

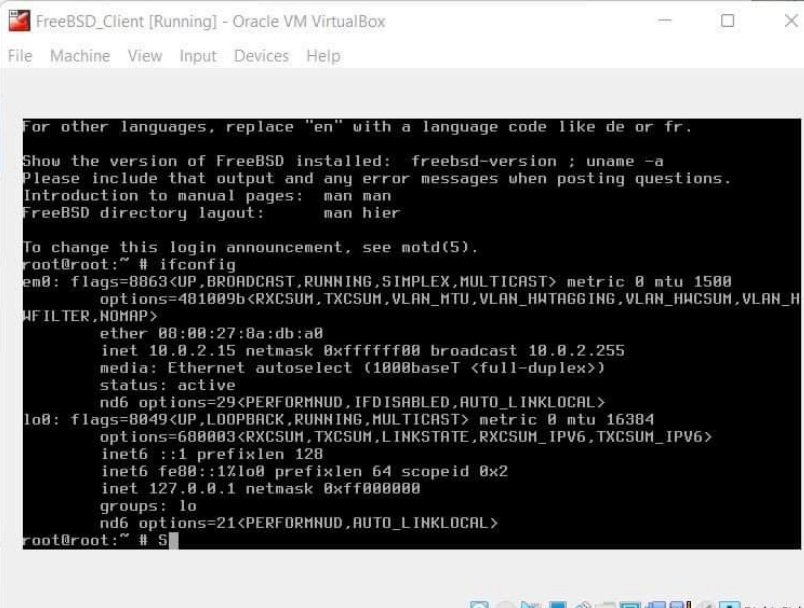
Introduction to Computer Networks - Programming Assignment 2

Anastasia Riana 邱慧莉 - 109006234

Part 1. Environment Setup

We can see from these figures that the FreeBSD client and server has been successfully installed. By using the **ifconfig** command, we are able to check both the server and client's IP address, which is 10.0.2.15 and 10.0.2.5 for client and server respectively.

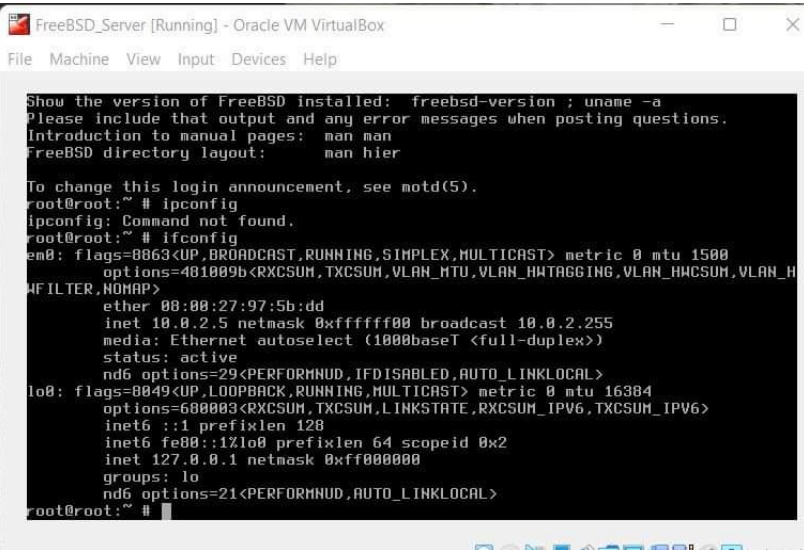
Result:



```
FreeBSD_Client [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

For other languages, replace "en" with a language code like de or fr.
Show the version of FreeBSD installed: freebsd-version ; uname -a
Please include that output and any error messages when posting questions.
Introduction to manual pages: man man
FreeBSD directory layout: man hier

To change this login announcement, see motd(5).
root@root:~ # ifconfig
em0: flags=8063<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    options=401009b<RXCSUM, TXCSUM, VLAN_MTU, VLAN_HWTAGGING, VLAN_HWCSUM, VLAN_H
    AFILTER, NMAP>
    ether 08:00:27:8a:db:a0
    inet 10.0.2.15 netmask 0xfffff000 broadcast 10.0.2.255
    media: Ethernet autoselect (1000baseT <full-duplex>)
    status: active
nd6 options=29<PERFORMNUD, IFDISABLED, AUTO_LINKLOCAL>
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> metric 0 mtu 16384
    options=600003<RXCSUM, TXCSUM, LINKSTATE, RXCSUM_IPV6, TXCSUM_IPV6>
    inet6 ::1 prefixlen 128
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x2
    inet 127.0.0.1 netmask 0xff000000
    groups: lo
    nd6 options=21<PERFORMNUD, AUTO_LINKLOCAL>
root@root:~ # S
```



```
FreeBSD_Server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

Show the version of FreeBSD installed: freebsd-version ; uname -a
Please include that output and any error messages when posting questions.
Introduction to manual pages: man man
FreeBSD directory layout: man hier

To change this login announcement, see motd(5).
root@root:~ # ipconfig
ipconfig: Command not found.
root@root:~ # ifconfig
em0: flags=8063<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    options=401009b<RXCSUM, TXCSUM, VLAN_MTU, VLAN_HWTAGGING, VLAN_HWCSUM, VLAN_H
    AFILTER, NMAP>
    ether 08:00:27:97:5b:dd
    inet 10.0.2.5 netmask 0xfffff000 broadcast 10.0.2.255
    media: Ethernet autoselect (1000baseT <full-duplex>)
    status: active
nd6 options=29<PERFORMNUD, IFDISABLED, AUTO_LINKLOCAL>
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> metric 0 mtu 16384
    options=600003<RXCSUM, TXCSUM, LINKSTATE, RXCSUM_IPV6, TXCSUM_IPV6>
    inet6 ::1 prefixlen 128
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x2
    inet 127.0.0.1 netmask 0xff000000
    groups: lo
    nd6 options=21<PERFORMNUD, AUTO_LINKLOCAL>
root@root:~ #
```

