Basant Solanky

12040430

Assignment- 2

Transport Layer and Network Simulations using NS-3

PART 3

Understanding TCP Congestion Window using NS-3

1. For each of the given congestion control algorithms, perform the simulations and answer the following questions.

a. Newreno

b. Highspeed

c. Veno

d. Vegas

Q1. Plot the cwnd vs. time graph, and describe what you observed, like slow start and congestion avoidance, in detail.

Using Gnuplot with the given script, plotting the graph between cwnd vs. time for each algorithm.

Steps to generate demo.cwnd file:-

- Copy Demo.cc file in **scratch/** folder.

```
❯ cd scratch/
❯ ls
CMakeLists.txt  Demo.cc  first.cc  nested-subdir  q2.cc  scratch-simulator.cc  subdir
~/Downloads/ns-allinone-3.40/ns-3.40/scratch ❯
```

- Run **.ns3 run scratch/Demo.cc**

```
❯ ./ns3 run scratch/Demo.cc
[0/2] Re-checking globbed directories...
[1/1] Linking CXX executable /home/mafia/Downloads/ns-allinone-3.40/ns-3.40/build/scratch/ns3.40-Demo-debug
~/Downloads/ns-allinone-3.40/ns-3.40 ❯
```

By changing the Demo.cc file in this line :

“Config::SetDefault("ns3::TcpL4Protocol::SocketType", TypeIdValue (<**Algorithm Name**>::GetTypeId()));”

We can observe for different algorithms.

- After that, we will get an executable file **ns3.40-Demo-debug** at build/scratch/<file>.

```
❯ cd Downloads/ns-allinone-3.40/ns-3.40/
❯ ls
__pycache__      Basant-2-1.pcap  cmake-cache        demo-file-1-1.pcap  examples          scratch         utils
AUTHORS          Basant-3-0.pcap  CMakeLists.txt     demo-file-2-0.pcap  LICENSE           setup.cfg       utils.py
Basant-0-0.pcap  bindings         contrib            demo-file-2-1.pcap  ns3               setup.py        VERSION
Basant-1-0.pcap  build            CONTRIBUTING.md    demo-file-3-0.pcap  pyproject.toml    src
Basant-1-1.pcap  build-support    demo-file-0-0.pcap demo.cwnd           README.md         test.py
Basant-2-0.pcap  CHANGES.md       demo-file-1-0.pcap doc                 RELEASE_NOTES.md  testpy-output
❯ cd build
❯ ls
examples  include  lib  scratch  src  utils
❯ cd scratch
❯ ls
nested-subdir  ns3.40-Demo-debug  subdir
~/Downloads/ns-allinone-3.40/ns-3.40/build/scratch ❯                                              16:07:26
```

Running this file will provide three files as follows: one is demo.cwnd:

```
❯ ls
examples  include  lib  scratch  src  utils
❯ cd scratch
❯ ls
demo-file-0-0.pcap  demo.cwnd      ns3.40-Demo-debug
demo-file-1-0.pcap  nested-subdir  subdir
~/Downloads/ns-allinone-3.40/ns-3.40/build/scratch ❯
```

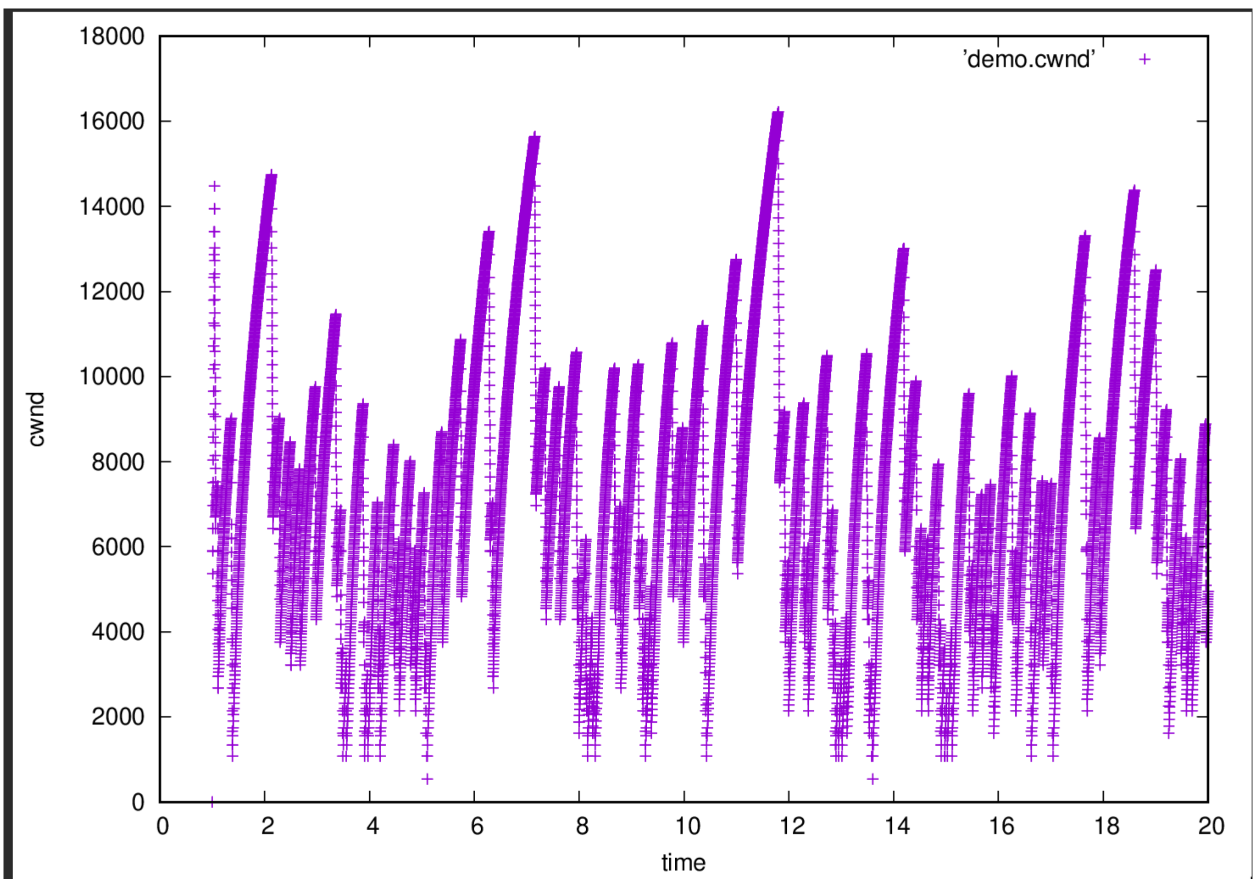
>> graph.pg -> uses gnuplot to generate graph.

```
set term postscript eps color
set output 'cwnd.eps'
set ylabel 'cwnd'
set xlabel 'time'
plot 'demo.cwnd'
```

Then run: **>> gnuplot graph.gp**

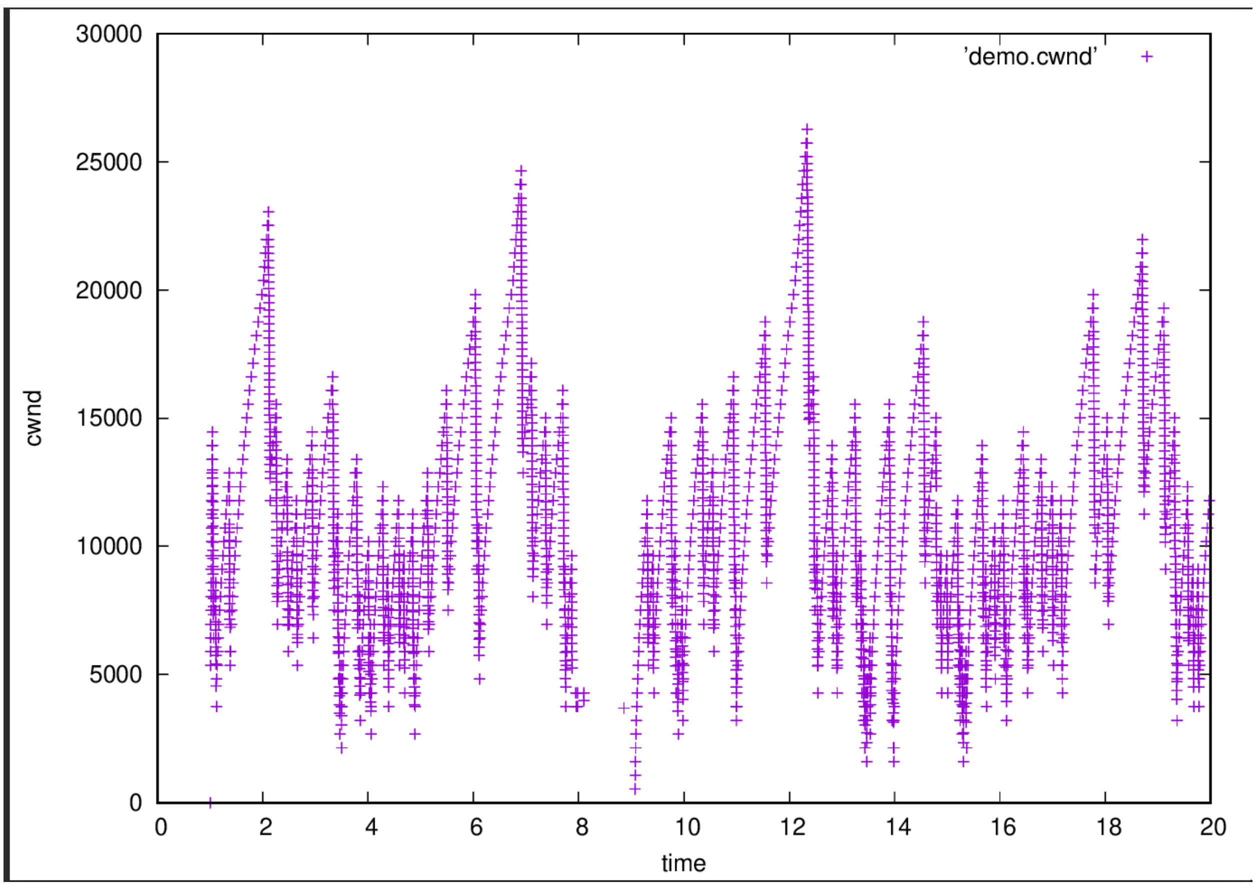
Doing the above steps for all the algorithms.

a. **Tcp NewReno**



The NewReno algorithm uses a fast retransmit technique to recover from loss events quickly. It immediately retransmits when a loss event is detected and starts a timer. Suppose the timer expires before an ACK is received for the retransmitted packet. In that case, NewReno assumes that the packet has been lost again and retransmits it repeatedly until an ACk is received for the retransmitted packet.

