NETWORK TOOLS AND PROTOCOLS

Lab 11: Router's Buffer Size

Document Version: 06-14-2019

Award 1829698
"CyberTraining CIP: Cyberinfrastructure Expertise on High-throughput Networks for Big Science Data Transfers"

Contents

Overview

This lab reviews the internal architecture of routers and switches. These devices are essential in high-speed networks, as they must be capable of absorbing transient packet bursts generated by large flows and thus avoid packet loss. The lab describes the buffer requirements to absorb such traffic fluctuations, which are then validated by experimental results.

Objectives

By the end of this lab, students should be able to:

1. Describe the internal architecture of routers and switches.
2. Understand the importance of buffers of routers and switches to prevent packet loss.
3. Conduct experiments with routers and switches of variable buffer sizes.
4. Calculate the buffer size required by routers and switches to absorb transient bursts.
5. Use experimental results to draw conclusions and make appropriate decision related to routers' and switches' buffers.

Lab settings

The information in Table 1 provides the credentials of the machine containing Mininet.

Table 1**.** Credentials to access Client1 machine.

Device	**Account**	**Password**
Client1	admin	password

Lab roadmap

This lab is organized as follows:

1. Section 1: Introduction.
2. Section 2: Lab topology.
3. Section 3: Testing throughput with 100*MTU switch's buffer size.
4. Section 4: Testing throughput with one BDP switch's buffer size.
5. Section 5: Emulating high-latency WAN with packet loss.

1 Introduction

1.1 Introduction to switching

Two essential functions performed by routers are routing and forwarding. Routing refers to the determination of the route taken by packets. Forwarding refers to the switching of a packet from the input port to the appropriate output port. The term switching is also used interchangeably with forwarding. Traditional routing approaches such as static and dynamic routing (e.g., Open Shortest Path First (OSPF)[1], BGP[2]) are used in the implementation of high-speed networks, e.g., Science DMZs. Routing events, such as routing table updates, occur at the millisecond, second, or minute timescale, and best practices used in regular enterprise networks are applicable to high-speed networks as well. These functions are sometimes collectively referred to as the control plane and are usually implemented in software and execute on the routing processor (typically a traditional CPU), see Figure 1. On the other hand, with transmission rates of 10 Gbps and above, the forwarding operations related to moving packets from input to output interfaces at very high speed must occur at the nanosecond timescale. Thus, forwarding operations, collectively referred to as forwarding or data plane, are executed in specialized hardware and optimized for performance.

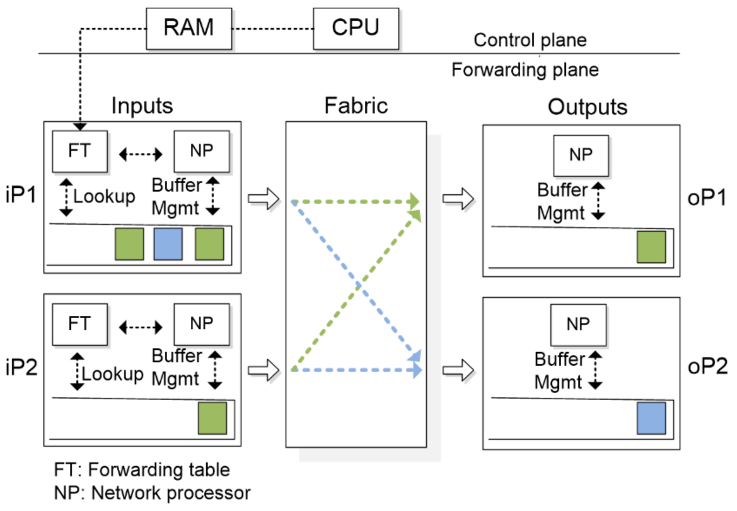


Figure 1. A generic router architecture.

Since forwarding functionality is common in both routers and switches, this lab reviews the architecture and forwarding-related attributes of switches. These attributes are applicable to routers as well; thus, for this lab, the terms switch and router are used interchangeably.

1.2 Router architecture

Consider the generic router architecture that is shown in Figure 1. Modern routers may have a network processor (NP) and a table derived from the routing table in each port, which is referred to as the forwarding table (FT) or forwarding information base (FIB). The router in Figure 1 has two input ports, iP1 and iP2, with their respective queues. iP1 has three packets in its queue, which will be forwarded to output ports oP1 (green packets) and oP2 (blue packet) by the fabric. A switch fabric moves packets from input to output ports. Switch fabric designs are shared memory, crossbar network, and bus. In shared memory switches, packets are written into a memory location by an input port and then read from that memory location by the output port. Crossbar switches implement a matrix of pathways that can be configured to connect any input port to any output port. Bus switches use a shared bus to move packets from the input ports to the output ports[3].

Router queues/buffers absorb traffic fluctuations. Even in the absence of congestion, fluctuations are present, resulting mostly from coincident traffic bursts[4]. Consider an input buffer implemented as a first-in first-out in the router of Figure 1. As iP1 and iP2 both have one packet to be forwarded to oP1 at the front of the buffer, only one of them, say the packet at iP2, will be forwarded to oP1. The consequence of this is that not only the first packet must wait at iP1. Also, the second packet that is queued at iP1 must wait, even though there is no contention for oP2. This phenomenon is known as Head-Of-Line (HOL) blocking[5]. To avoid HOL blocking, many switches use output buffering, a mixture of internal and output buffering, or techniques emulating output buffering such as Virtual Output Queueing (VOQ).

1.3 Where does packet loss occur?

Packet queues may form at both the input ports and the output ports. The location and extent of queueing (either at the input port queues or the output port queues) will depend on the traffic load, the relative speed of the switching fabric, and the line speed[5]. However, in modern switches with large switching rate capability, queues are commonly formed at output or transmission ports. A main contributing factor is the coincident arrivals of traffic bursts from different input ports that must be forwarded to the same output port. If transmission rates of input and output ports are the same, then packets from coincident arrivals must be momentarily buffered.

Note, however, that buffers will only prevent packet losses in case of transient traffic bursts. If those were not transient but permanent, such as approximately constant bit rates from large file transfers, the aggregate rate of input ports will surpass the rate of the output port. Thus, the output buffer would be permanently full, and the router would drop packets.

Packet loss occurs when a router drops the packet. It is the queues within a router, where such packets are dropped and lost.

1.4 Buffer size

From the above observation, a key question is how large should buffers be to absorb the fluctuations generated by TCP flows. The rule of thumb has been that the amount of buffering (in bits) in a router's port should equal the average Round-Trip Time (RTT) (in seconds) multiplied by the capacity C (in bits per seconds) of the port[6, 7].

$$\text{Router's buffer size} = C \cdot \text{RTT [bits]} \quad \text{(single / small number of flows)}$$

Note that RTT is the average of individual RTTs. For example, if there are five TCP flows sharing a router's link (port), the RTT used in the equation above is the average value of the five flows, and the capacity C is the router's port capacity. E.g., for 250 millisecond connections and a 10 Gbps port, the router's buffer size equals 2.5 Gbits. The above quantity is a conservative value that can be used in high-throughput high-latency networks.

In 2004, Appenzeller et al.[8] presented a study that suggests that when there is a large number of TCP flows passing through a link, say N (e.g., hundreds, thousands or more), the amount of buffering can be reduced to:

$$\text{Router's buffer size} = \frac{C \cdot \text{RTT}}{\sqrt{N}} \text{ [bits]} \quad \text{(large number of flows N)}$$

This result is observed when there is no dominant flow and the router aggregates hundreds, thousands, or more flows. The observed effect is that the fluctuation of the sum of congestion windows are smoothed, and the buffer size at an output port can be reduced to the expression given above. Note that N can be very large for campus and backbone networks, and the reduction in needed buffer size can become considerable.

2 Lab topology

Let's get started with creating a simple Mininet topology using Miniedit. The topology uses 10.0.0.0/8 which is the default network assigned by Mininet.

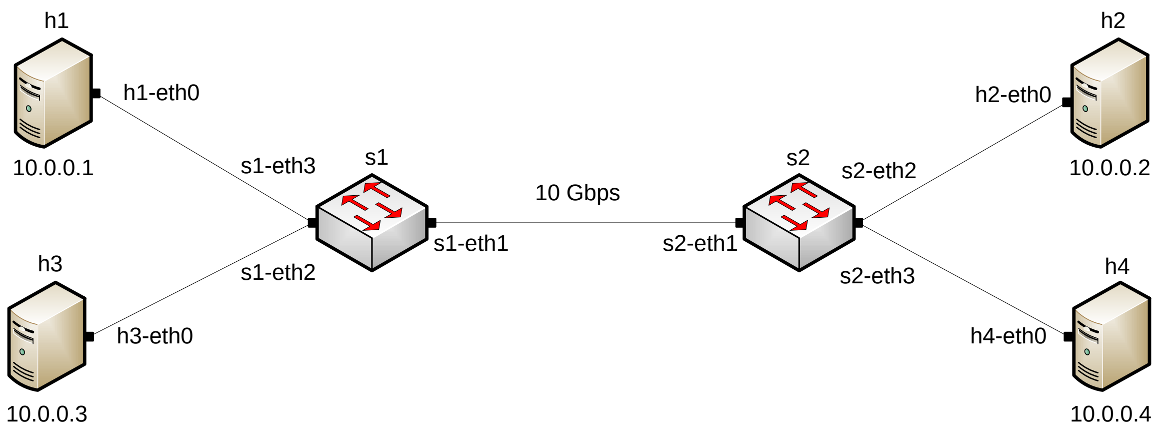


Figure 2. Lab topology.

The above topology uses 10.0.0.0/8 which is the default network assigned by Mininet.

Step 1. A shortcut to MiniEdit is located on the machine's Desktop. Start MiniEdit by clicking on MiniEdit's shortcut. When prompted for a password, type `password`.

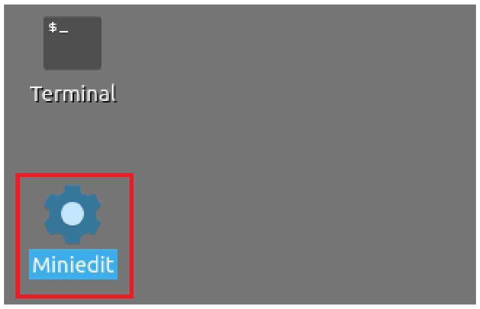


Figure 3. MiniEdit shortcut.

Step 2. On MiniEdit's menu bar, click on *File* then *Open* to load the lab's topology. Locate the *Lab 11.mn* topology file and click on *Open*.

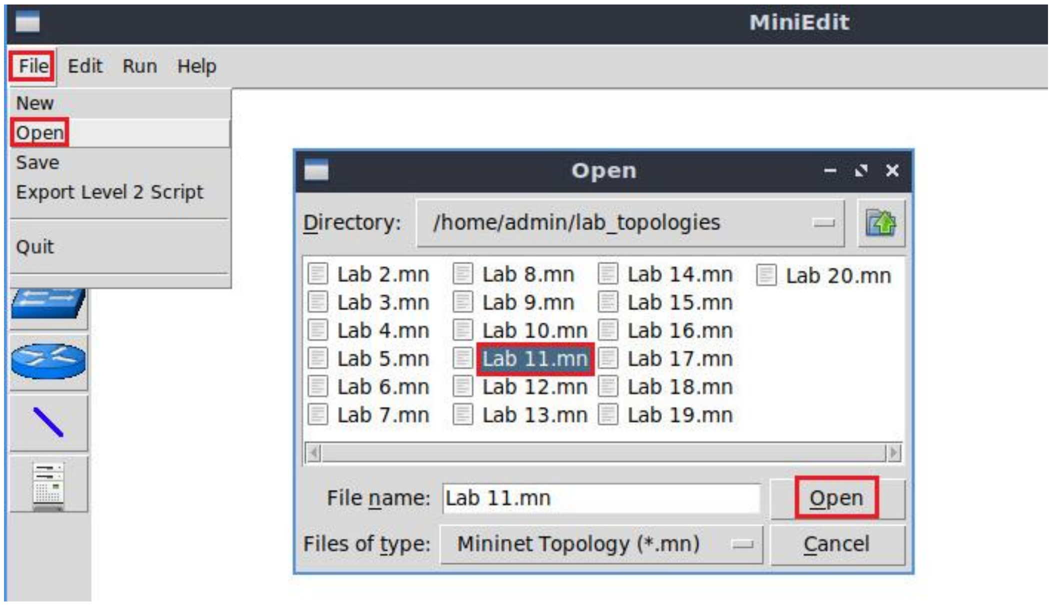


Figure 4. Miniedit's *Open* dialog.

Step 3. Before starting the measurements between host h1 and host h2, the network must be started. Click on the *Run* button located at the bottom left of MiniEdit's window to start the emulation.

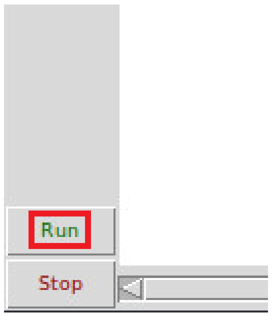


Figure 5. Running the emulation.

The above topology uses 10.0.0.0/8 which is the default network assigned by Mininet.

2.1 Starting host h1, host h2, host h3 and host h4

Step 1. Hold the right-click on host h1 and select *Terminal*. This opens the terminal of host h1 and allows the execution of commands on that host.

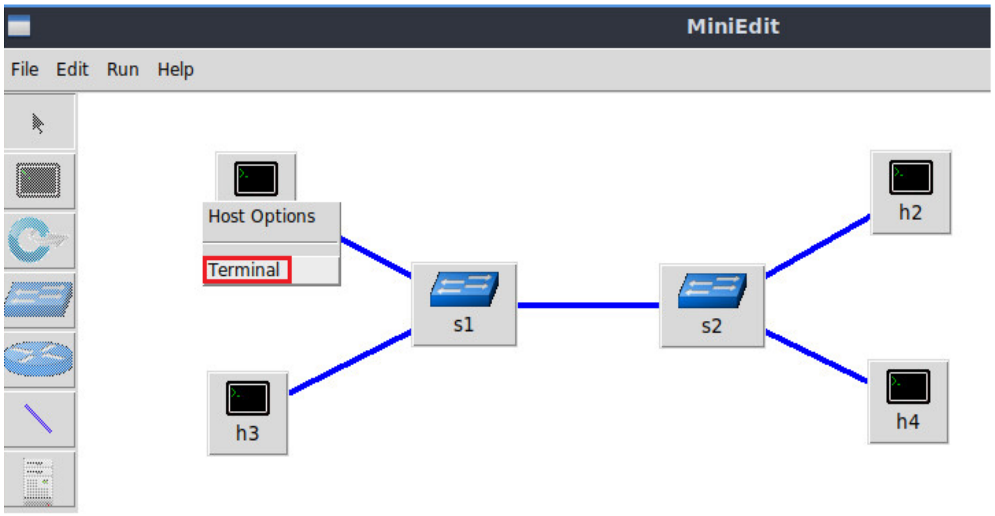
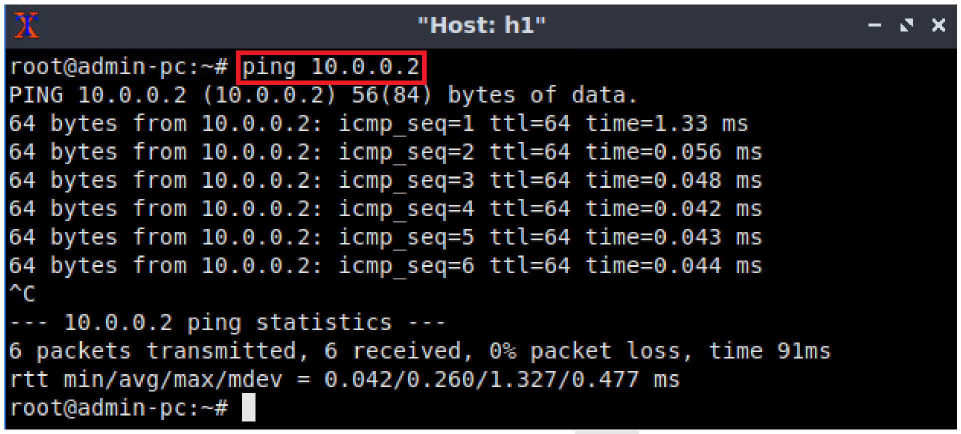


Figure 6. Opening a terminal on host h1.