Part 2. TCP Data Transmission

The command used to run the **iperf3** command is:

```
iperf3 {a} {b} {c}
```

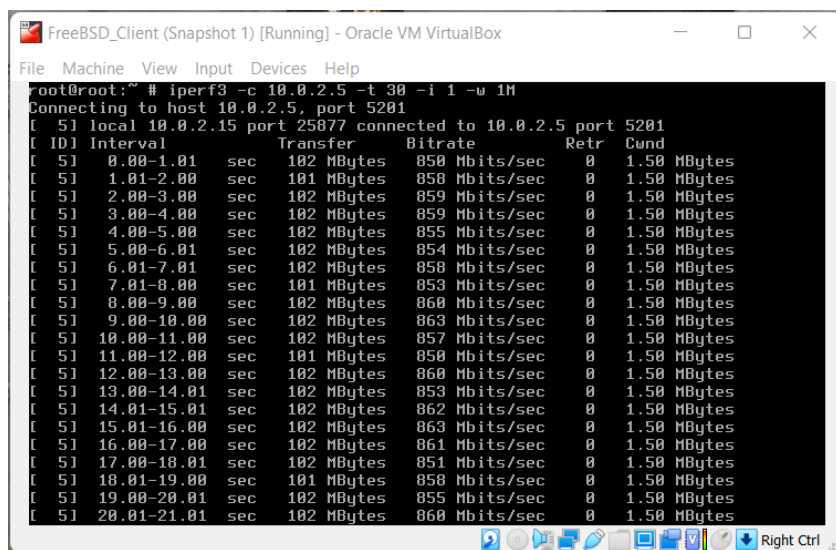
```
iperf3 {-c 10.0.2.5} {-t 30} {-i 1} -w 1M
```

{-c 10.0.2.5} -c 10.0.2.5: indicates that run iperf3 to be run in client mode and connecting to an iperf3 server that is running on host

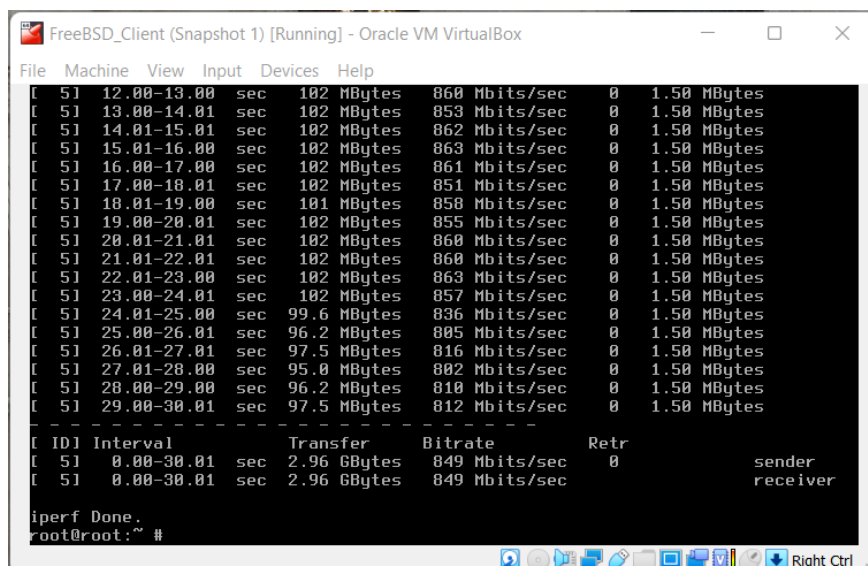
{-t 30}: indicates the time to transmit the data in seconds.

{-i 1} : set the interval time in seconds between the periodic bandwidth, jitter and loss report as requested by the homework question.

