Assignment – 3

COL334

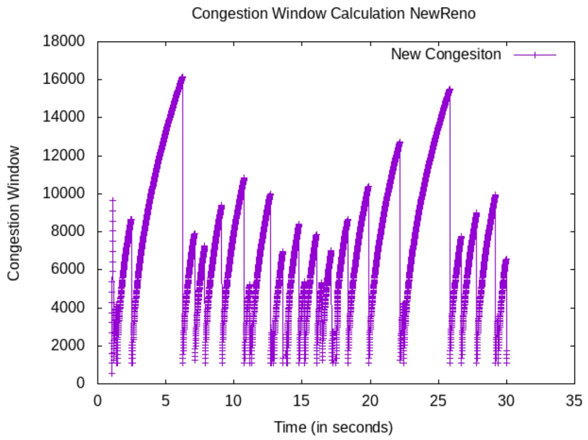
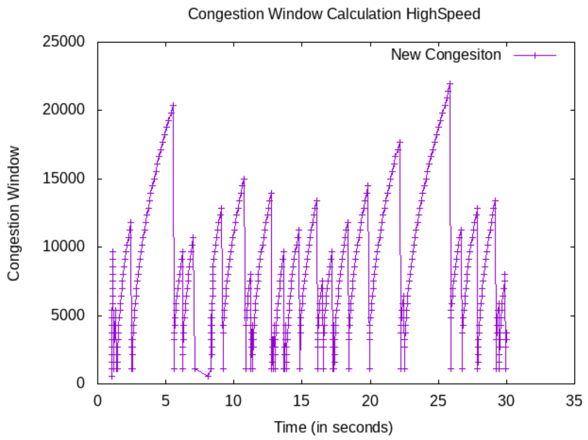
Anirudha Kulkarni

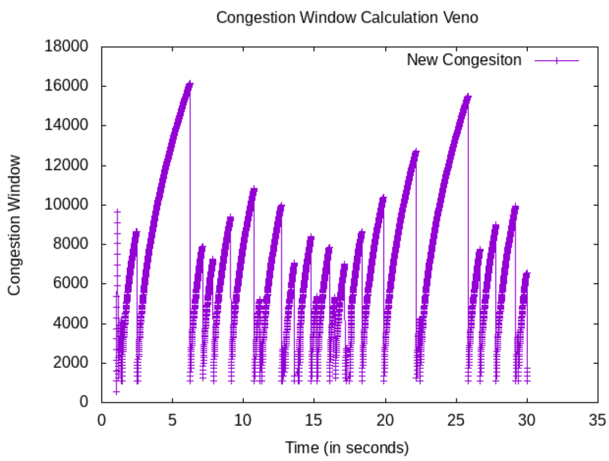
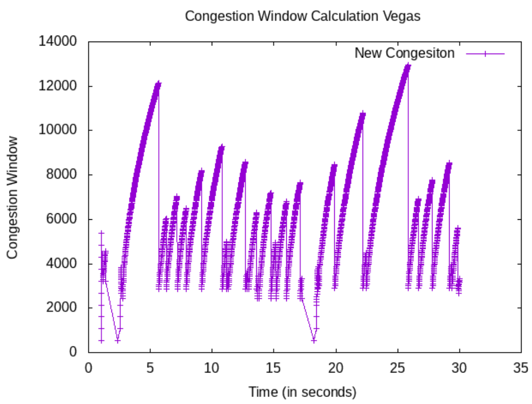
2019CS50421

Instructions:

1. run q1.sh q2.sh q3.sh to run 1st,2nd,3rd question respectively. Plots are plotted by gnuplot with script provided in respective directories.

Q1

<table>
<tr><td>Protocol</td><td>Packets dropped</td><td>Plot</td></tr>
<tr><td>NewReno</td><td>38</td><td>
</td></tr>
<tr><td>HighSpeed</td><td>38</td><td>
</td></tr>
</table>

<table>
<tr><td>Veno</td><td>38</td><td>
</td></tr>
<tr><td>Vegas</td><td>39</td><td>
</td></tr>
</table>

Observations:

We see almost similar packet losses in all the 4 protocols. We find that HighSpeed is with largest congestion window and hence has fastest recovery phase. Veno is derived from NewReno and hence we see almost similar graphs with around 10 different congestion window sizes. We also a stop in packet transmission at t=17 seconds in vegas.

Max value reached is 1600, 2200, 1600, 1350 respectively in each of the protocols.

The SMSS is the size of the largest segment that the sender can transmit.

Newreno:

- Congestion avoidance – congestion window is increased by 1 full sized segment
- Slow start – congestion window is increased by min(number of bytes unacknowledged, maximum segment size)
- We can see that till ssthresh value the points are separated by much larger distance that is jump is higher. But when ssthresh is reached points are close as the increment is by 1 – fixed.