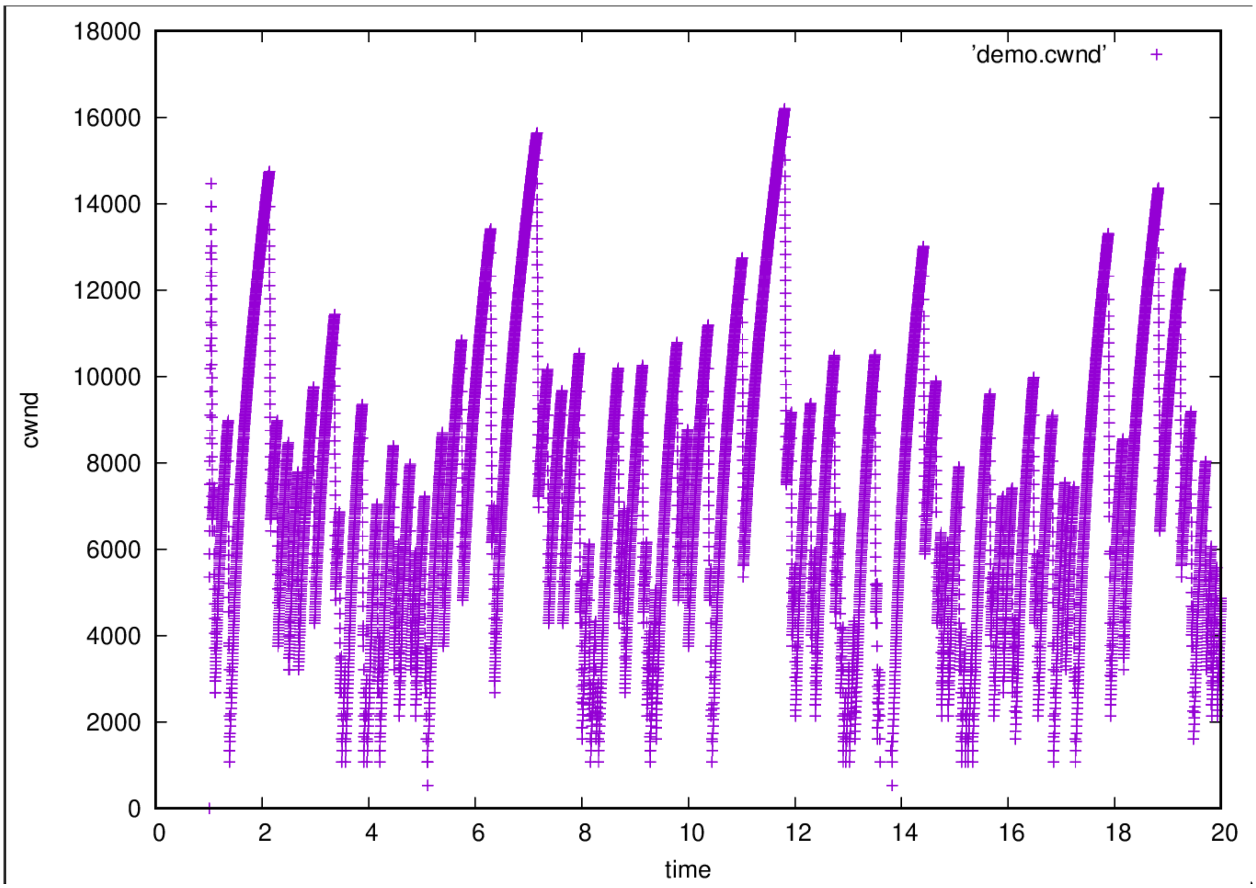
b. Tcp HighSpeed



In the Highspeed algorithm, a slow start restart is used to recover from the loss events and avoid congestion quickly. When Highspeed detects a loss event, it retransmits the lost packet and resets the cwnd to half its previous value.

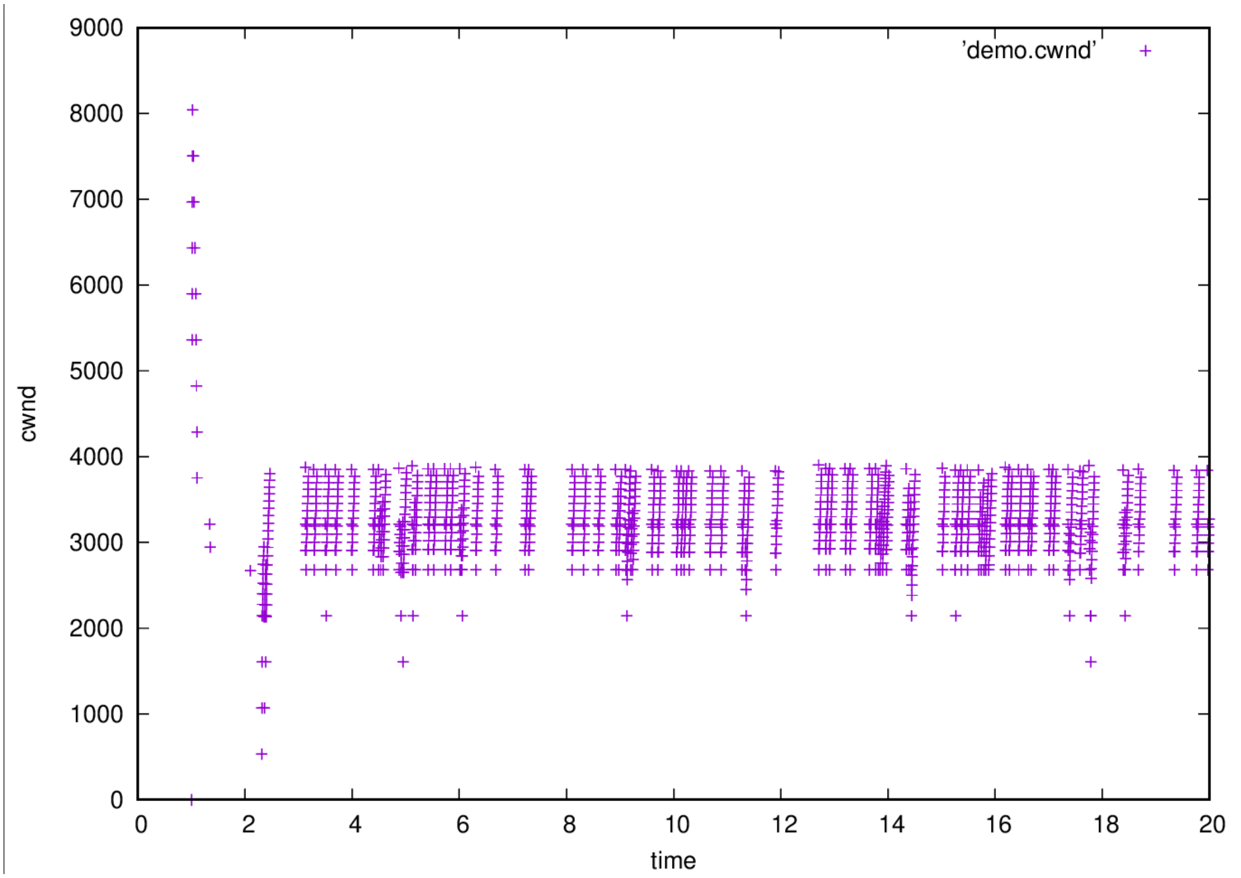
Highspeed then enters a slow start phase, doubling the cwnd each time it receives an ACK. This continues until the cwnd reaches its previous value.

c. **Tcp Veno**



In the Tcp Veno algorithm, a combination of slow start and congestion avoidance is used. In the slow start phase, it doubles the cwnd each time it receives an ACK. In the congestion avoidance mode, it increases the cwnd by one each time it receives ACK. But this algorithm also uses an Adaptive exponential increase/multiplicative decrease (AIMD) to adjust the rate at which the cwnd will be increased.

d. Tcp Vegas



The Tcp Vegas algorithm uses rate-based pacing to avoid congestion. It maintains a running estimate of the available bandwidth and uses this estimate to adjust the rate at which it sends packets. If it detects that the network is congested, it will slow down the rate at which it sends packets.

Q2. Find the average throughput for each of the congestion control algorithms using tshark from the pcap files generated, and state which algorithm performed the best.

Command: tshark -r demo-file-1-0.pcap -qz io,stat,1,BYTES

a. **Tcp NewReno**

```
===========================
| IO Statistics           |
|                         |
| Duration: 18.999916 secs|
| Interval:  1 secs       |
|                         |
| Col 1: BYTES            |
|-------------------------|
|          |1       |     |
| Interval |  BYTES |     |
|------------------|      |
|  0 <>  1 | 634884 |     |
|  1 <>  2 | 656156 |     |
|  2 <>  3 | 579056 |     |
|  3 <>  4 | 621386 |     |
|  4 <>  5 | 622864 |     |
|  5 <>  6 | 652898 |     |
|  6 <>  7 | 655580 |     |
|  7 <>  8 | 587708 |     |
|  8 <>  9 | 623674 |     |
|  9 <> 10 | 631268 |     |
| 10 <> 11 | 655066 |     |
| 11 <> 12 | 606032 |     |
| 12 <> 13 | 581242 |     |
| 13 <> 14 | 614302 |     |
| 14 <> 15 | 589370 |     |
| 15 <> 16 | 631798 |     |
| 16 <> 17 | 627064 |     |
| 17 <> 18 | 654294 |     |
| 18 <> Dur| 632360 |     |
===========================
```

Average Throughput: Total Bytes/ Duration

$$= \frac{11857002}{18.999916}$$

= **624055.4958** Bytes/sec

b. Tcp Highspeed

```
==========================
| IO Statistics          |
|                        |
| Duration: 18.999132 secs|
| Interval:  1 secs      |
|                        |
| Col 1: BYTES           |
|------------------------|
|          |1        |   |
| Interval |  BYTES  |   |
|------------------|     |
|  0 <>  1 | 653810 |    |
|  1 <>  2 | 659430 |    |
|  2 <>  3 | 656704 |    |
|  3 <>  4 | 658756 |    |
|  4 <>  5 | 655390 |    |
|  5 <>  6 | 657086 |    |
|  6 <>  7 | 661036 |    |
|  7 <>  8 |  66804 |    |
|  8 <>  9 | 606828 |    |
|  9 <> 10 | 657716 |    |
| 10 <> 11 | 654760 |    |
| 11 <> 12 | 658556 |    |
| 12 <> 13 | 653886 |    |
| 13 <> 14 | 656788 |    |
| 14 <> 15 | 656096 |    |
| 15 <> 16 | 657664 |    |
| 16 <> 17 | 657568 |    |
| 17 <> 18 | 656104 |    |
| 18 <> Dur| 658346 |    |
==========================
```

Average Throughput: Total Bytes/ Duration

$$= \frac{11843328}{18.999132}$$

= 623361.5304 Bytes/sec

c. Tcp Veno

```
=============================
| IO Statistics              |
|                            |
| Duration: 18.999724 secs|
| Interval:  1 secs          |
|                            |
| Col 1: BYTES               |
|--------------------------|
|          |1         |      |
| Interval |  BYTES |        |
|-------------------|        |
|  0 <>  1 | 634240 |        |
|  1 <>  2 | 656156 |        |
|  2 <>  3 | 579056 |        |
|  3 <>  4 | 618972 |        |
|  4 <>  5 | 622810 |        |
|  5 <>  6 | 652952 |        |
|  6 <>  7 | 657140 |        |
|  7 <>  8 | 581794 |        |
|  8 <>  9 | 623022 |        |
|  9 <> 10 | 631260 |        |
| 10 <> 11 | 656494 |        |
| 11 <> 12 | 606498 |        |
| 12 <> 13 | 442436 |        |
| 13 <> 14 | 644692 |        |
| 14 <> 15 | 572312 |        |
| 15 <> 16 | 614150 |        |
| 16 <> 17 | 626300 |        |
| 17 <> 18 | 654966 |        |
| 18 <> Dur| 628028 |        |
=============================
```

Average Throughput: Total Bytes/ Duration

$$= \frac{11703278}{18.999724}$$

= **615970.9478** Bytes/sec

d. Tcp Vegas

```
=============================
| IO Statistics              |
|                            |
| Duration: 18.998761 secs|
| Interval:  1 secs          |
|                            |
| Col 1: BYTES               |
|----------------------------|
|           |1         |     |
| Interval  |  BYTES   |     |
|----------------------|     |
|  0 <>  1 | 223458 |        |
|  1 <>  2 | 421666 |        |
|  2 <>  3 | 634590 |        |
|  3 <>  4 | 614282 |        |
|  4 <>  5 | 627404 |        |
|  5 <>  6 | 636072 |        |
|  6 <>  7 | 645298 |        |
|  7 <>  8 | 632448 |        |
|  8 <>  9 | 625254 |        |
|  9 <> 10 | 628444 |        |
| 10 <> 11 | 634620 |        |
| 11 <> 12 | 641876 |        |
| 12 <> 13 | 619760 |        |
| 13 <> 14 | 635256 |        |
| 14 <> 15 | 613250 |        |
| 15 <> 16 | 632564 |        |
| 16 <> 17 | 618744 |        |
| 17 <> 18 | 639434 |        |
| 18 <> Dur| 641232 |        |
=============================
```

Average Throughput: Total Bytes/ Duration

$$= \frac{11365652}{18.998761}$$

= **598231.2215** Bytes/sec

Tcp NewReno Performed best among all the algorithms with an average throughput of **624055.4958** Bytes/sec.

