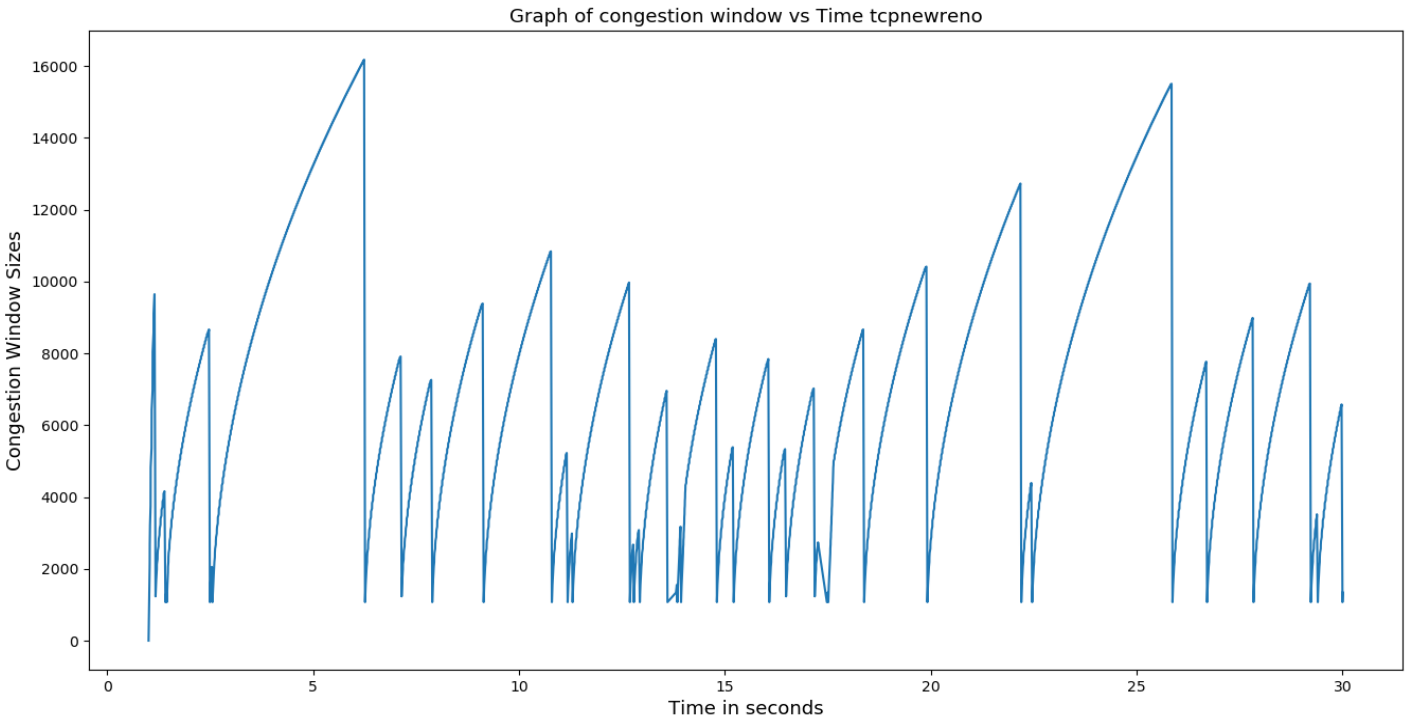
Assignment3:COL672
Entry no: 2021MCS2132
Name: Gaurav Milind Chaudhari
Email: mcs212132@iitd.ac.in

TcpNewReno :
packetLossCount TcpNewReno 38



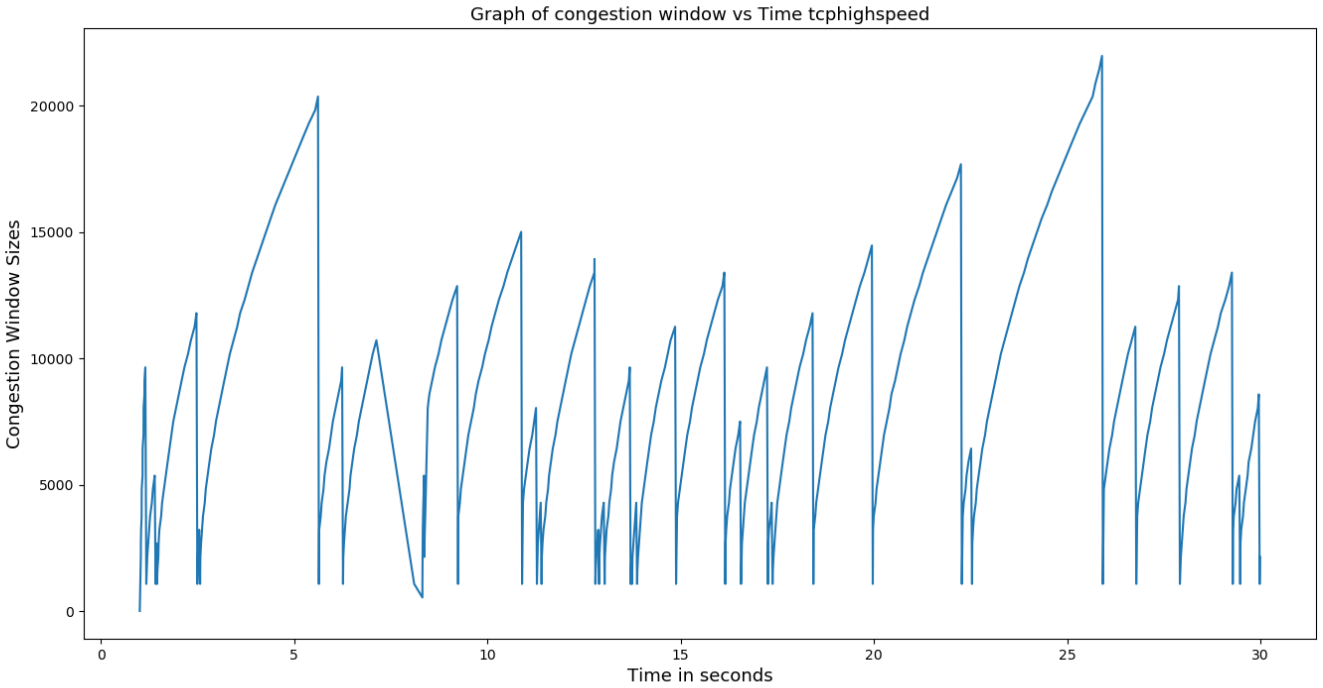
The basic idea of these extensions to the fast retransmit and fast recovery algorithms are as The TCP sender can infer, from the arrival of duplicate acknowledgments, whether multiple losses in the same window of data have most likely occurred, and avoid taking a retransmit timeout or making various congestion window reductions due to such an event.

- Looking at the plot, we observe that the slow start phase is linear in the collision avoidance phase. We get curved output at the end; hence as soon as we experience the congestion window again drop minimum value.
- We also observed that at once congestion detected slope of the congestion window is high when it is low but as it gets high gradient also get lesser anticipating congestion.

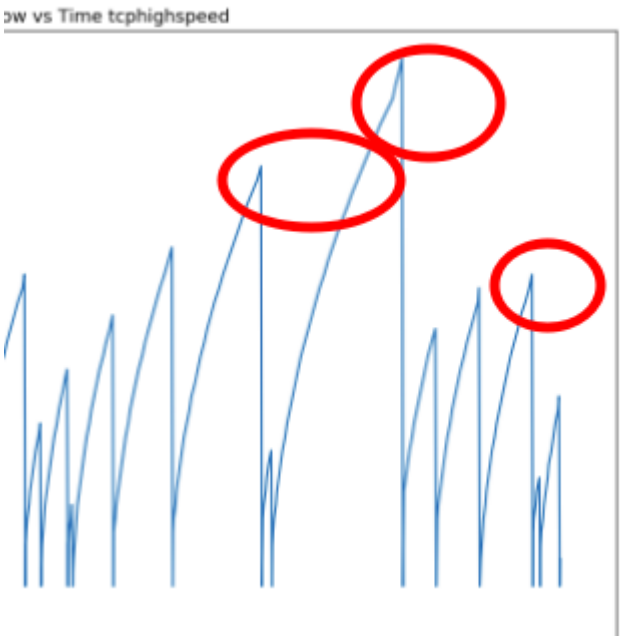
TcpHighSpeed :
packetLossCount TcpHighSpeed 38

Tip highspeed Is a modification of the TCP-Reno congestion control mechanism with TCP connections with large congestion windows. H-TCP is a loss-based algorithm, using additive-increase/multiplicative- decrease to