Result:



```
FreeBSD_Client (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
root@root:~ # iperf3 -c 10.0.2.5 -t 30 -i 1 -w 1M
Connecting to host 10.0.2.5, port 5201
[ 5] local 10.0.2.15 port 25877 connected to 10.0.2.5 port 5201
[ ID] Interval           Transfer     Bitrate      Retr    Cwnd
[ 5]  0.00-1.01   sec    102 MBytes  850 Mbits/sec    0    1.50 MBytes
[ 5]  1.01-2.00   sec    101 MBytes  858 Mbits/sec    0    1.50 MBytes
[ 5]  2.00-3.00   sec    102 MBytes  859 Mbits/sec    0    1.50 MBytes
[ 5]  3.00-4.00   sec    102 MBytes  859 Mbits/sec    0    1.50 MBytes
[ 5]  4.00-5.00   sec    102 MBytes  855 Mbits/sec    0    1.50 MBytes
[ 5]  5.00-6.01   sec    102 MBytes  854 Mbits/sec    0    1.50 MBytes
[ 5]  6.01-7.01   sec    102 MBytes  858 Mbits/sec    0    1.50 MBytes
[ 5]  7.01-8.00   sec    101 MBytes  853 Mbits/sec    0    1.50 MBytes
[ 5]  8.00-9.00   sec    102 MBytes  860 Mbits/sec    0    1.50 MBytes
[ 5]  9.00-10.00  sec    102 MBytes  863 Mbits/sec    0    1.50 MBytes
[ 5] 10.00-11.00  sec    102 MBytes  857 Mbits/sec    0    1.50 MBytes
[ 5] 11.00-12.00  sec    101 MBytes  850 Mbits/sec    0    1.50 MBytes
[ 5] 12.00-13.00  sec    102 MBytes  860 Mbits/sec    0    1.50 MBytes
[ 5] 13.00-14.01  sec    102 MBytes  853 Mbits/sec    0    1.50 MBytes
[ 5] 14.01-15.01  sec    102 MBytes  862 Mbits/sec    0    1.50 MBytes
[ 5] 15.01-16.00  sec    102 MBytes  863 Mbits/sec    0    1.50 MBytes
[ 5] 16.00-17.00  sec    102 MBytes  861 Mbits/sec    0    1.50 MBytes
[ 5] 17.00-18.01  sec    102 MBytes  851 Mbits/sec    0    1.50 MBytes
[ 5] 18.01-19.00  sec    101 MBytes  858 Mbits/sec    0    1.50 MBytes
[ 5] 19.00-20.01  sec    102 MBytes  855 Mbits/sec    0    1.50 MBytes
[ 5] 20.01-21.01  sec    102 MBytes  860 Mbits/sec    0    1.50 MBytes
```



```
FreeBSD_Client (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
[ 5] 12.00-13.00  sec    102 MBytes  860 Mbits/sec    0    1.50 MBytes
[ 5] 13.00-14.01  sec    102 MBytes  853 Mbits/sec    0    1.50 MBytes
[ 5] 14.01-15.01  sec    102 MBytes  862 Mbits/sec    0    1.50 MBytes
[ 5] 15.01-16.00  sec    102 MBytes  863 Mbits/sec    0    1.50 MBytes
[ 5] 16.00-17.00  sec    102 MBytes  861 Mbits/sec    0    1.50 MBytes
[ 5] 17.00-18.01  sec    102 MBytes  851 Mbits/sec    0    1.50 MBytes
[ 5] 18.01-19.00  sec    101 MBytes  858 Mbits/sec    0    1.50 MBytes
[ 5] 19.00-20.01  sec    102 MBytes  855 Mbits/sec    0    1.50 MBytes
[ 5] 20.01-21.01  sec    102 MBytes  860 Mbits/sec    0    1.50 MBytes
[ 5] 21.01-22.01  sec    102 MBytes  860 Mbits/sec    0    1.50 MBytes
[ 5] 22.01-23.00  sec    102 MBytes  863 Mbits/sec    0    1.50 MBytes
[ 5] 23.00-24.01  sec    102 MBytes  857 Mbits/sec    0    1.50 MBytes
[ 5] 24.01-25.00  sec    99.6 MBytes  836 Mbits/sec    0    1.50 MBytes
[ 5] 25.00-26.01  sec    96.2 MBytes  805 Mbits/sec    0    1.50 MBytes
[ 5] 26.01-27.01  sec    97.5 MBytes  816 Mbits/sec    0    1.50 MBytes
[ 5] 27.01-28.00  sec    95.0 MBytes  802 Mbits/sec    0    1.50 MBytes
[ 5] 28.00-29.00  sec    96.2 MBytes  810 Mbits/sec    0    1.50 MBytes
[ 5] 29.00-30.01  sec    97.5 MBytes  812 Mbits/sec    0    1.50 MBytes
-- -- -- -- --
[ ID] Interval           Transfer     Bitrate      Retr
[ 5]  0.00-30.01  sec    2.96 GBytes  849 Mbits/sec    0
[ 5]  0.00-30.01  sec    2.96 GBytes  849 Mbits/sec    0
sender
receiver

iperf Done.
root@root:~ #
```