Q3. How many times did the TCP algo reduce the cwnd and why?

Counting the approx. number of peaks for each algorithm.

a. Newreno:- 81
b. Highspeed:- 75
c. Veno:- 79
d. Vegas:- 67

There can be various reasons for reducing cwnd. Some are mentioned below.

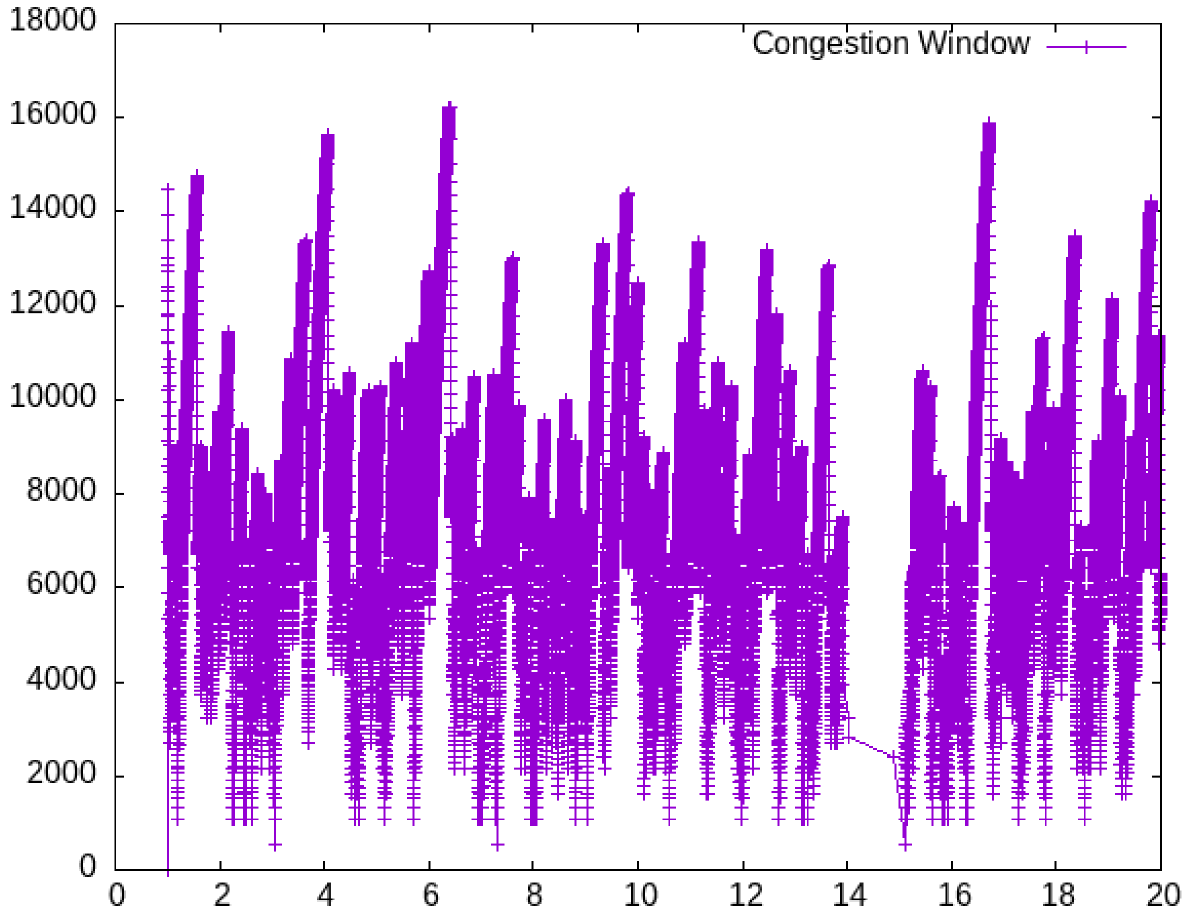
- **Packet loss:** TCp reduces cwnd when it detects packet loss.
- **Explicit Congestion Notification (ECN):** Enabled ECN can lead to cwnd reduction when network congestion is signed without packet loss.
- **Timeout:** A timeout event also indicates that congestion leads to cwnd reduction.
- **Triple Duplicate ACKs:** After detecting three ACK, some TCP variants reduce cwd as a congestion signal.

Q4. Check the effect of changing the bandwidth and latency of point-to-point connection and explain its effect on average throughput.

a. Newreno

- Increased bandwidth typically increases throughput, while decreased bandwidth produces lower throughput.
- Latency: As Newreno adjusts to network conditions, higher latency can result in lower throughput, and lower latency can result in higher throughput.

Simulation:- bw = 10mbps, latency = 1ms



```
===========================
| IO Statistics           |
|                         |
| Duration: 18.998698 secs|
| Interval:  1 secs       |
|                         |
| Col 1: BYTES            |
|-------------------------|
|           |1        |   |
| Interval |  BYTES   |   |
|--------------------|    |
|  0 <>  1 | 1283478 |    |
|  1 <>  2 | 1191646 |    |
|  2 <>  3 | 1271578 |    |
|  3 <>  4 | 1236798 |    |
|  4 <>  5 | 1249182 |    |
|  5 <>  6 | 1255004 |    |
|  6 <>  7 | 1188464 |    |
|  7 <>  8 | 1213552 |    |
|  8 <>  9 | 1275466 |    |
|  9 <> 10 | 1247658 |    |
| 10 <> 11 | 1244948 |    |
| 11 <> 12 | 1265906 |    |
| 12 <> 13 | 1218598 |    |
| 13 <> 14 |   34338 |    |
| 14 <> 15 |  997378 |    |
| 15 <> 16 | 1238250 |    |
| 16 <> 17 | 1240786 |    |
| 17 <> 18 | 1251930 |    |
| 18 <> Dur| 1274142 |    |
===========================
```

Average Throughput: Total Bytes/ Duration

$$= \frac{22179102}{18.998698}$$

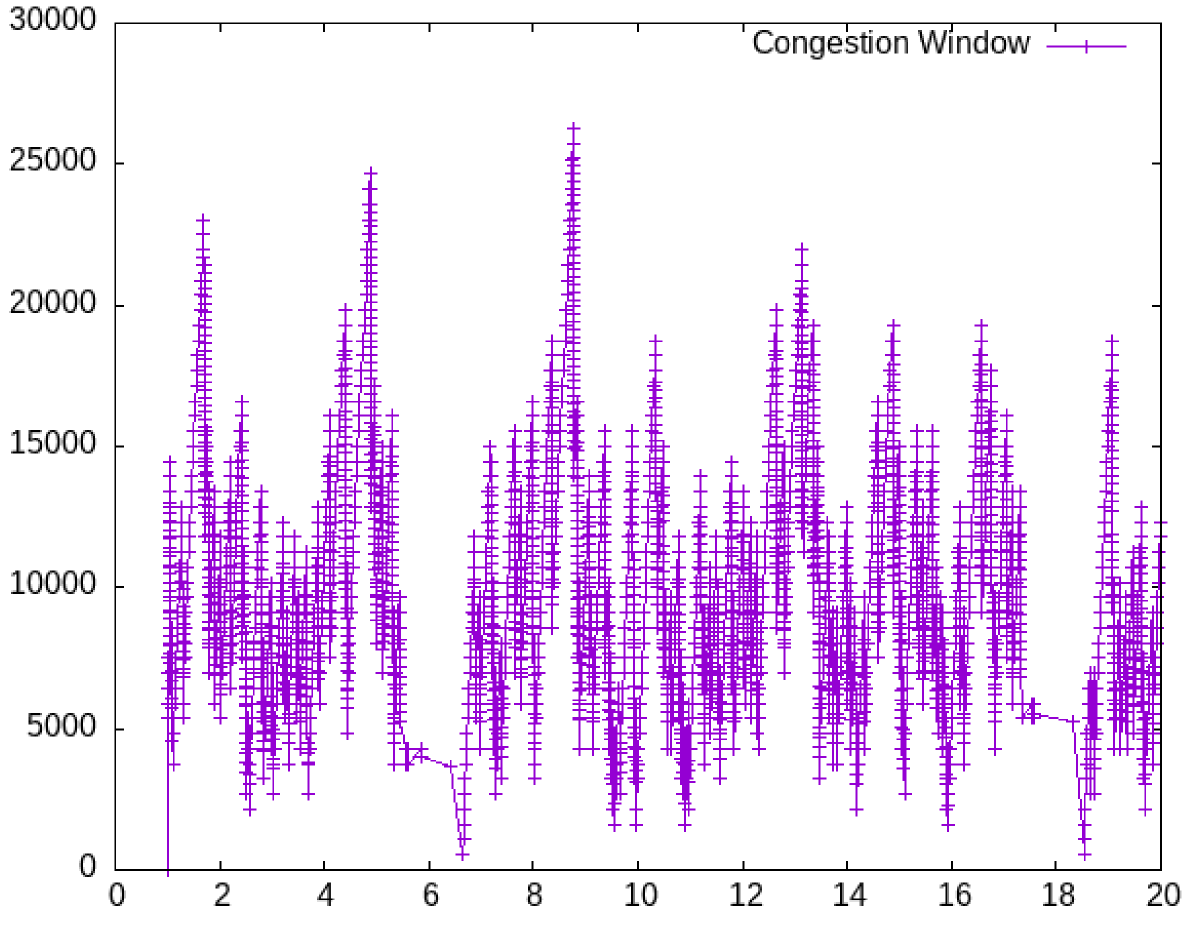
= **1167401.156** Bytes/sec

Average throughput increases on an increase in bandwidth and a decrease in the latency from the Q2 TcpNewReno throughput (624055.4958).

b. HighSpeed

- Throughput should increase when bandwidth is increased, but there may be diminishing returns at very high bandwidth levels. Throughput will decline as bandwidth is reduced.
- Low-latency environments should allow high speed to remain steady and high throughput. Because round-trip times increase with increased latency, throughput may suffer.