control the TCP congestion window

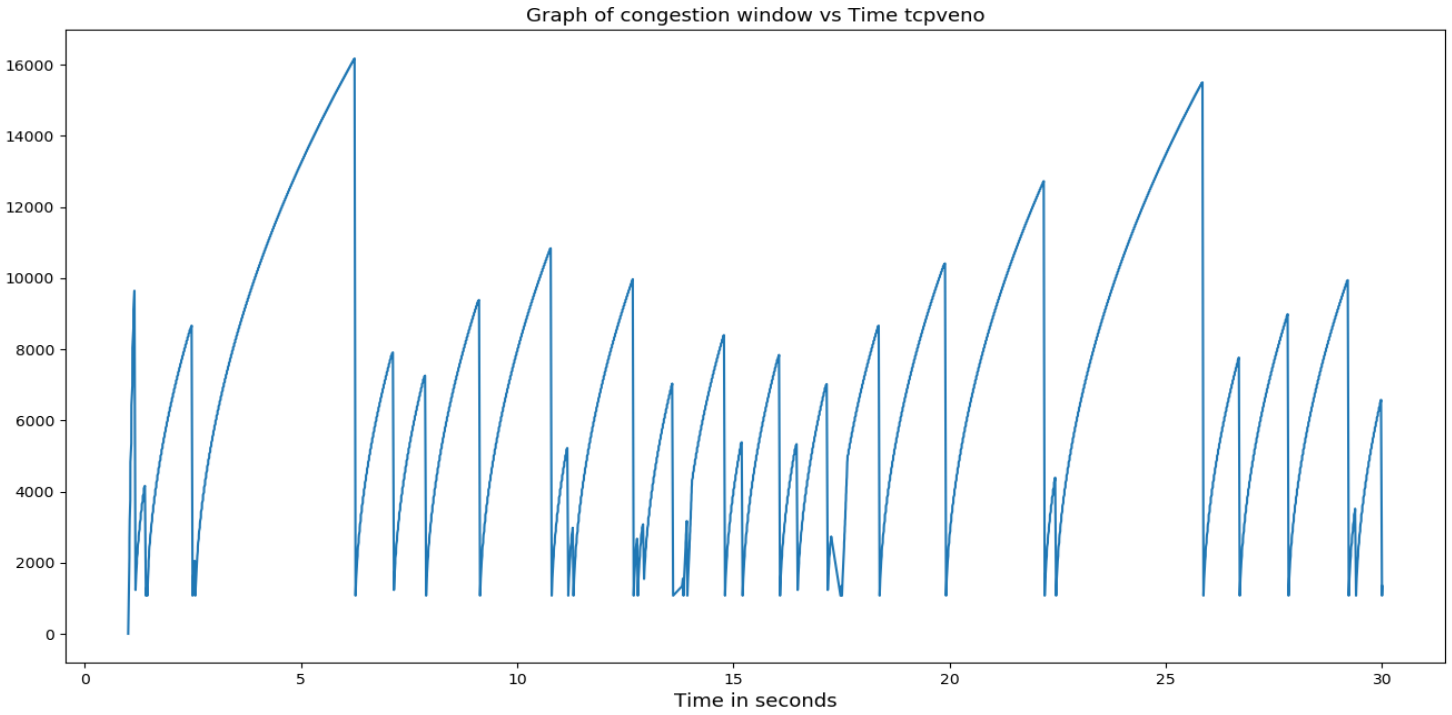


- In the graph, we don't see much difference with New Reno.
- But at the top at TCP acknowledgment loss or timeout, we see there are sharper edges than usual.



TcpVeno :
Packet Loss count TCP Veno is 38

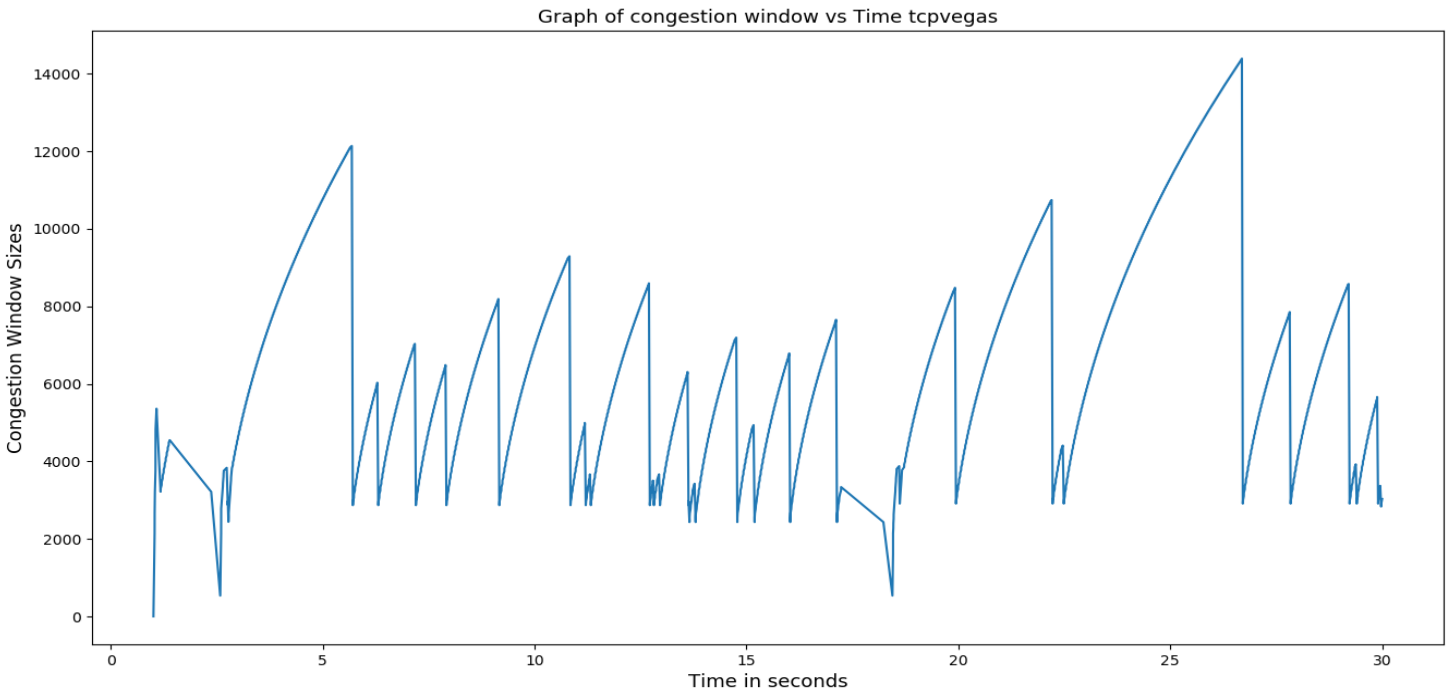
TCP Veno module is a congestion control module to improve TCP performance over wireless networks. It improves over the TCP Reno congestion control algorithm by using the estimated state of a connection based on TCP Vegas.TCP Veno reduces the "blind" reduction of TCP window regardless of the cause of packet loss. This TCP version distinguishes between accidental loss(non- congestion state) and congestion loss (congestion state). Also, depending on this difference, it refines the congestion window adjustment. In the wireless environment, the packet loss is because of the noise and link error.



- If we compare TCP new reno, there is not much difference in the graph, but we can see that curves and lines are much smoother for TCP Veno.

TcpVegas:
Packet loss Count 38

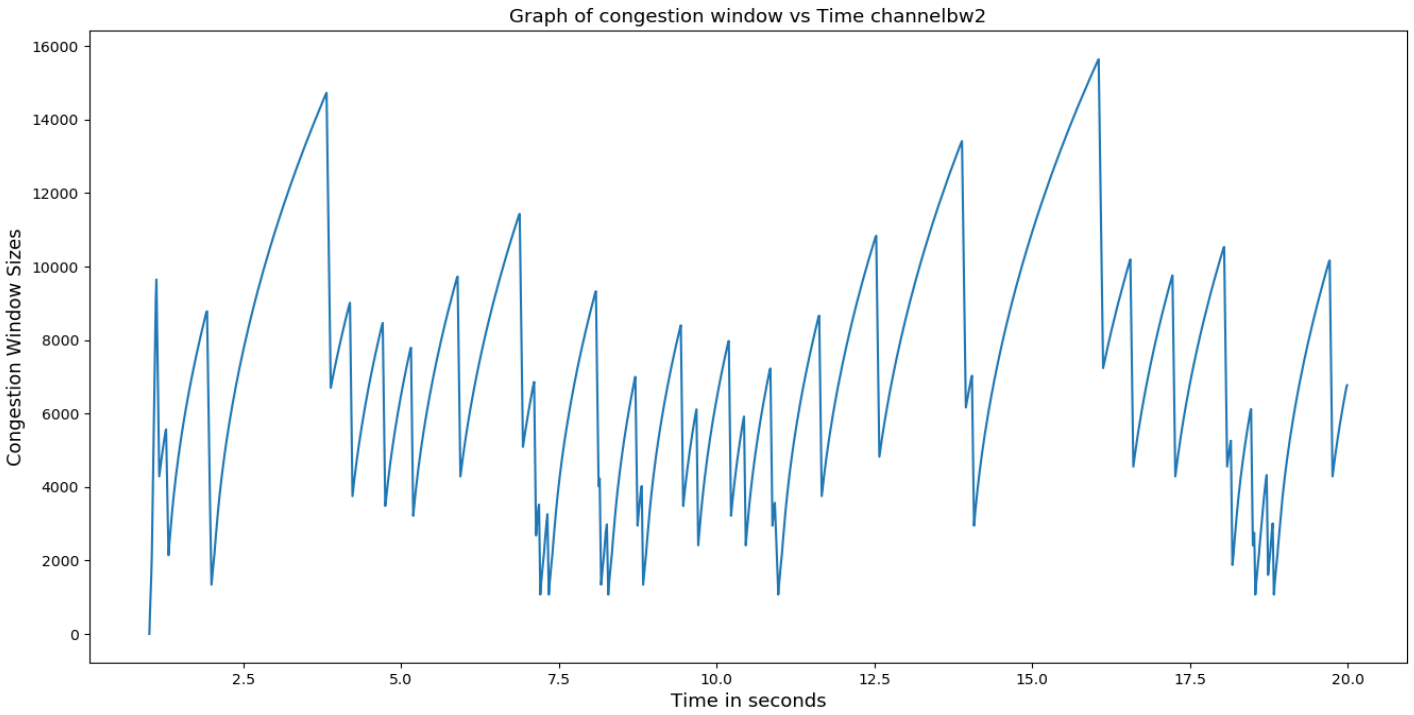
TCP Vegas emphasizes packet delay, rather than packet loss, as a signal to determine the rate at which to send packets. Unlike TCP-Reno, which detects congestion only after it has happened via packet drops, TCP-Vegas detects congestion at a nascent stage based on increasing RTT values of the packets in the connection y l