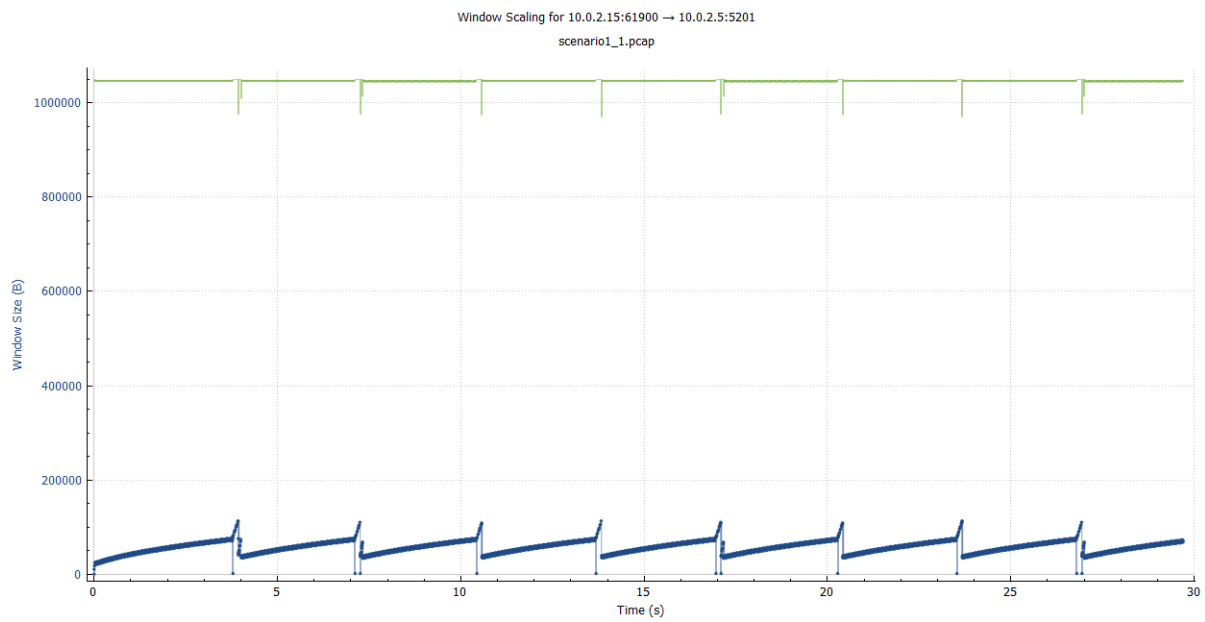
Part 3. TCP Congestion Control

Scenarios

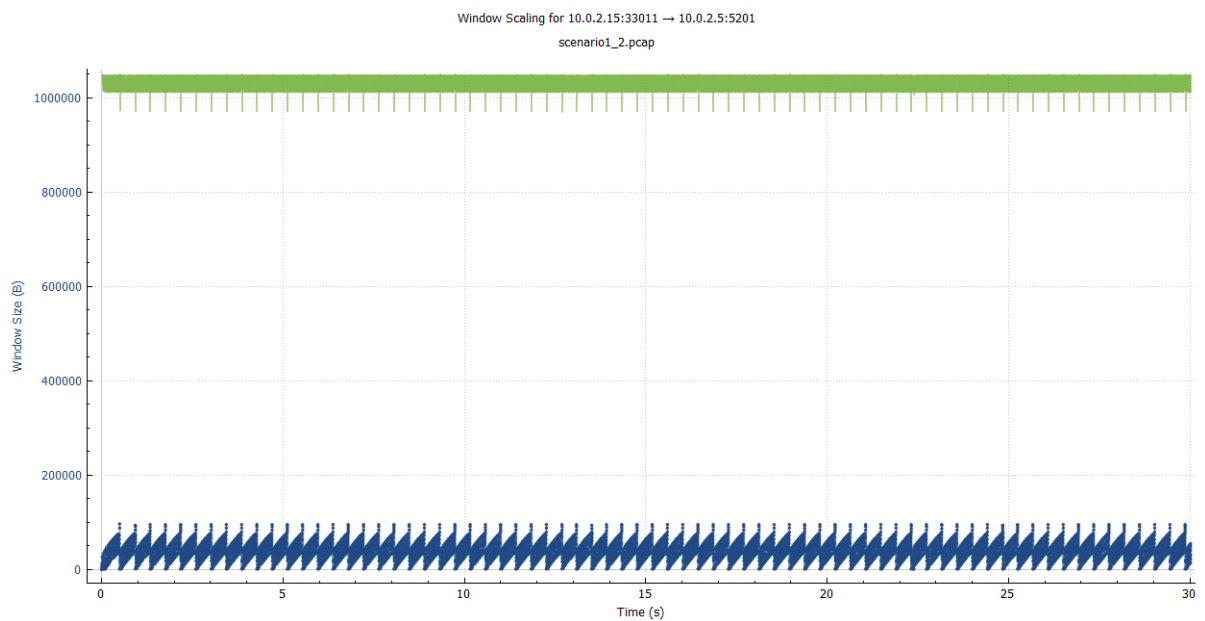
	Bandwidth	Delay	Packet Loss Rate	algorithm
Scenario 1	4 and 32 Mbit/s	default	default	newreno
Scenario 2	8 Mbit/s	10 and 100 ms	default	newreno
Scenario 3	8 Mbit/s	default	0.1 and 1%	newreno
Scenario 4	8 Mbit/s	100 ms	default	newreno and cubic
Scenario 5	8 Mbit/s	default	1%	newreno and cubic