Step 2. Apply the same steps on host h2 and open its *Terminal*.

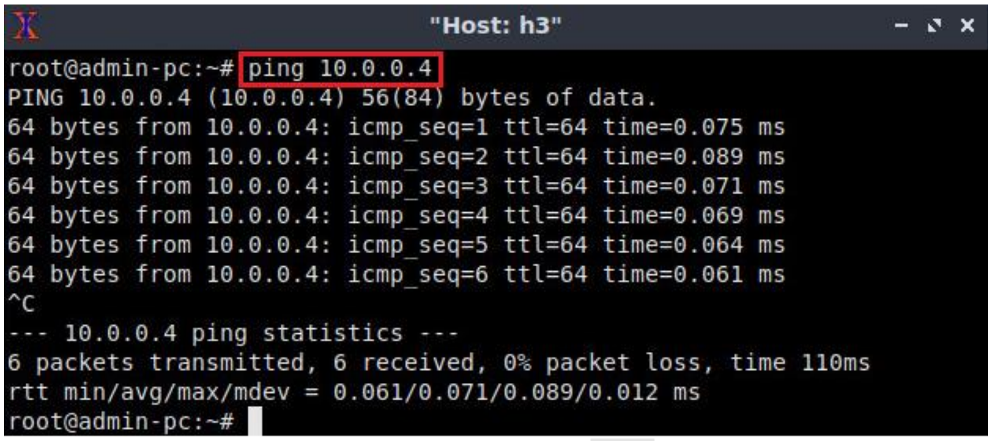
Step 3. Test connectivity between the end-hosts using the `ping` command. On host h1, type the command `ping 10.0.0.2`. This command tests the connectivity between host h1 and host h2. To stop the test, press `Ctrl+c`. The figure below shows a successful connectivity test.



```
root@admin-pc:~# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1.33 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.056 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.048 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.042 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.043 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.044 ms
^C
--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 91ms
rtt min/avg/max/mdev = 0.042/0.260/1.327/0.477 ms
root@admin-pc:~#
```

Figure 7. Connectivity test using `ping` command.

Step 4. Test connectivity between the end-hosts using the `ping` command. On host h3, type the command `ping 10.0.0.4`. This command tests the connectivity between host h3 and host h4. To stop the test, press `Ctrl+c`. The figure below shows a successful connectivity test.



```
root@admin-pc:~# ping 10.0.0.4
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=0.075 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=0.089 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.071 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=0.069 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=0.064 ms
64 bytes from 10.0.0.4: icmp_seq=6 ttl=64 time=0.061 ms
^C
--- 10.0.0.4 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 110ms
rtt min/avg/max/mdev = 0.061/0.071/0.089/0.012 ms
root@admin-pc:~#
```

Figure 8. Connectivity test using `ping` command.

2.2 Modifying hosts' buffer size

The following tests the bandwidth is limited to 10 Gbps, and the RTT (delay or latency) is 20ms.

In order to have enough TCP buffer size, we will set the sending and receiving buffer to $5 \cdot \text{BDP}$ in all hosts.

$$\text{BW} = 10{,}000{,}000{,}000 \text{ bits/second}$$

$$\text{RTT} = 0.02 \text{ seconds}$$

$$\begin{aligned}\text{BDP} &= 10{,}000{,}000{,}000 \cdot 0.02 = 200{,}000{,}000 \text{ bits} \\ &= 25{,}000{,}000 \text{ bytes} \approx 25 \text{ Mbytes}\end{aligned}$$

The send and receive buffer sizes should be set to $5 \cdot \text{BDP}$. We will use the 25 Mbytes value for the BDP instead of 25,000,000 bytes.

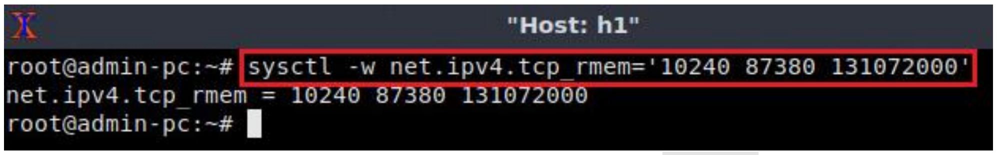
$$1 \text{ Mbyte} = 1024^2 \text{ bytes}$$

$$\text{BDP} = 25 \text{ Mbytes} = 25 \cdot 1024^2 \text{ bytes} = 26{,}214{,}400 \text{ bytes}$$

$$5 \cdot \text{BDP} = 5 \cdot 26{,}214{,}400 \text{ bytes} = 131{,}072{,}000 \text{ bytes}$$

Step 1. Now, we have calculated the maximum value of the TCP sending and receiving buffer size. In order to change the receiving buffer size, on host h1's terminal type the command shown below. The values set are: 10,240 (minimum), 87,380 (default), and 131,072,000 (maximum).

```
sysctl -w net.ipv4.tcp_rmem='10240 87380 131072000'
```



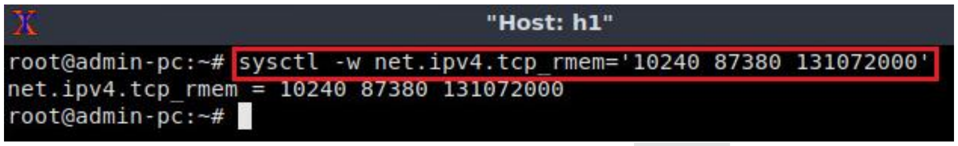
```
"Host: h1"
root@admin-pc:~# sysctl -w net.ipv4.tcp_rmem='10240 87380 131072000'
net.ipv4.tcp_rmem = 10240 87380 131072000
root@admin-pc:~#
```

Figure 9. Receive window change in `sysctl`.

The returned values are measured in bytes. 10,240 represents the minimum buffer size that is used by each TCP socket. 87,380 is the default buffer which is allocated when applications create a TCP socket. 131,072,000 is the maximum receive buffer that can be allocated for a TCP socket.

Step 2. To change the current send-window size value(s), use the following command on host h1's terminal. The values set are: 10,240 (minimum), 87,380 (default), and 131,072,000 (maximum).

```
sysctl -w net.ipv4.tcp_wmem='10240 87380 131072000'
```



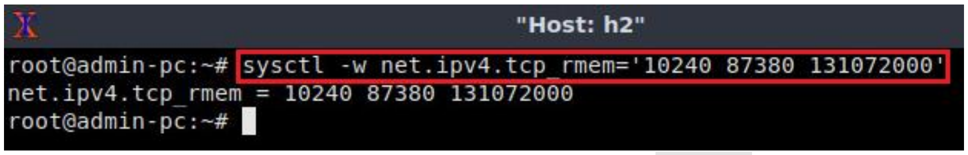
```
"Host: h1"
root@admin-pc:~# sysctl -w net.ipv4.tcp_rmem='10240 87380 131072000'
net.ipv4.tcp_rmem = 10240 87380 131072000
root@admin-pc:~#
```

Figure 10. Send window change in `sysctl`.

Next, the same commands must be configured on host h2, host h3, and host h4.

Step 3. To change the current receiver-window size value(s), use the following command on host h2's terminal. The values set are: 10,240 (minimum), 87,380 (default), and 131,072,000 (maximum).

```
sysctl -w net.ipv4.tcp_rmem='10240 87380 131072000'
```

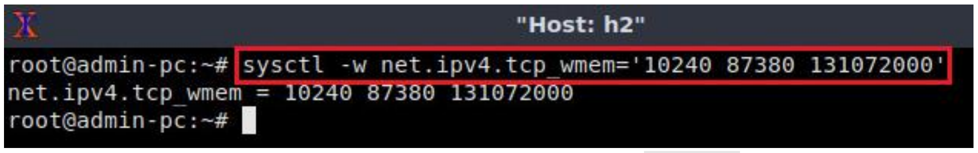


```
"Host: h2"
root@admin-pc:~# sysctl -w net.ipv4.tcp_rmem='10240 87380 131072000'
net.ipv4.tcp_rmem = 10240 87380 131072000
root@admin-pc:~#
```

Figure 11. Receive window change in `sysctl`.

Step 4. To change the current send-window size value(s), use the following command on host h2's terminal. The values set are: 10,240 (minimum), 87,380 (default), and 131,072,000 (maximum).

```
sysctl -w net.ipv4.tcp_wmem='10240 87380 131072000'
```

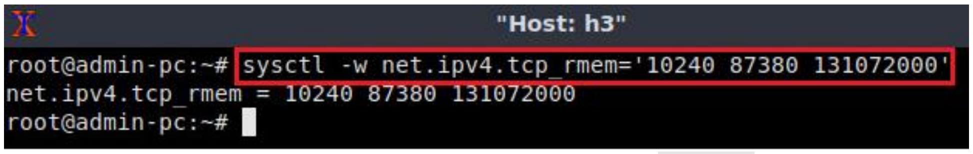


```
"Host: h2"
root@admin-pc:~# sysctl -w net.ipv4.tcp_wmem='10240 87380 131072000'
net.ipv4.tcp_wmem = 10240 87380 131072000
root@admin-pc:~#
```

Figure 12. Send window change in `sysctl`.

Step 5. To change the current receiver-window size value(s), use the following command on host h3's terminal. The values set are: 10,240 (minimum), 87,380 (default), and 131,072,000 (maximum).

```
sysctl -w net.ipv4.tcp_rmem='10240 87380 131072000'
```

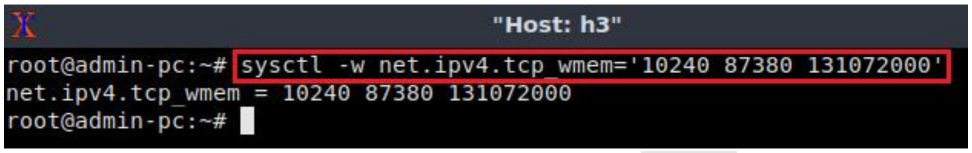


```
root@admin-pc:~# sysctl -w net.ipv4.tcp_rmem='10240 87380 131072000'
net.ipv4.tcp_rmem = 10240 87380 131072000
root@admin-pc:~# 
```

Figure 13. Receive window change in `sysctl`.

Step 6. To change the current send-window size value(s), use the following command on host h3's terminal. The values set are: 10,240 (minimum), 87,380 (default), and 131,072,000 (maximum).

```
sysctl -w net.ipv4.tcp_wmem='10240 87380 131072000'
```

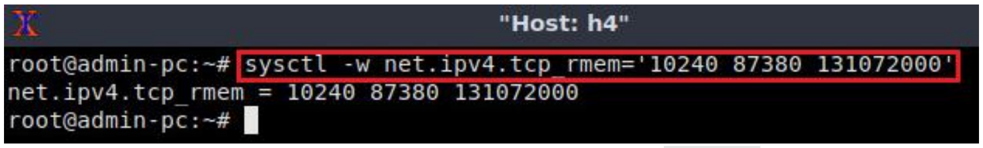


```
root@admin-pc:~# sysctl -w net.ipv4.tcp_wmem='10240 87380 131072000'
net.ipv4.tcp_wmem = 10240 87380 131072000
root@admin-pc:~# 
```

Figure 14. Send window change in `sysctl`.

Step 7. To change the current receiver-window size value(s), use the following command on host h4's terminal. The values set are: 10,240 (minimum), 87,380 (default), and 131,072,000 (maximum).

```
sysctl -w net.ipv4.tcp_rmem='10240 87380 131072000'
```

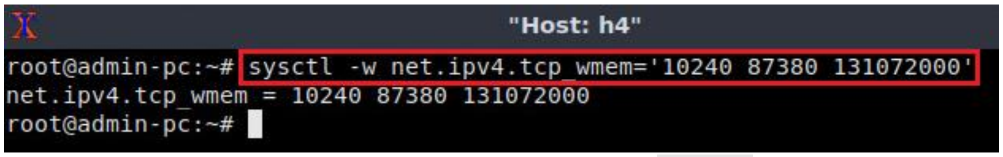


```
root@admin-pc:~# sysctl -w net.ipv4.tcp_rmem='10240 87380 131072000'
net.ipv4.tcp_rmem = 10240 87380 131072000
root@admin-pc:~# 
```

Figure 15. Receive window change in `sysctl`.

Step 8. To change the current send-window size value(s), use the following command on host h4's terminal. The values set are: 10,240 (minimum), 87,380 (default), and 131,072,000 (maximum).

```
sysctl -w net.ipv4.tcp_wmem='10240 87380 131072000'
```



```
root@admin-pc:~# sysctl -w net.ipv4.tcp_wmem='10240 87380 131072000'
net.ipv4.tcp_wmem = 10240 87380 131072000
root@admin-pc:~# 
```

Figure 16. Send window change in `sysctl`.

2.3 Emulating high-latency WAN

This section emulates a high-latency WAN. We will first emulate 20ms delay between switches, setting 10ms delay on switch S1 and 10ms delay on switch S2, resulting in 20ms of Round-Trip Time (*RTT)*.

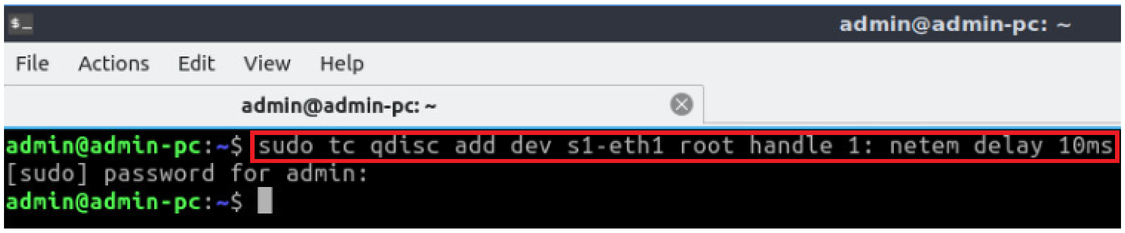
Step 1. Launch a Linux terminal by holding the `Ctrl+Alt+T` keys or by clicking on the Linux terminal icon.

Figure 17. Shortcut to open a Linux terminal.

The Linux terminal is a program that opens a window and permits you to interact with a command-line interface (CLI). A CLI is a program that takes commands from the keyboard and sends them to the operating system to perform.

Step 2. In the terminal, type the command below. When prompted for a password, type `password` and hit *Enter*. This command introduces 10ms delay to switch S1's *s1-eth1* interface.

```
sudo tc qdisc add dev s1-eth1 root handle 1: netem delay 10ms
```

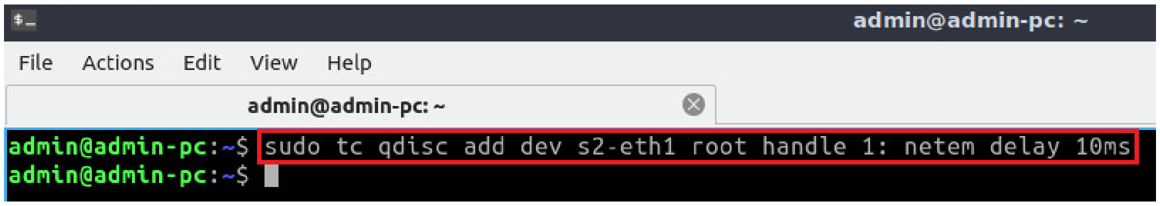


```
admin@admin-pc:~$ sudo tc qdisc add dev s1-eth1 root handle 1: netem delay 10ms
[sudo] password for admin:
admin@admin-pc:~$
```

Figure 18. Adding delay of 10ms to switch S1's *s1-eth1* interface.