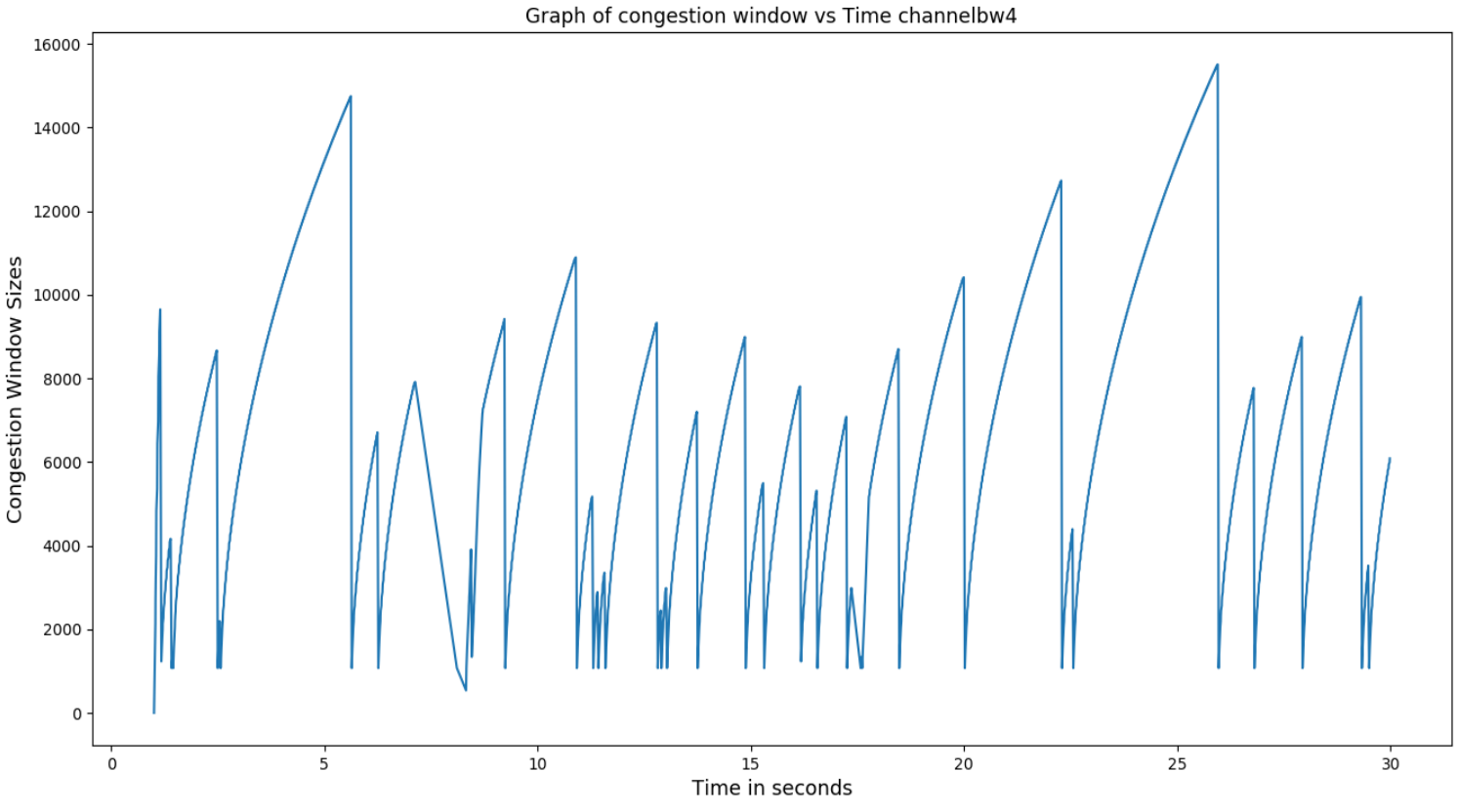
looking at the graph, we can see that the congestion window not dropping below a certain threshold throughout minimum congestion window base is higher as compared to other algorithms

PART 2

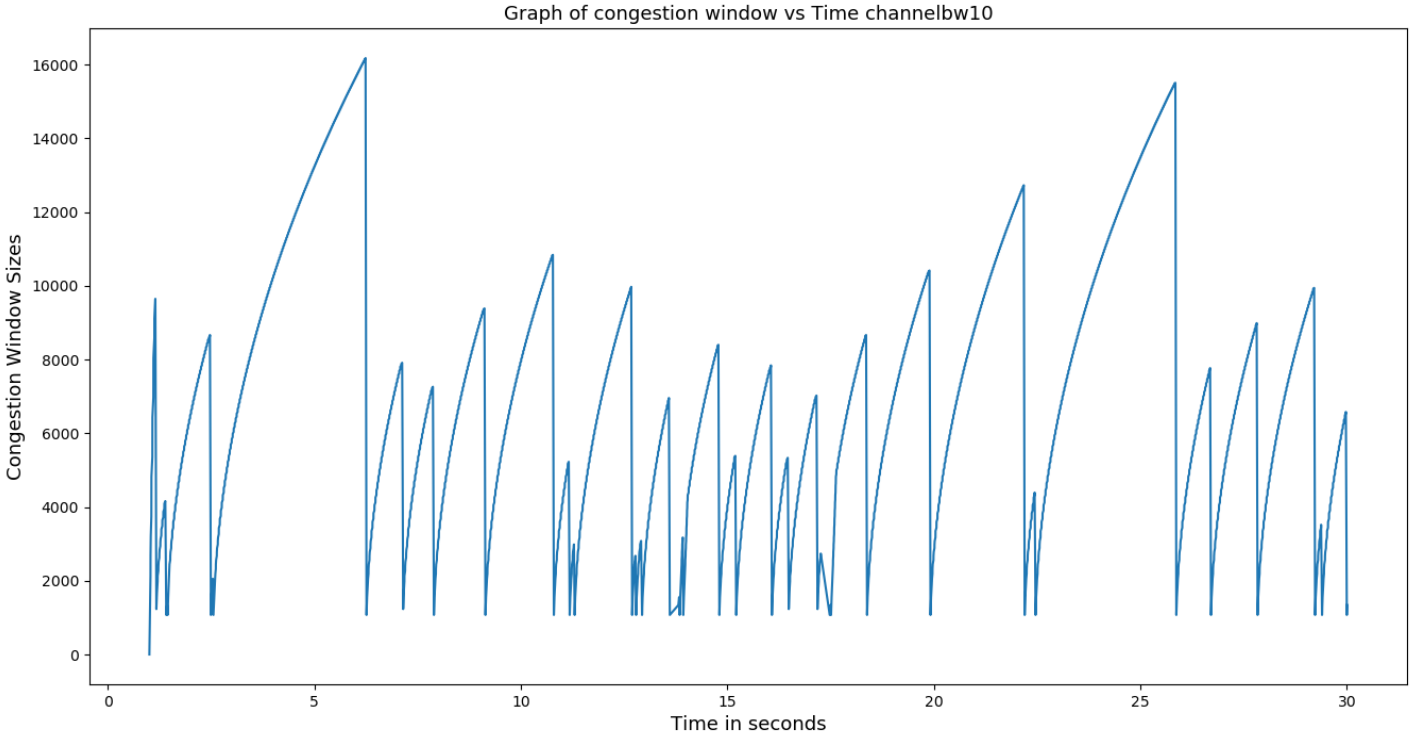
Channel Bandwidth 2Mbps Application Datarate 2Mbps



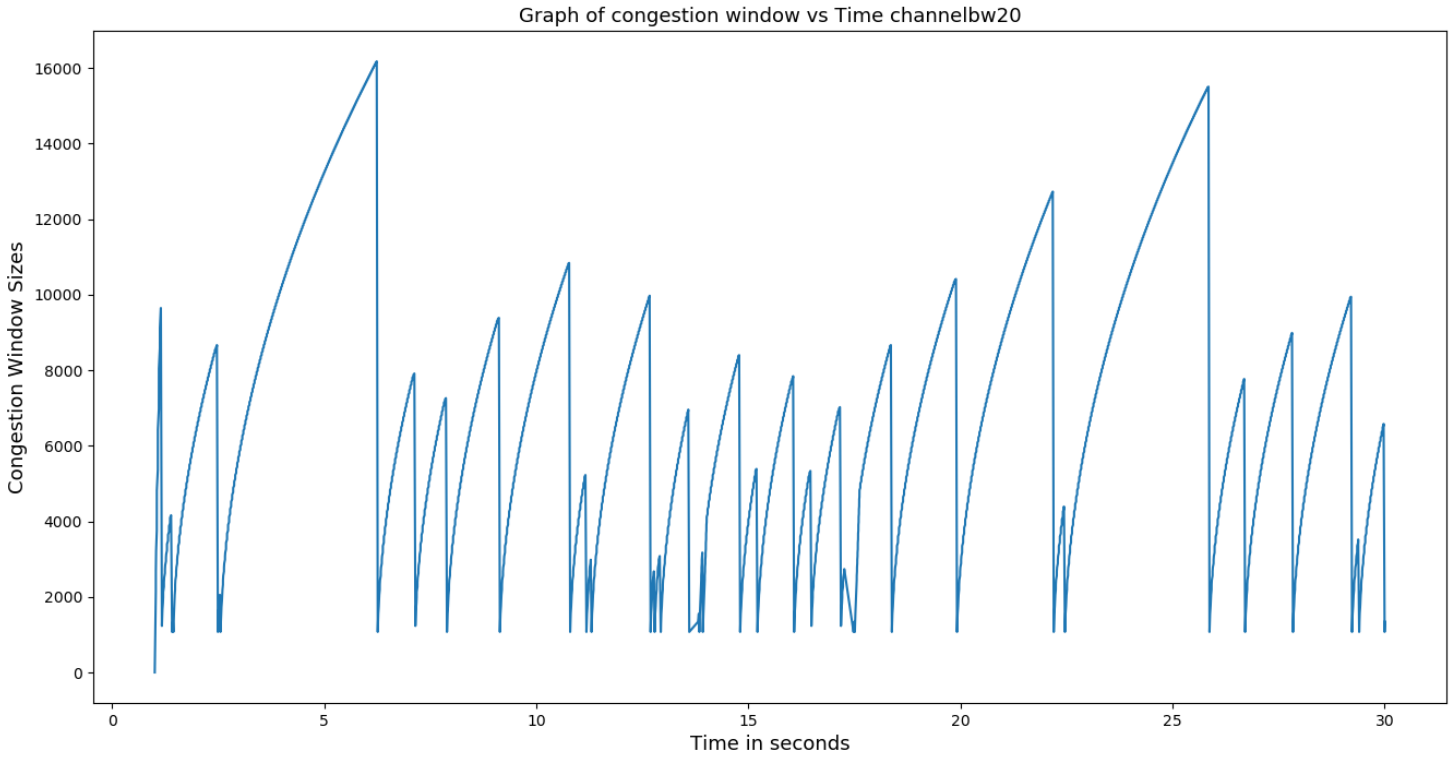
Channel Bandwidth 4 Mbps Application Datarate 2 Mbps