Note: violet - slow start ; blue - congestion avoidance ; maroon - fast recovery

Scenario 1_1: 4Mbit/s Bandwidth



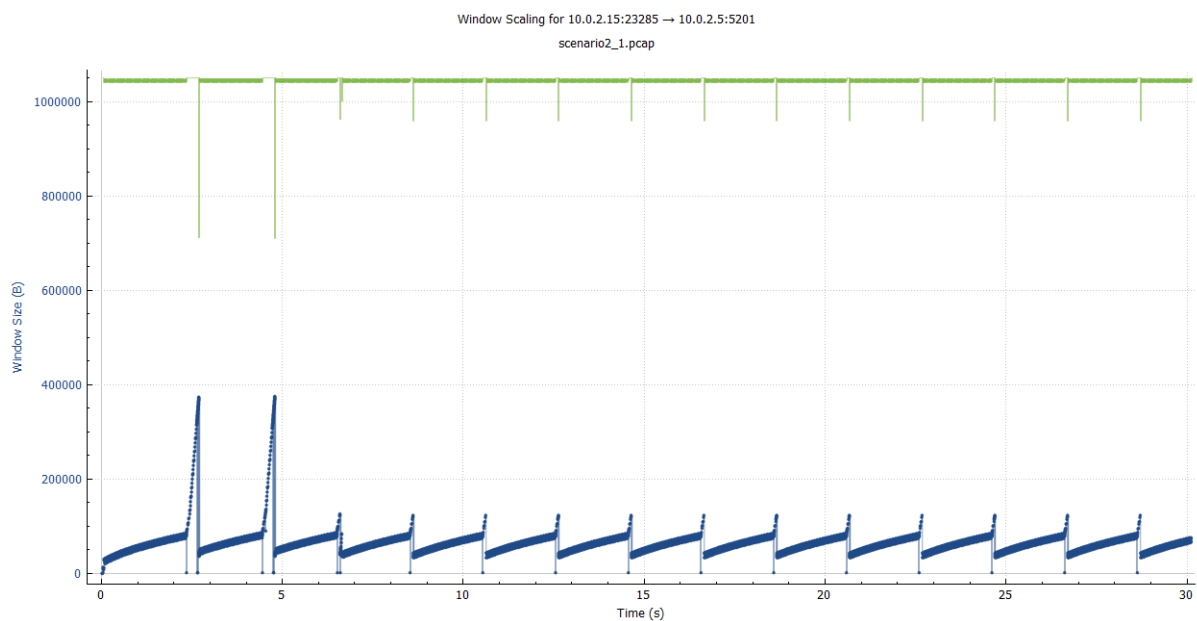
Scenario 1_2: 32Mbit/s Bandwidth



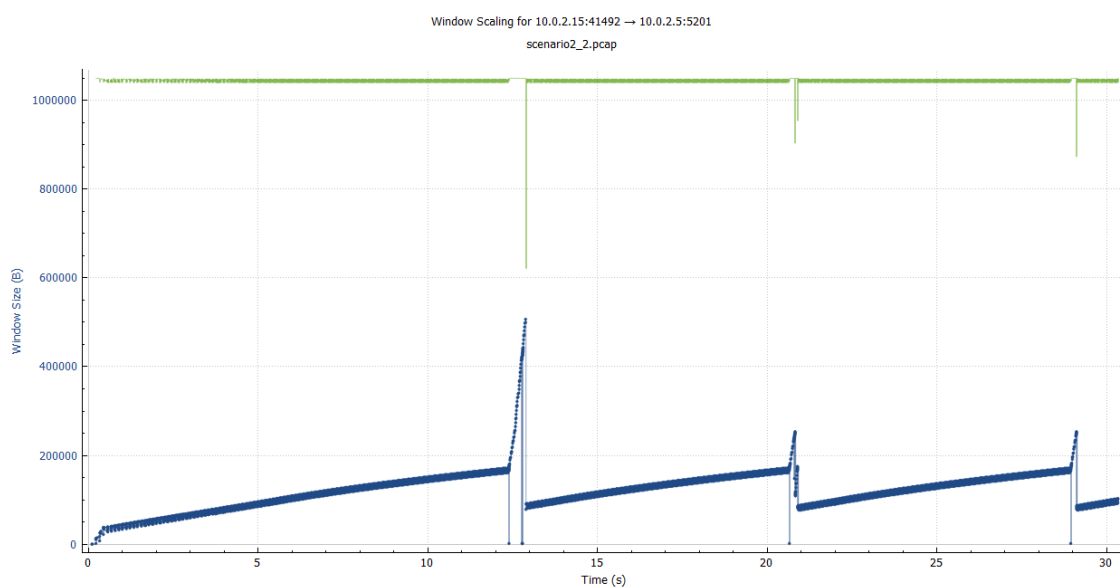
Analysis:

From the first scenario, we can see one slow start in the beginning. We see multiple congestion avoidance, indicated by the rising spike. Lastly, we see multiple fast recovery, indicated by the break of the spike drop. From the second scenario, we can see one slow start in the beginning. We see multiple congestion avoidance, indicated by the rising spike. Lastly, we don't really see any fast recovery. Then, we can conclude that with higher bandwidth, packets are sent more efficiently and can be sent faster from server to client and vice versa.