Step 3. Similarly, repeat again the previous step to set a 10ms delay to switch S2's interface. When prompted for a password, type `password` and hit *Enter*. This command introduces 10ms delay on switch S2's *s2-eth1* interface.

```
sudo tc qdisc add dev s2-eth1 root handle 1: netem delay 10ms
```



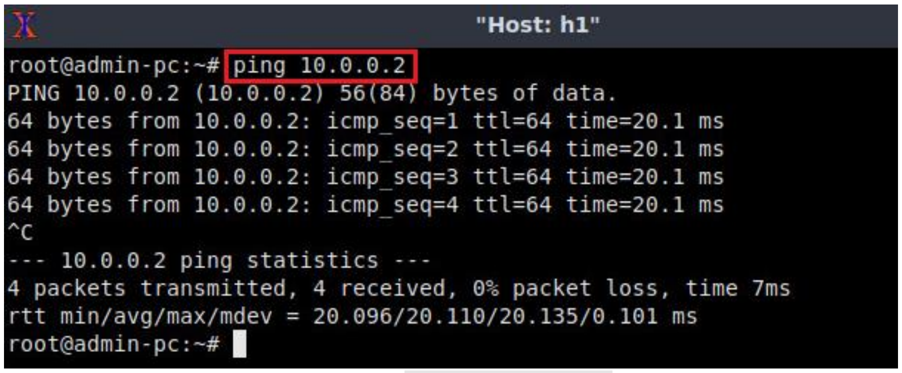
```
admin@admin-pc:~$ sudo tc qdisc add dev s2-eth1 root handle 1: netem delay 10ms
admin@admin-pc:~$
```

Figure 19. Adding delay of 10ms to switch S2's *s2-eth1* interface.

2.4 Testing connection

To test connectivity, you can use the command `ping`.

Step 1. On the terminal of host h1, type `ping 10.0.0.2`. To stop the test, press `Ctrl+c`. The figure below shows a successful connectivity test. Host h1 (10.0.0.1) sent four packets to host h2 (10.0.0.2), successfully receiving responses back.

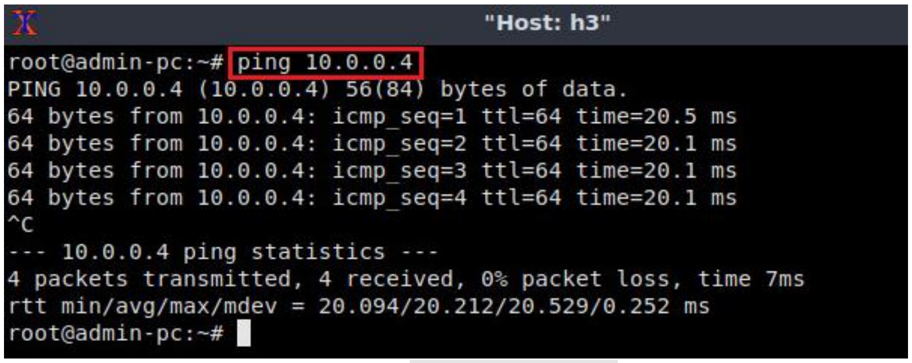


```
root@admin-pc:~# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=20.1 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=20.1 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=20.1 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=20.1 ms
^C
--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 7ms
rtt min/avg/max/mdev = 20.096/20.110/20.135/0.101 ms
root@admin-pc:~#
```

Figure 20. Output of `ping 10.0.0.2` command.

The result above indicates that all four packets were received successfully (0% packet loss) and that the minimum, average, maximum, and standard deviation of the Round-Trip Time (RTT) were 20.096, 20.110, 20.135, and 0.101 milliseconds, respectively. The output above verifies that delay was injected successfully, as the RTT is approximately 20ms.

Step 2. On the terminal of host h3, type `ping 10.0.0.4`. The ping output in this test should be relatively similar to the results of the test initiated by host h1 in Step 1. To stop the test, press `Ctrl+c`.



```
root@admin-pc:~# ping 10.0.0.4
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=20.5 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=20.1 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=20.1 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=20.1 ms
^C
--- 10.0.0.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 7ms
rtt min/avg/max/mdev = 20.094/20.212/20.529/0.252 ms
root@admin-pc:~#
```

Figure 21. Output of `ping 10.0.0.4` command.

The result above indicates that all four packets were received successfully (0% packet loss) and that the minimum, average, maximum, and standard deviation of the Round-Trip Time (RTT) were 20.094, 20.212, 20.529, and 0.252 milliseconds, respectively. The output above verifies that delay was injected successfully, as the RTT is approximately 20ms.

3 Testing throughput with 100·MTU switch's buffer size

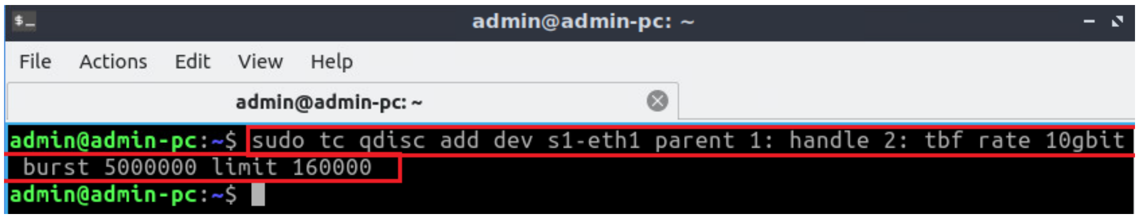
In this section, you are going to change the switch S1's buffer size to 100·MTU and emulate a 10 Gbps Wide Area Network (*WAN*) using the Token Bucket Filter (`tbf`). Then, you will test the throughput between host h1 and host h2 while there is another TCP flow between host h3 and host h4. On each test, you will modify the congestion control algorithm in host h1, namely, *cubic*, *reno* and *bbr*. The congestion control algorithm will still be *cubic* in host h3 for all tests. In this section, the MTU is 1600 bytes, thus the `tbf` limit value will be set to 100 · MTU = 160,000 bytes.

3.1 Setting switch S1's buffer size to 100·MTU

Step 1. Apply `tbf` rate limiting rule on switch S1's *s1-eth1* interface. In the client's terminal, type the command below. When prompted for a password, type `password` and hit *Enter*.

- `rate`: 10gbit
- `burst`: 5,000,000
- `limit`: 160,000

```
sudo tc qdisc add dev s1-eth1 parent 1: handle 2: tbf rate 10gbit burst 5000000
limit 160000
```



```
admin@admin-pc:~$ sudo tc qdisc add dev s1-eth1 parent 1: handle 2: tbf rate 10gbit
 burst 5000000 limit 160000
admin@admin-pc:~$
```

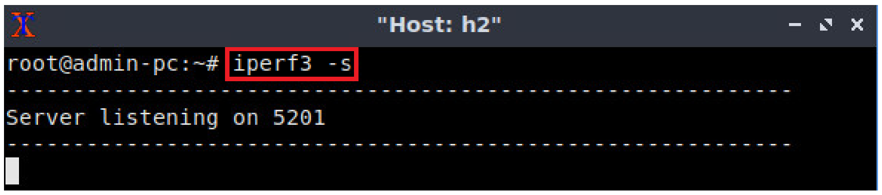
Figure 22. Limiting rate to 10 Gbps and setting the buffer size to 100·MTU on switch S1's interface.

3.2 TCP Cubic

The default congestion avoidance algorithm in the following test is *cubic* thus, there is no need to specify it manually.

Step 1. Launch iPerf3 in server mode on host h2's terminal.

```
iperf3 -s
```

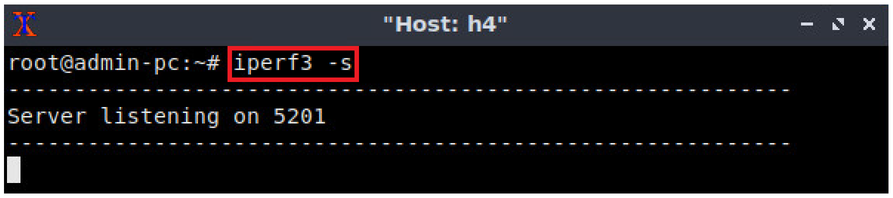


```
root@admin-pc:~# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Figure 23. Starting iPerf3 server on host h2.

Step 2. Launch iPerf3 in server mode on host h4's terminal.

```
iperf3 -s
```



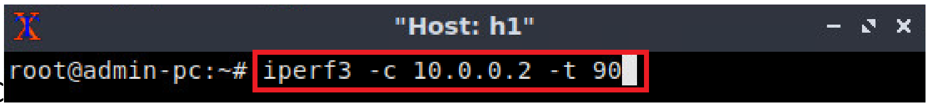
```
root@admin-pc:~# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Figure 24. Starting iPerf3 server on host h4.

The following two steps should be executed almost simultaneously thus, you will type the commands displayed in Step 3 and Step 4 then, in Step 5 you will execute them.

Step 3. Type the following iPerf3 command in host h1's terminal without executing it.

```
iperf3 -c 10.0.0.2 -t 90
```

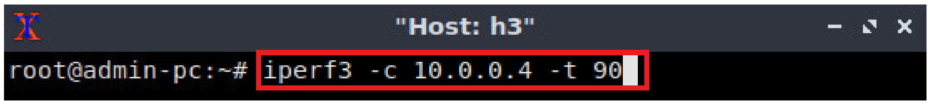


```
root@admin-pc:~# iperf3 -c 10.0.0.2 -t 90
```

Figure 25. Typing iPerf3 client command on host h1.

Step 4. Type the following iPerf3 command in host h3's terminal without executing it.

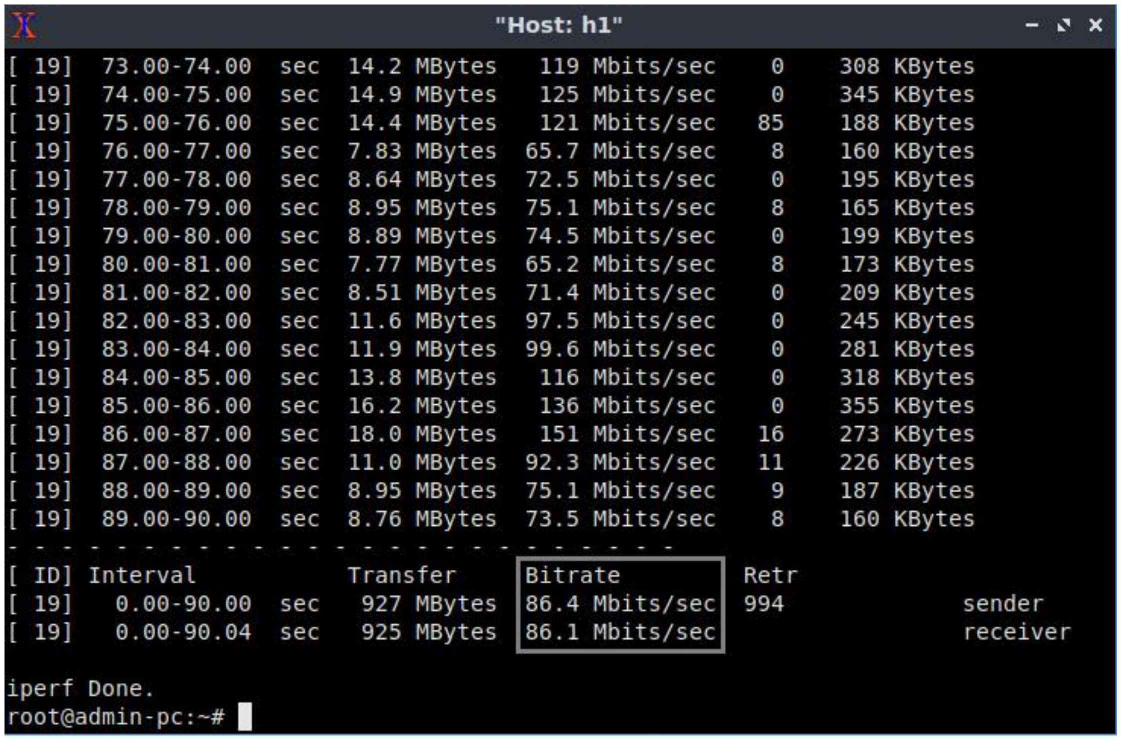
```
iperf3 -c 10.0.0.2 -t 90
```



```
root@admin-pc:~# iperf3 -c 10.0.0.4 -t 90
```

Figure 26. Typing iPerf3 client command on host h3.

Step 5. Press *Enter* to execute the commands, first in host h1 terminal then, in host h3 terminal.



```
"Host: h1"
[ 19]  73.00-74.00  sec  14.2 MBytes   119 Mbits/sec    0    308 KBytes
[ 19]  74.00-75.00  sec  14.9 MBytes   125 Mbits/sec    0    345 KBytes
[ 19]  75.00-76.00  sec  14.4 MBytes   121 Mbits/sec   85    188 KBytes
[ 19]  76.00-77.00  sec  7.83 MBytes  65.7 Mbits/sec    8    160 KBytes
[ 19]  77.00-78.00  sec  8.64 MBytes  72.5 Mbits/sec    0    195 KBytes
[ 19]  78.00-79.00  sec  8.95 MBytes  75.1 Mbits/sec    8    165 KBytes
[ 19]  79.00-80.00  sec  8.89 MBytes  74.5 Mbits/sec    0    199 KBytes
[ 19]  80.00-81.00  sec  7.77 MBytes  65.2 Mbits/sec    8    173 KBytes
[ 19]  81.00-82.00  sec  8.51 MBytes  71.4 Mbits/sec    0    209 KBytes
[ 19]  82.00-83.00  sec  11.6 MBytes  97.5 Mbits/sec    0    245 KBytes
[ 19]  83.00-84.00  sec  11.9 MBytes  99.6 Mbits/sec    0    281 KBytes
[ 19]  84.00-85.00  sec  13.8 MBytes   116 Mbits/sec    0    318 KBytes
[ 19]  85.00-86.00  sec  16.2 MBytes   136 Mbits/sec    0    355 KBytes
[ 19]  86.00-87.00  sec  18.0 MBytes   151 Mbits/sec   16    273 KBytes
[ 19]  87.00-88.00  sec  11.0 MBytes  92.3 Mbits/sec   11    226 KBytes
[ 19]  88.00-89.00  sec  8.95 MBytes  75.1 Mbits/sec    9    187 KBytes
[ 19]  89.00-90.00  sec  8.76 MBytes  73.5 Mbits/sec    8    160 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[ 19]   0.00-90.00  sec   927 MBytes  86.4 Mbits/sec  994             sender
[ 19]   0.00-90.04  sec   925 MBytes  86.1 Mbits/sec                  receiver

iperf Done.
root@admin-pc:~# 
```

Figure 27. Running iPerf3 client on host h1.

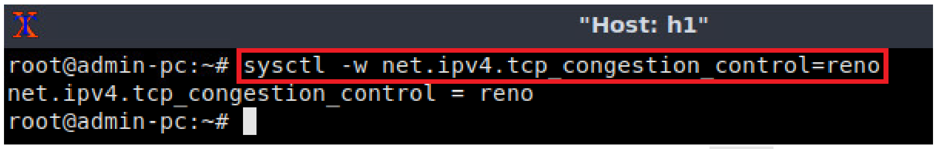
The figure above shows the iPerf3 test output report by the last 20 seconds. The average achieved throughput is 86.4 Mbps (sender) and 86.1 Mbps (receiver), and the number of retransmissions is 994. Host h3's results are similar to the above, however we are just focused on host h1's results.

Step 6. In order to stop the server, press `Ctrl+c` in host h2's and host h4's terminals. The user can see the throughput results in the server side too.