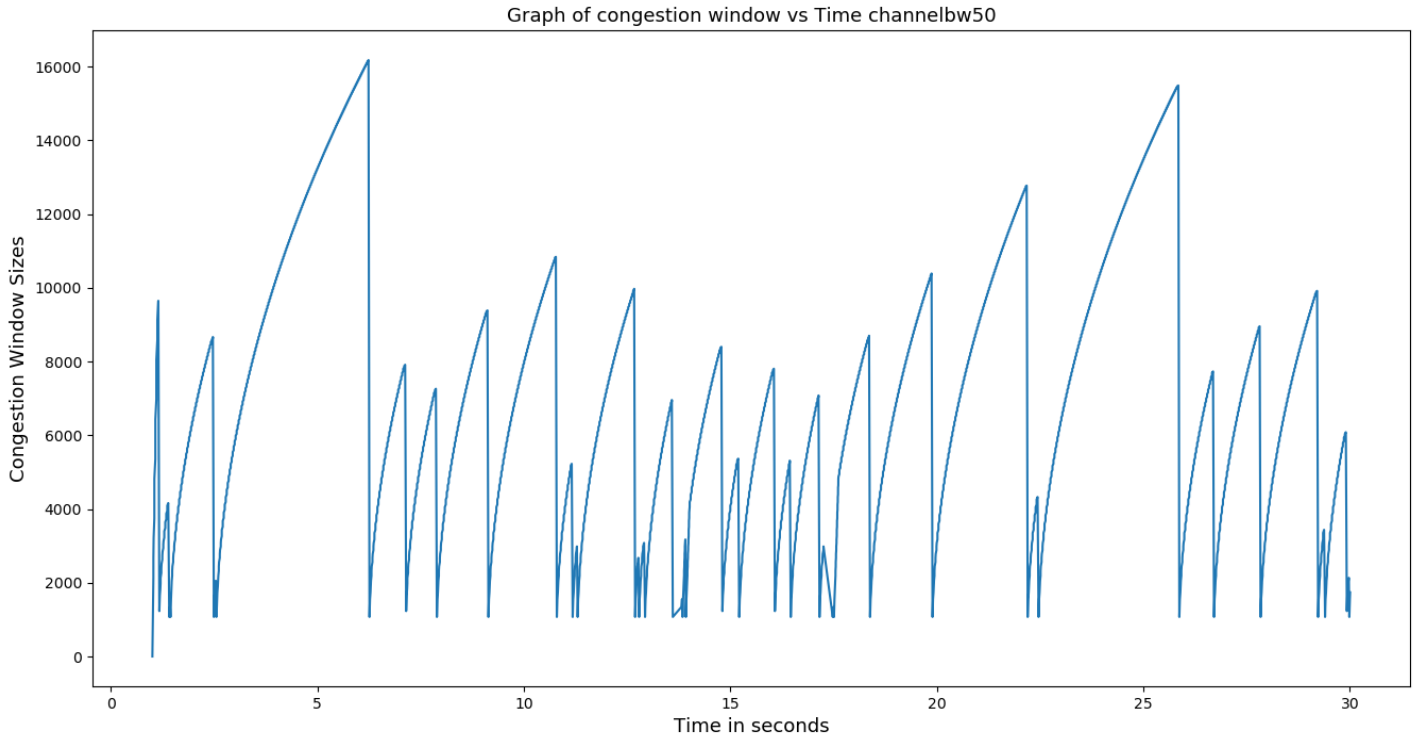
Channel Bandwidth 10 Mbps Application Datarate 2 Mbps



Channel Bandwidth 20 Mbps Application Datarate 20 Mbps



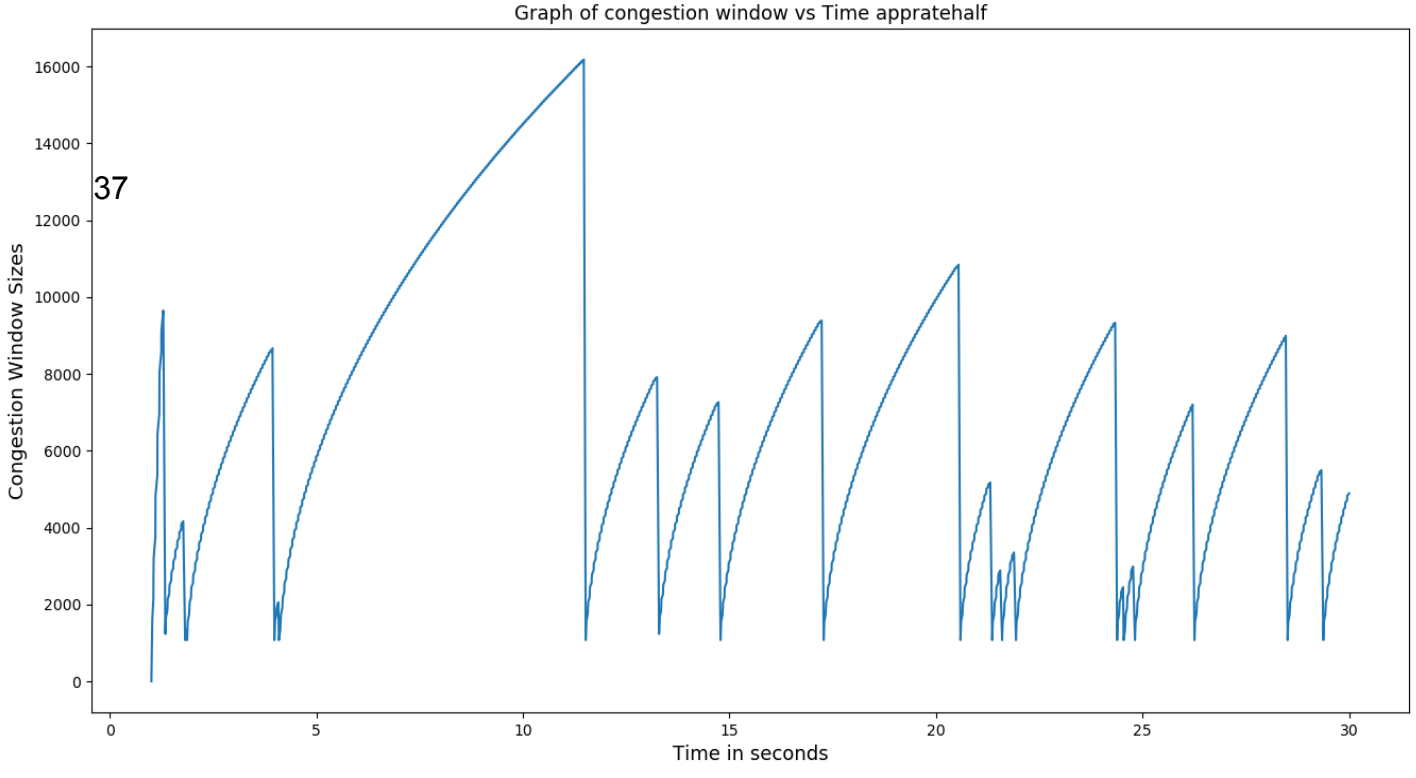
Channel Bandwidth 50 Mbps Application Datarate 50 Mbps



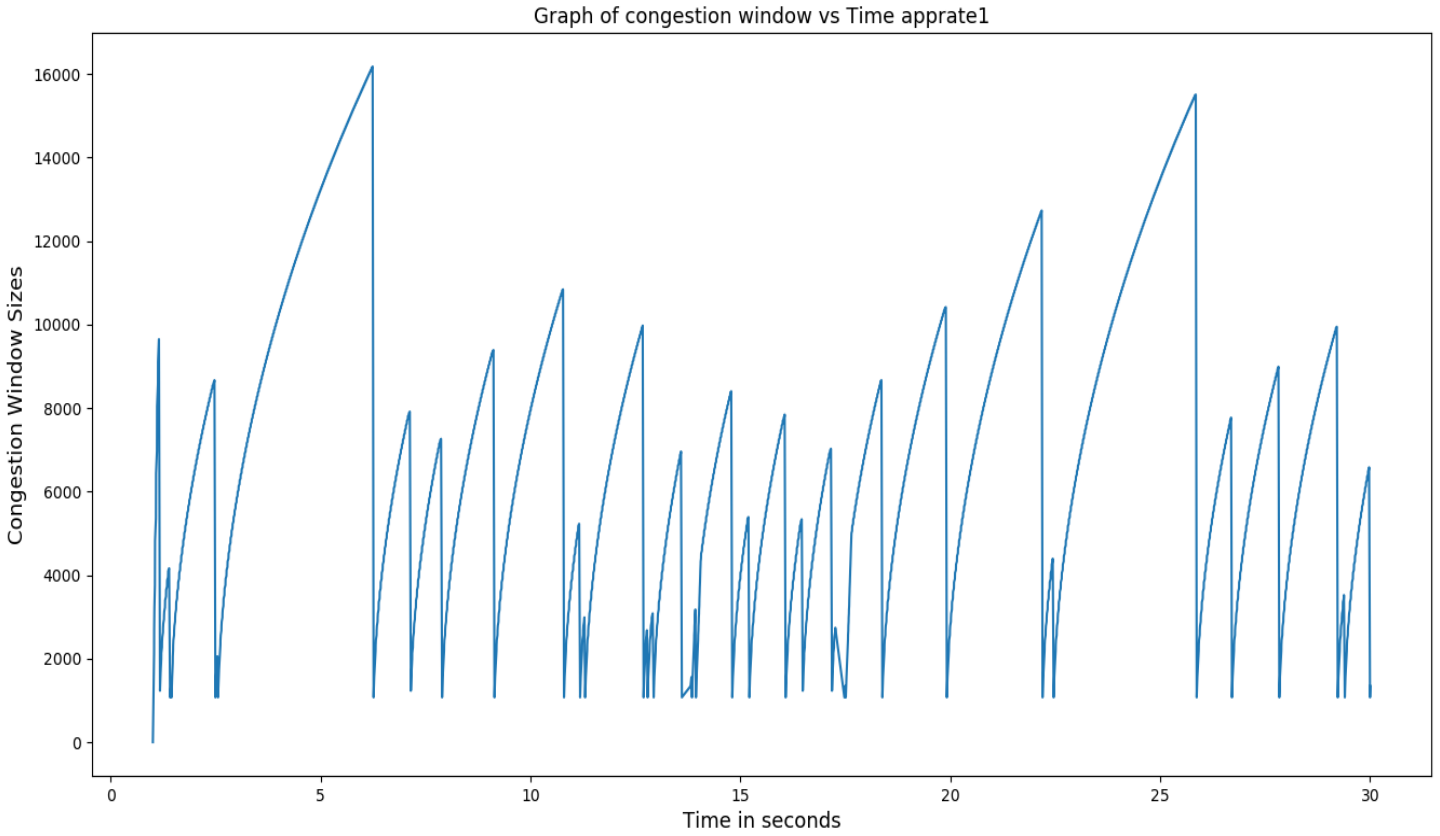
- When channel bandwidth is almost the same with app datarate 2 Mbps, we see a more random variable drop after collision base return.
- 2 Mbps bandwidth we see reached highest congestion window close to 16000 that is
- The highest congestion window throughout the transfer is almost the same for all configurations.
- According to graphs, we can deduce that after some threshold value of bandwidth of channel congestion doesn't get affected by that much margin.