Simulation:- bw = 10mbps, latency = 5ms



```
=============================
| IO Statistics              |
|                            |
| Duration: 18.994624 secs|
| Interval:  1 secs          |
|                            |
| Col 1: BYTES               |
|-------------------------|
|           |1         |     |
| Interval |  BYTES   |     |
|--------------------|     |
|  0 <>   1 | 1057656 |     |
|  1 <>   2 |  923076 |     |
|  2 <>   3 |  827538 |     |
|  3 <>   4 | 1219670 |     |
|  4 <>   5 |  614472 |     |
|  5 <>   6 |  250792 |     |
|  6 <>   7 | 1000368 |     |
|  7 <>   8 | 1125776 |     |
|  8 <>   9 |  838398 |     |
|  9 <>  10 |  901156 |     |
| 10 <>  11 |  901894 |     |
| 11 <>  12 | 1104676 |     |
| 12 <>  13 | 1044064 |     |
| 13 <>  14 | 1022382 |     |
| 14 <>  15 |  888132 |     |
| 15 <>  16 | 1065078 |     |
| 16 <>  17 |  496590 |     |
| 17 <>  18 |  371878 |     |
| 18 <>  Dur|  847452 |     |
=============================
```

Average Throughput: Total Bytes/ Duration

$$= \frac{16501048}{18.994624}$$

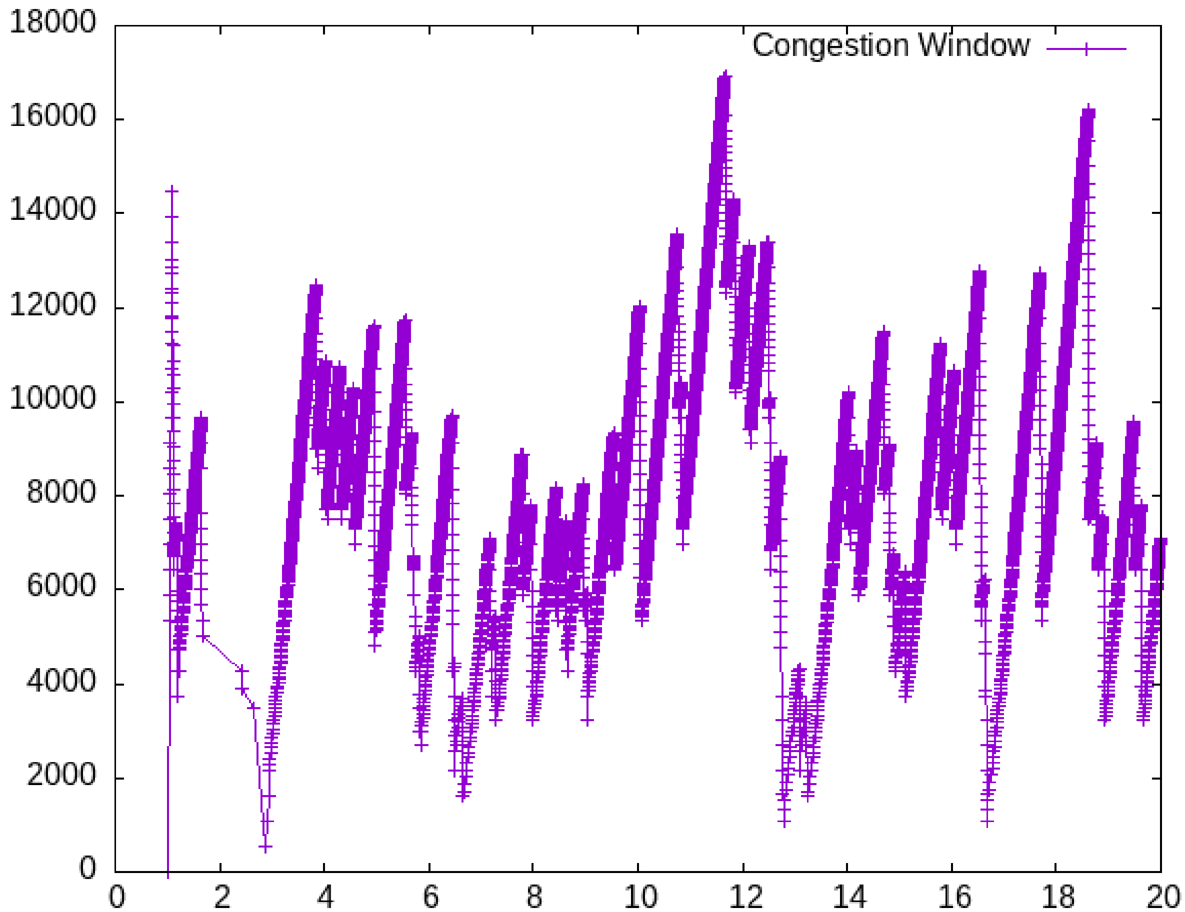
= **868722.0131** Bytes/sec

Avg. throughput increase on the increase in bandwidth from 5bytes to 10 bytes (previous Throughput=623361.5304 Bytes/sec)

c. Veno

- Throughput is typically improved when bandwidth is increased, while throughput is typically decreased when bandwidth is decreased.
- Throughput may be less susceptible to changes in latency since Veno can manage varying latencies, although throughput can still be negatively impacted by very high latency.

Simulation:- bw = 10mbps, latency = 1ms



```
=============================
| IO Statistics             |
|                           |
| Duration: 18.981829 secs|
| Interval:  1 secs         |
|                           |
| Col 1: BYTES              |
|-------------------------|
|            |1        |    |
| Interval |  BYTES |       |
|------------------|        |
|  0 <>  1 | 306654 |       |
|  1 <>  2 |  96222 |       |
|  2 <>  3 | 416116 |       |
|  3 <>  4 | 471444 |       |
|  4 <>  5 | 374256 |       |
|  5 <>  6 | 248084 |       |
|  6 <>  7 | 305758 |       |
|  7 <>  8 | 312252 |       |
|  8 <>  9 | 395240 |       |
|  9 <> 10 | 486170 |       |
| 10 <> 11 | 678914 |       |
| 11 <> 12 | 433428 |       |
| 12 <> 13 | 252296 |       |
| 13 <> 14 | 410952 |       |
| 14 <> 15 | 386834 |       |
| 15 <> 16 | 363266 |       |
| 16 <> 17 | 429682 |       |
| 17 <> 18 | 547078 |       |
| 18 <> Dur| 314028 |       |
=============================
```

Average Throughput: Total Bytes/ Duration

$$= \frac{7228674}{18.981829}$$

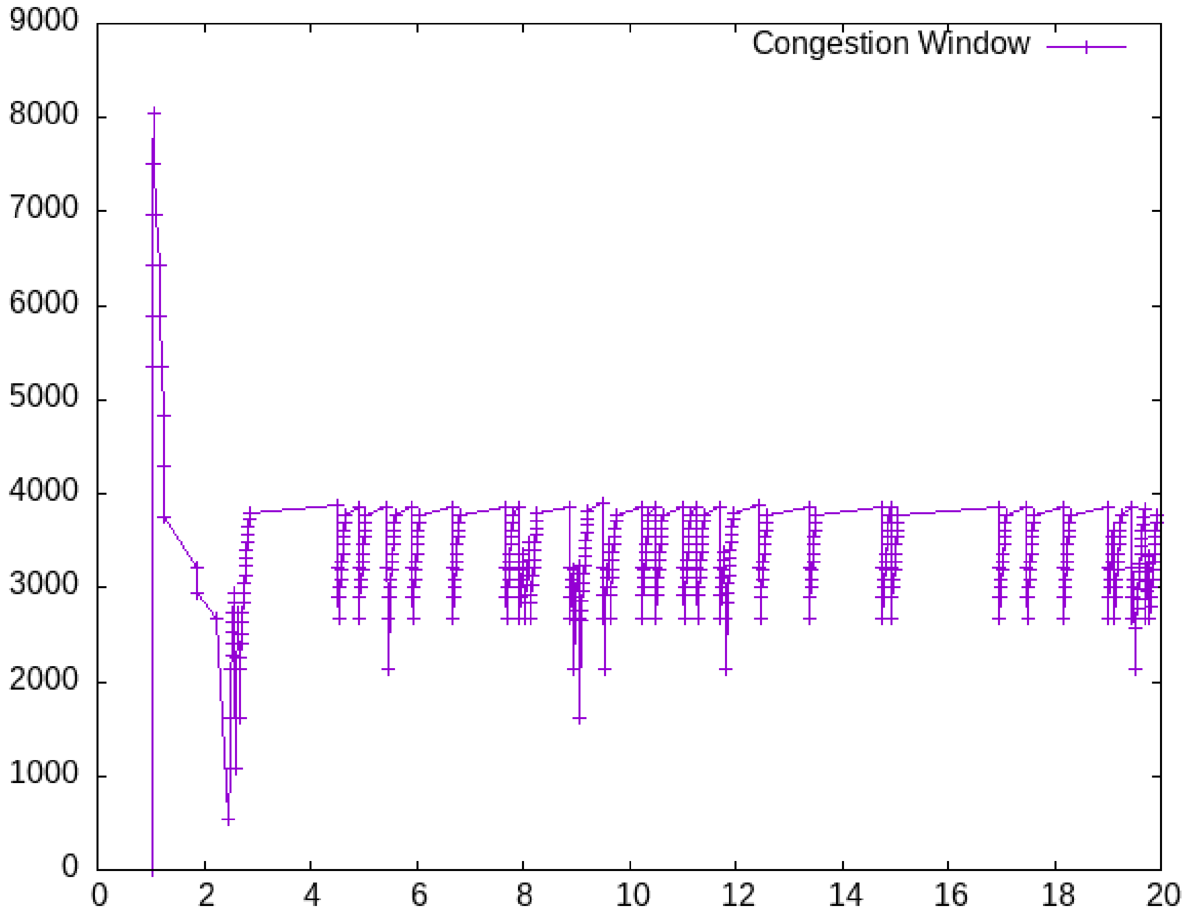
= **380820.7312** Bytes/sec

Average throughput decreases on an increase in latency from 2ms to 10ms. (prev throughput =615970.9478 Bytes/sec).

d. Vegas

- Increased bandwidth leads to increased throughput, whereas decreased bandwidth may temporarily lower performance before Vegas adapts.
- Vegas is built to accommodate changes in bandwidth and latency, thus, changes in latency have less of an impact on throughput, although highly high latency may cause some throughput drop as Vegas adapts.

Simulation:- bw = 10mbps, latency = 1ms



```
============================
| IO Statistics            |
|                          |
| Duration: 18.992449 secs|
| Interval:  1 secs        |
|                          |
| Col 1: BYTES             |
|--------------------------|
|          |1        |     |
| Interval |  BYTES  |     |
|--------------------|     |
|  0 <>  1 | 221688 |      |
|  1 <>  2 | 105388 |      |
|  2 <>  3 | 261608 |      |
|  3 <>  4 | 252296 |      |
|  4 <>  5 | 250084 |      |
|  5 <>  6 | 256308 |      |
|  6 <>  7 | 251714 |      |
|  7 <>  8 | 242246 |      |
|  8 <>  9 | 237932 |      |
|  9 <> 10 | 252350 |      |
| 10 <> 11 | 241408 |      |
| 11 <> 12 | 256308 |      |
| 12 <> 13 | 256362 |      |
| 13 <> 14 | 252940 |      |
| 14 <> 15 | 260964 |      |
| 15 <> 16 | 258186 |      |
| 16 <> 17 | 254484 |      |
| 17 <> 18 | 256642 |      |
| 18 <> Dur| 228370 |      |
============================
```

Average Throughput: Total Bytes/ Duration

$$= \frac{4597278}{18.992449}$$

= **242058.1990** Bytes/sec

Average throughput decreases drastically on the decrease in bandwidth and an increase in latency. (prev throughput = 598231.2215 Bytes/sec).

Q5. Explain in short what is the effect of changing the default MTU size.

MTU:- The maximum transmission unit is the largest size frame or packet in bytes that can be transmitted across a data link. It is most used for packet size on an Ethernet network using the Internet Protocol.
To achieve the same network speed, the operating system can deliver fewer packets of a bigger size when using large MTU sizes. If the workload permits sending large messages, the larger packets significantly reduce the processing the operating system needs. The greater MTU size won't help if all of the workload consists of sending brief messages.