3.3 TCP Reno

Step 1. In host h1's terminal, change the TCP congestion control algorithm to Reno by typing the following command:

```
sysctl -w net.ipv4.tcp_congestion_control=reno
```



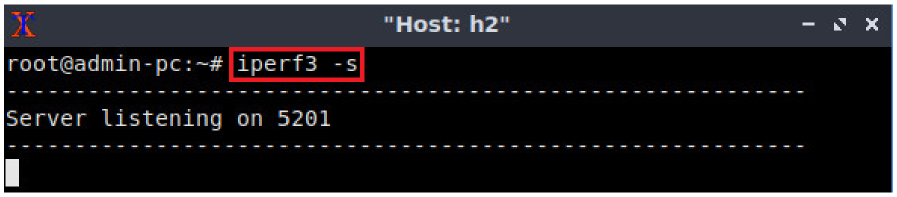
```
"Host: h1"
root@admin-pc:~# sysctl -w net.ipv4.tcp_congestion_control=reno
net.ipv4.tcp_congestion_control = reno
root@admin-pc:~# 
```

Figure 28. Changing TCP congestion control algorithm to `reno` in host h1.

Note that host h3's congestion control algorithm is cubic by default.

Step 2. Launch iPerf3 in server mode on host h2's terminal.

```
iperf3 -s
```

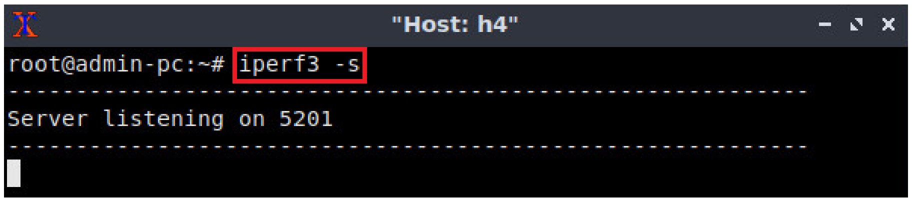


```
"Host: h2"
root@admin-pc:~# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Figure 29. Starting iPerf3 server on host h2.

Step 3. Launch iPerf3 in server mode on host h4's terminal.

```
iperf3 -s
```



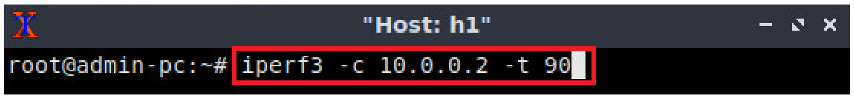
```
"Host: h4"
root@admin-pc:~# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Figure 30. Starting iPerf3 server on host h4.

The following two steps should be executed almost simultaneously thus, you will type the commands displayed in Step 3 and Step 4 then, in Step 5 you will execute them.

Step 4. Type the following iPerf3 command in host h1's terminal without executing it.

```
iperf3 -c 10.0.0.2 -t 90
```

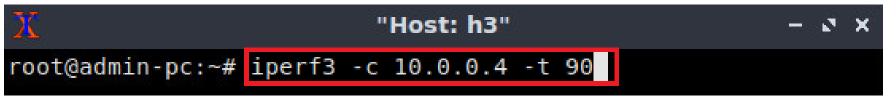


```
"Host: h1"
root@admin-pc:~# iperf3 -c 10.0.0.2 -t 90
```

Figure 31. Typing iPerf3 client command on host h1.

Step 5. Type the following iPerf3 command in host h3's terminal without executing it.

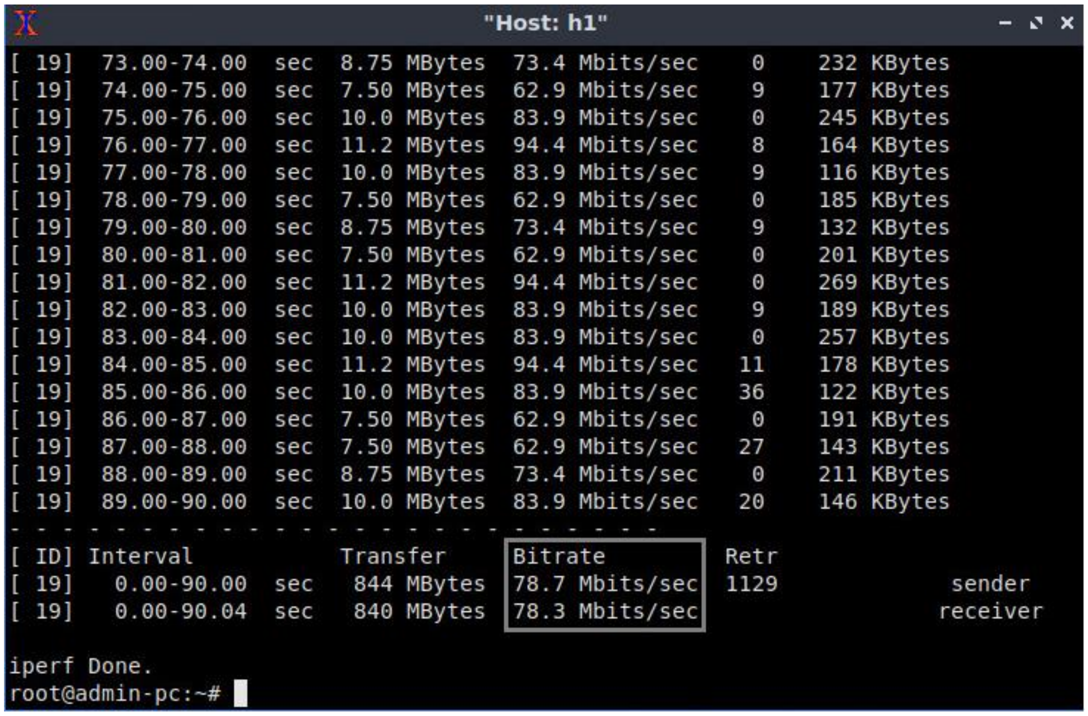
```
iperf3 -c 10.0.0.2 -t 90
```



```
"Host: h3"
root@admin-pc:~# iperf3 -c 10.0.0.4 -t 90
```

Figure 32. Typing iPerf3 client command on host h3.

Step 6. Press *Enter* to execute the commands, first in host h1 terminal then, in host h3 terminal.



```
[ 19]  73.00-74.00  sec  8.75 MBytes  73.4 Mbits/sec    0    232 KBytes
[ 19]  74.00-75.00  sec  7.50 MBytes  62.9 Mbits/sec    9    177 KBytes
[ 19]  75.00-76.00  sec  10.0 MBytes  83.9 Mbits/sec    0    245 KBytes
[ 19]  76.00-77.00  sec  11.2 MBytes  94.4 Mbits/sec    8    164 KBytes
[ 19]  77.00-78.00  sec  10.0 MBytes  83.9 Mbits/sec    9    116 KBytes
[ 19]  78.00-79.00  sec  7.50 MBytes  62.9 Mbits/sec    0    185 KBytes
[ 19]  79.00-80.00  sec  8.75 MBytes  73.4 Mbits/sec    9    132 KBytes
[ 19]  80.00-81.00  sec  7.50 MBytes  62.9 Mbits/sec    0    201 KBytes
[ 19]  81.00-82.00  sec  11.2 MBytes  94.4 Mbits/sec    0    269 KBytes
[ 19]  82.00-83.00  sec  10.0 MBytes  83.9 Mbits/sec    9    189 KBytes
[ 19]  83.00-84.00  sec  10.0 MBytes  83.9 Mbits/sec    0    257 KBytes
[ 19]  84.00-85.00  sec  11.2 MBytes  94.4 Mbits/sec   11    178 KBytes
[ 19]  85.00-86.00  sec  10.0 MBytes  83.9 Mbits/sec   36    122 KBytes
[ 19]  86.00-87.00  sec  7.50 MBytes  62.9 Mbits/sec    0    191 KBytes
[ 19]  87.00-88.00  sec  7.50 MBytes  62.9 Mbits/sec   27    143 KBytes
[ 19]  88.00-89.00  sec  8.75 MBytes  73.4 Mbits/sec    0    211 KBytes
[ 19]  89.00-90.00  sec  10.0 MBytes  83.9 Mbits/sec   20    146 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[ 19]   0.00-90.00  sec   844 MBytes  78.7 Mbits/sec  1129             sender
[ 19]   0.00-90.04  sec   840 MBytes  78.3 Mbits/sec                  receiver

iperf Done.
root@admin-pc:~#
```

Figure 33. Running iPerf3 client on host h1.

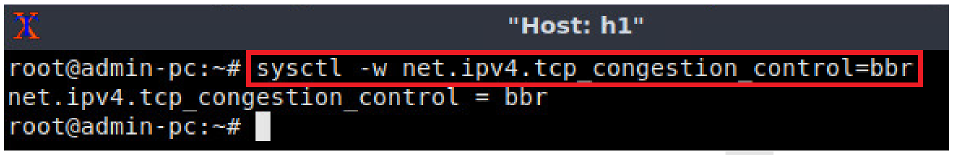
The figure above shows the iPerf3 test output report by the last 20 seconds. The average achieved throughput is 78.7 Mbps (sender) and 78.3 Mbps (receiver), and the number of retransmissions is 1129. Host h3's results are similar to the figure above, however we are just focused on host h1's results.

Step 7. In order to stop the server, press `Ctrl+c` in host h2's and host h4's terminals. The user can see the throughput results in the server side too.

3.4 TCP BBR

Step 1. In host h1's terminal, change the TCP congestion control algorithm to BBR by typing the following command:

```
sysctl -w net.ipv4.tcp_congestion_control=bbr
```



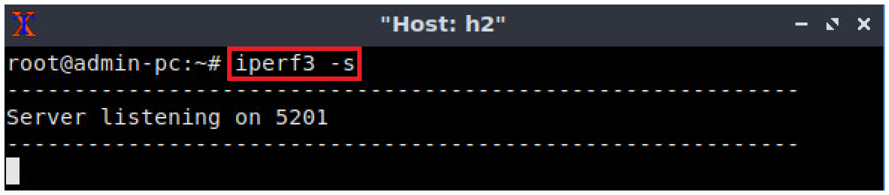
```
root@admin-pc:~# sysctl -w net.ipv4.tcp_congestion_control=bbr
net.ipv4.tcp_congestion_control = bbr
root@admin-pc:~#
```

Figure 34. Changing TCP congestion control algorithm to `bbr` in host h1.

Note that host h3's congestion control algorithm is cubic by default.

Step 2. Launch iPerf3 in server mode on host h2's terminal.

```
iperf3 -s
```

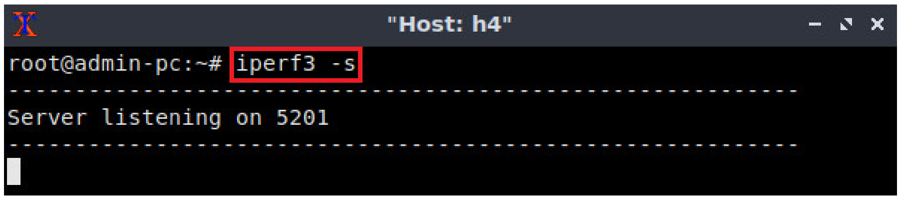


```
root@admin-pc:~# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Figure 35. Starting iPerf3 server on host h2.

Step 3. Launch iPerf3 in server mode on host h4's terminal.

```
iperf3 -s
```



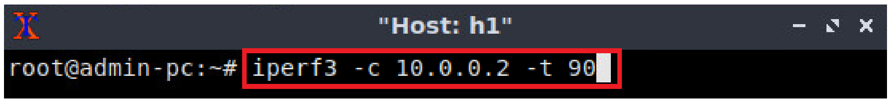
```
root@admin-pc:~# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Figure 36. Starting iPerf3 server on host h4.

The following two steps should be executed almost simultaneously, thus you will type the commands displayed in Step 3 and Step 4, then in Step 5 you will execute them.

Step 4. Type the following iPerf3 command in host h1's terminal without executing it.

```
iperf3 -c 10.0.0.2 -t 90
```

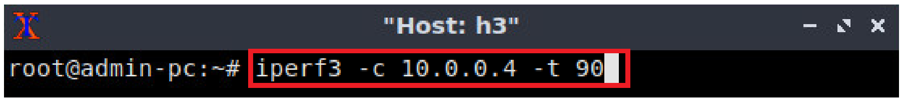


```
root@admin-pc:~# iperf3 -c 10.0.0.2 -t 90
```

Figure 37. Typing iPerf3 client command on host h1.

Step 5. Type the following iPerf3 command in host h3's terminal without executing it.

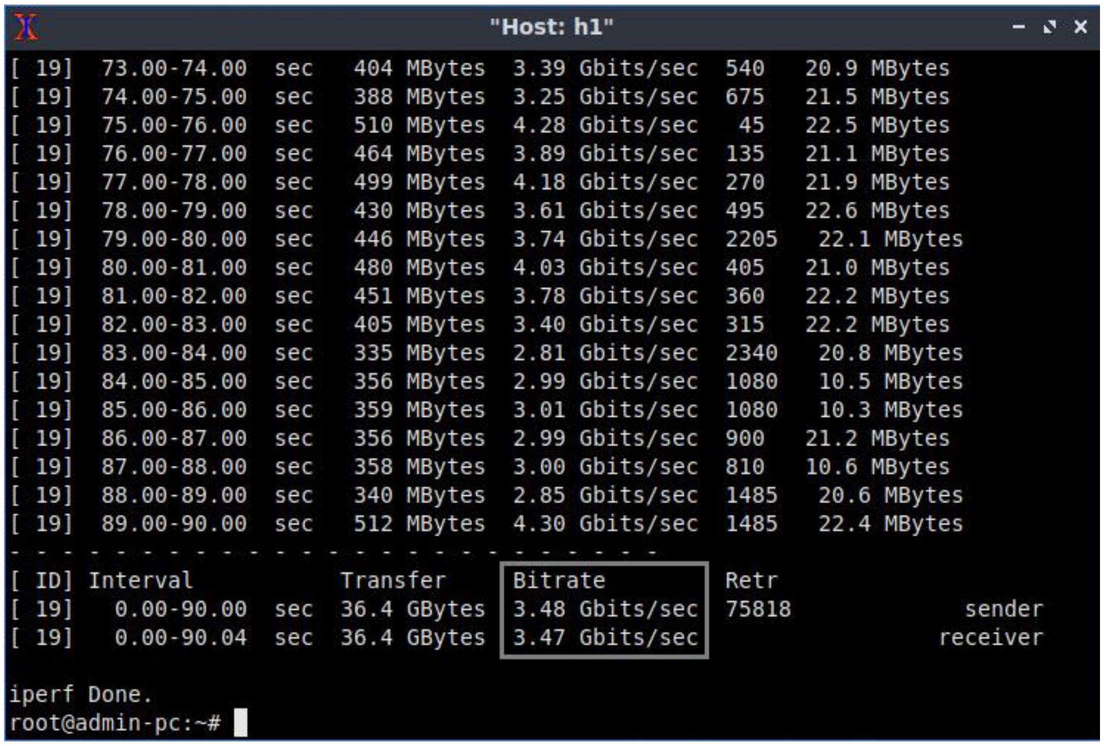
```
iperf3 -c 10.0.0.2 -t 90
```



```
root@admin-pc:~# iperf3 -c 10.0.0.4 -t 90
```

Figure 38. Typing iPerf3 client command on host h3.