2)

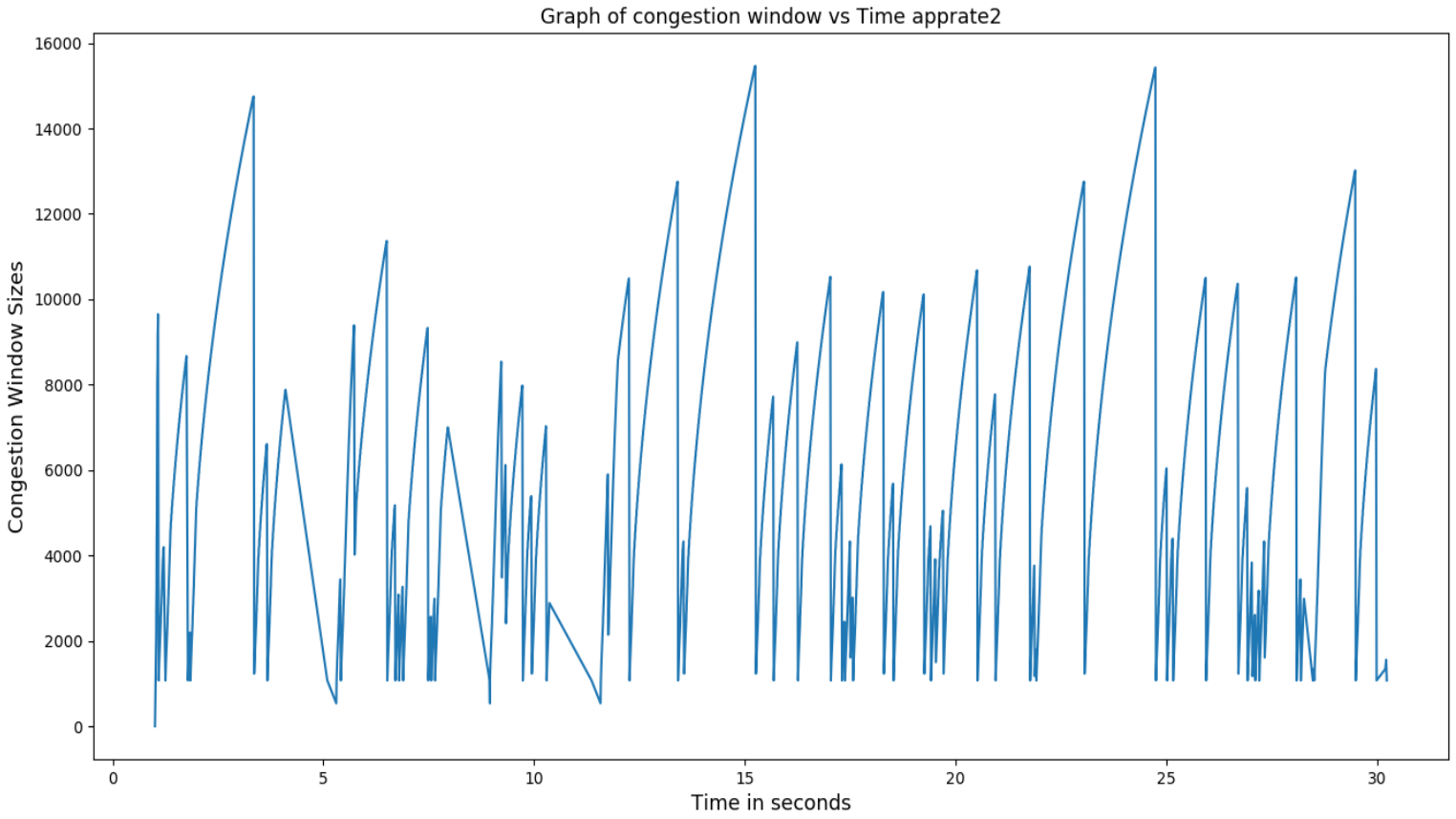
App Datarate 0.5 Mbps Channel Bandwidth 6Mbps



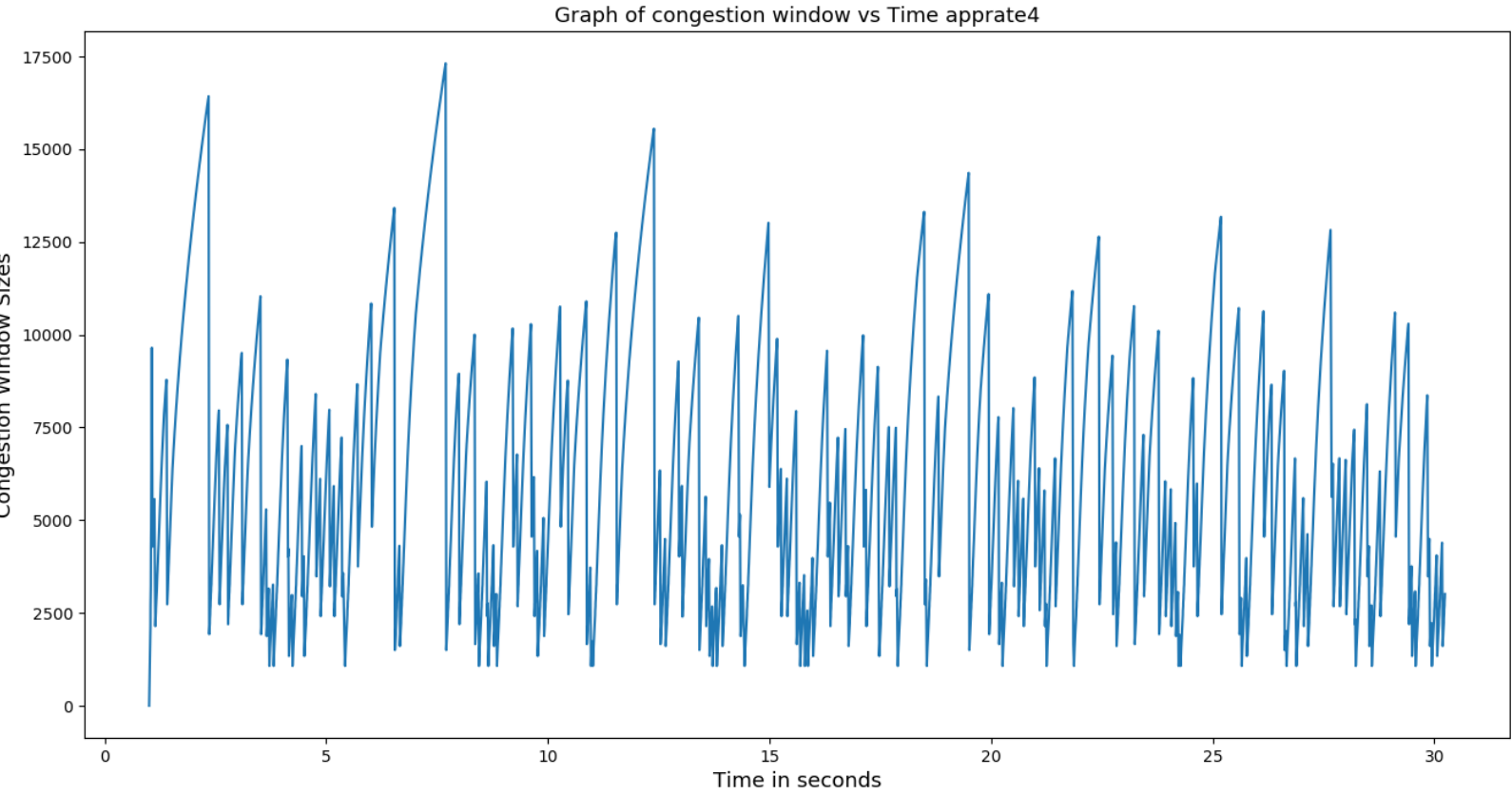
App Datarate 1 Mbps Channel Bandwidth 6Mbps



