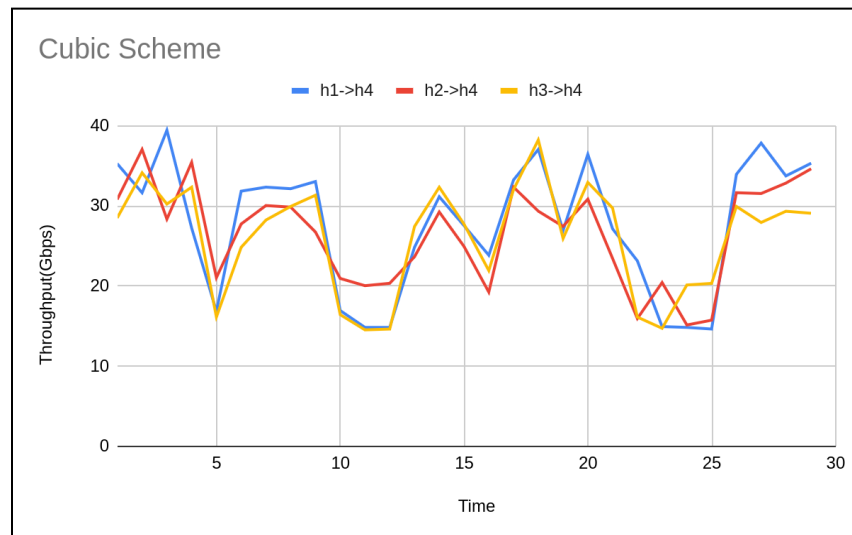
```
"Node: h2"
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sysctl net.ipv4.tcp_congestion_control
net.ipv4.tcp_congestion_control = bbr
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -c 10.0.0.4 -t 30
Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 128 KByte (default)
[ 1] local 10.0.0.2 port 40570 connected with 10.0.0.4 port 5001

"Node: h1"
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sysctl net.ipv4.tcp_congestion_control
net.ipv4.tcp_congestion_control = bbr
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -c 10.0.0.4 -t 30
tcp connect failed: Connection refused
Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 1,00 Byte (default)
[ 1] local 0.0.0.0 port 0 connected with 10.0.0.4 port 5001
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -c 10.0.0.4 -t 30
Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 128 KByte (default)
[ 1] local 10.0.0.1 port 50818 connected with 10.0.0.4 port 5001

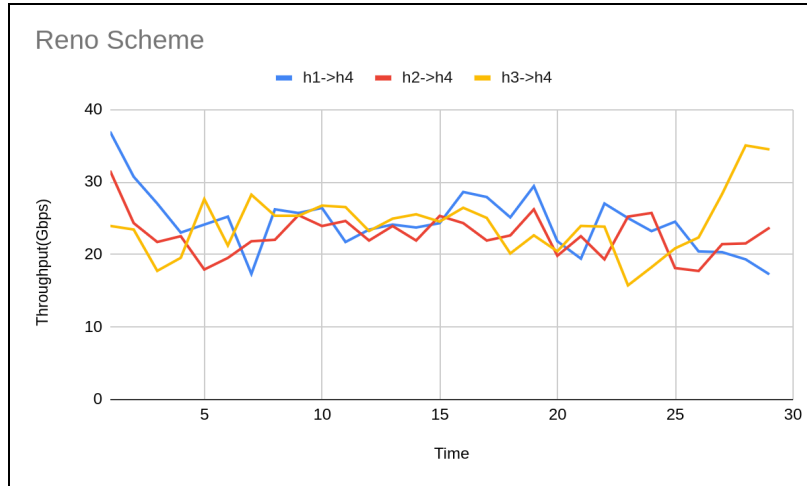
"Node: h4"
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sysctl net.ipv4.tcp_congestion_control
net.ipv4.tcp_congestion_control = bbr
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -s -i 1 > p3-bbr.txt

"Node: h3"
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# sysctl net.ipv4.tcp_congestion_control
net.ipv4.tcp_congestion_control = bbr
root@karan-pc:/home/rentalspy/Desktop/CNAssign-2# iperf -c 10.0.0.4 -t 30
Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 128 KByte (default)
[ 1] local 10.0.0.3 port 47198 connected with 10.0.0.4 port 5001
```