Step 6. Press *Enter* to execute the commands, first in host h1 terminal then, in host h3 terminal.



```
[ 19]  73.00-74.00  sec   404 MBytes  3.39 Gbits/sec  540    20.9 MBytes
[ 19]  74.00-75.00  sec   388 MBytes  3.25 Gbits/sec  675    21.5 MBytes
[ 19]  75.00-76.00  sec   510 MBytes  4.28 Gbits/sec   45    22.5 MBytes
[ 19]  76.00-77.00  sec   464 MBytes  3.89 Gbits/sec  135    21.1 MBytes
[ 19]  77.00-78.00  sec   499 MBytes  4.18 Gbits/sec  270    21.9 MBytes
[ 19]  78.00-79.00  sec   430 MBytes  3.61 Gbits/sec  495    22.6 MBytes
[ 19]  79.00-80.00  sec   446 MBytes  3.74 Gbits/sec  2205    22.1 MBytes
[ 19]  80.00-81.00  sec   480 MBytes  4.03 Gbits/sec  405    21.0 MBytes
[ 19]  81.00-82.00  sec   451 MBytes  3.78 Gbits/sec  360    22.2 MBytes
[ 19]  82.00-83.00  sec   405 MBytes  3.40 Gbits/sec  315    22.2 MBytes
[ 19]  83.00-84.00  sec   335 MBytes  2.81 Gbits/sec  2340    20.8 MBytes
[ 19]  84.00-85.00  sec   356 MBytes  2.99 Gbits/sec  1080    10.5 MBytes
[ 19]  85.00-86.00  sec   359 MBytes  3.01 Gbits/sec  1080    10.3 MBytes
[ 19]  86.00-87.00  sec   356 MBytes  2.99 Gbits/sec  900    21.2 MBytes
[ 19]  87.00-88.00  sec   358 MBytes  3.00 Gbits/sec  810    10.6 MBytes
[ 19]  88.00-89.00  sec   340 MBytes  2.85 Gbits/sec  1485    20.6 MBytes
[ 19]  89.00-90.00  sec   512 MBytes  4.30 Gbits/sec  1485    22.4 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[ 19]   0.00-90.00  sec  36.4 GBytes  3.48 Gbits/sec  75818             sender
[ 19]   0.00-90.04  sec  36.4 GBytes  3.47 Gbits/sec                  receiver

iperf Done.
root@admin-pc:~#
```

Figure 39. Running iPerf3 client on host h1.

The figure above shows the iPerf3 test output report by the last 20 seconds. The average achieved throughput is 3.48 Gbps (sender) and 3.47 Gbps (receiver), and the number of retransmissions is 75818. Note that the congestion control algorithm used in host h1 is *bbr* and in host h3 is *cubic*.

Step 7. In order to stop the server, press `Ctrl+c` in host h2's and host h4's terminals. The user can see the throughput results in the server side too.

4 Testing throughput with one BDP switch's buffer size

In this section, you are going to change the switch S1 buffer size to one BDP (26,214,400) using the Token Bucket Filter (`tbf`). Then, you will test the throughput between host h1 and host h2 while there is another TCP flow between host h3 and host h4. On each test, you will modify the congestion control algorithm in host h1 namely, *cubic*, *reno* and *bbr*. The congestion control algorithm will still *cubic* in host 3 for all tests. In this section, the `tbf` limit value will be set to one BDP = 26,214,400 bytes.

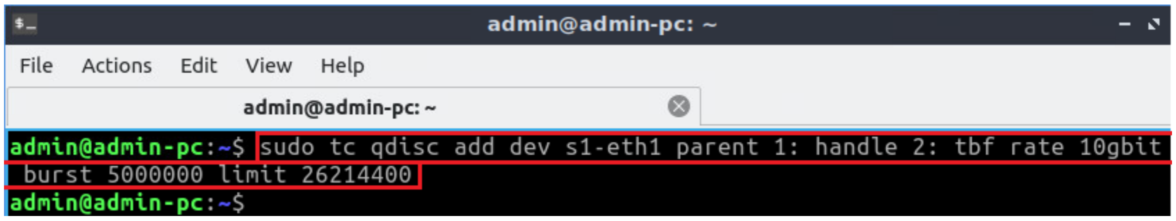
4.1 Changing switch S1's buffer size to one BDP

Step 1. Apply `tbf` rate limiting rule on switch S1's *s1-eth1* interface. In the client's terminal, type the command below. When prompted for a password, type `password` and hit *Enter*.

- `rate`: 10gbit

- `burst`: 5,000,000
- `limit`: 26,214,400

```
sudo tc qdisc change dev s1-eth1 parent 1: handle 2: tbf rate 10gbit burst
5000000 limit 26214400
```



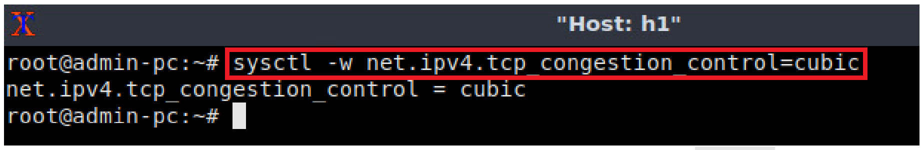
```
admin@admin-pc:~$ sudo tc qdisc add dev s1-eth1 parent 1: handle 2: tbf rate 10gbit
 burst 5000000 limit 26214400
admin@admin-pc:~$
```

Figure 40. Changing the buffer size to one BDP on switch S1's *s1-eth1* interface.

4.2 TCP Cubic

Step 1. In host h1's terminal, change the TCP congestion control algorithm to Cubic by typing the following command:

```
sysctl -w net.ipv4.tcp_congestion_control=cubic
```



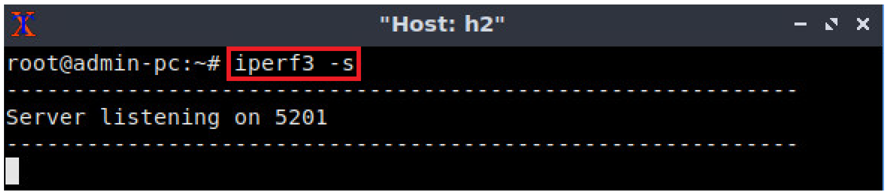
```
root@admin-pc:~# sysctl -w net.ipv4.tcp_congestion_control=cubic
net.ipv4.tcp_congestion_control = cubic
root@admin-pc:~#
```

Figure 41. Changing TCP congestion control algorithm to `cubic` in host h1.

Note that host h3's congestion control algorithm is cubic by default.

Step 2. Launch iPerf3 in server mode on host h2's terminal.

```
iperf3 -s
```

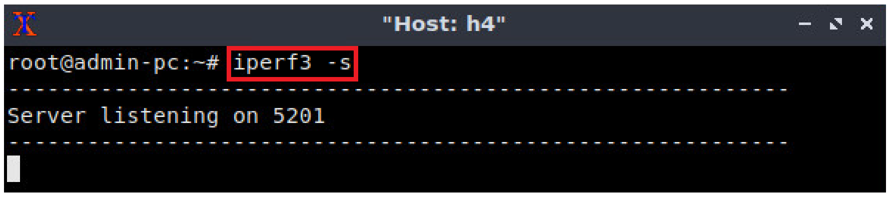


```
root@admin-pc:~# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Figure 42. Starting iPerf3 server on host h2.

Step 3. Launch iPerf3 in server mode on host h4's terminal.

```
iperf3 -s
```



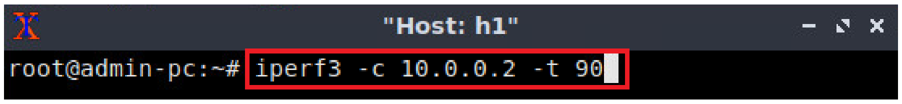
```
root@admin-pc:~# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Figure 43. Starting iPerf3 server on host h4.

The following two steps should be executed almost simultaneously, thus you will type the commands displayed in Step 3 and Step 4, then in Step 5 you will execute them.

Step 4. Type the following iPerf3 command in host h1's terminal without executing it.

```
iperf3 -c 10.0.0.2 -t 90
```

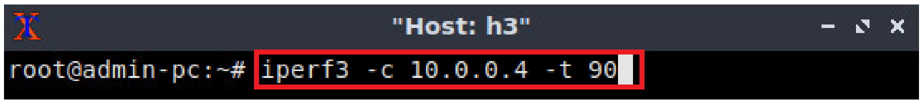


```
root@admin-pc:~# iperf3 -c 10.0.0.2 -t 90
```

Figure 44. Typing iPerf3 client command on host h1.

Step 5. Type the following iPerf3 command in host h3's terminal without executing it.

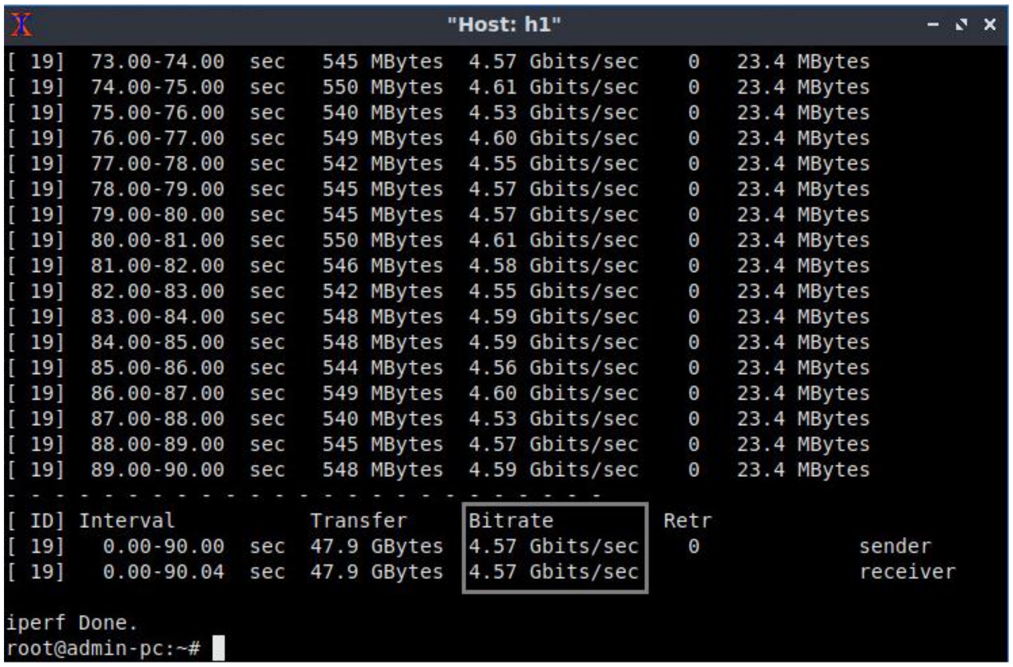
```
iperf3 -c 10.0.0.2 -t 90
```



```
root@admin-pc:~# iperf3 -c 10.0.0.4 -t 90
```

Figure 45. Typing iPerf3 client command on host h3.

Step 6. Press *Enter* to execute the commands, first in host h1 terminal then, in host h3 terminal.



```
[ 19]  73.00-74.00  sec   545 MBytes  4.57 Gbits/sec    0   23.4 MBytes
[ 19]  74.00-75.00  sec   550 MBytes  4.61 Gbits/sec    0   23.4 MBytes
[ 19]  75.00-76.00  sec   540 MBytes  4.53 Gbits/sec    0   23.4 MBytes
[ 19]  76.00-77.00  sec   549 MBytes  4.60 Gbits/sec    0   23.4 MBytes
[ 19]  77.00-78.00  sec   542 MBytes  4.55 Gbits/sec    0   23.4 MBytes
[ 19]  78.00-79.00  sec   545 MBytes  4.57 Gbits/sec    0   23.4 MBytes
[ 19]  79.00-80.00  sec   545 MBytes  4.57 Gbits/sec    0   23.4 MBytes
[ 19]  80.00-81.00  sec   550 MBytes  4.61 Gbits/sec    0   23.4 MBytes
[ 19]  81.00-82.00  sec   546 MBytes  4.58 Gbits/sec    0   23.4 MBytes
[ 19]  82.00-83.00  sec   542 MBytes  4.55 Gbits/sec    0   23.4 MBytes
[ 19]  83.00-84.00  sec   548 MBytes  4.59 Gbits/sec    0   23.4 MBytes
[ 19]  84.00-85.00  sec   548 MBytes  4.59 Gbits/sec    0   23.4 MBytes
[ 19]  85.00-86.00  sec   544 MBytes  4.56 Gbits/sec    0   23.4 MBytes
[ 19]  86.00-87.00  sec   549 MBytes  4.60 Gbits/sec    0   23.4 MBytes
[ 19]  87.00-88.00  sec   540 MBytes  4.53 Gbits/sec    0   23.4 MBytes
[ 19]  88.00-89.00  sec   545 MBytes  4.57 Gbits/sec    0   23.4 MBytes
[ 19]  89.00-90.00  sec   548 MBytes  4.59 Gbits/sec    0   23.4 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[ 19]   0.00-90.00  sec  47.9 GBytes  4.57 Gbits/sec    0             sender
[ 19]   0.00-90.04  sec  47.9 GBytes  4.57 Gbits/sec                  receiver

iperf Done.
root@admin-pc:~#
```

Figure 46. Running iPerf3 client on host h1.

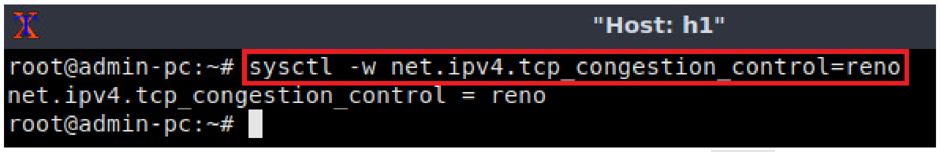
The figure above shows the iPerf3 test output report by the last 20 seconds. The average achieved throughput is 4.57 Gbps (sender) and 4.57 Gbps (receiver), and the number of retransmissions is 0. Note that the congestion avoidances algorithm used in host h1 and host h2 is *cubic*. Similar results are found in host h3 terminal.

Step 7. In order to stop the server, press `Ctrl+c` in host h2's and host h4's terminals. The user can see the throughput results in the server side too.

4.3 TCP Reno

Step 1. In host h1's terminal, change the TCP congestion control algorithm to Reno by typing the following command:

```
sysctl -w net.ipv4.tcp_congestion_control=reno
```



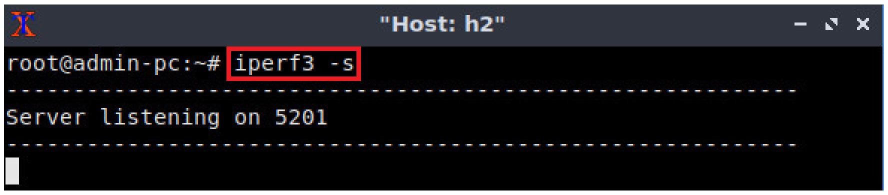
```
root@admin-pc:~# sysctl -w net.ipv4.tcp_congestion_control=reno
net.ipv4.tcp_congestion_control = reno
root@admin-pc:~#
```

Figure 47. Changing TCP congestion control algorithm to `reno` in host h1.

Note that host h3's congestion control algorithm is cubic by default.

Step 2. Launch iPerf3 in server mode on host h2's terminal.

```
iperf3 -s
```

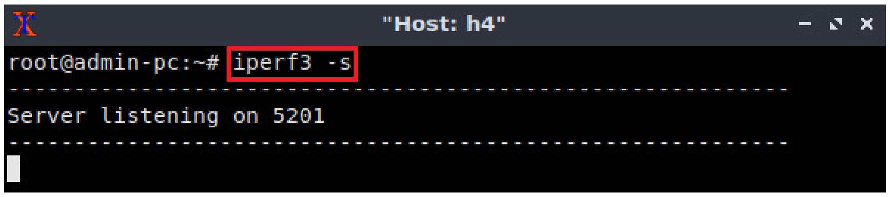


```
root@admin-pc:~# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Figure 48. Starting iPerf3 server on host h2.

Step 3. Launch iPerf3 in server mode on host h4's terminal.