HighSpeed:

- Designed for high capacity channels, with large congestion windows.
- Cwnd grows much faster and accelerates from recovery faster.
- We see much higher jumps in the plot. Less noisy compared to others
- The average window size is higher compare with any other protocol

Vegas:

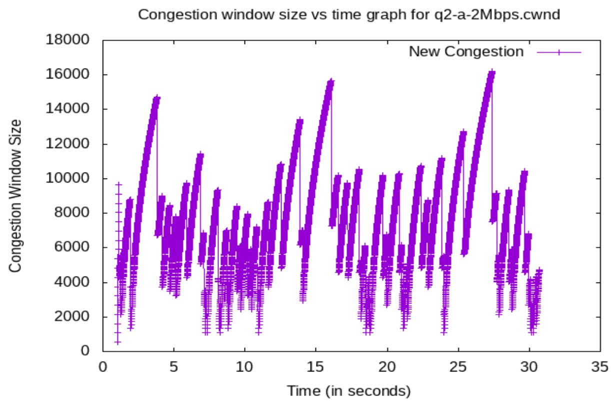
- Delay based congestion control protocol. Expected and actual throughputs are calculated to get the congestion queue at bottleneck and accordingly cwnd is adjusted
- Linear increase and linear decrease
- Plots show lowest average cwnd size

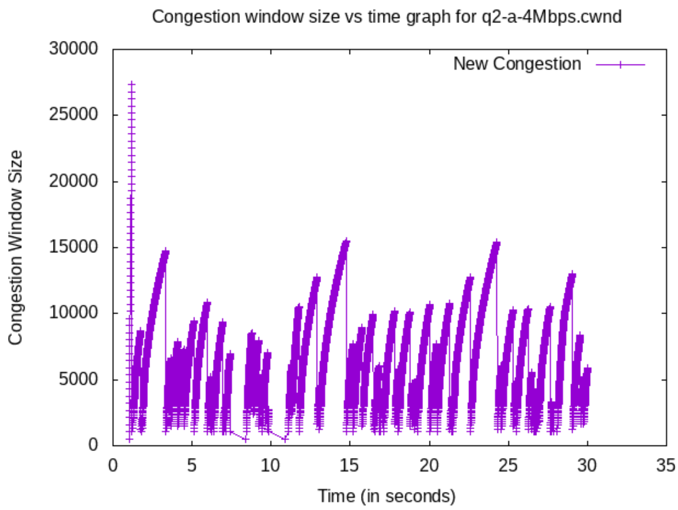
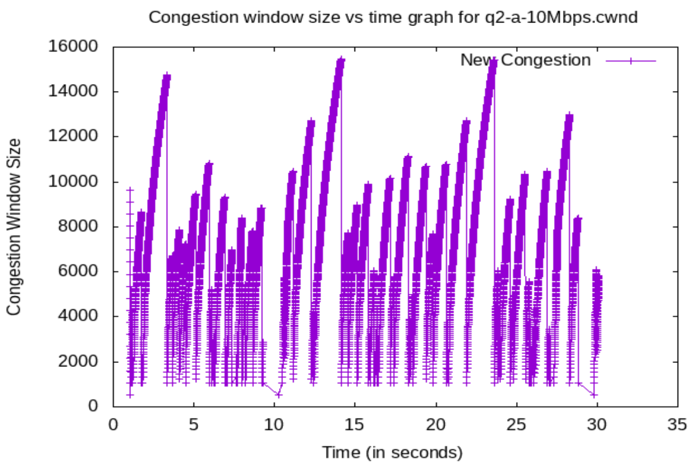
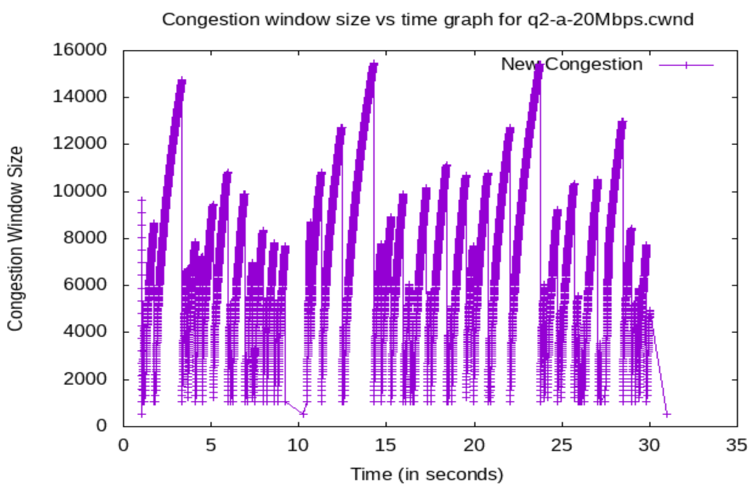
Veno:

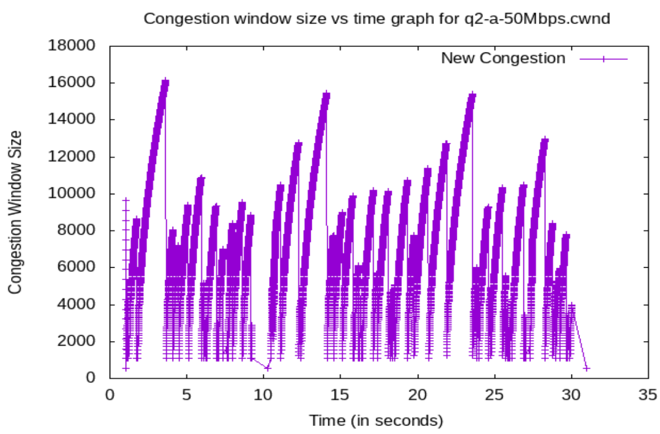
- Uses prediction method similar to vegas.
- Refines additive increase algo of reno to increase duration of connection in stable state by increasing cwnd by 1/cwnd for every new ack after bandwidth is fully utilized
- Multiplicative decrease is by 1/5 as loss is assumed to be more likely due to corruption.
- The average window size is higher compare with any other protocol
- Plot is similar to newreno except for fully utilization part