Scenario 2_1: 8Mbit/s Bandwidth, 10 msec delay



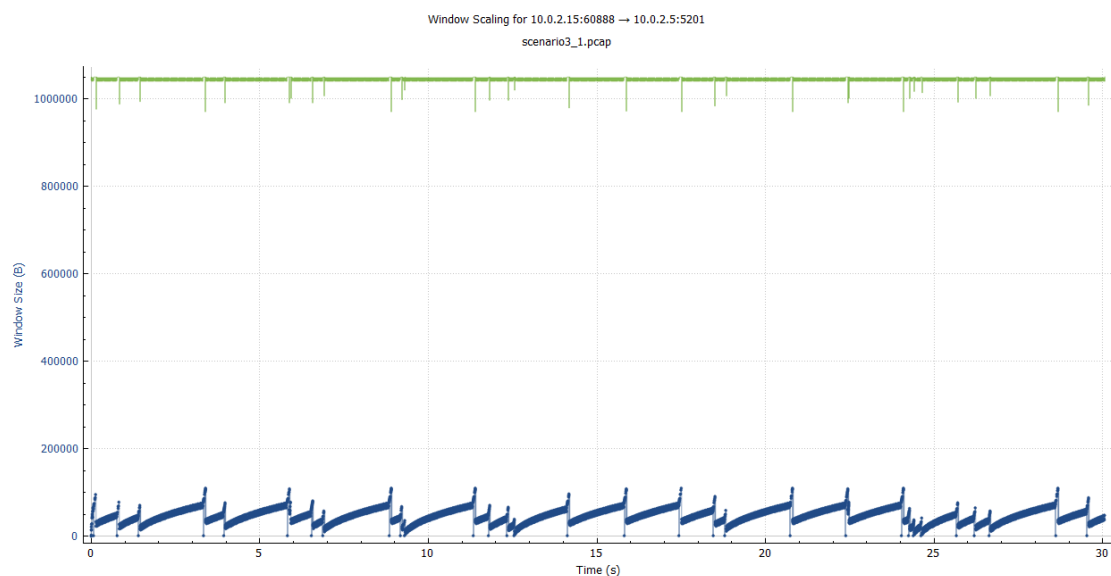
Scenario 2_2: 8Mbit/s Bandwidth, 100 msec delay



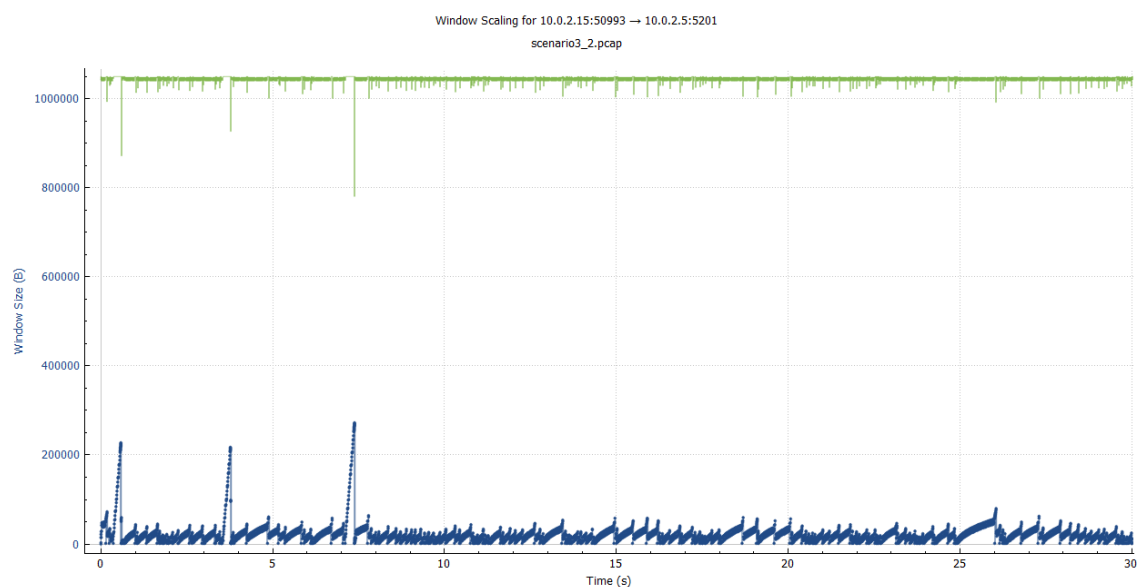
Analysis:

From the first scenario, we can see one slow start in the beginning. We see multiple congestion avoidance, indicated by the rising spike. Lastly, we see multiple fast recovery, indicated by the break of the spike drop. From the second scenario, we can see one slow start in the beginning. We see multiple congestion avoidance, indicated by the rising spike, however it is less than the first scenario. Lastly, we also see many fast recoveries. Then, we can conclude that the greater the delay, the sender must spend more time being idle which reduces how fast throughput grows. Hence, the smaller delay is preferred, which is the first scenario.

Scenario 3_1: 8Mbit/s Bandwidth, 0.1% Packet Loss Rate



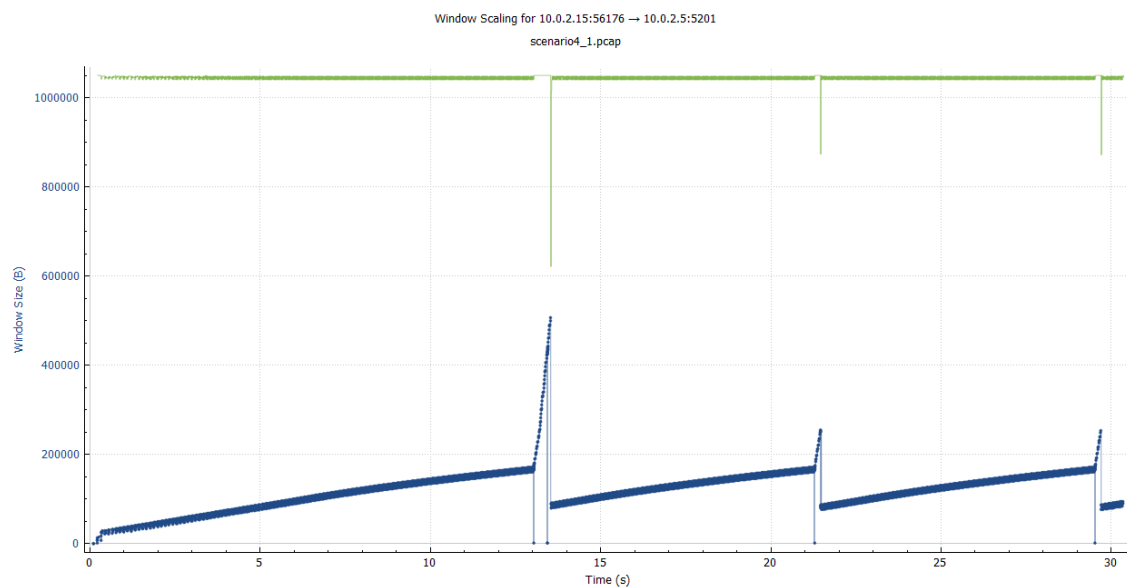
Scenario 3_2: 8Mbit/s Bandwidth, 1% Packet Loss Rate