```
iperf3 -s
```



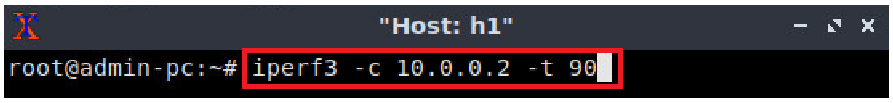
```
root@admin-pc:~# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Figure 49. Starting iPerf3 server on host h4.

The following two steps should be executed almost simultaneously, thus you will type the commands displayed in Step 3 and Step 4, then in Step 5 you will execute them.

Step 4. Type the following iPerf3 command in host h1's terminal without executing it.

```
iperf3 -c 10.0.0.2 -t 90
```

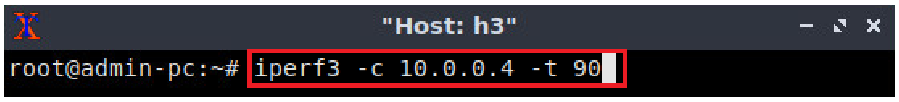


```
root@admin-pc:~# iperf3 -c 10.0.0.2 -t 90
```

Figure 50. Typing iPerf3 client command on host h1.

Step 5. Type the following iPerf3 command in host h3's terminal without executing it.

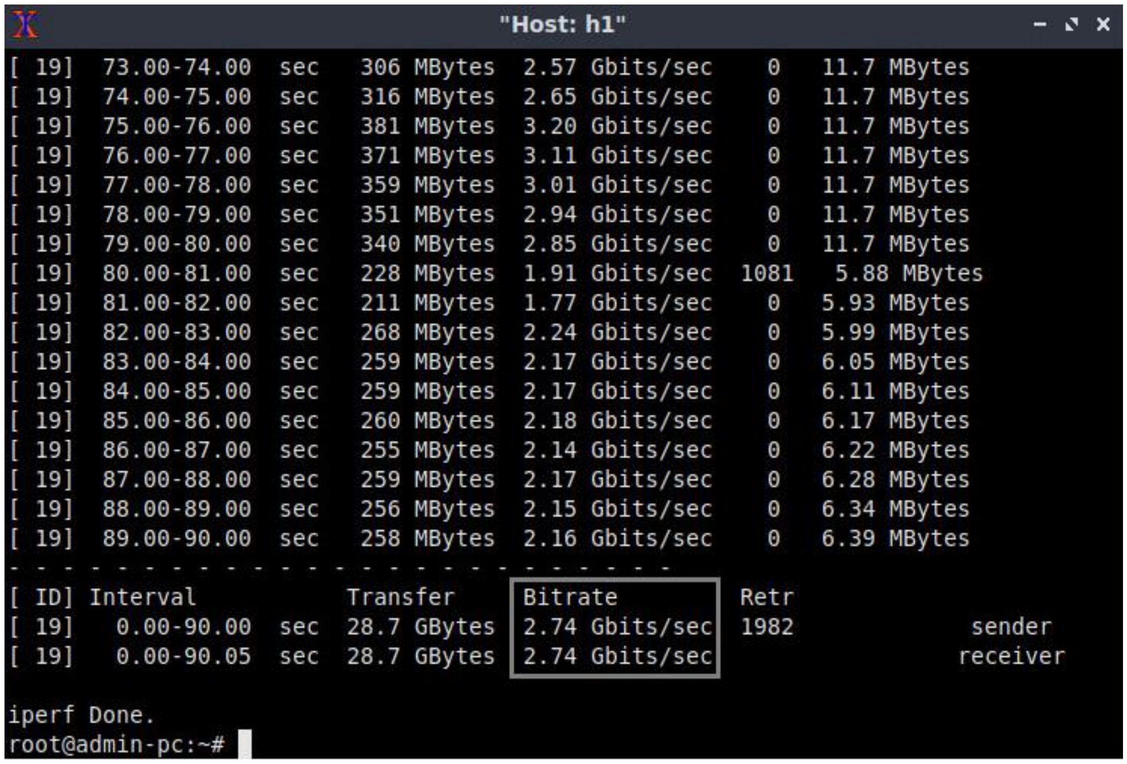
```
iperf3 -c 10.0.0.2 -t 90
```



```
root@admin-pc:~# iperf3 -c 10.0.0.4 -t 90
```

Figure 51. Typing iPerf3 client command on host h3.

Step 6. Press *Enter* to execute the commands, first in host h1 terminal then, in host h3 terminal.



```
[ 19]  73.00-74.00  sec   306 MBytes  2.57 Gbits/sec    0   11.7 MBytes
[ 19]  74.00-75.00  sec   316 MBytes  2.65 Gbits/sec    0   11.7 MBytes
[ 19]  75.00-76.00  sec   381 MBytes  3.20 Gbits/sec    0   11.7 MBytes
[ 19]  76.00-77.00  sec   371 MBytes  3.11 Gbits/sec    0   11.7 MBytes
[ 19]  77.00-78.00  sec   359 MBytes  3.01 Gbits/sec    0   11.7 MBytes
[ 19]  78.00-79.00  sec   351 MBytes  2.94 Gbits/sec    0   11.7 MBytes
[ 19]  79.00-80.00  sec   340 MBytes  2.85 Gbits/sec    0   11.7 MBytes
[ 19]  80.00-81.00  sec   228 MBytes  1.91 Gbits/sec  1081   5.88 MBytes
[ 19]  81.00-82.00  sec   211 MBytes  1.77 Gbits/sec    0   5.93 MBytes
[ 19]  82.00-83.00  sec   268 MBytes  2.24 Gbits/sec    0   5.99 MBytes
[ 19]  83.00-84.00  sec   259 MBytes  2.17 Gbits/sec    0   6.05 MBytes
[ 19]  84.00-85.00  sec   259 MBytes  2.17 Gbits/sec    0   6.11 MBytes
[ 19]  85.00-86.00  sec   260 MBytes  2.18 Gbits/sec    0   6.17 MBytes
[ 19]  86.00-87.00  sec   255 MBytes  2.14 Gbits/sec    0   6.22 MBytes
[ 19]  87.00-88.00  sec   259 MBytes  2.17 Gbits/sec    0   6.28 MBytes
[ 19]  88.00-89.00  sec   256 MBytes  2.15 Gbits/sec    0   6.34 MBytes
[ 19]  89.00-90.00  sec   258 MBytes  2.16 Gbits/sec    0   6.39 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[ 19]   0.00-90.00  sec  28.7 GBytes  2.74 Gbits/sec  1982             sender
[ 19]   0.00-90.05  sec  28.7 GBytes  2.74 Gbits/sec                  receiver

iperf Done.
root@admin-pc:~#
```

Figure 52. Running iPerf3 client on host h1.

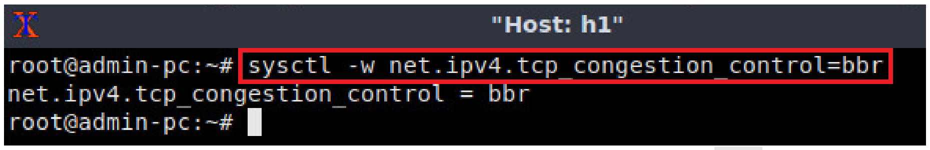
The figure above shows the iPerf3 test output report by the last 20 seconds. The average achieved throughput is 2.74 Gbps (sender) and 2.74 Gbps (receiver), and the number of retransmissions is 1982. Note that the congestion avoidances algorithm used in host h1 is *reno* and in host h2 is *cubic*. Host h3's results are similar to the figure above, however we are just focused on host h1's results.

Step 7. In order to stop the server, press `Ctrl+c` in host h2's and host h4's terminals. The user can see the throughput results in the server side too.

4.4 TCP BBR

Step 1. In host h1's terminal, change the TCP congestion control algorithm to BBR by typing the following command:

```
sysctl -w net.ipv4.tcp_congestion_control=bbr
```



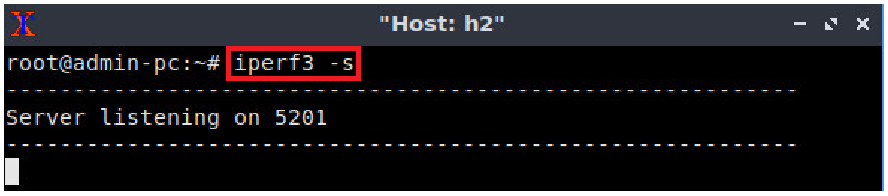
```
root@admin-pc:~# sysctl -w net.ipv4.tcp_congestion_control=bbr
net.ipv4.tcp_congestion_control = bbr
root@admin-pc:~#
```

Figure 53. Changing TCP congestion control algorithm to `bbr` in host h1.

Note that host h3's congestion control algorithm is cubic by default.

Step 2. Launch iPerf3 in server mode on host h2's terminal.

```
iperf3 -s
```

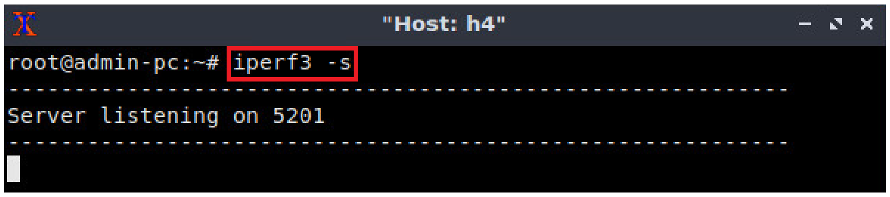


```
root@admin-pc:~# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Figure 54. Starting iPerf3 server on host h2.

Step 3. Launch iPerf3 in server mode on host h4's terminal.

```
iperf3 -s
```



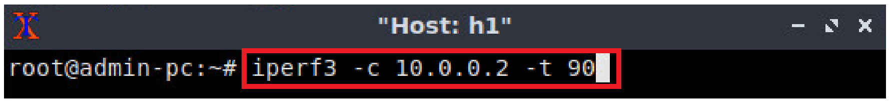
```
root@admin-pc:~# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Figure 55. Starting iPerf3server on host h4.

The following two steps should be executed almost simultaneously thus, you will type the commands displayed in Step 3 and Step 4 then, in Step 5 you will execute them.

Step 4. Type the following iPerf3 command in host h1's terminal without executing it.

```
iperf3 -c 10.0.0.2 -t 90
```

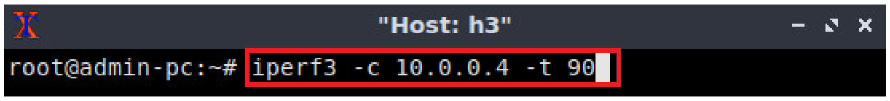


```
"Host: h1"
root@admin-pc:~# iperf3 -c 10.0.0.2 -t 90
```

Figure 56. Typing iPerf3 client command on host h1.

Step 5. Type the following iPerf3 command in host h3's terminal without executing it.

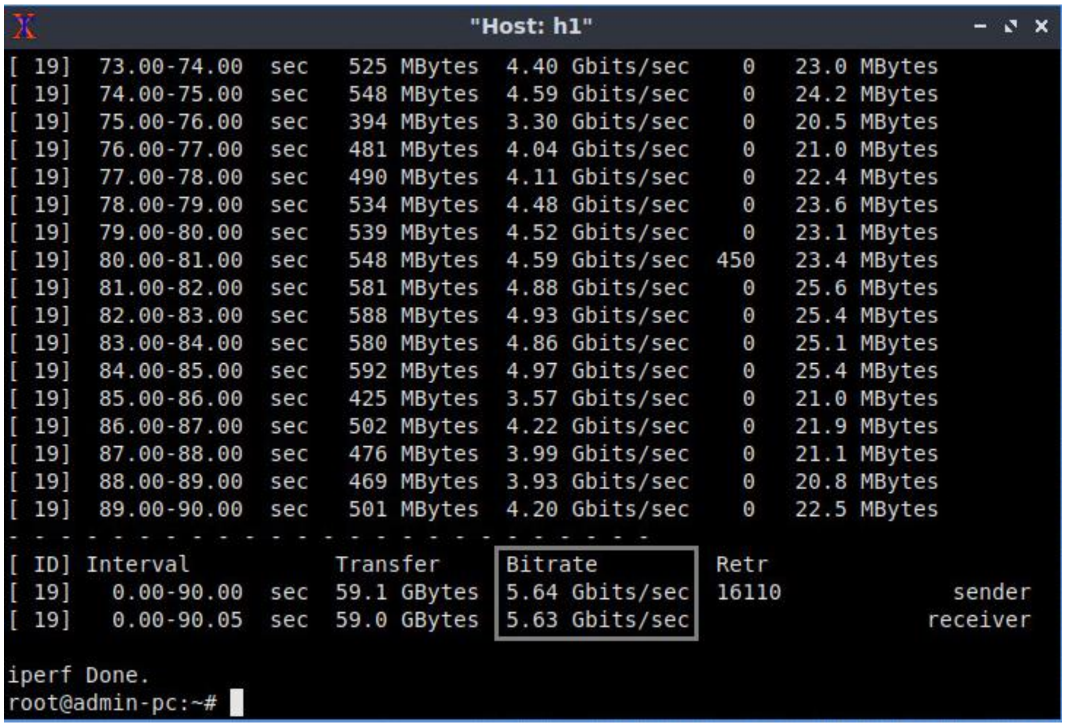
```
iperf3 -c 10.0.0.2 -t 90
```



```
"Host: h3"
root@admin-pc:~# iperf3 -c 10.0.0.4 -t 90
```

Figure 57. Typing iPerf3 client command on host h3.

Step 6. Press *Enter* to execute the commands, first in host h1 terminal then, in host h3 terminal.



```
"Host: h1"
[ 19]  73.00-74.00  sec  525 MBytes  4.40 Gbits/sec    0   23.0 MBytes
[ 19]  74.00-75.00  sec  548 MBytes  4.59 Gbits/sec    0   24.2 MBytes
[ 19]  75.00-76.00  sec  394 MBytes  3.30 Gbits/sec    0   20.5 MBytes
[ 19]  76.00-77.00  sec  481 MBytes  4.04 Gbits/sec    0   21.0 MBytes
[ 19]  77.00-78.00  sec  490 MBytes  4.11 Gbits/sec    0   22.4 MBytes
[ 19]  78.00-79.00  sec  534 MBytes  4.48 Gbits/sec    0   23.6 MBytes
[ 19]  79.00-80.00  sec  539 MBytes  4.52 Gbits/sec    0   23.1 MBytes
[ 19]  80.00-81.00  sec  548 MBytes  4.59 Gbits/sec  450   23.4 MBytes
[ 19]  81.00-82.00  sec  581 MBytes  4.88 Gbits/sec    0   25.6 MBytes
[ 19]  82.00-83.00  sec  588 MBytes  4.93 Gbits/sec    0   25.4 MBytes
[ 19]  83.00-84.00  sec  580 MBytes  4.86 Gbits/sec    0   25.1 MBytes
[ 19]  84.00-85.00  sec  592 MBytes  4.97 Gbits/sec    0   25.4 MBytes
[ 19]  85.00-86.00  sec  425 MBytes  3.57 Gbits/sec    0   21.0 MBytes
[ 19]  86.00-87.00  sec  502 MBytes  4.22 Gbits/sec    0   21.9 MBytes
[ 19]  87.00-88.00  sec  476 MBytes  3.99 Gbits/sec    0   21.1 MBytes
[ 19]  88.00-89.00  sec  469 MBytes  3.93 Gbits/sec    0   20.8 MBytes
[ 19]  89.00-90.00  sec  501 MBytes  4.20 Gbits/sec    0   22.5 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[ 19]   0.00-90.00  sec  59.1 GBytes  5.64 Gbits/sec  16110             sender
[ 19]   0.00-90.05  sec  59.0 GBytes  5.63 Gbits/sec                  receiver

iperf Done.
root@admin-pc:~#
```

Figure 58. Running iPerf3 client on host h1.