Effects of changing the default MTU size:-

- Affects network performance and compatibility.
- Larger MTUs can improve performance but may lead to fragmentation.

2. Using the Demo.cc file as reference, create a new scenario topology consisting of 4 nodes as shown below.

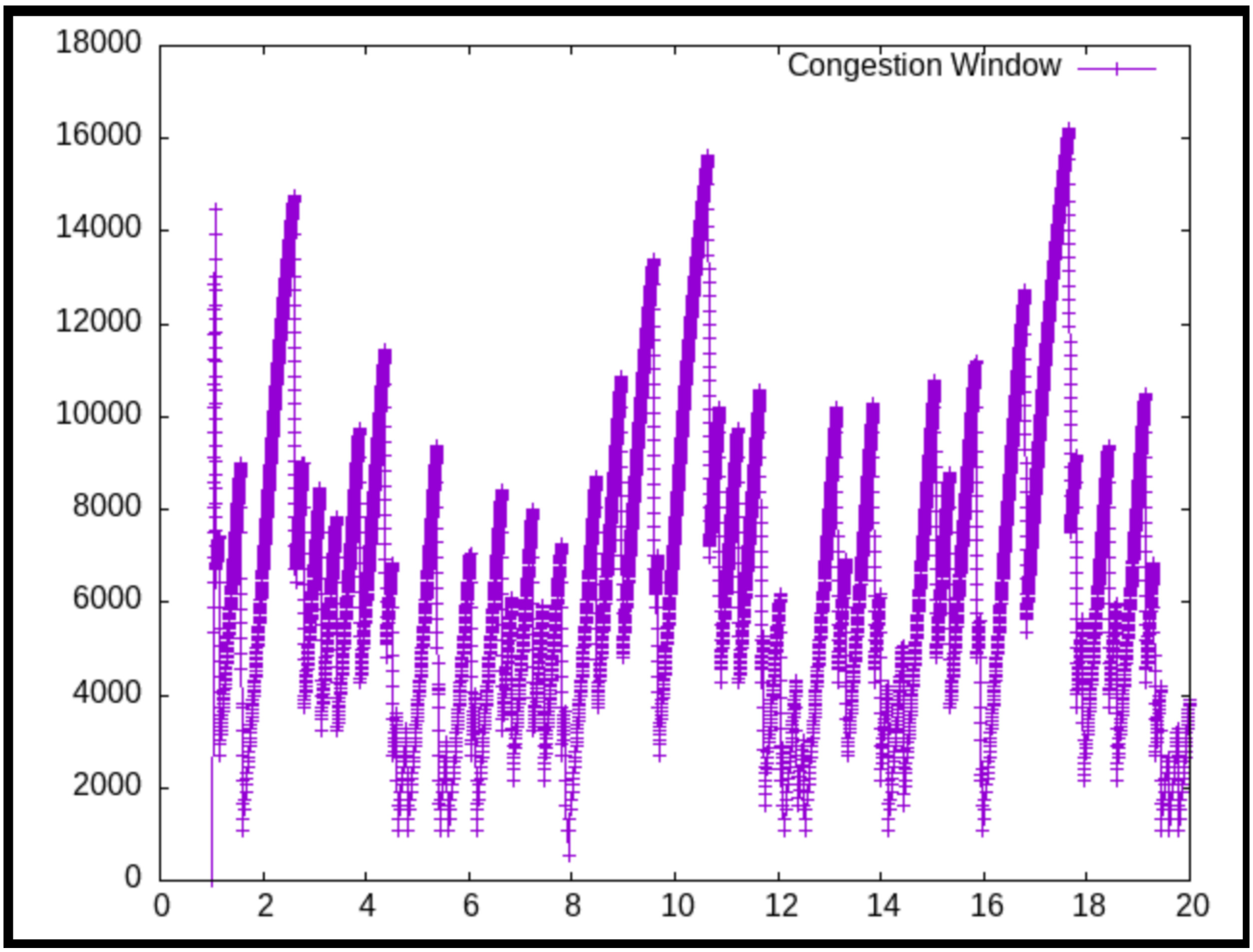
Q1. Find the average throughput for the given parameters.

bw1 = 5mbps, latency1 = 2ms

bw2 = 5mbps, latency2 = 2ms

bw3 =5mbps, latency3 =2ms

Using TcpNewReno algorithm for performing this simulation.



```
============================
| IO Statistics            |
|                          |
| Duration: 18.992858 secs|
| Interval:  1 secs        |
|                          |
| Col 1: BYTES             |
|--------------------------|
|          |1         |    |
| Interval |   BYTES  |    |
|-------------------|      |
|  0 <>  1 | 344574 |      |
|  1 <>  2 | 551966 |      |
|  2 <>  3 | 413442 |      |
|  3 <>  4 | 341736 |      |
|  4 <>  5 | 295480 |      |
|  5 <>  6 | 290026 |      |
|  6 <>  7 | 293672 |      |
|  7 <>  8 | 414774 |      |
|  8 <>  9 | 491942 |      |
|  9 <> 10 | 597128 |      |
| 10 <> 11 | 426882 |      |
| 11 <> 12 | 227648 |      |
| 12 <> 13 | 434202 |      |
| 13 <> 14 | 304194 |      |
| 14 <> 15 | 453760 |      |
| 15 <> 16 | 453718 |      |
| 16 <> 17 | 561746 |      |
| 17 <> 18 | 376884 |      |
| 18 <> Dur| 252438 |      |
============================
```

Avg. throughput = **Total Bytes/ Duration**

$$= \frac{7526212}{18.992858}$$

$$= \frac{512338}{3}$$

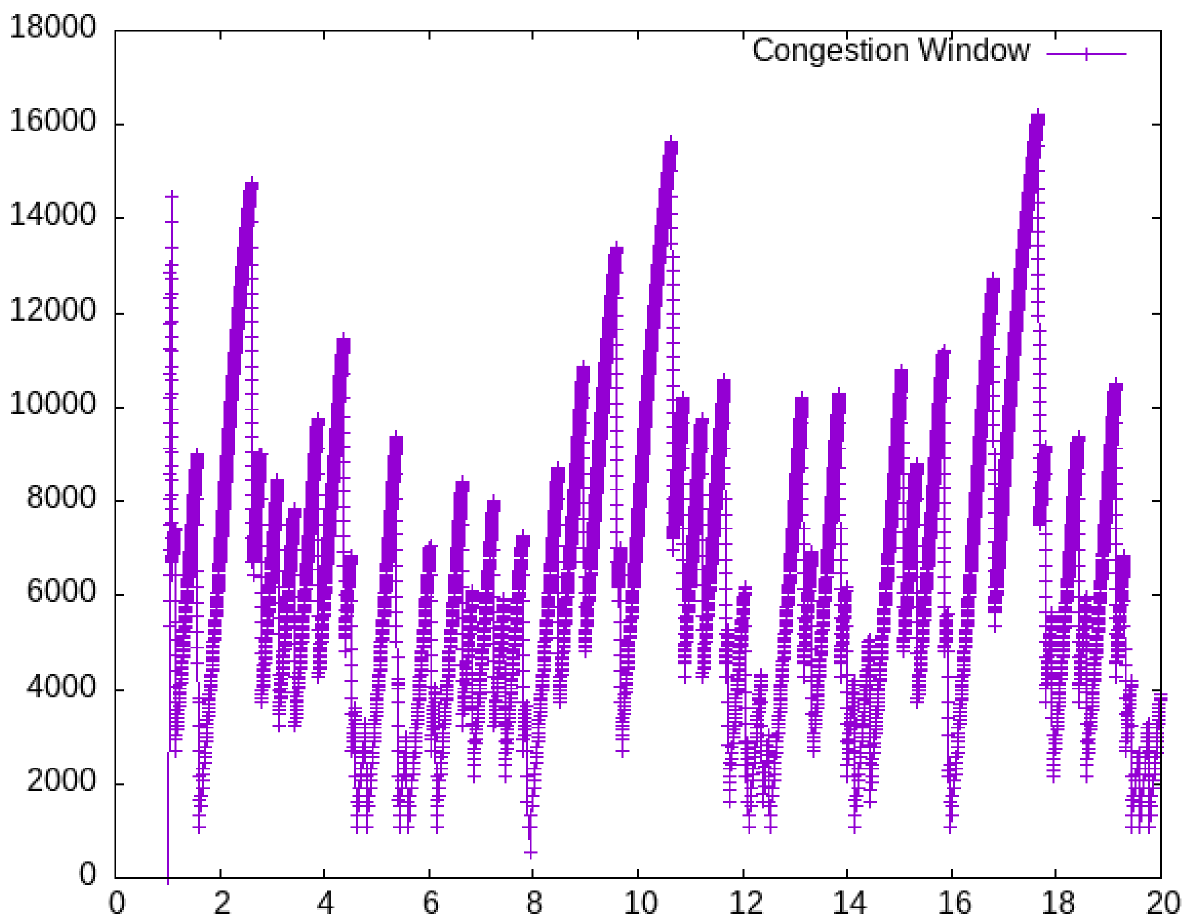
= 396265.3751 bps

Q2. Now change the values of latency2 from 2ms to 10ms and plot and compare the cwnd shape. Do this for two different congestion control algorithms.

For this part, I used the modified version of [Demo.cc]() script to find the throughput. The screenshot shows the throughput output for delay values at latency2 2ms,4ms,6ms,8ms, and 10ms, respectively, for both TcpNewReno and TcpVegas.