Q2

A) Different channel rate

Channel data rate	Plot	Packet dropped
2 Mbps		66

4 Mbps		72
10 Mbps		73
20 Mbps		74

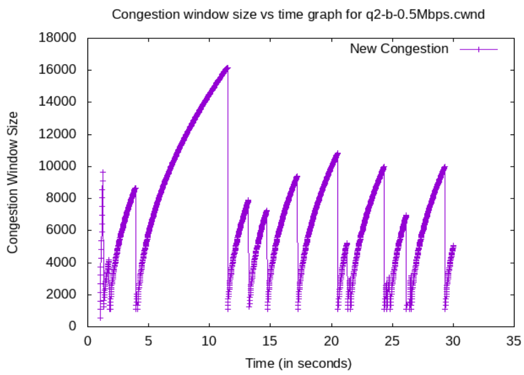
50 Mbps		75

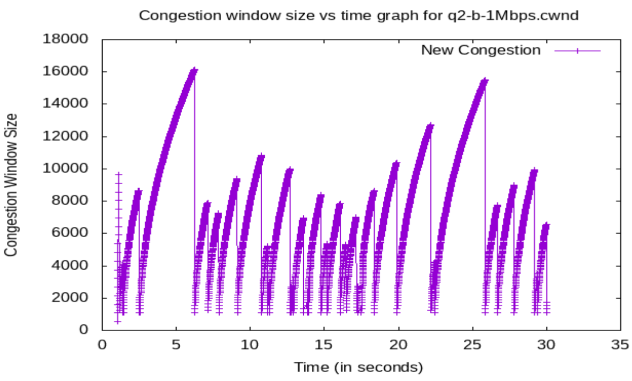
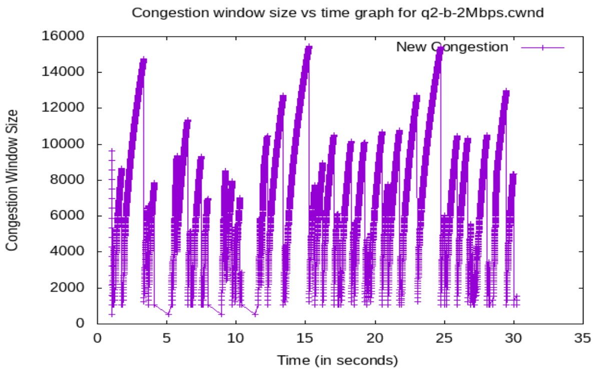
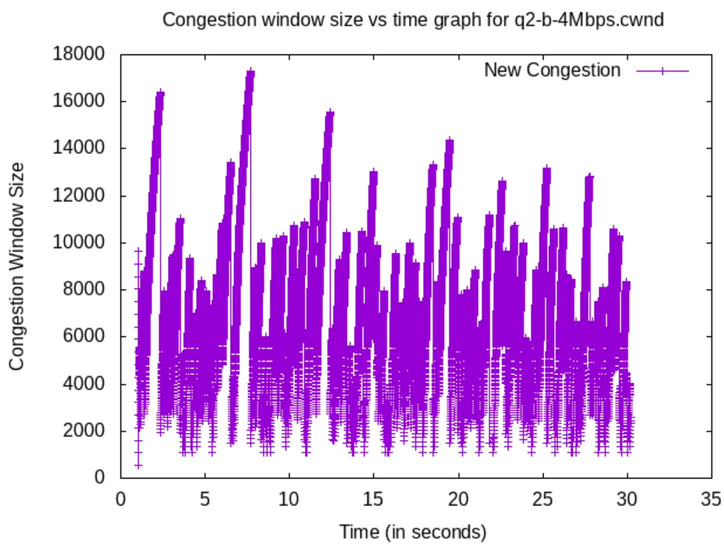
Channel rate is varied but the rate at which application is sending data is constant.

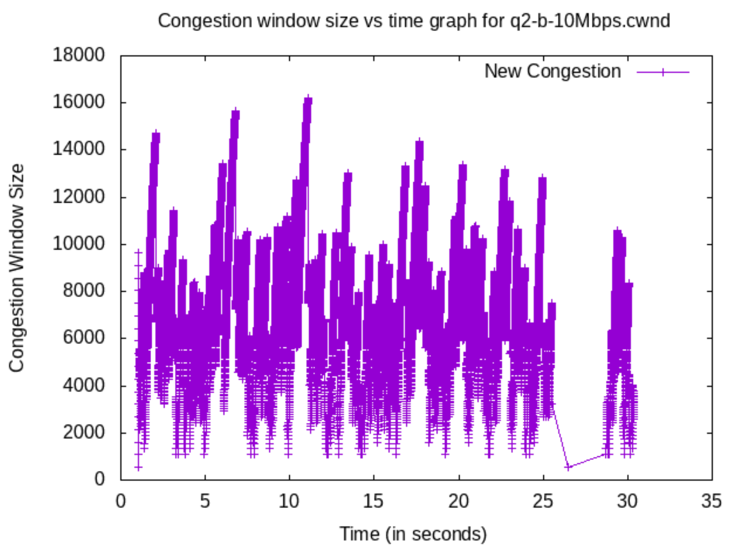
The total packets loss increases slightly as we increase the channel rate but the increase is not much.