The figure above shows the iPerf3 test output report by the last 20 seconds. The average achieved throughput is 5.64 Gbps (sender) and 5.63 Gbps (receiver), and the number of retransmissions is 16,110. Note that the congestion avoidances algorithm used in host h1 is *bbr* and in host h3 is *cubic*. Host h3's results are similar to the figure above, however we are just focused on host h1's results.

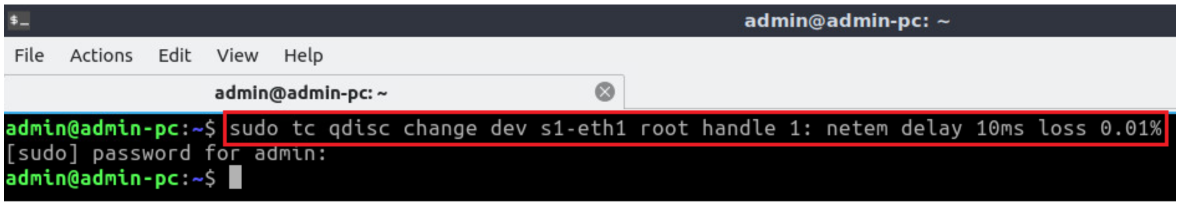
Step 7. In order to stop the server, press `Ctrl+c` in host h2's and host h4's terminals. The user can see the throughput results in the server side too.

5 Emulating high-latency WAN with packet loss

This section emulates a high-latency WAN with packet loss. We already have set a 20ms RTT on the switches. Now, you will add 0.01% packet loss on the switch S1. Note that the switch S1's buffer size is set to one BDP.

Step 1. In the terminal, type the command below. When prompted for a password, type `password` and hit *Enter*. This command introduces 0.01% packet loss on switch S1's *s1-eth1* interface.

```
sudo tc qdisc change dev s1-eth1 root handle 1: netem delay 10ms loss 0.01%
```



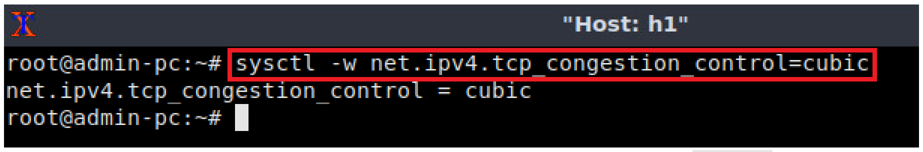
```
admin@admin-pc:~$ sudo tc qdisc change dev s1-eth1 root handle 1: netem delay 10ms loss 0.01%
[sudo] password for admin:
admin@admin-pc:~$
```

Figure 59. Adding delay of 0.01% to switch S1's *s1-eth1* interface.

5.1 TCP Cubic

Step 1. In host h1's terminal, change the TCP congestion control algorithm to Cubic by typing the following command:

```
sysctl -w net.ipv4.tcp_congestion_control=cubic
```



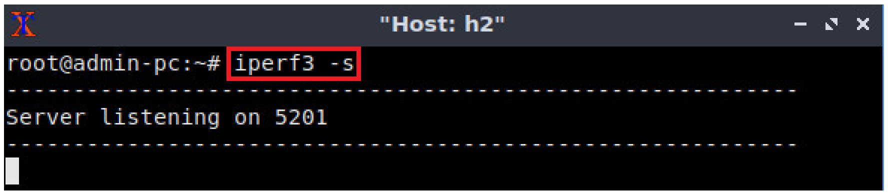
```
root@admin-pc:~# sysctl -w net.ipv4.tcp_congestion_control=cubic
net.ipv4.tcp_congestion_control = cubic
root@admin-pc:~#
```

Figure 60. Changing TCP congestion control algorithm to `cubic` in host h1.

Note that host h3's congestion control algorithm is Cubic by default.

Step 2. Launch iPerf3 in server mode on host h2's terminal.

```
iperf3 -s
```

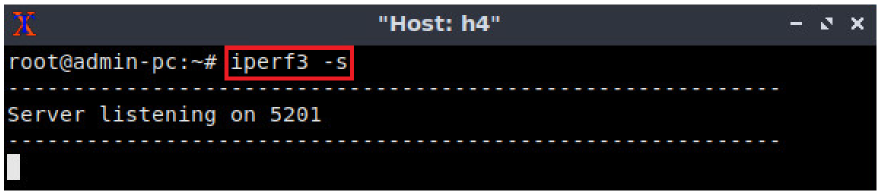


```
root@admin-pc:~# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Figure 61. Starting iPerf3 server on host h2.

Step 3. Launch iPerf3 in server mode on host h4's terminal.

```
iperf3 -s
```



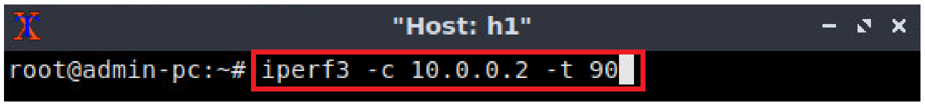
```
root@admin-pc:~# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Figure 62. Starting iPerf3 server on host h4.

The following two steps should be executed almost simultaneously thus, you will type the commands displayed in Step 3 and Step 4 then, in Step 5 you will execute them.

Step 4. Type the following iPerf3 command in host h1's terminal without executing it.

```
iperf3 -c 10.0.0.2 -t 90
```

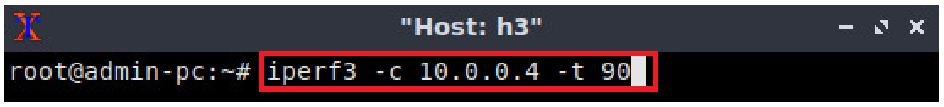


```
root@admin-pc:~# iperf3 -c 10.0.0.2 -t 90
```

Figure 63. Typing iPerf3 client command on host h1.

Step 5. Type the following iPerf3 command in host h3's terminal without executing it.

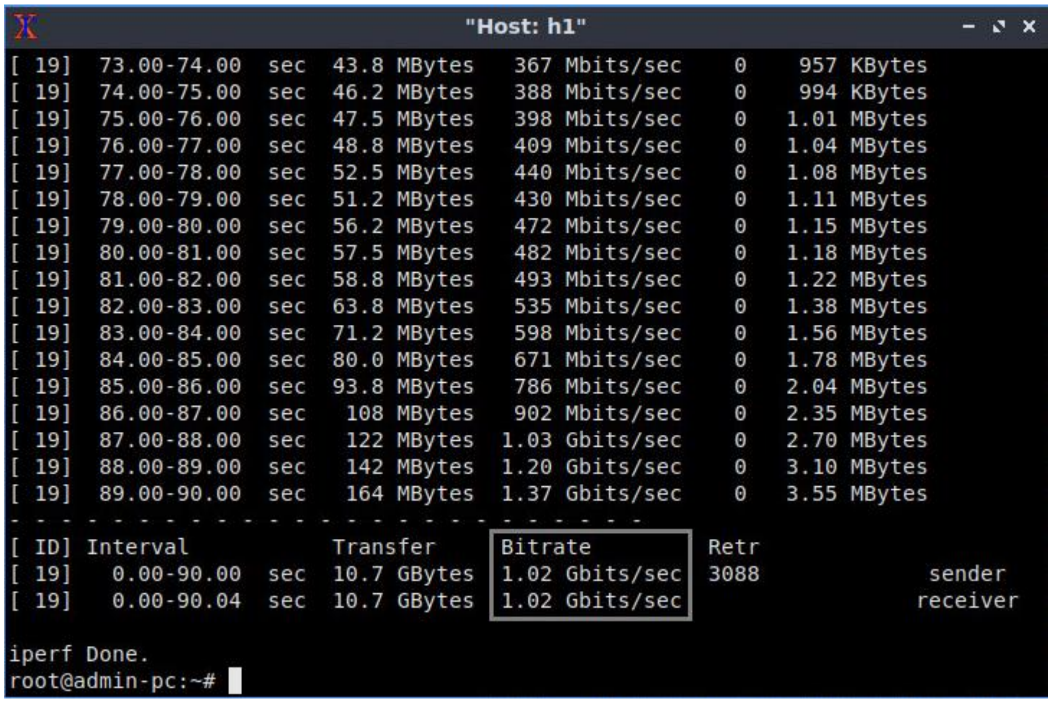
```
iperf3 -c 10.0.0.2 -t 90
```



```
root@admin-pc:~# iperf3 -c 10.0.0.4 -t 90
```

Figure 64. Typing iPerf3 client command on host h3.

Step 6. Press *Enter* to execute the commands, first in host h1 terminal then, in host h3 terminal.



```
"Host: h1"
[ 19]  73.00-74.00  sec  43.8 MBytes   367 Mbits/sec    0    957 KBytes
[ 19]  74.00-75.00  sec  46.2 MBytes   388 Mbits/sec    0    994 KBytes
[ 19]  75.00-76.00  sec  47.5 MBytes   398 Mbits/sec    0   1.01 MBytes
[ 19]  76.00-77.00  sec  48.8 MBytes   409 Mbits/sec    0   1.04 MBytes
[ 19]  77.00-78.00  sec  52.5 MBytes   440 Mbits/sec    0   1.08 MBytes
[ 19]  78.00-79.00  sec  51.2 MBytes   430 Mbits/sec    0   1.11 MBytes
[ 19]  79.00-80.00  sec  56.2 MBytes   472 Mbits/sec    0   1.15 MBytes
[ 19]  80.00-81.00  sec  57.5 MBytes   482 Mbits/sec    0   1.18 MBytes
[ 19]  81.00-82.00  sec  58.8 MBytes   493 Mbits/sec    0   1.22 MBytes
[ 19]  82.00-83.00  sec  63.8 MBytes   535 Mbits/sec    0   1.38 MBytes
[ 19]  83.00-84.00  sec  71.2 MBytes   598 Mbits/sec    0   1.56 MBytes
[ 19]  84.00-85.00  sec  80.0 MBytes   671 Mbits/sec    0   1.78 MBytes
[ 19]  85.00-86.00  sec  93.8 MBytes   786 Mbits/sec    0   2.04 MBytes
[ 19]  86.00-87.00  sec   108 MBytes   902 Mbits/sec    0   2.35 MBytes
[ 19]  87.00-88.00  sec   122 MBytes  1.03 Gbits/sec    0   2.70 MBytes
[ 19]  88.00-89.00  sec   142 MBytes  1.20 Gbits/sec    0   3.10 MBytes
[ 19]  89.00-90.00  sec   164 MBytes  1.37 Gbits/sec    0   3.55 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[ 19]   0.00-90.00  sec  10.7 GBytes  1.02 Gbits/sec  3088             sender
[ 19]   0.00-90.04  sec  10.7 GBytes  1.02 Gbits/sec                  receiver

iperf Done.
root@admin-pc:~#
```

Figure 65. Running iPerf3 client on host h1.

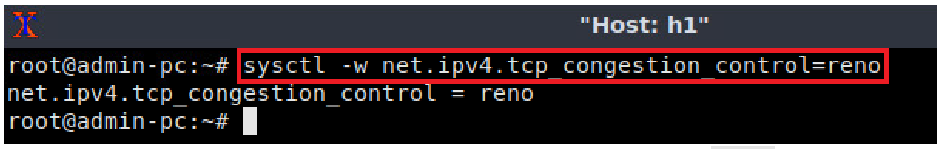
The figure above shows the iPerf3 test output report by the last 20 seconds. The average achieved throughput is 1.02 Gbps (sender) and 1.02 Gbps (receiver), and the number of retransmissions is 3088. Note that the congestion control algorithm used in host h1 and host h2 is *cubic*. Host h3’s results are similar to the figure above, however we are just focused on host h1’s results.

Step 7. In order to stop the server, press `Ctrl+c` in host h2’s and host h4’s terminals. The user can see the throughput results in the server side too.

5.2 TCP Reno

Step 1. In host h1’s terminal, change the TCP congestion control algorithm to Reno by typing the following command:

```
sysctl -w net.ipv4.tcp_congestion_control=reno
```



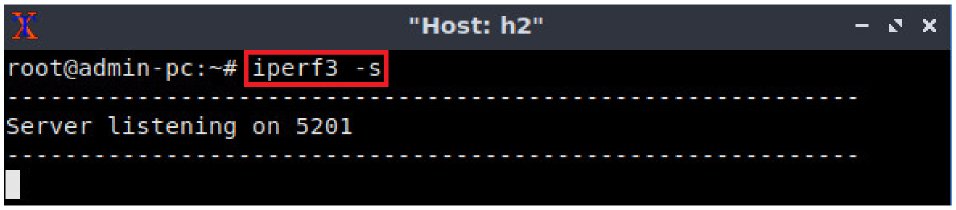
```
"Host: h1"
root@admin-pc:~# sysctl -w net.ipv4.tcp_congestion_control=reno
net.ipv4.tcp_congestion_control = reno
root@admin-pc:~#
```

Figure 66. Changing TCP congestion control algorithm to `reno` in host h1.

Note that host h3’s congestion control algorithm is cubic by default.

Step 2. Launch iPerf3 in server mode on host h2’s terminal.

```
iperf3 -s
```

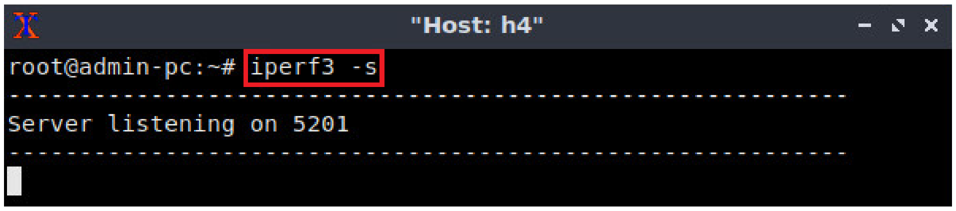


```
"Host: h2"
root@admin-pc:~# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Figure 67. Starting iPerf3 server on host h2.

Step 3. Launch iPerf3 in server mode on host h4's terminal.

```
iperf3 -s
```



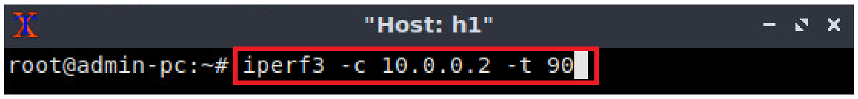
```
"Host: h4"
root@admin-pc:~# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Figure 68. Starting iPerf3 server on host h4.

The following two steps should be executed almost simultaneously, thus you will type the commands displayed in Step 3 and Step 4, then in Step 5 you will execute them.

Step 4. Type the following iPerf3 command in host h1's terminal without executing it.

```
iperf3 -c 10.0.0.2 -t 90
```

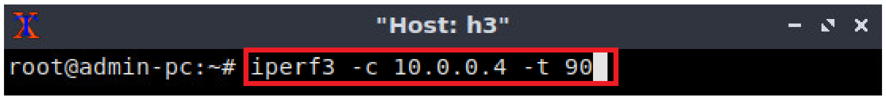


```
"Host: h1"
root@admin-pc:~# iperf3 -c 10.0.0.2 -t 90
```

Figure 69. Typing iPerf3 client command on host h1.

Step 5. Type the following iPerf3 command in host h3's terminal without executing it.