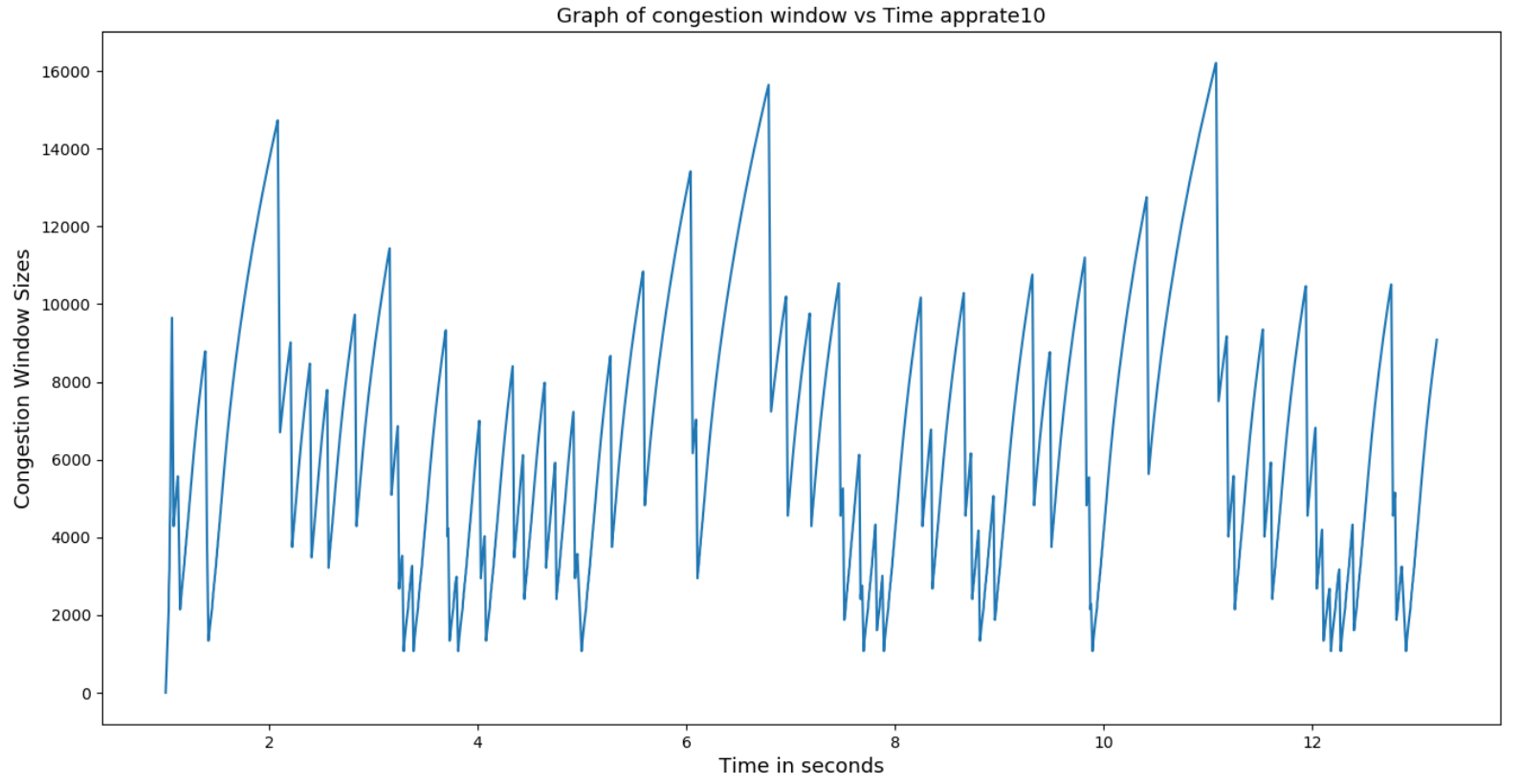
App Datarate 2 Mbps Channel Bandwidth 6Mbps



App Datarate 4 Mbps Channel Bandwidth 6Mbps



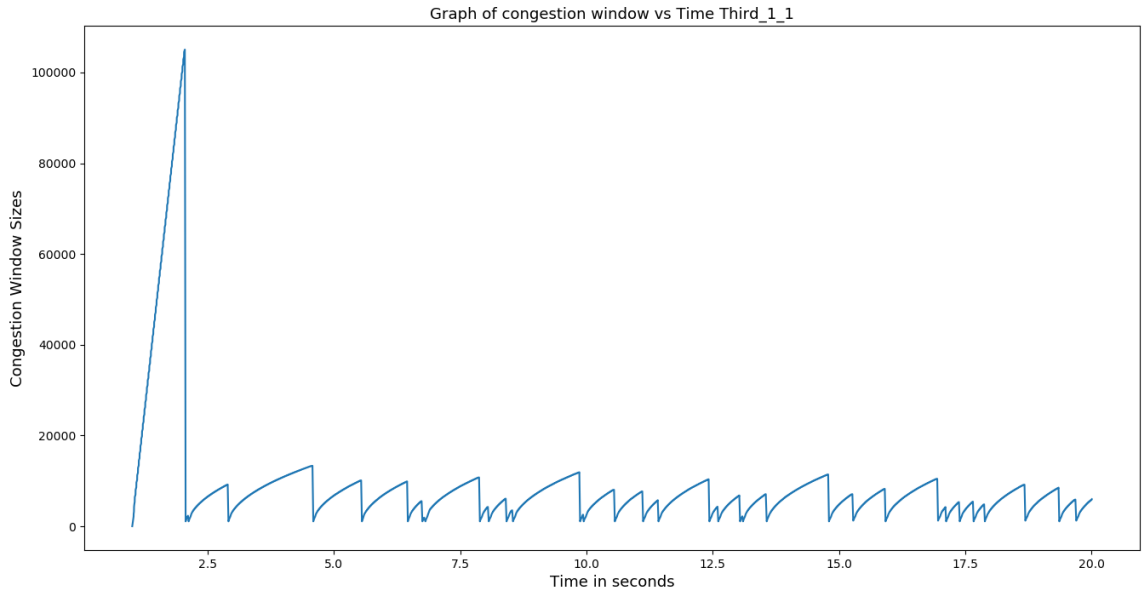
App Datarate 10 Mbps Channel Bandwidth 6Mbps



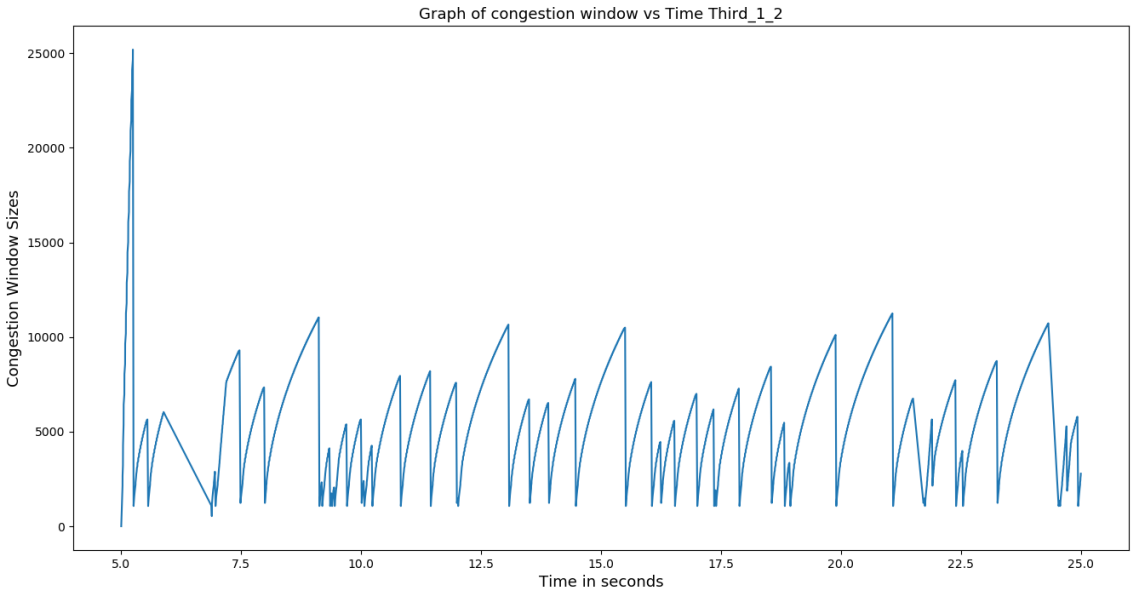
- By looking at graph trends we observe that as we keep as the application data rate increases we see number of congestion occurrences also increasing
- For high app data rate congestion window size varies more than compared with lower app datarate
- Congestion window for datarate more than bandwidth becomes unstable and we can see congestion window size not always decreasing to one