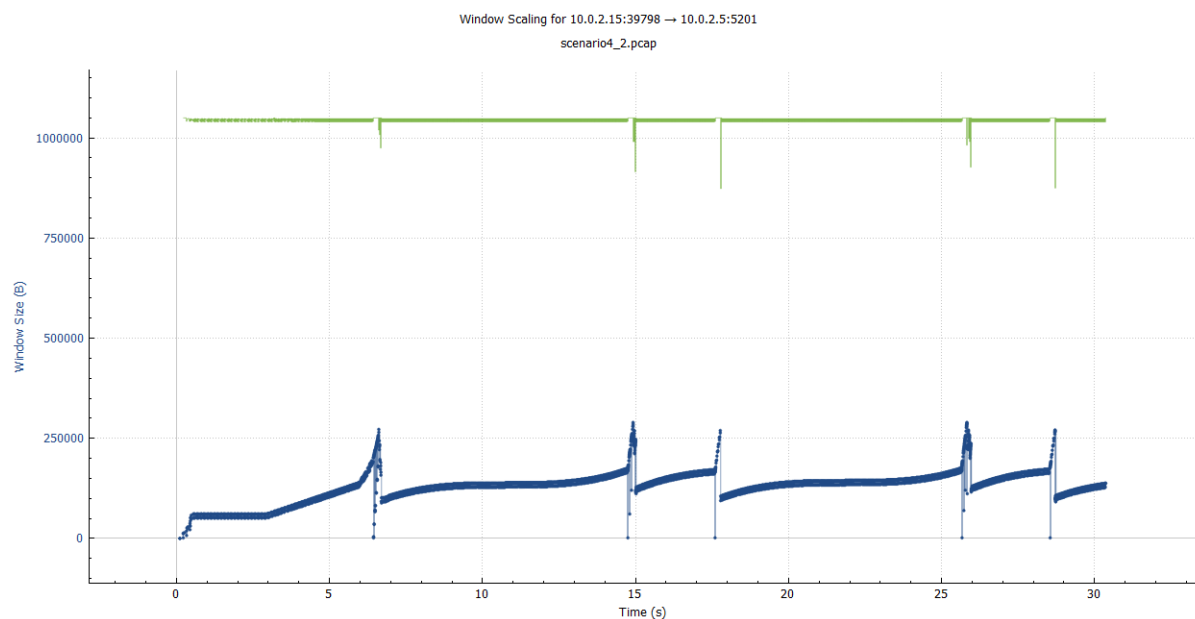
Analysis:

From the first scenario, we can see one slow start in the beginning. We see multiple congestion avoidance, indicated by the rising spike. Lastly, we see multiple fast recovery, indicated by the break of the spike drop. From the second scenario, we can see one slow start in the beginning. We see multiple congestion avoidance, indicated by the rising spike, compared to the first scenario. Lastly, we also see a lot of fast recoveries, compared to the first scenario. Then, we can conclude that the lower the packet loss rate, the better.