Most of the graphs look almost same in all cases. The packet losses are determined by buffer size at receiver and not much affected by channel transmission rate. We see slightly less maximum congestion window size in case of application data rate == channel data rate as both are the same. But as the channel rate start to increase there is slightly large congestion window size but after 10 Mbps the effect is negligible and the graph is almost the same except for error model

B) Different application data rate

Channel data rate	Plot	Packet dropped
0.5 Mbps		22
1 Mbps		38

		
2 Mbps		71
4 Mbps		158

10 Mbps		100

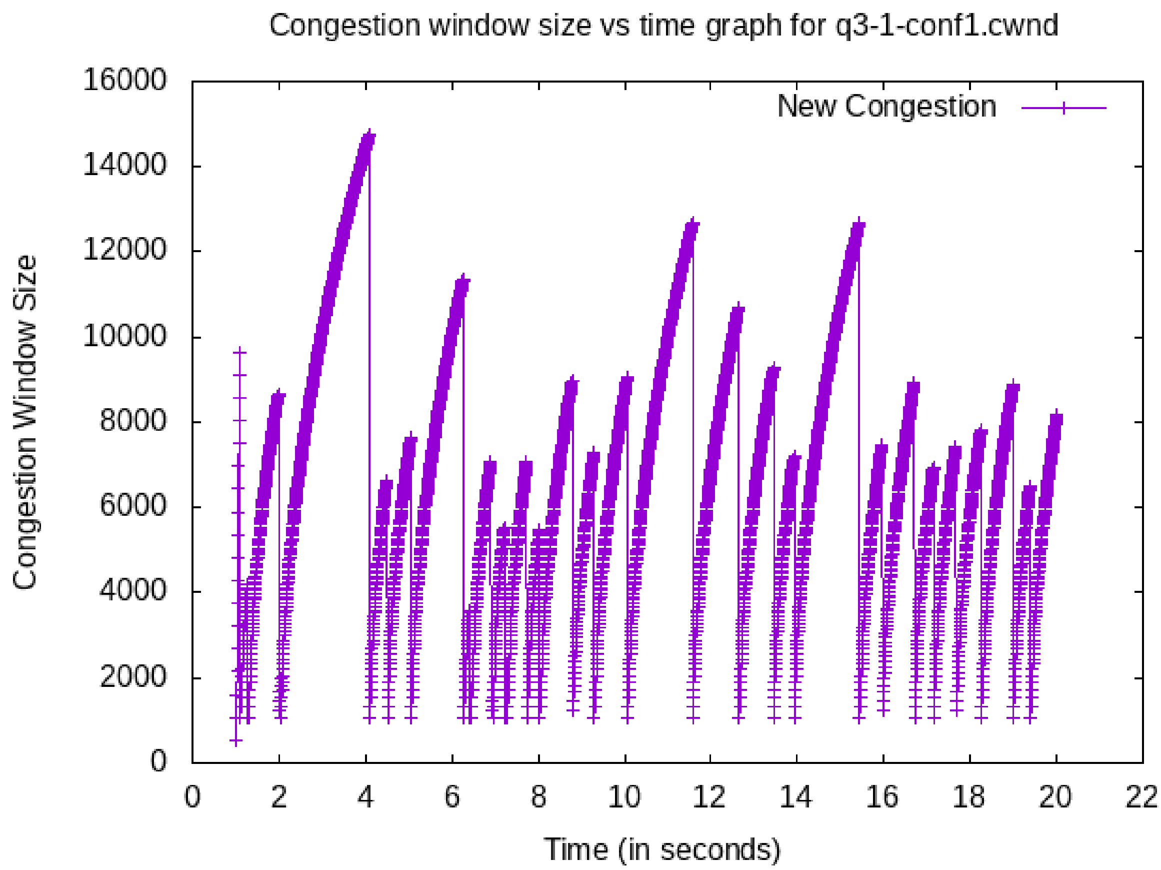
Here channel rate is fixed but application rate is varied.

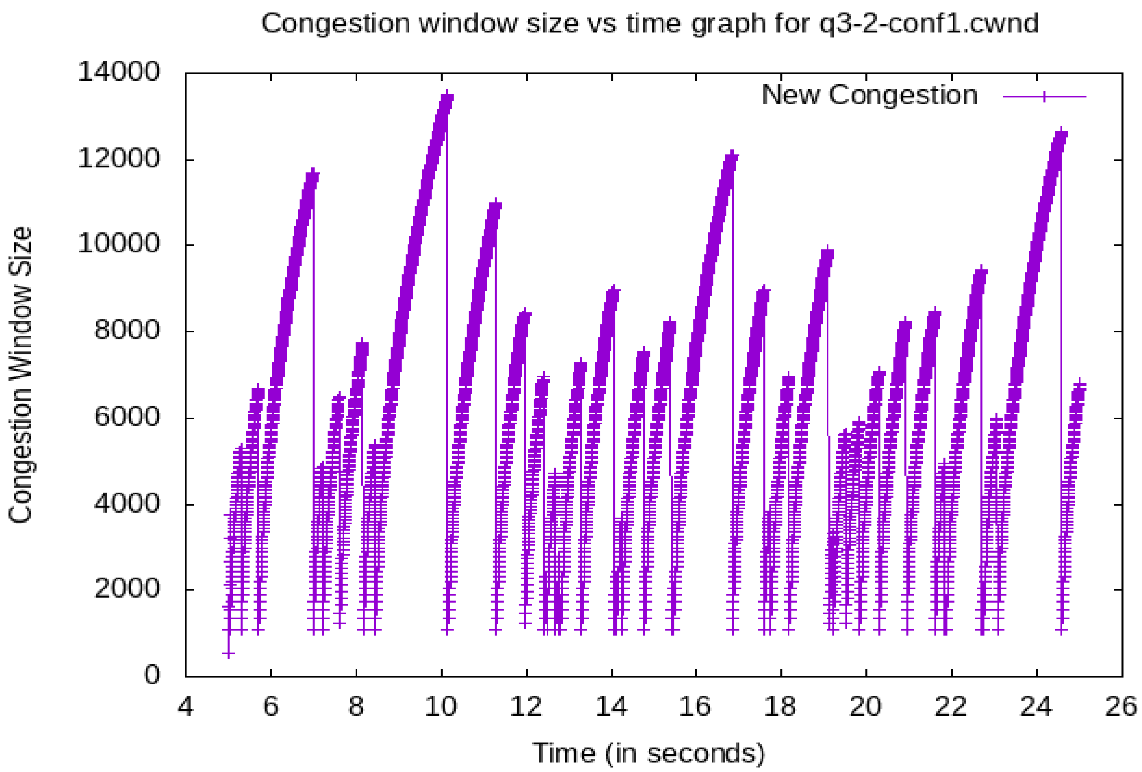
Total number of packets lost increases with the increase in application data rate. Large number of packets lost increases the noise in the graph. This is expected as the channel gets over occupied for more time than less application rate.