a. **TcpNewReno**

1. **Latency2=2ms**

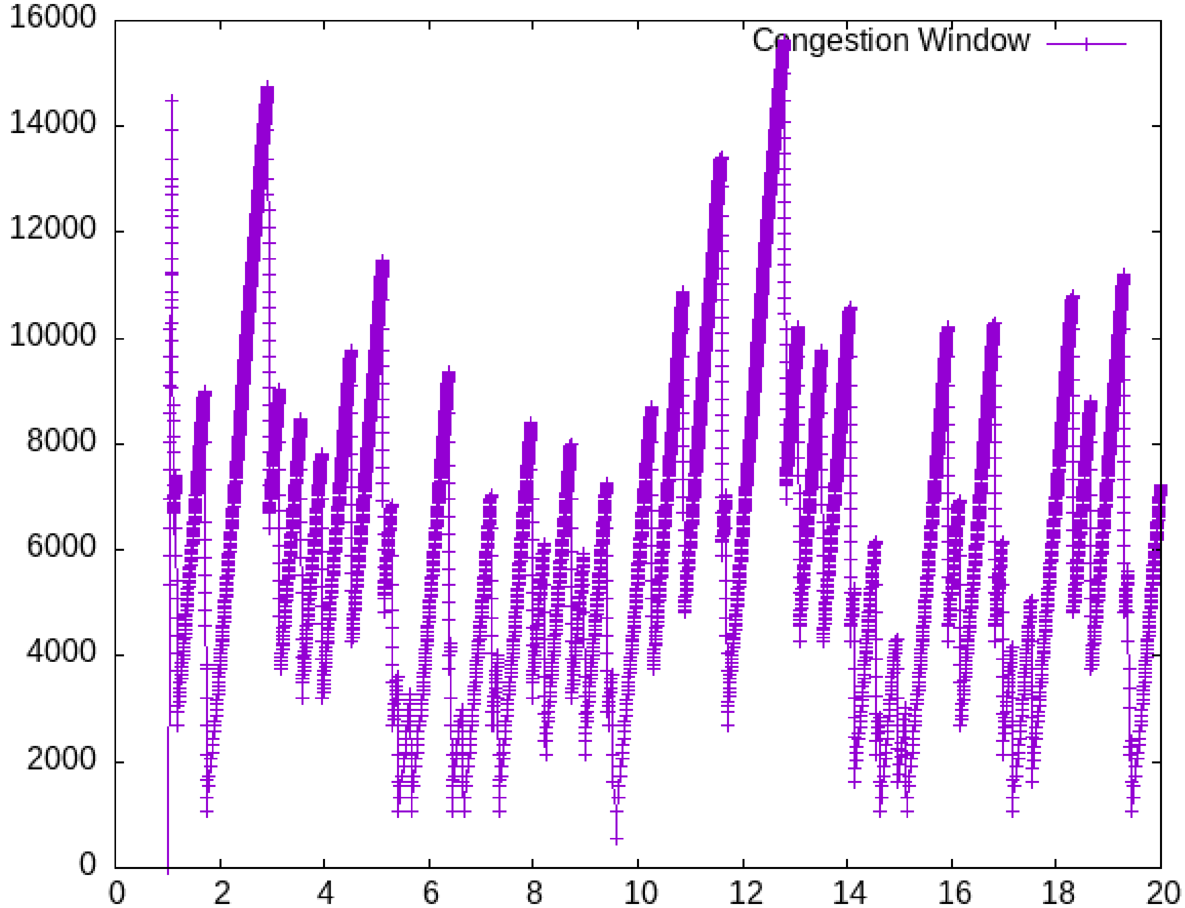


Average Throughput: Total Bytes/ Duration

$$= \frac{7526212}{18.992858}$$

= 396265.3751 bps

2. 4ms

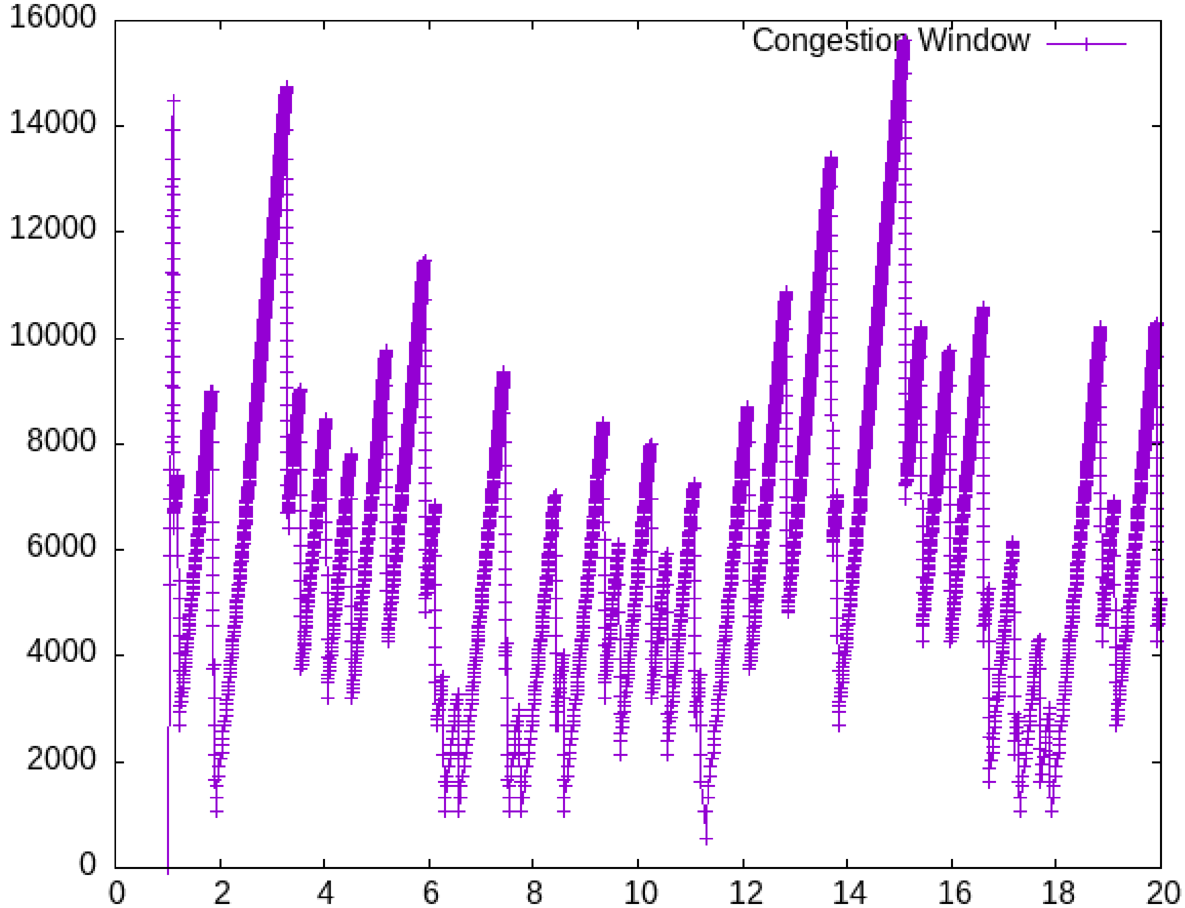


Average Throughput: Total Bytes/ Duration

$$= \frac{5993720}{18.990784}$$

= 315612.0358 bps

3. 6ms

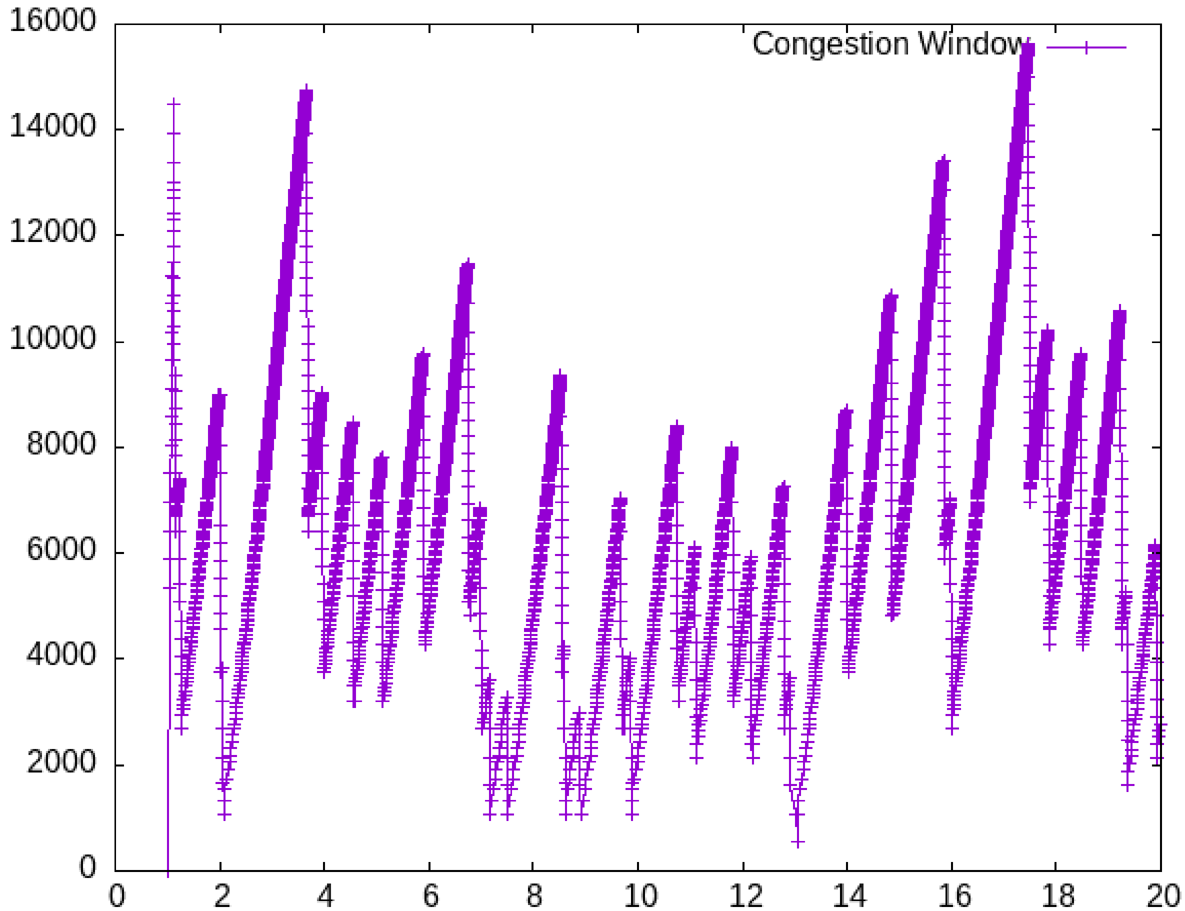


Average Throughput: Total Bytes/ Duration

$$= \frac{5090324}{18.981917}$$

= 268167.0139 bps

4. 8ms

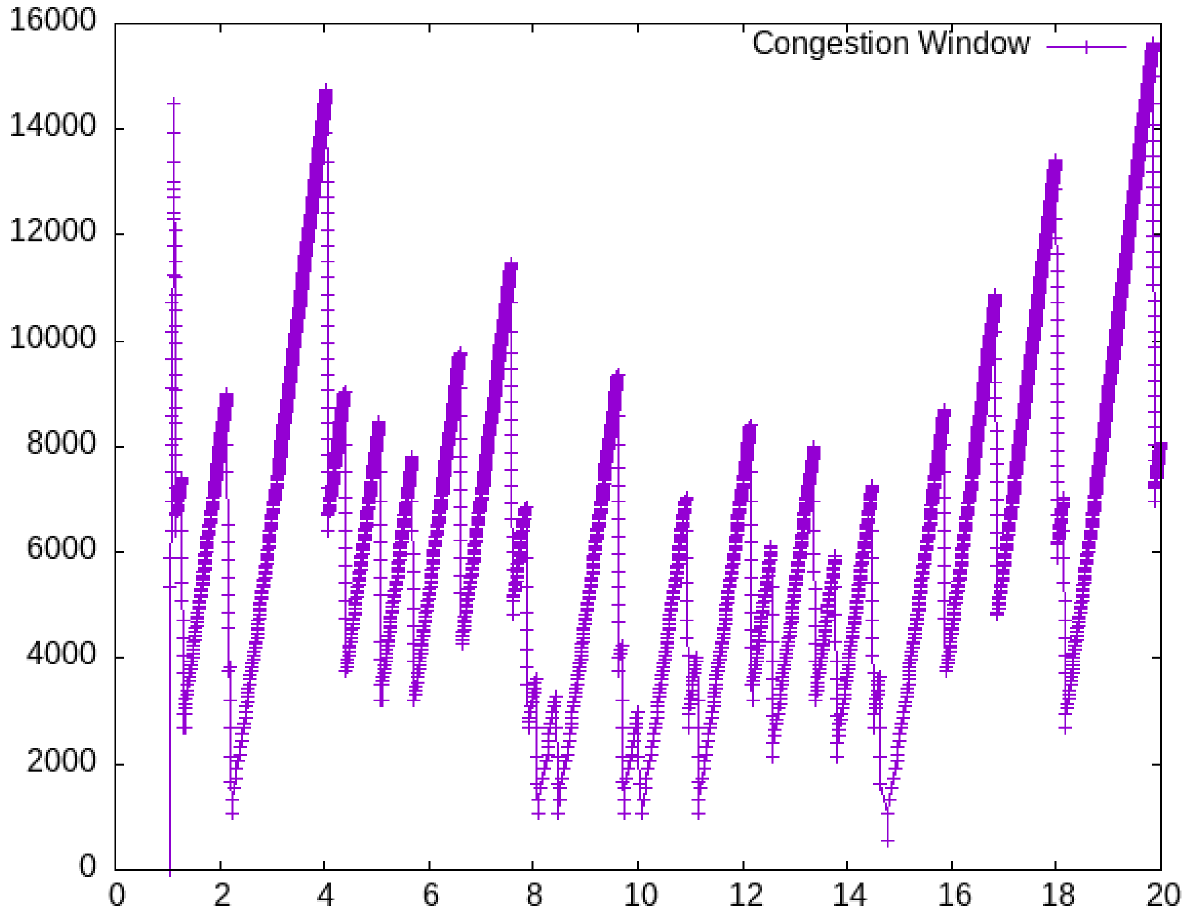


Average Throughput: Total Bytes/ Duration

$$= \frac{4484968}{18.963590}$$

= 236504.1640 bps