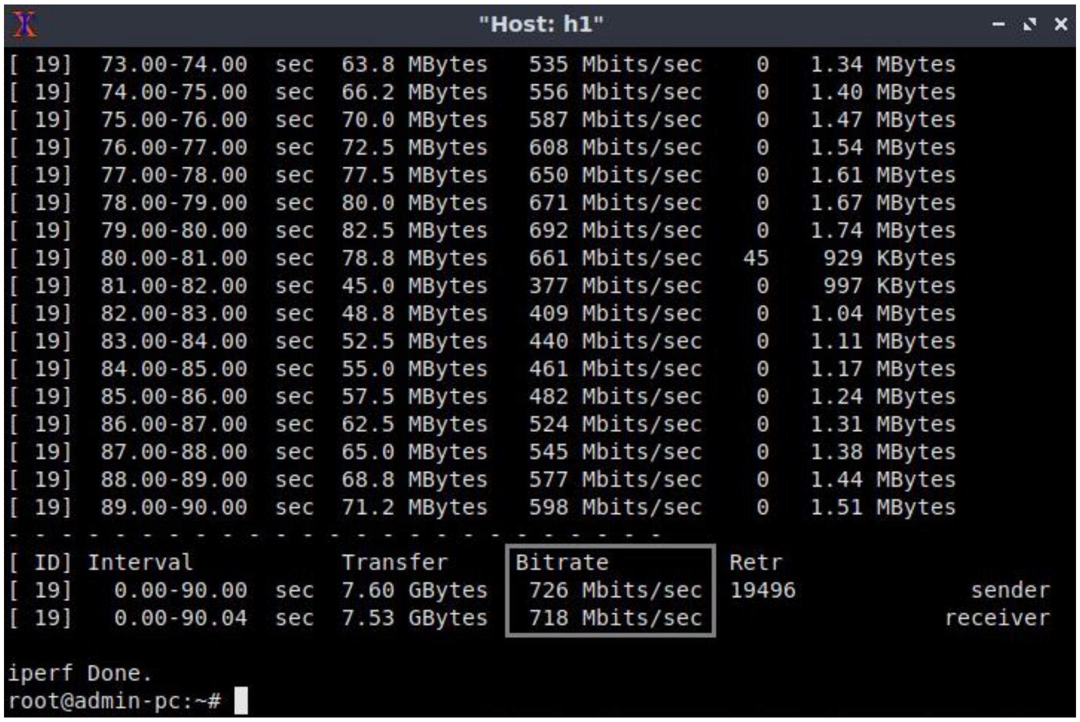
```
iperf3 -c 10.0.0.2 -t 90
```



```
"Host: h3"
root@admin-pc:~# iperf3 -c 10.0.0.4 -t 90
```

Figure 70. Typing iPerf3 client command on host h3.

Step 6. Press *Enter* to execute the commands, first in host h1 terminal then, in host h3 terminal.



```
[ 19]  73.00-74.00  sec  63.8 MBytes   535 Mbits/sec    0   1.34 MBytes
[ 19]  74.00-75.00  sec  66.2 MBytes   556 Mbits/sec    0   1.40 MBytes
[ 19]  75.00-76.00  sec  70.0 MBytes   587 Mbits/sec    0   1.47 MBytes
[ 19]  76.00-77.00  sec  72.5 MBytes   608 Mbits/sec    0   1.54 MBytes
[ 19]  77.00-78.00  sec  77.5 MBytes   650 Mbits/sec    0   1.61 MBytes
[ 19]  78.00-79.00  sec  80.0 MBytes   671 Mbits/sec    0   1.67 MBytes
[ 19]  79.00-80.00  sec  82.5 MBytes   692 Mbits/sec    0   1.74 MBytes
[ 19]  80.00-81.00  sec  78.8 MBytes   661 Mbits/sec   45    929 KBytes
[ 19]  81.00-82.00  sec  45.0 MBytes   377 Mbits/sec    0    997 KBytes
[ 19]  82.00-83.00  sec  48.8 MBytes   409 Mbits/sec    0   1.04 MBytes
[ 19]  83.00-84.00  sec  52.5 MBytes   440 Mbits/sec    0   1.11 MBytes
[ 19]  84.00-85.00  sec  55.0 MBytes   461 Mbits/sec    0   1.17 MBytes
[ 19]  85.00-86.00  sec  57.5 MBytes   482 Mbits/sec    0   1.24 MBytes
[ 19]  86.00-87.00  sec  62.5 MBytes   524 Mbits/sec    0   1.31 MBytes
[ 19]  87.00-88.00  sec  65.0 MBytes   545 Mbits/sec    0   1.38 MBytes
[ 19]  88.00-89.00  sec  68.8 MBytes   577 Mbits/sec    0   1.44 MBytes
[ 19]  89.00-90.00  sec  71.2 MBytes   598 Mbits/sec    0   1.51 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[ 19]   0.00-90.00  sec  7.60 GBytes   726 Mbits/sec  19496             sender
[ 19]   0.00-90.04  sec  7.53 GBytes   718 Mbits/sec                  receiver

iperf Done.
root@admin-pc:~#
```

Figure 71. Running iPerf3 client on host h1.

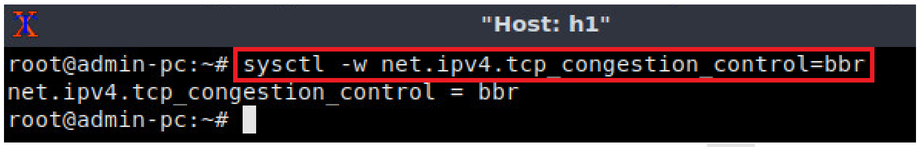
The figure above shows the iPerf3 test output report by the last 20 seconds. The average achieved throughput is 726 Mbps (sender) and 718 Mbps (receiver), and the number of retransmissions is 19,496. Note that the congestion control algorithm used in host h1 is *reno* and in host h2 is *cubic*. Host h3's results are similar to the figure above, however we are just focused on host h1's results.

Step 7. In order to stop the server, press `Ctrl+c` in host h2's and host h4's terminals. The user can see the throughput results in the server side too.

5.3 TCP BBR

Step 1. In host h1's terminal, change the TCP congestion control algorithm to BBR by typing the following command:

```
sysctl -w net.ipv4.tcp_congestion_control=bbr
```



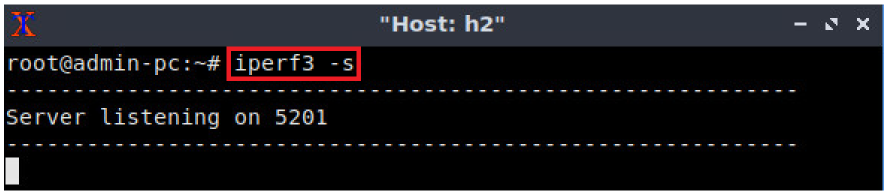
```
root@admin-pc:~# sysctl -w net.ipv4.tcp_congestion_control=bbr
net.ipv4.tcp_congestion_control = bbr
root@admin-pc:~#
```

Figure 72. Changing TCP congestion control algorithm to `bbr` in host h1.

Note that host h3's congestion control algorithm is cubic by default.

Step 2. Launch iPerf3 in server mode on host h2's terminal.

```
iperf3 -s
```

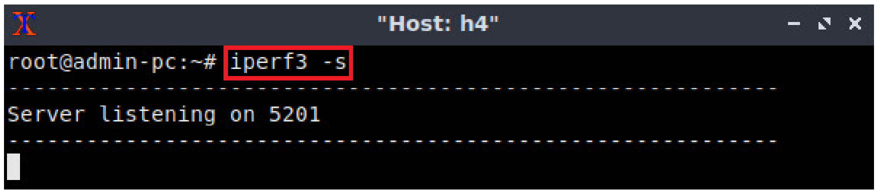


```
"Host: h2"
root@admin-pc:~# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Figure 73. Starting iPerf3 server on host h2.

Step 3. Launch iPerf3 in server mode on host h4's terminal.

```
iperf3 -s
```



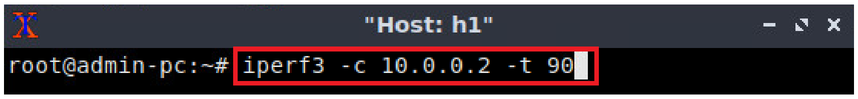
```
"Host: h4"
root@admin-pc:~# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Figure 74. Starting iPerf3 server on host h4.

The following two steps should be executed almost simultaneously, thus you will type the commands displayed in Step 3 and Step 4, then in Step 5 you will execute them.

Step 4. Type the following iPerf3 command in host h1's terminal without executing it.

```
iperf3 -c 10.0.0.2 -t 90
```

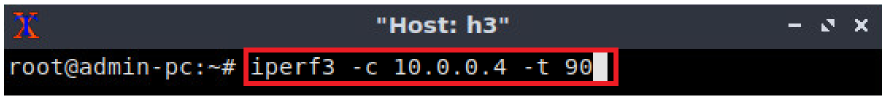


```
"Host: h1"
root@admin-pc:~# iperf3 -c 10.0.0.2 -t 90
```

Figure 75. Typing iPerf3 client command on host h1.

Step 5. Type the following iPerf3 command in host h3's terminal without executing it.

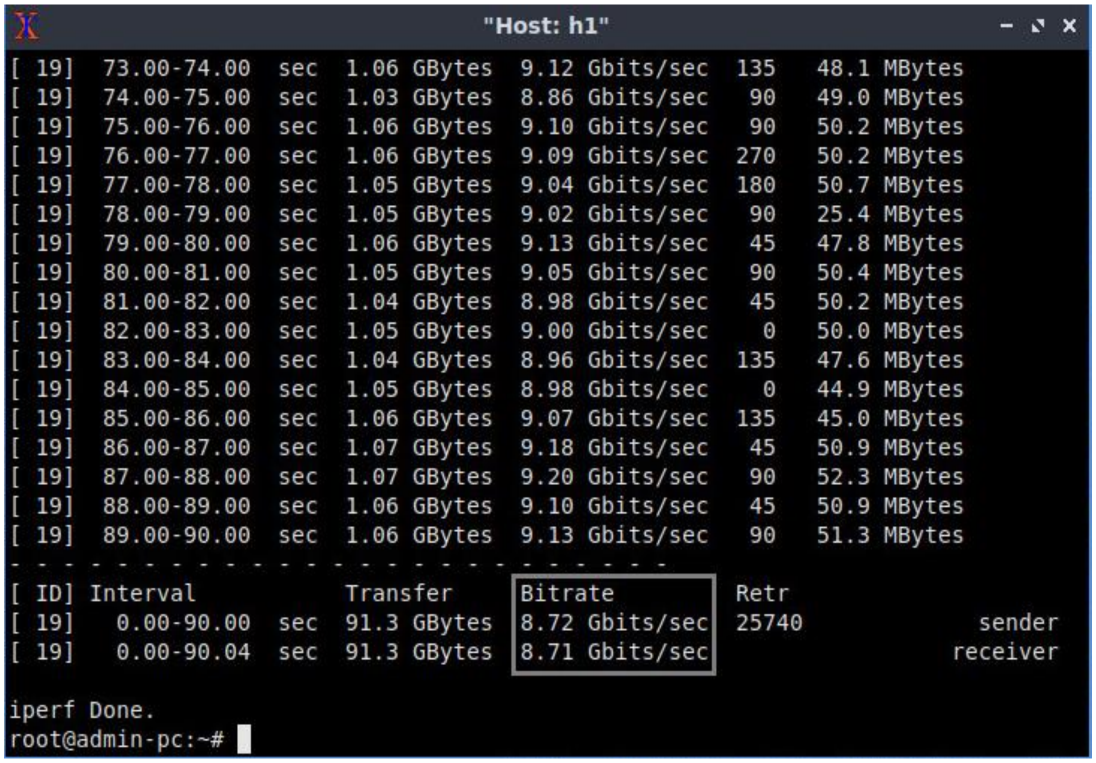
```
iperf3 -c 10.0.0.2 -t 90
```



```
"Host: h3"
root@admin-pc:~# iperf3 -c 10.0.0.4 -t 90
```

Figure 76. Typing iPerf3 client command on host h3.

Step 6. Press *Enter* to execute the commands, first in host h1 terminal then, in host h3 terminal.



```
[ 19]  73.00-74.00  sec  1.06 GBytes  9.12 Gbits/sec  135   48.1 MBytes
[ 19]  74.00-75.00  sec  1.03 GBytes  8.86 Gbits/sec   90   49.0 MBytes
[ 19]  75.00-76.00  sec  1.06 GBytes  9.10 Gbits/sec   90   50.2 MBytes
[ 19]  76.00-77.00  sec  1.06 GBytes  9.09 Gbits/sec  270   50.2 MBytes
[ 19]  77.00-78.00  sec  1.05 GBytes  9.04 Gbits/sec  180   50.7 MBytes
[ 19]  78.00-79.00  sec  1.05 GBytes  9.02 Gbits/sec   90   25.4 MBytes
[ 19]  79.00-80.00  sec  1.06 GBytes  9.13 Gbits/sec   45   47.8 MBytes
[ 19]  80.00-81.00  sec  1.05 GBytes  9.05 Gbits/sec   90   50.4 MBytes
[ 19]  81.00-82.00  sec  1.04 GBytes  8.98 Gbits/sec   45   50.2 MBytes
[ 19]  82.00-83.00  sec  1.05 GBytes  9.00 Gbits/sec    0   50.0 MBytes
[ 19]  83.00-84.00  sec  1.04 GBytes  8.96 Gbits/sec  135   47.6 MBytes
[ 19]  84.00-85.00  sec  1.05 GBytes  8.98 Gbits/sec    0   44.9 MBytes
[ 19]  85.00-86.00  sec  1.06 GBytes  9.07 Gbits/sec  135   45.0 MBytes
[ 19]  86.00-87.00  sec  1.07 GBytes  9.18 Gbits/sec   45   50.9 MBytes
[ 19]  87.00-88.00  sec  1.07 GBytes  9.20 Gbits/sec   90   52.3 MBytes
[ 19]  88.00-89.00  sec  1.06 GBytes  9.10 Gbits/sec   45   50.9 MBytes
[ 19]  89.00-90.00  sec  1.06 GBytes  9.13 Gbits/sec   90   51.3 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[ 19]   0.00-90.00  sec  91.3 GBytes  8.72 Gbits/sec  25740             sender
[ 19]   0.00-90.04  sec  91.3 GBytes  8.71 Gbits/sec                  receiver

iperf Done.
root@admin-pc:~# 
```

Figure 77. Running iPerf3 client on host h1.

The figure above shows the iPerf3 test output report by the last 20 seconds. The average achieved throughput is 8.72 Gbps (sender) and 8.71 Gbps (receiver), and the number of retransmissions is 25,740. Note that the congestion avoidances algorithm used in host h1 is *bbr* and in host h3 is *cubic*.

Step 7. In order to stop the server, press `Ctrl+c` in host h2's and host h4's terminals. The user can see the throughput results in the server side too.

This concludes Lab 11. Stop the emulation and then exit out of MiniEdit.

References

1. J. Moy, "Open shortest path first (OSPF) Version 2," Internet Request for Comments, RFC Editor, RFC 2328, Apr. 1998. [Online]. Available: https://www.ietf.org/rfc/rfc2328.txt.
2. Y. Rekhter, T. Li, S. Hares, "Border gateway protocol 4," Internet Request for Comments, RFC Editor, RFC 4271, Jan. 2006. [Online]. Available: https://tools.ietf.org/html/rfc4271.
3. J. Crichigno, E. Bou-Harb, N. Ghani, "A comprehensive tutorial on Science DMZ," IEEE Communications Surveys and Tutorials, 2019.
4. N. Cardwell, Y. Cheng, C. Gunn, S. Yeganeh, V. Jacobson, "BBR: congestion-based congestion control," Communications of the ACM, vol 60, no. 2, pp. 58-66, Feb. 2017.
5. J. Kurose, K. Ross, "Computer networking: a top-down approach," 7th Edition, Pearson, 2017.

6. C. Villamizar, C. Song, "High performance TCP in ansnet," ACM Computer Communications Review, vol. 24, no. 5, pp. 45-60, Oct. 1994.
7. R. Bush, D. Meyer, "Some internet architectural guidelines and philosophy," Internet Request for Comments, RFC Editor, RFC 3439, Dec. 2003. [Online]. Available: https://www.ietf.org/rfc/rfc3439.txt.
8. G. Appenzeller, I. Keslassy, N. McKeown, "Sizing router buffers," in Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 281-292, Oct. 2004.