Q3

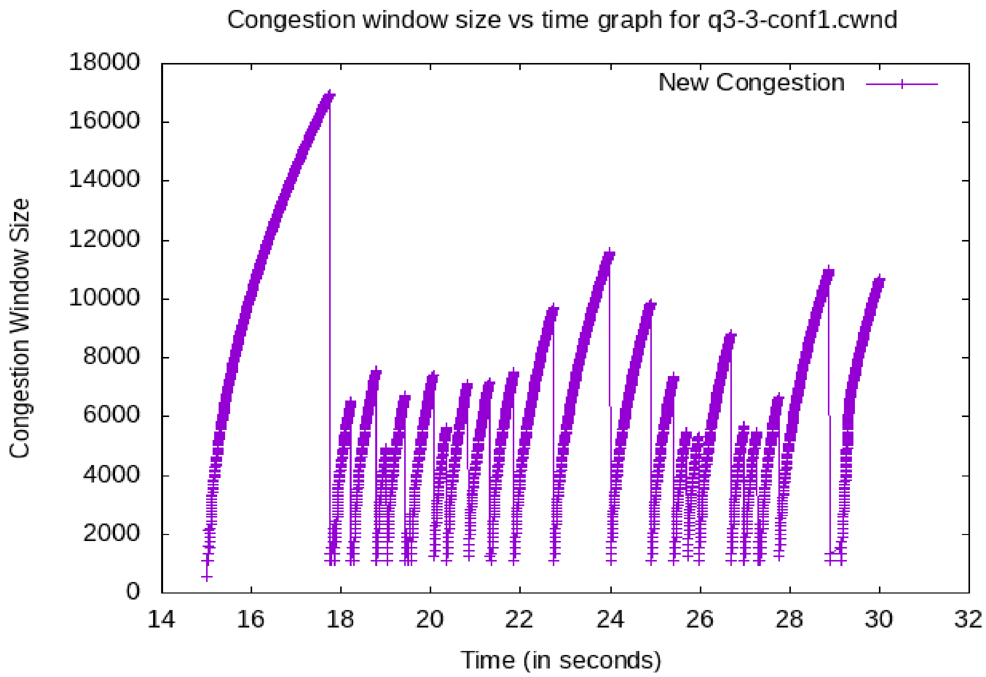
Implementation: By inheritance

./waf --run "scratch/Third –conf_num=3"