5. 10ms



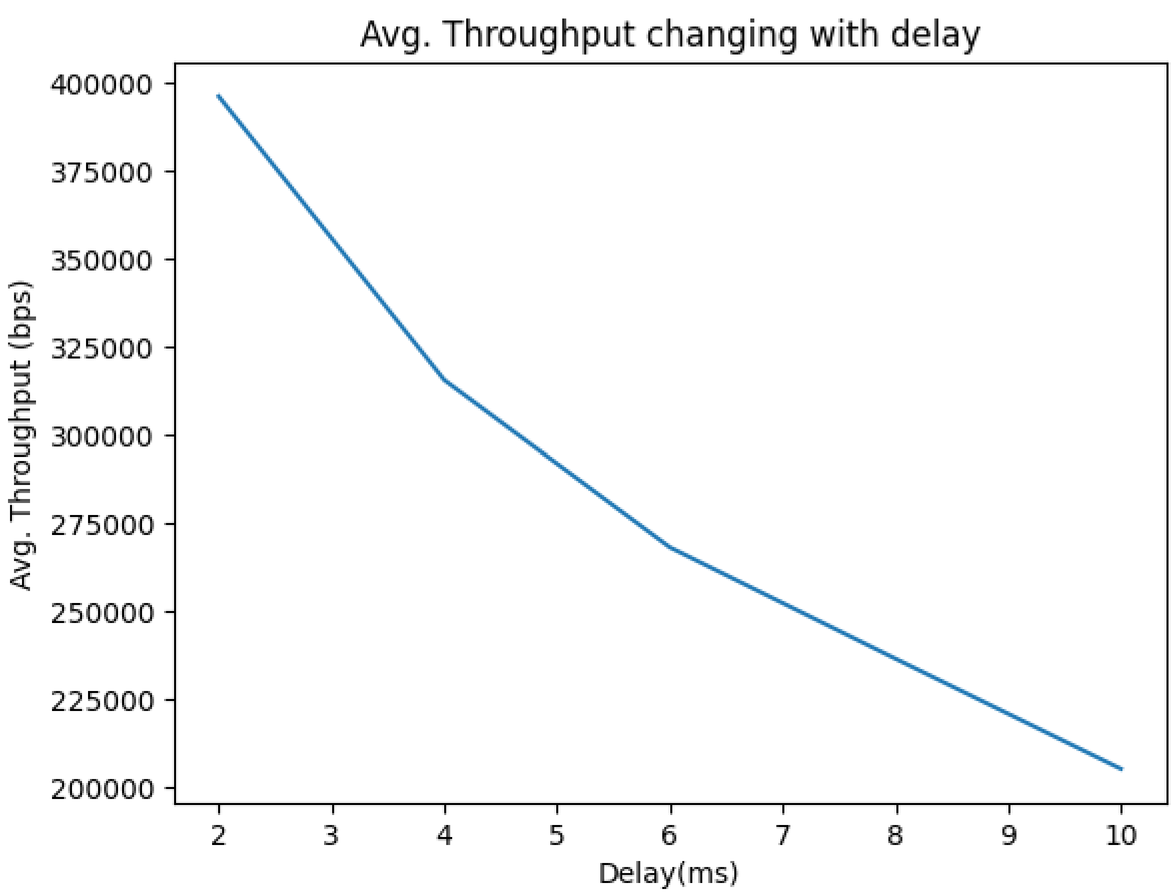
Average Throughput: Total Bytes/ Duration

$$= \frac{3895242}{18.985334}$$

= 205171.1073 bps

The graph uses the values from 2ms to 10ms of latency2 using a Python script.

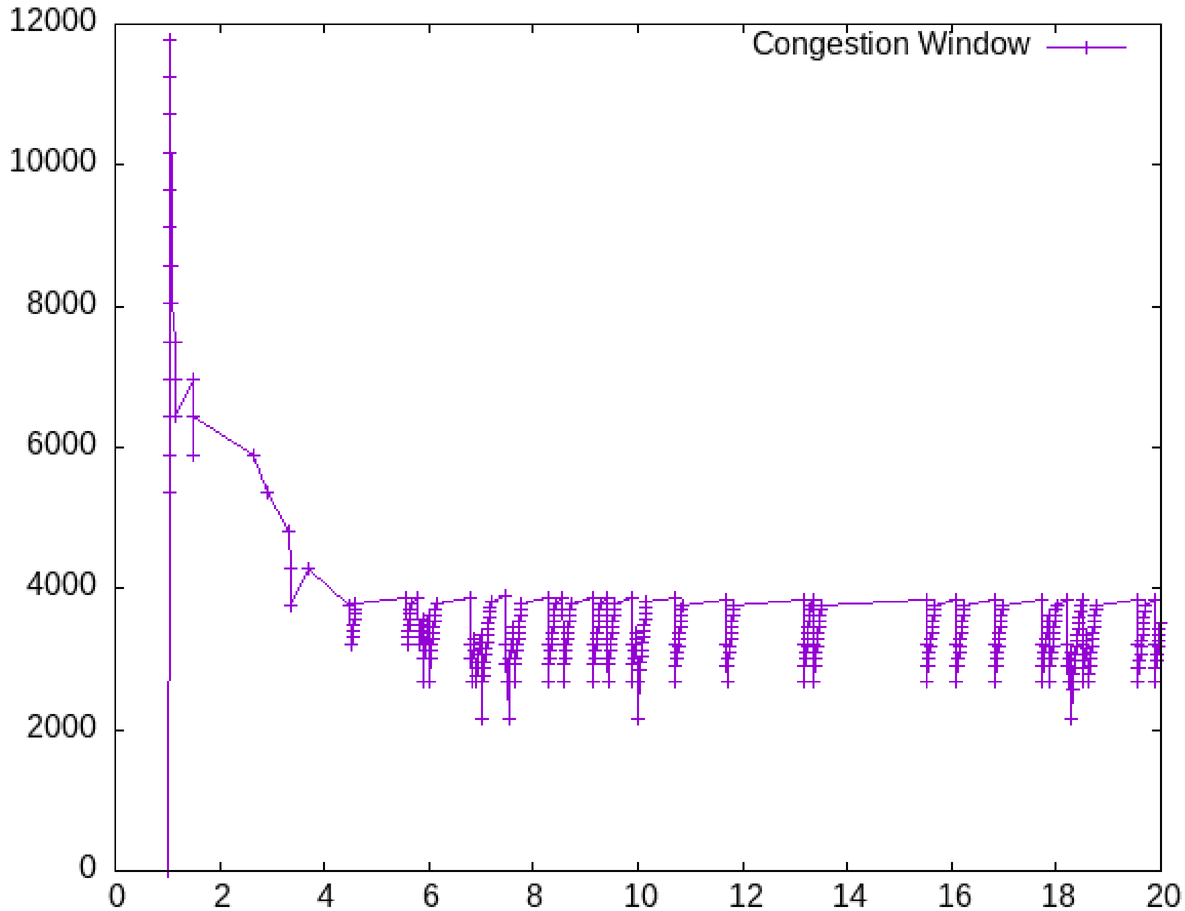
```python
import matplotlib.pyplot as plt

x = [2,4,6,8,10]
y = [396265.3751,315612.0358,268167.0139,236504.1640,205171.1073]

plt.plot(x,y)

plt.xlabel('Delay(ms)')
plt.ylabel('Avg. Throughput (bps)')

plt.title('Avg. Throughput changing with delay')

plt.show()
```



B. TcpVegas

Repeating the same above step for this Algorithm.