Scenario 4_1: 8Mbit/s Bandwidth, Delay 100 msec, Algorithm newreno



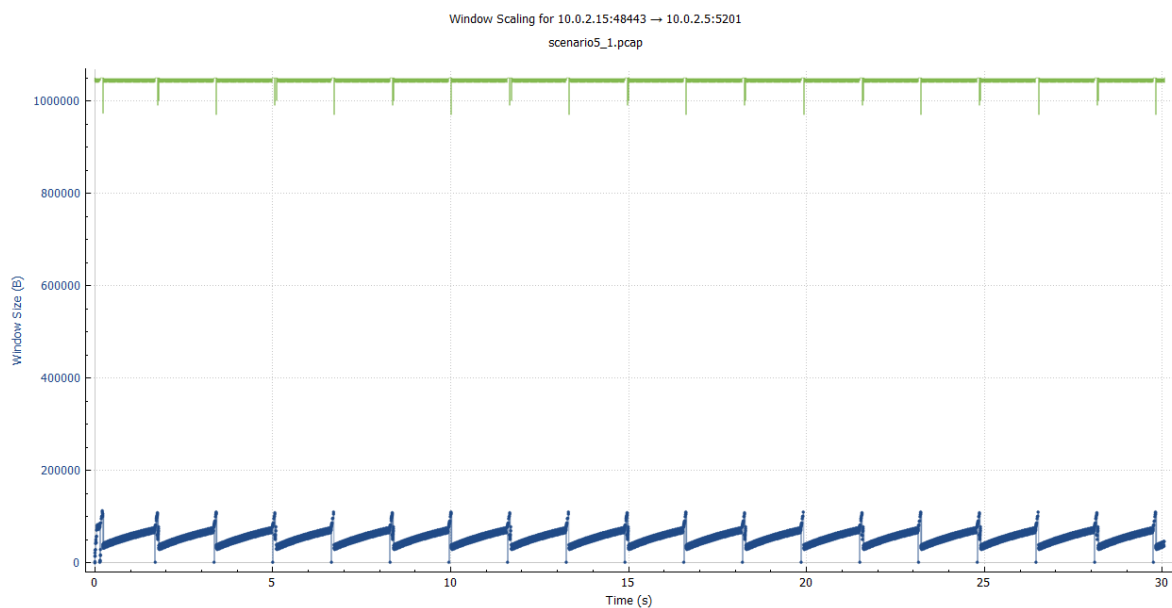
Scenario 4_2: 8Mbit/s Bandwidth, Delay 100 msec, Algorithm cubic



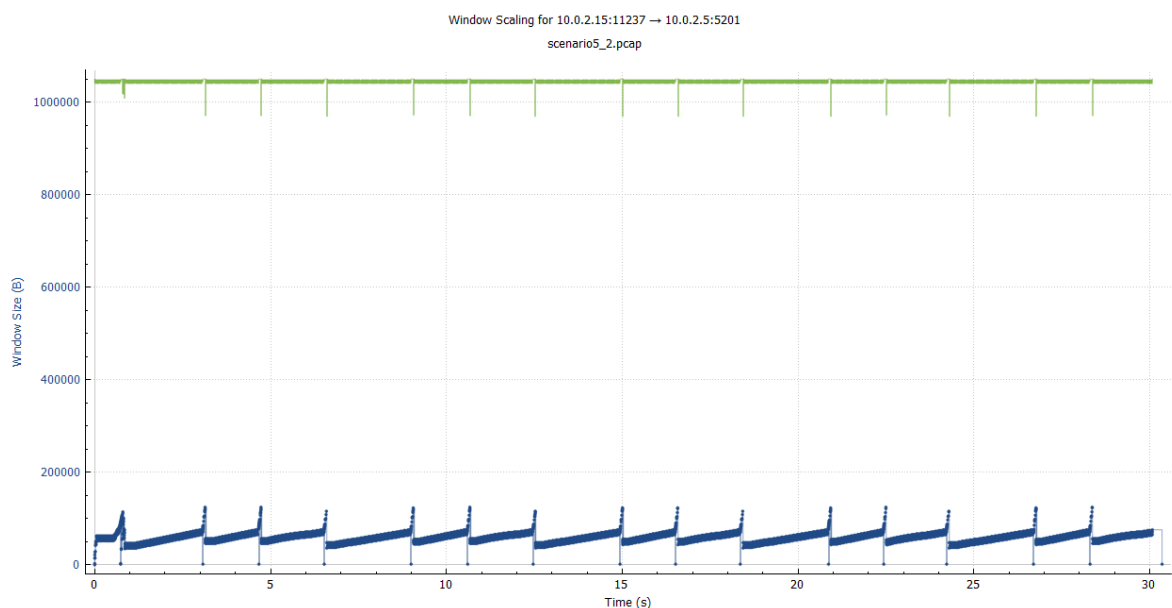
Analysis:

From the first scenario, we can see one slow start in the beginning. We see about three congestion avoidance, indicated by the rising spike. Lastly, we see three fast recovery, indicated by the break of the spike drop. From the second scenario, we can see one slow start in the beginning. We see much more congestion avoidance, indicated by the rising spike, compared to the first scenario. Lastly, we also see a lot of fast recoveries, compared to the first scenario. We can conclude that both algorithms are optimized, however the cubic algorithm performs better as it can reach the available bandwidth much faster than newreno.

Scenario 5_1: 8Mbit/s Bandwidth, 1% Packet Loss Rate, Algorithm newreno



Scenario 5_2: 8Mbit/s Bandwidth, 1% Packet Loss Rate, Algorithm cubic



Analysis:

From the first scenario, we can see one slow start in the beginning. We see multiple congestion avoidance, indicated by the rising spike. Lastly, we see multiple fast recovery, indicated by the break of the spike drop. From the second scenario, we can see one slow start in the beginning. We see less congestion avoidance, indicated by the rising spike, compared to the first scenario. Lastly, we also see less fast recoveries, compared to the first scenario. We can conclude that both algorithms are optimized, however in this scenario, the newreno is performing better.

Problem Faced

During the environment setup process, I struggled a little bit as I have no experience with using virtual machines because I didn't use virtual machines to do Assignment 1. I happened to have a spare harddrive, hence for Assignment 1, I installed ubuntu to that spare harddrive. Due to those reasons, I had to spend more time reading documentation on the internet to make sure that everything would be installed correctly.

During the process of saving captured packets to a .pcap file, I also had a few issues as I can't seem to save the file to the shared_data folder. Then, I realized that I have to do the mount command everytime I reload the VM server. After realizing that, I had no more problems about saving the .pcap file.