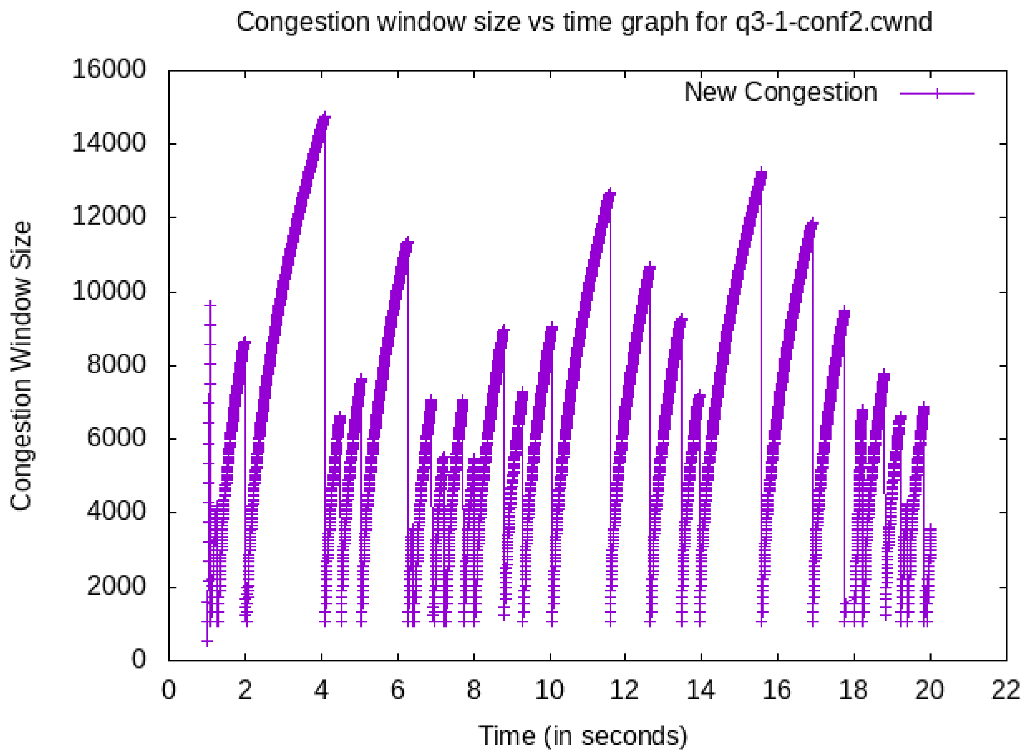


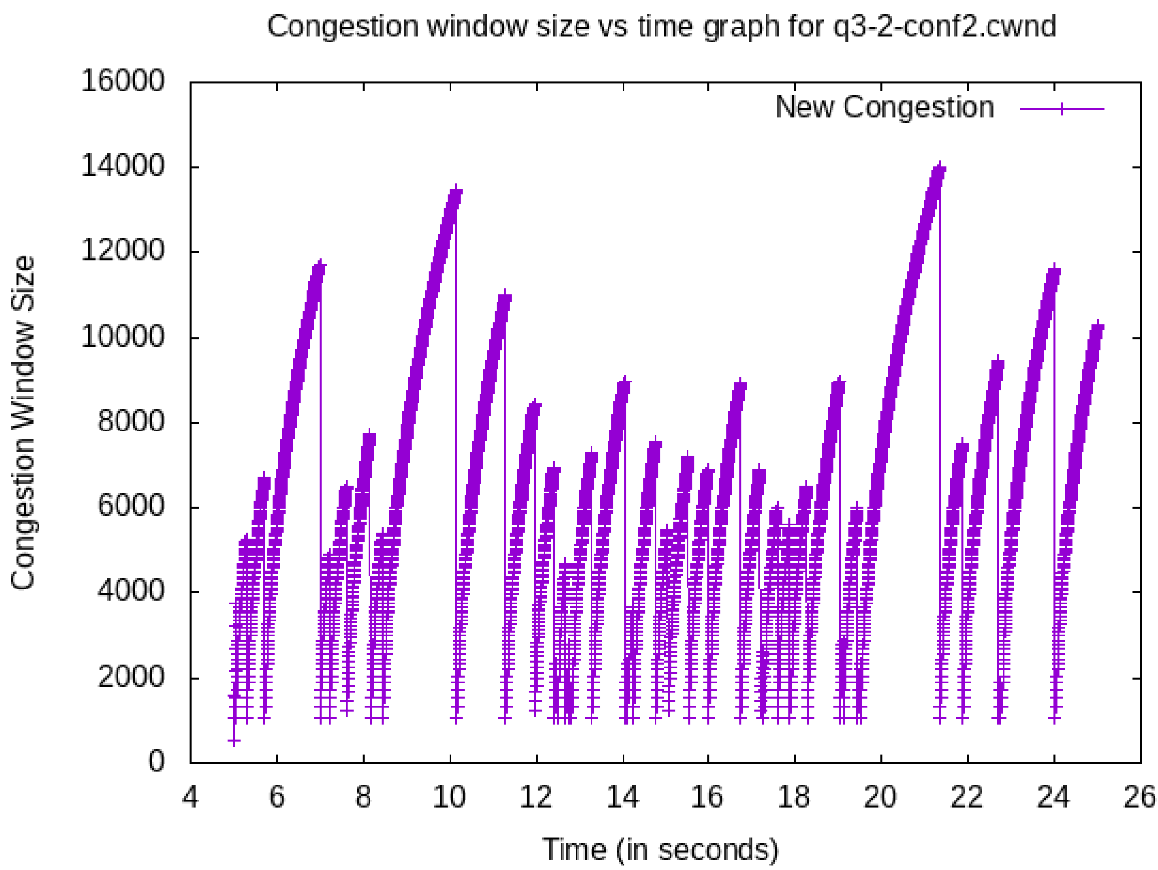
Congestion window size vs time graph for q3-2-conf1.cwnd
New Congestion
Congestion Window Size
Time (in seconds)