PART 3

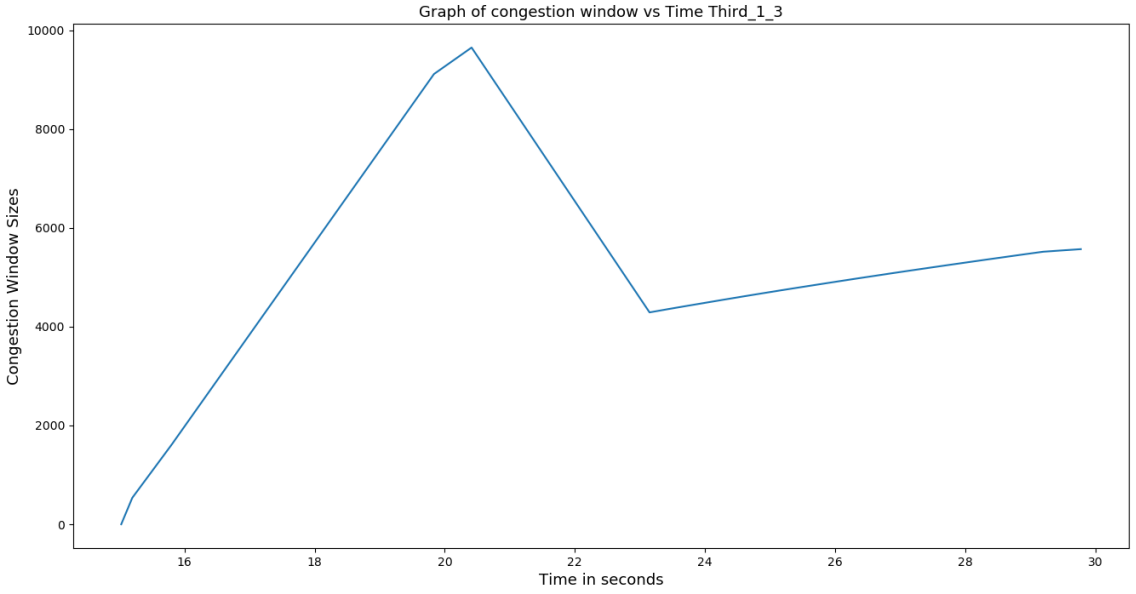
TcpNewReno connection 1



Tcp NewReno connection 2

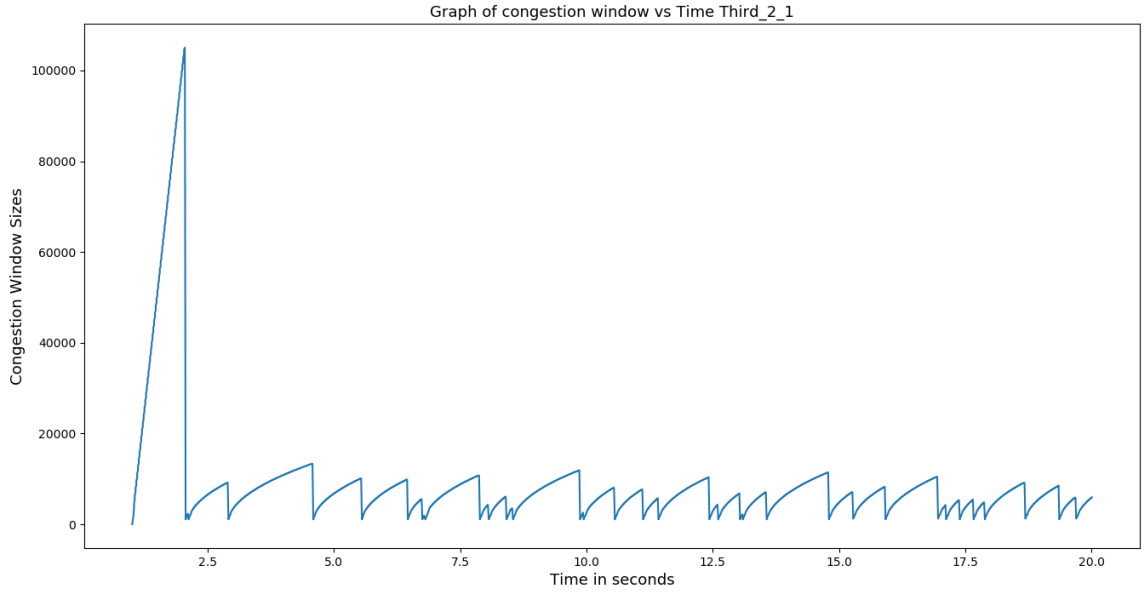


TCP New reno connection3

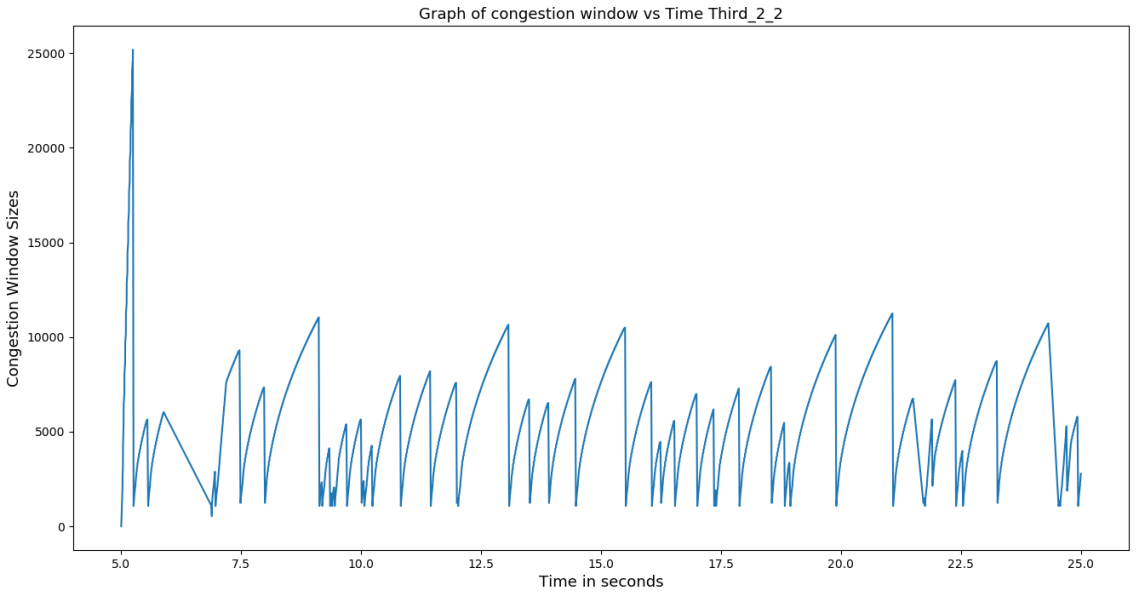


2 :: Connection 3 is TCPNewRenoCSE and others are TcpNewReno

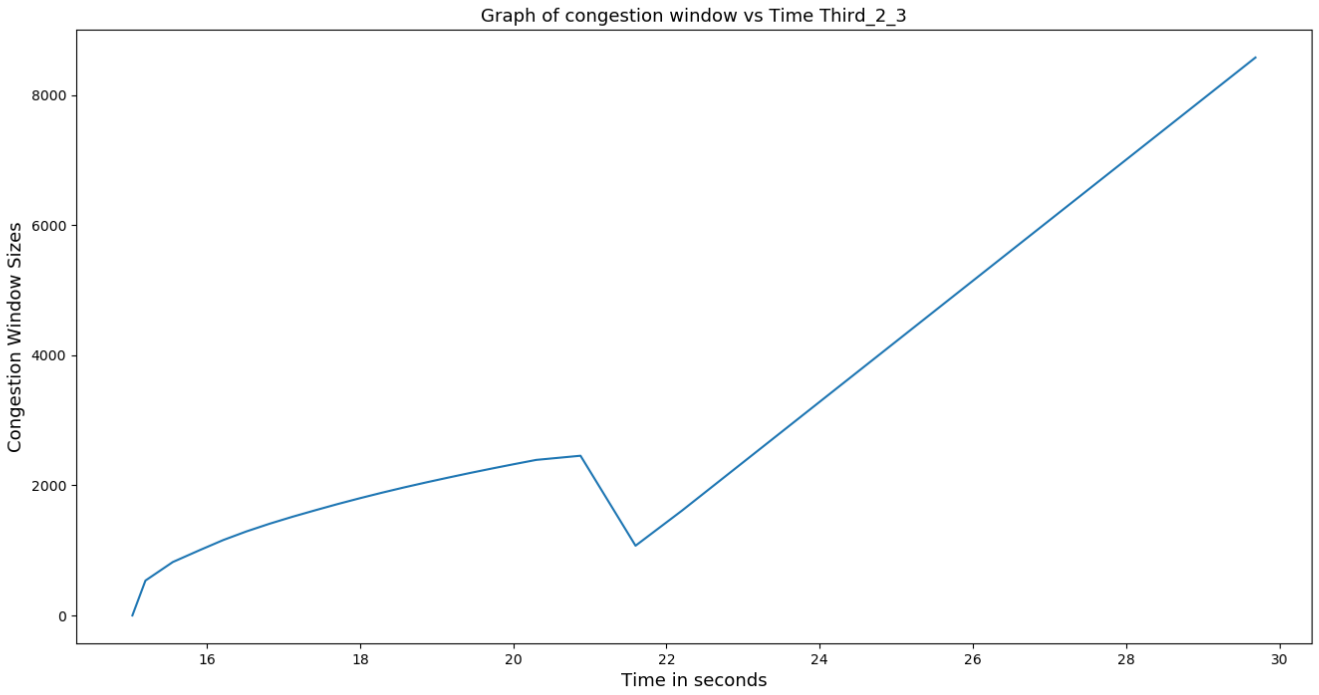
Connection 1 TcpNewReno



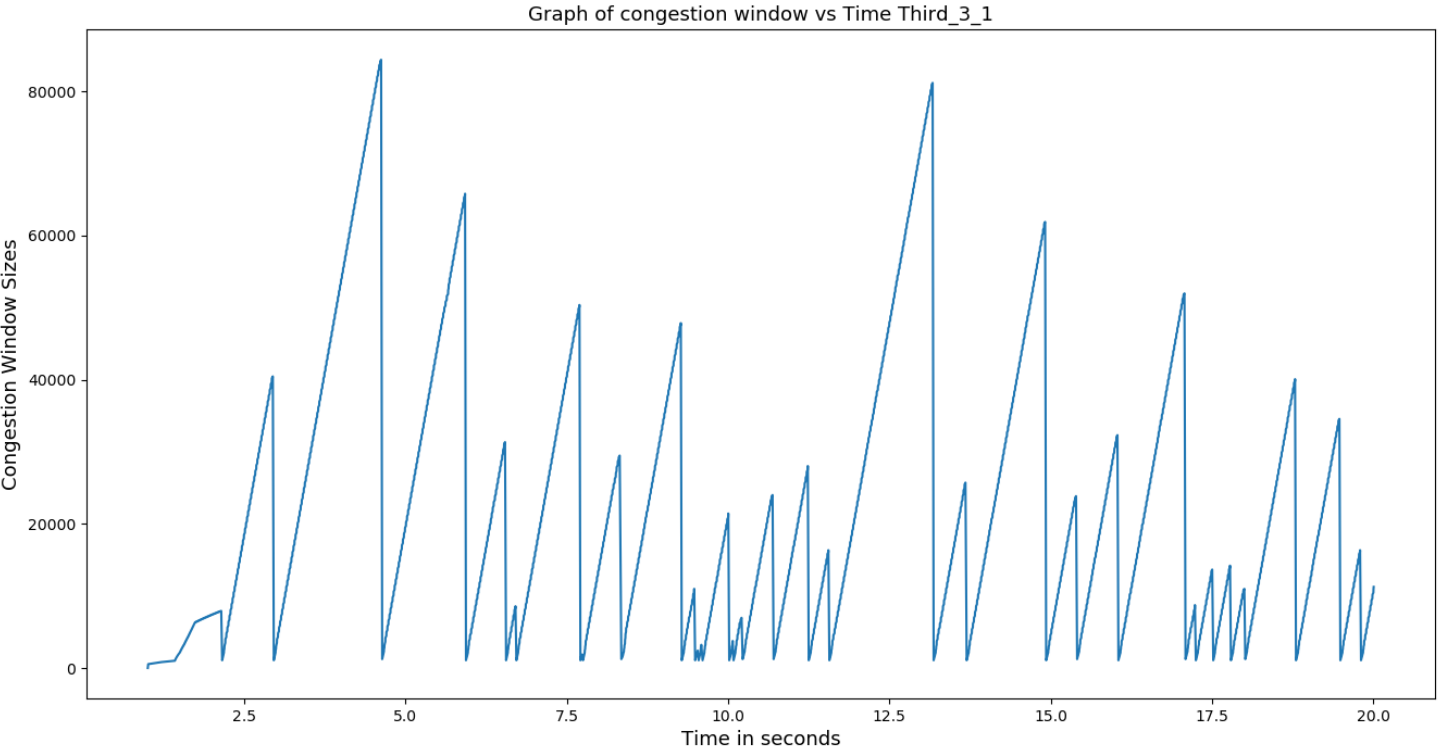
Connection TcpNewReno



Connection 3 Tcp NewRenoCse

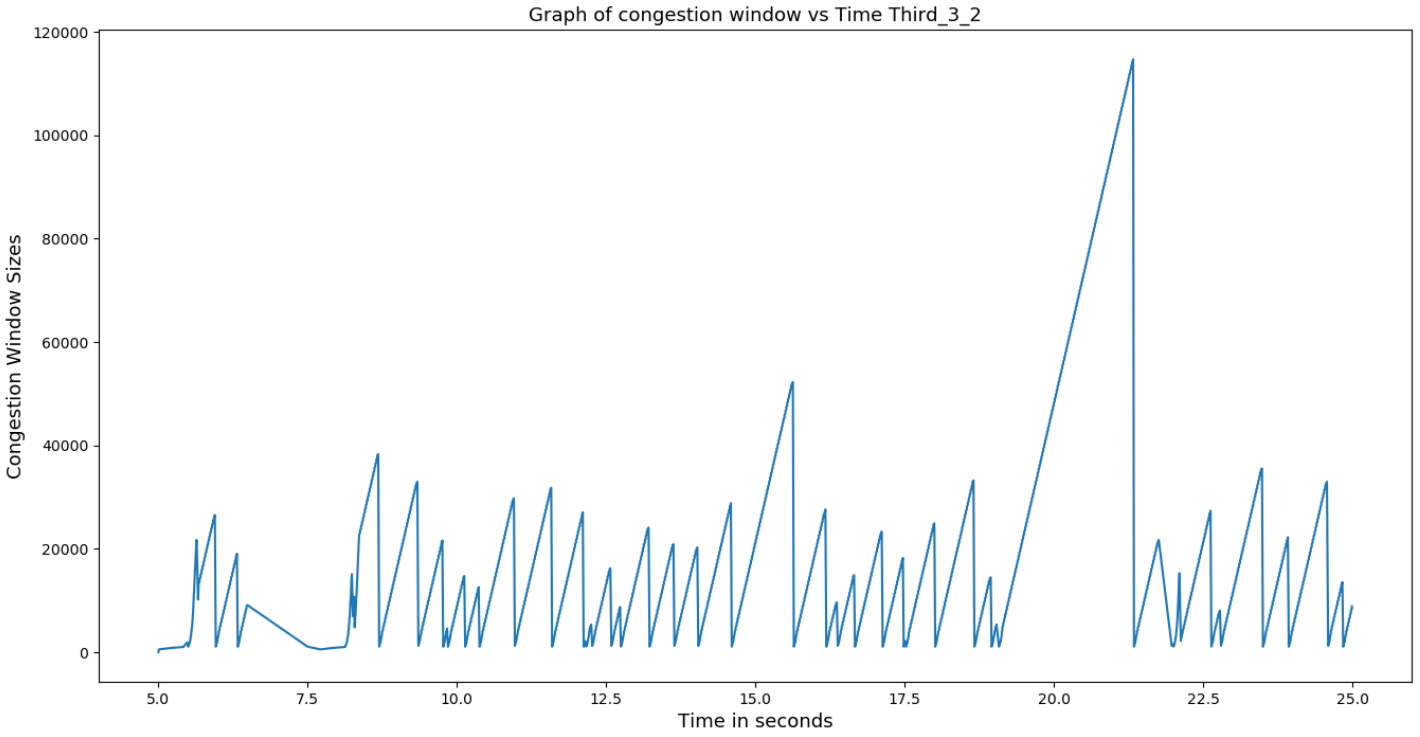


3:: all are TCPNewRenoCse

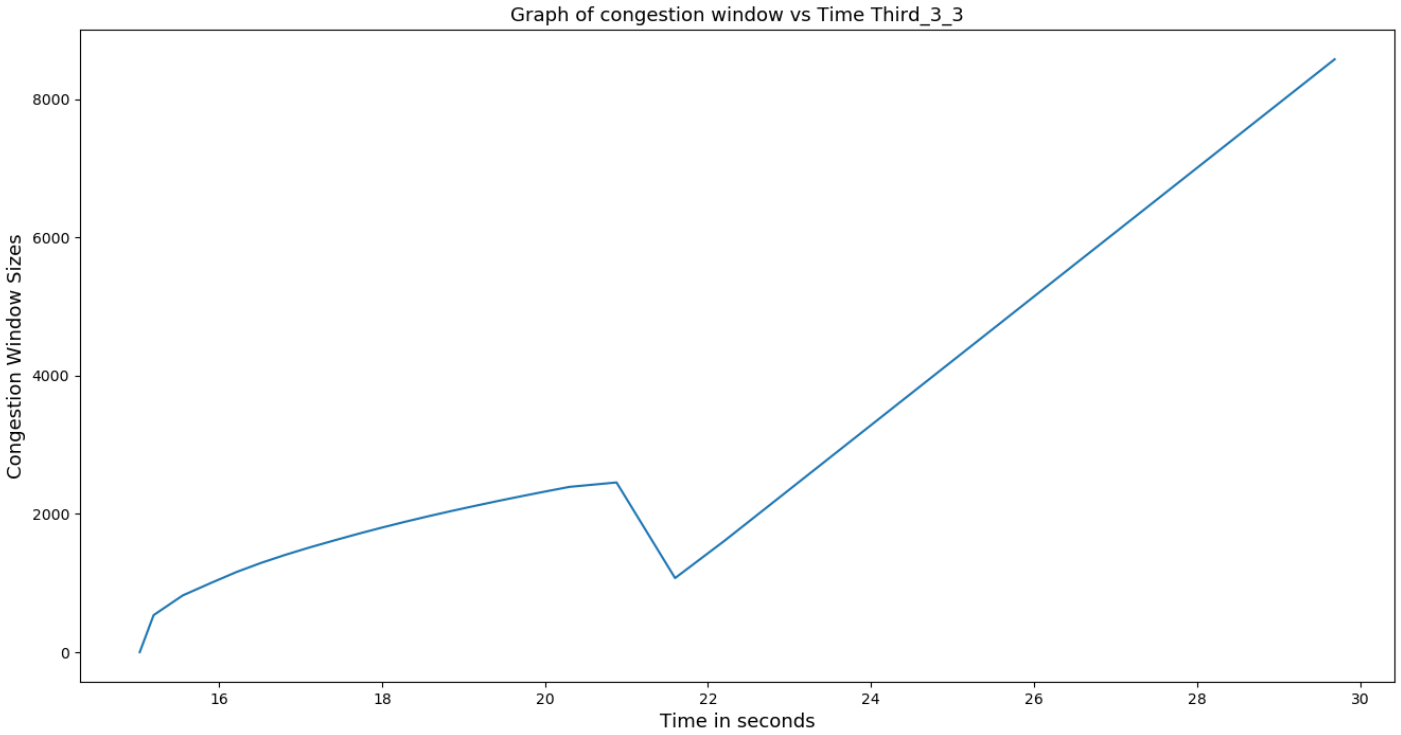


Connection 1::TcpNewRenoCse

Connection 2 ::TcpNewRenoCse



Conection3 TcpNewRenoCse



- We can see the colligion avoidance phase varies for TcpNewReno and TcpNewRenoCse
- Tcp NewReno avoidance phase is curved while for TcpNewRenoCse is a straight line this reason because we interchanged part of tcpnewreno collision avoidance and slow start phase
- TcpNewRenoCse when used alongside TcpNewReno it does not affect other connection congestionwindow it is similar to tcp new reno congestion control window

First_1 ……..First_2
File contains for1st question contains
1. Newreno
2. Highspeed
3. Veno
4. Vegas

Second_1...Second_10 conatins script for 2nd part of assingmets it contains each different datarate of channel or app data

Third_1….Third_3 contains the implementation of 3rd part of assignment

TcpNewRenoCse.cc TcpNewReno.cse.h

I added them in ns-allinone-3.29/ns-3.29/src/internet/model
And added 'model/TcpNewRenoCse.cc',
  'model/TcpNewRenoCse.h',

In wsscript file to link with other header file

Python matplot lib is used to plot the graph