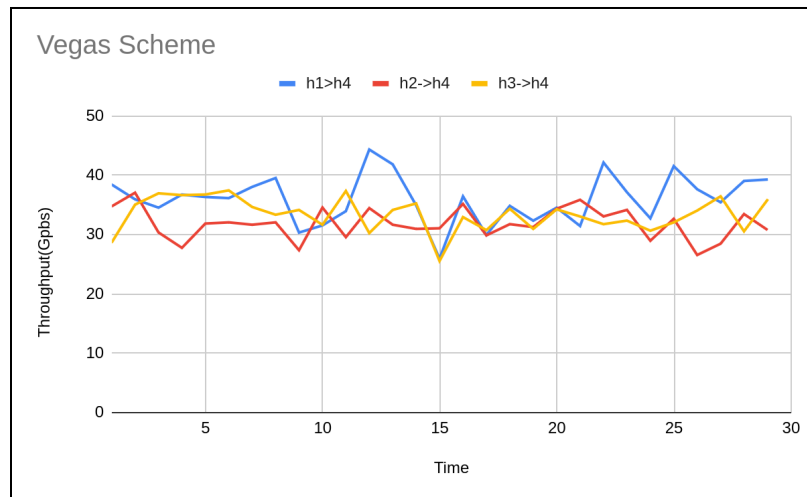
Graph



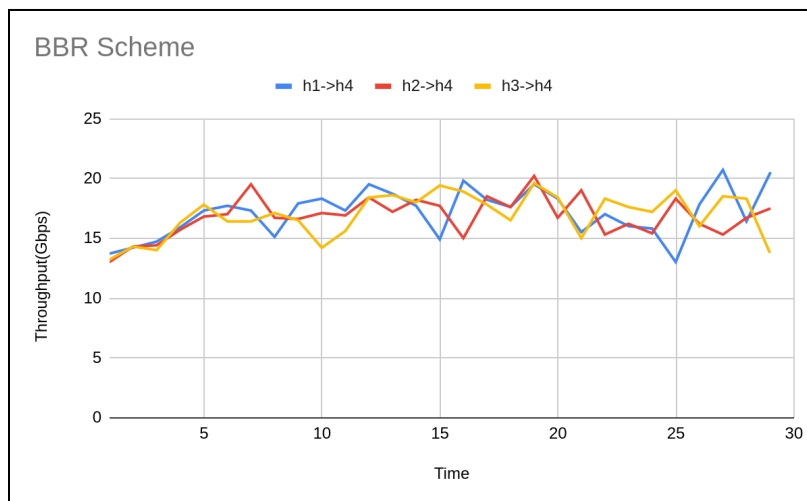
Avg. Throughput: 27.5 Gbps



Avg.Throughput: 24.1 Gbps



Avg.Throughput: 35.6 Gbps



Avg.Throughput: 15.7 Gbps

Observation: The Avg. Throughput for each scheme has reduced. The reason being that there are multiple connections and each connection competes for the available bandwidth, and as the number of connections increases, the available bandwidth per connection decreases, leading to lower throughput.

Avg. Throughput Order: Vegas > Cubic > Reno > Bbr

d. Link Loss Parameter

Command Used: `sudo mn -c; python3 Q2.py --config=d --congestion={congestion} --loss={1 or 3}`

The command on execution generates a txt file containing the Throughput information along with bandwidth.

Graph

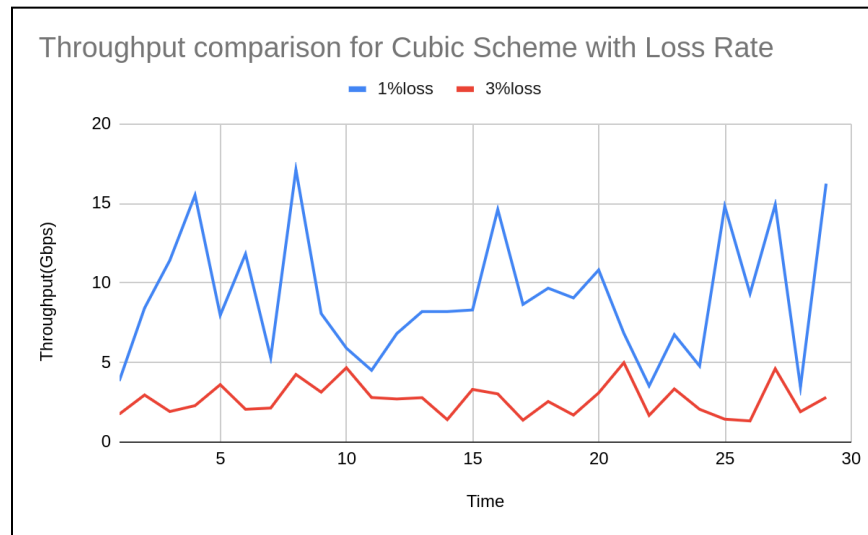
Bbr congestion scheme:



Avg. Throughput with 1% Loss: 14.1 Gbps

Avg. Throughput with 3% Loss: 2.34 Gbps

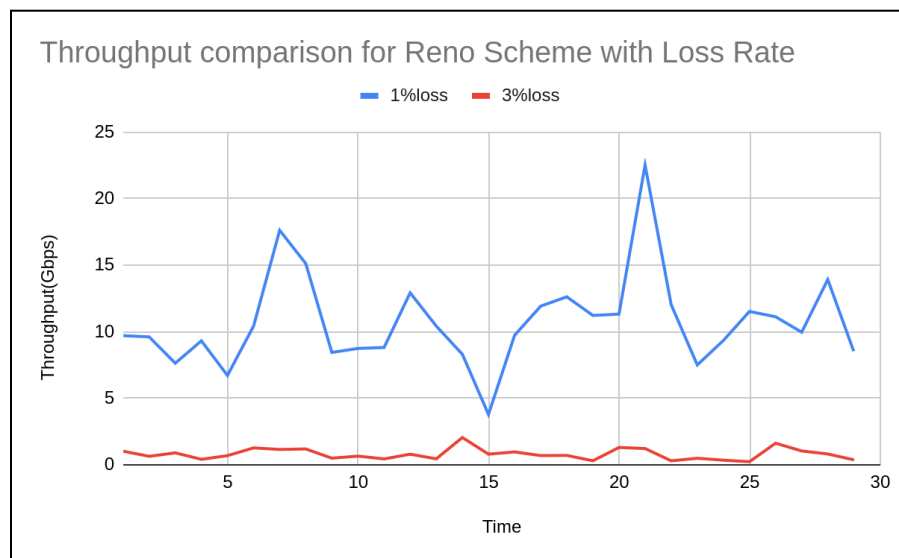
Cubic congestion scheme:



Avg. Throughput with 1% Loss: 9.05 Gbps

Avg. Throughput with 3% Loss: 2.41 Gbps

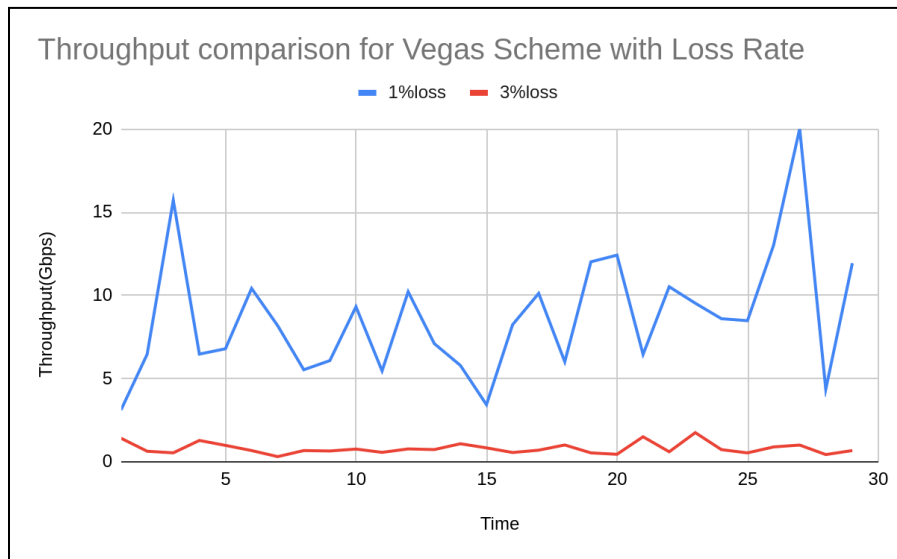
Reno congestion scheme:



Avg. Throughput with 1% Loss: 10.8 Gbps

Avg. Throughput with 3% Loss: 0.776 Gbps

Vegas congestion scheme:



Avg.Throughput with 1%Loss: 8.58 Gbps

Avg.Throughput with 3%Loss: 0.765 Gbps

Observation: The throughput gets reduced to $\frac{1}{3}$ when there is 1% loss. Also, there is a significant amount of throughput reduction as the loss went from 1% to 3%, around 10 times reduction.