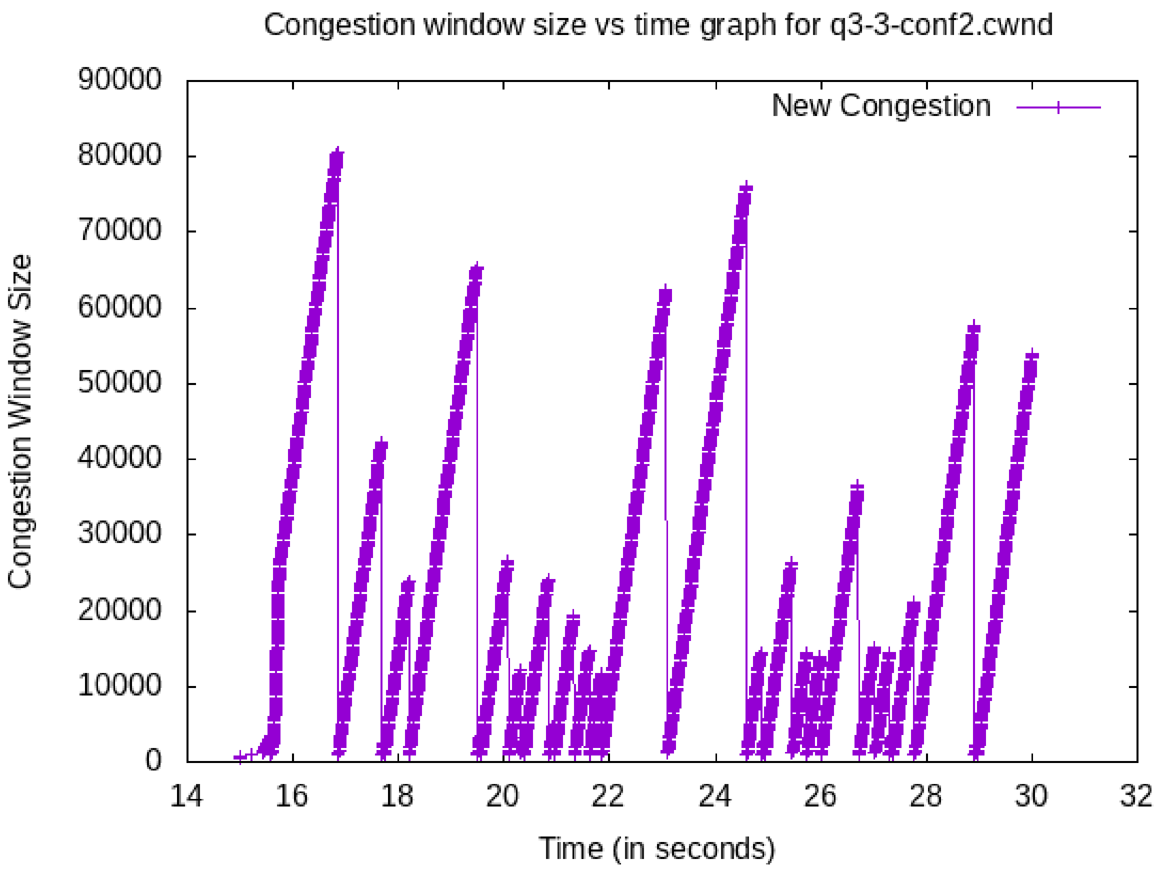


Congestion window size vs time graph for q3-3-conf1.cwnd
New Congestion
Congestion Window Size
Time (in seconds)

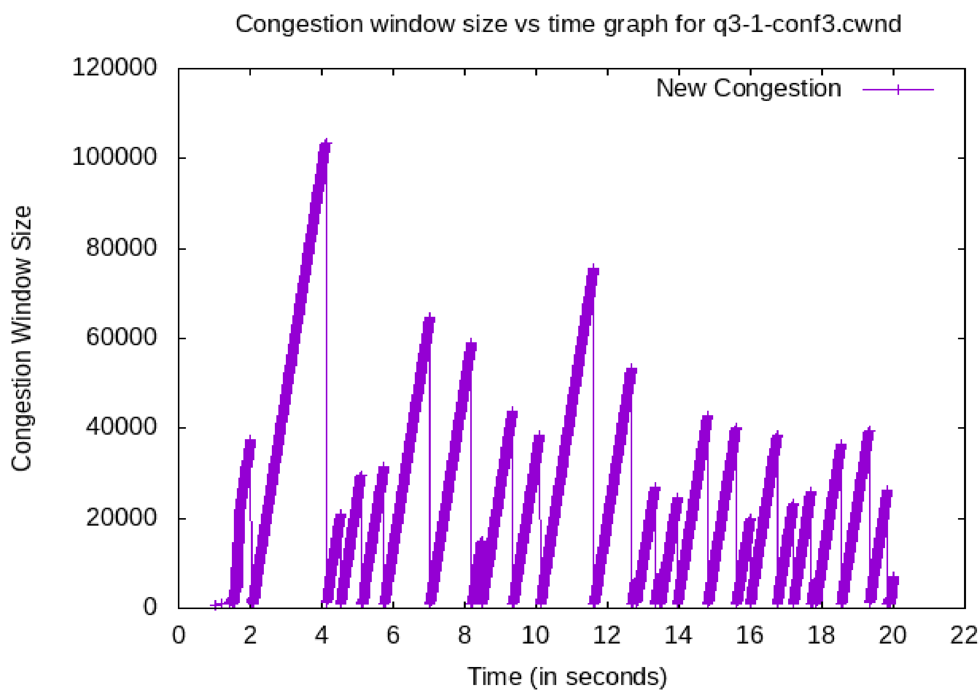
Configuration 2:

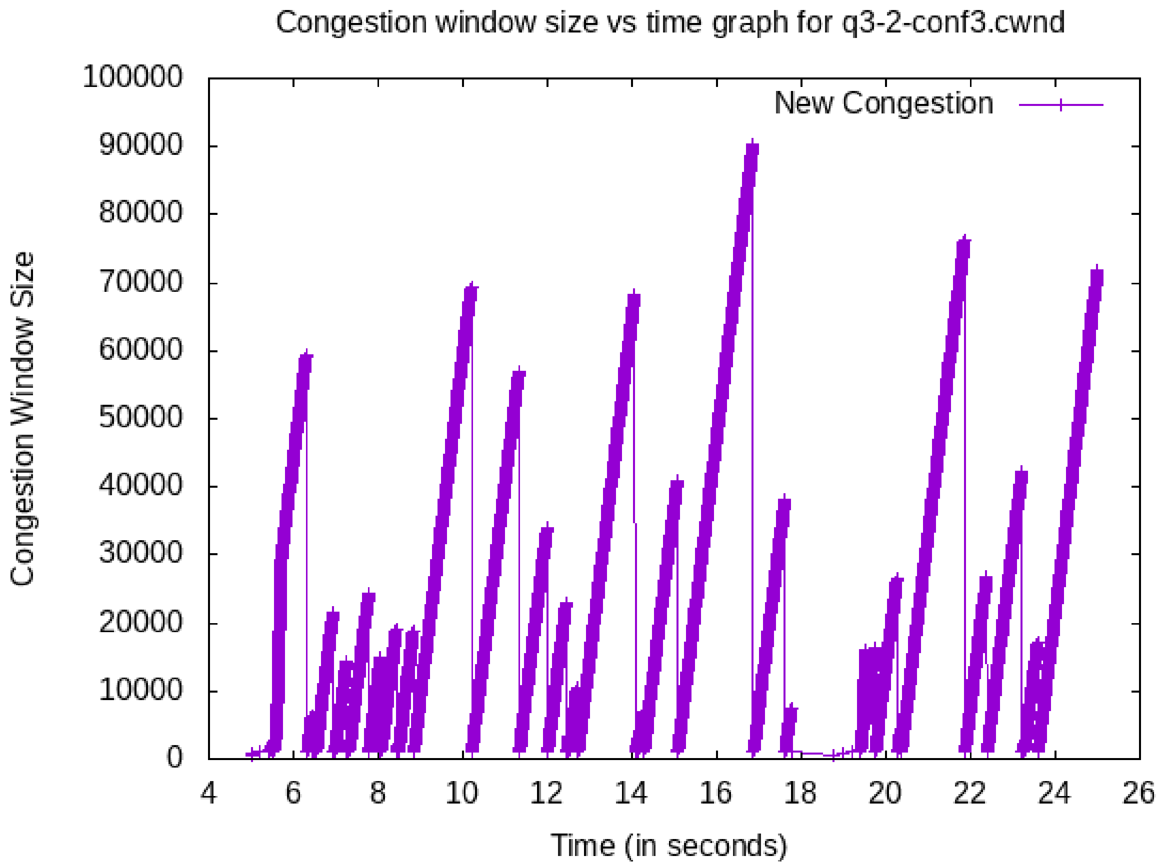


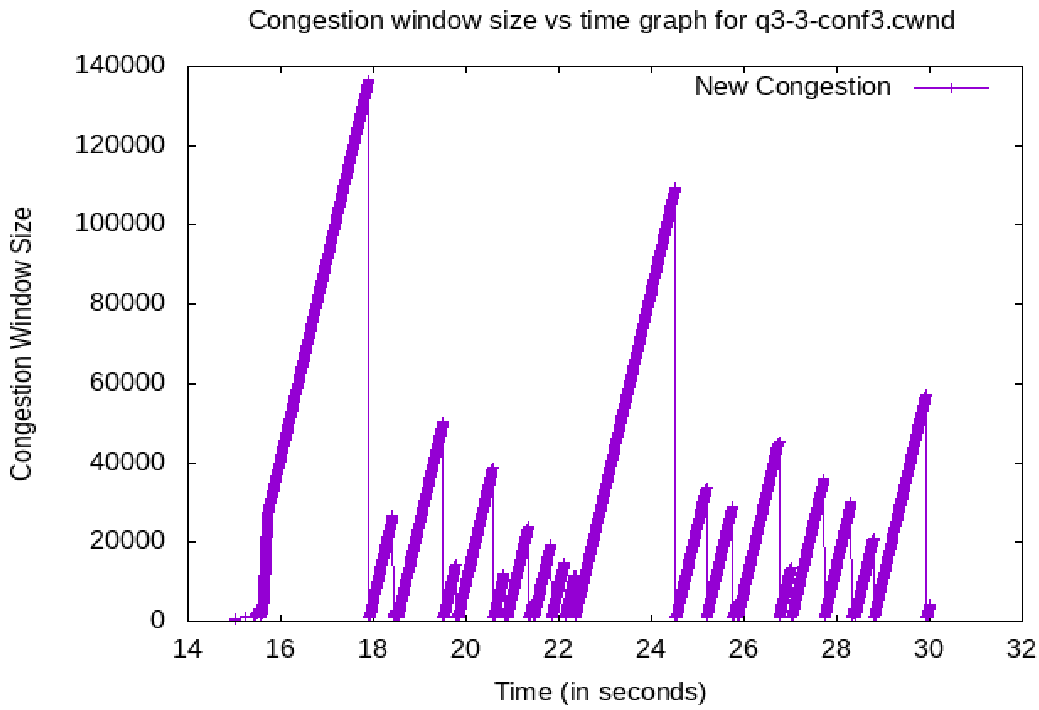




Configuration 3:




Congestion window size vs time graph for q3-2-conf3.cwnd
New Congestion
Congestion Window Size
100000
90000
80000
70000
60000
50000
40000
30000
20000
10000
0
4
6
8
10
12
14
16
18
20
22
24
26
Time (in seconds)


Congestion window size vs time graph for q3-3-conf3.cwnd
New Congestion
Congestion Window Size
140000
120000
100000
80000
60000
40000
20000
0
14
16
18
20
22
24
26
28
30
32
Time (in seconds)

Q2.

Packet Loss

Config	Channel 1-3 loss	Channel 2-3 loss	Total Loss
1	76	30	106
2	80	30	110
3	78	27	105

When connection 3 started to use TCPNewRenoCSE the packet loss in channel 1-3 started to increase. Channel 2-3 loss remained constant as there was not any diference. But when all the nodes used TCPNewReno the overall packet loss decreased. Channel 1-3 got higher number of packet losses but channel 2-3 got reduced number of packet losses.

Q3.

We see congestion avoidance in TCPNewReno is linear and hence we see a straight line in Conf2 after t=15 seconds where node with new protocol starts whereas we see non linear behaviour in earlier congestion avoidance phases.

Congestion avoidance phase of TCPNewRenoCSE is 0.5*segmentsize but in TcpNewReno we have segmentsize* segmentsize/cwndsize. Hence we see parabolic decrease in the cwnd in the congestion avoidance phase. Connection 3 in configuration 2 shows this trend. We see sudden change in growth of congestion window size in congestion avoidance phase from non linear to linear.
Initially the growth rate is higher in NewRenoCSE but after a point the square term in denominator dominates and we see the NewRenoCSE dominates. Hence at higher cwnd size the newrenocse can reach to much higher values than newreno.

The slow start phase is +=seqment size which is linear whereas NewRenoCSE has $(\text{segmentsize})^{1.9}/_{\text{cwnd}}$ which is non linear.

The overall network congestion improves as at higher congestion window size the rate of increase is much higher in NewRenoCSE. The packet lost decreases when all nodes use the same protocol.