1. **2ms**

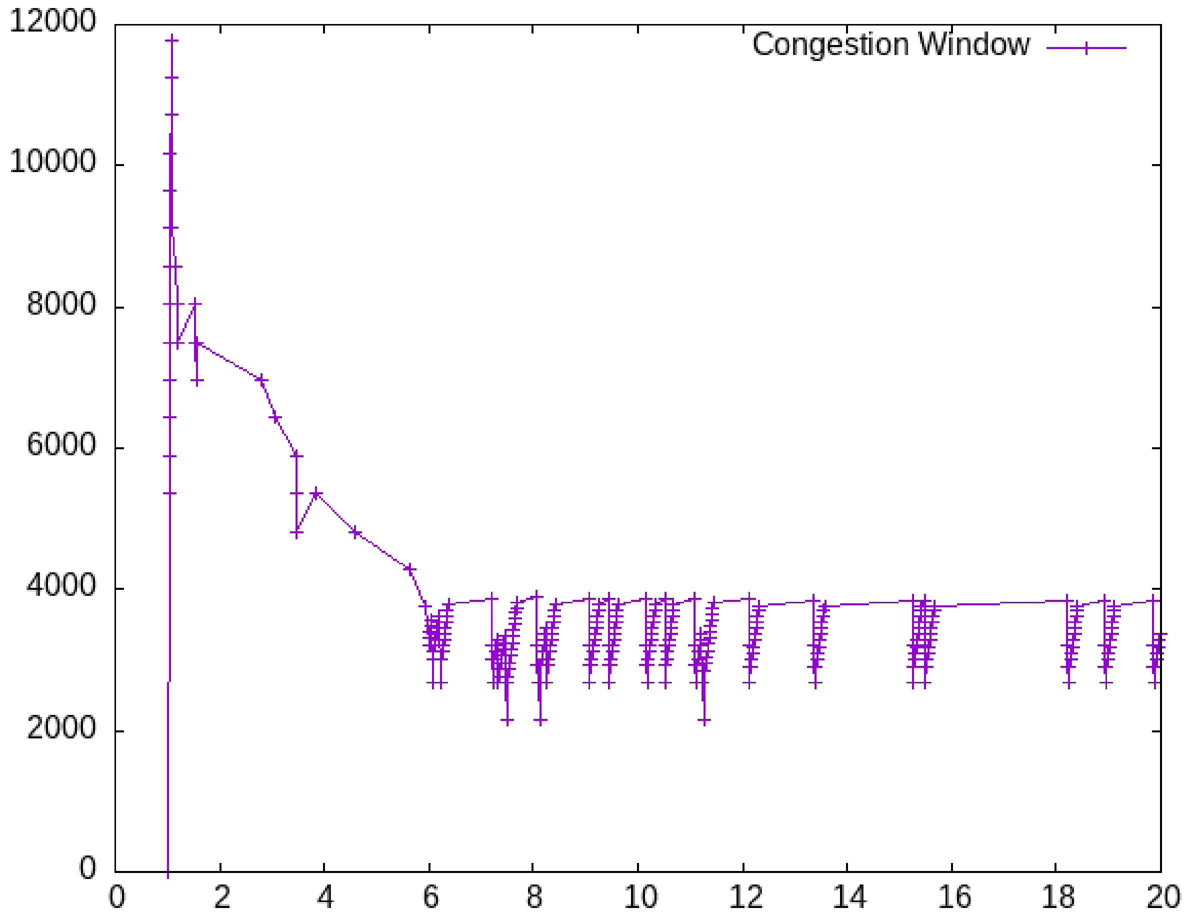


Average Throughput: Total Bytes/ Duration

$$= \frac{4862082}{18.988787}$$

= 256050.1627 bps

2. 4ms

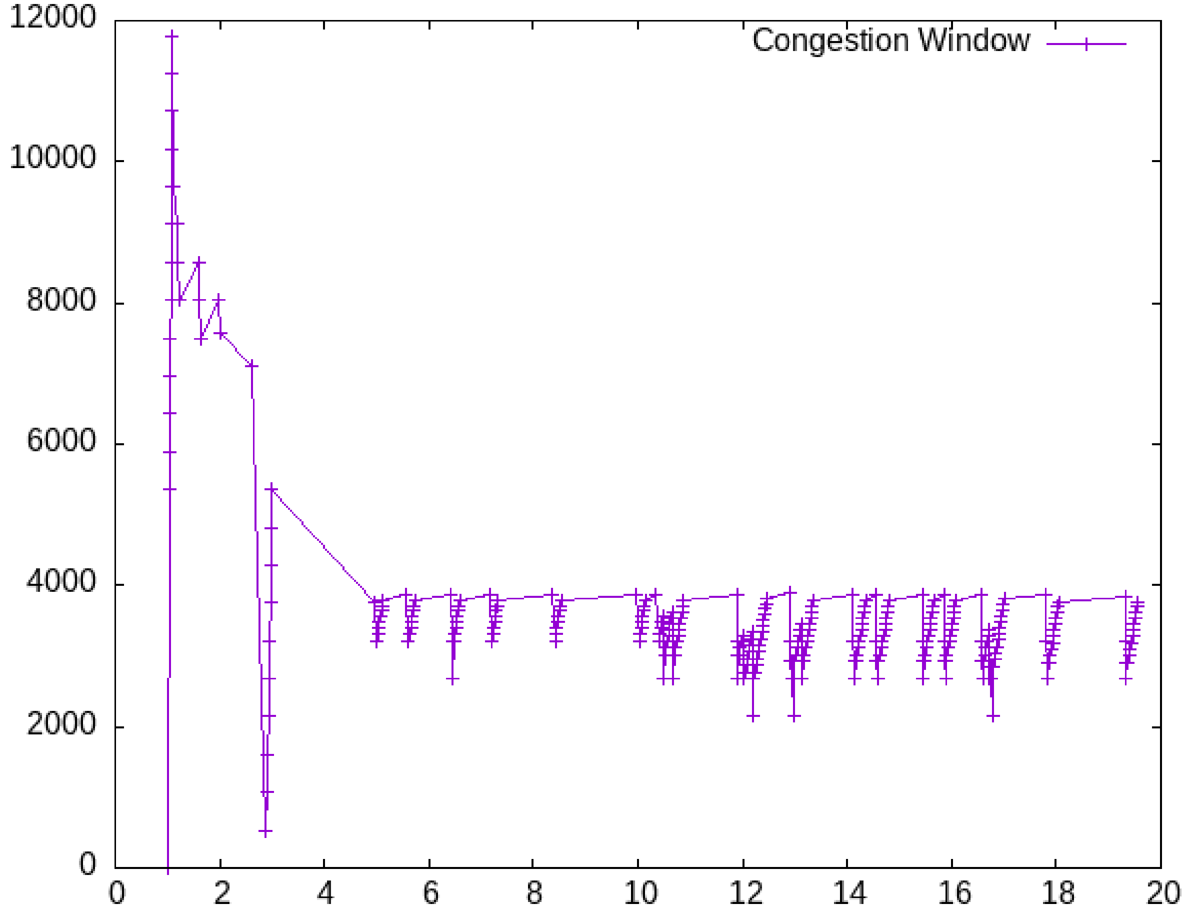


Average Throughput: Total Bytes/ Duration

$$= \frac{1364656}{3.997699}$$

= 341360.3676 bps

3. 6ms

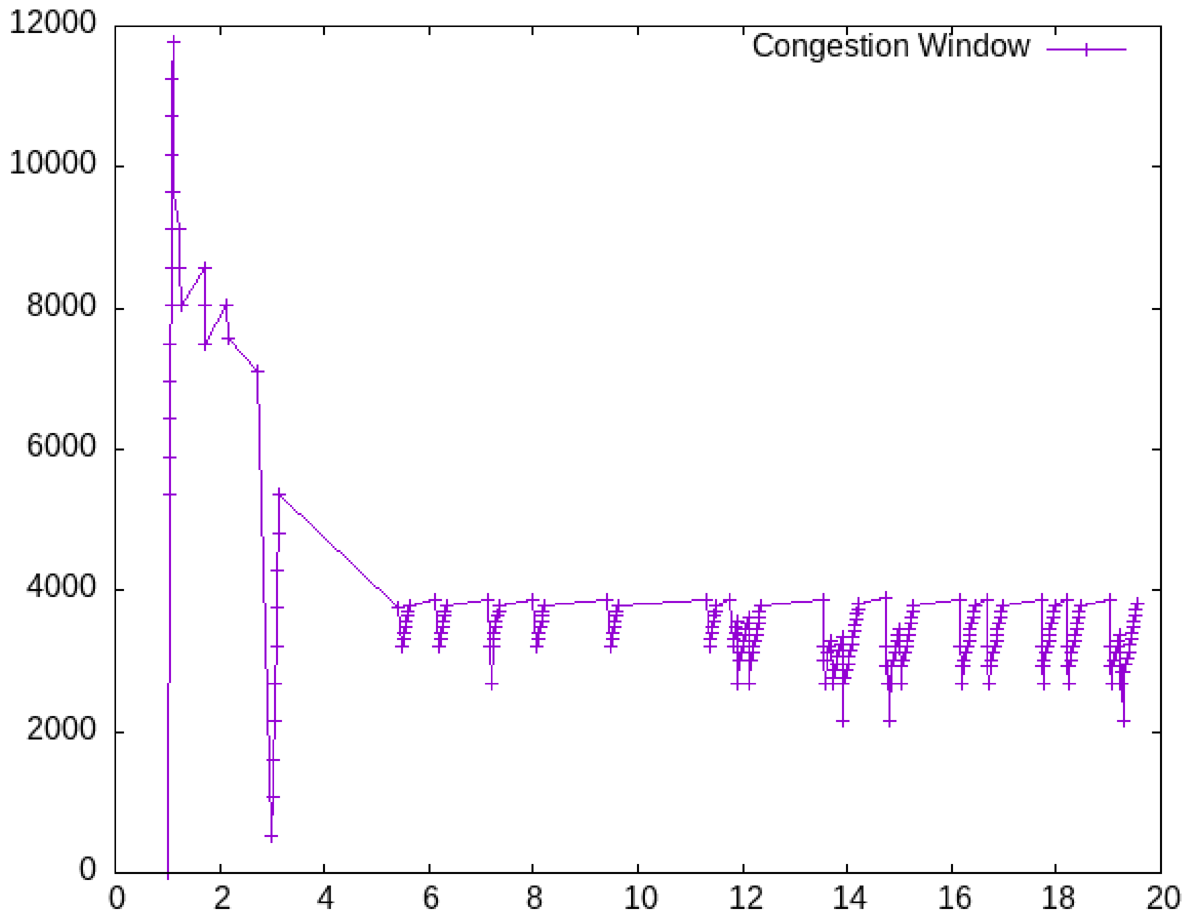


Average Throughput: Total Bytes/ Duration

$$= \frac{3029296}{18.977834}$$

= 159622.8526 bps

4. 8ms

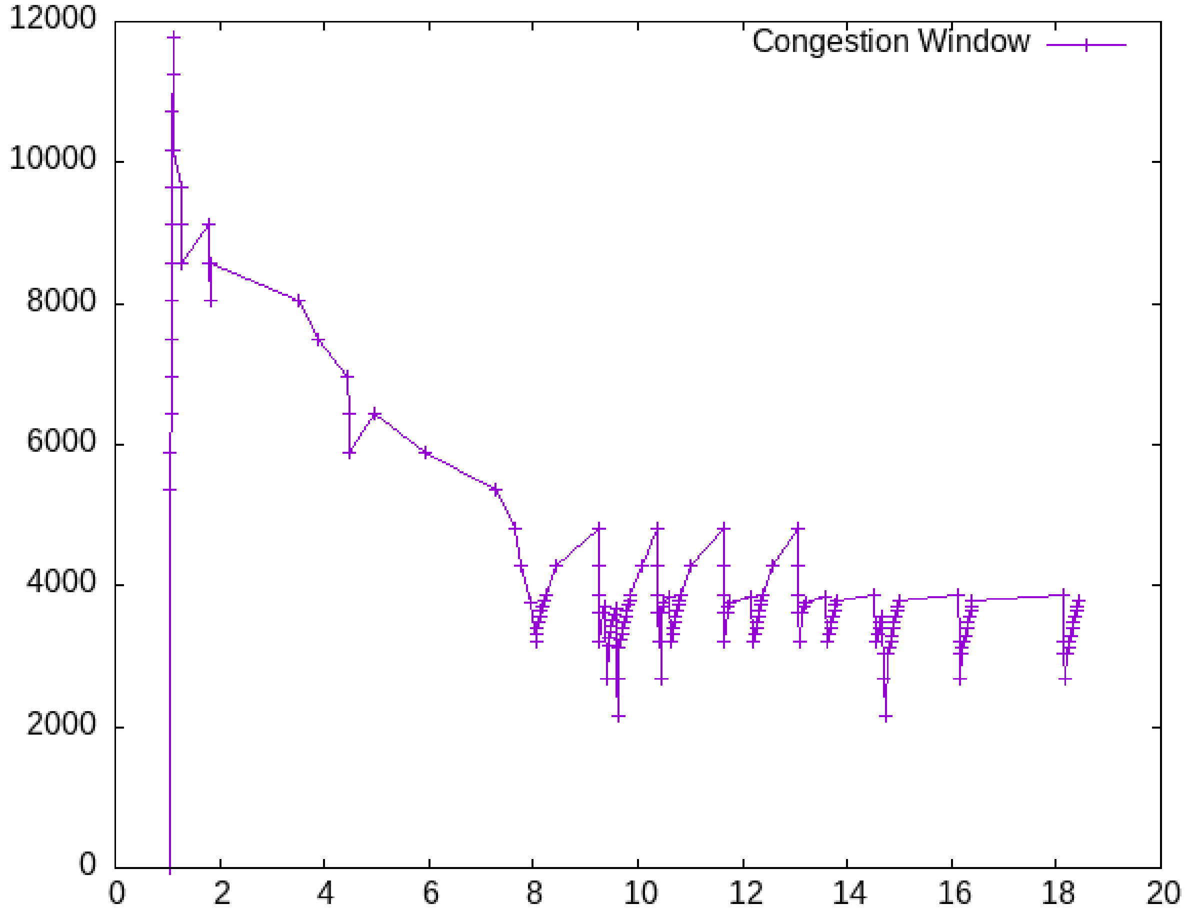


Average Throughput: Total Bytes/ Duration

$$= \frac{2623984}{18.987187}$$

= 138197.6172 bps

5. 10ms



Average Throughput: Total Bytes/ Duration

$$= \frac{1826194}{8.334672}$$

= 219108.0825 bps

The graph is made using the values from 2ms to 10ms of latency2 using the same Python script.