COL 334 Assingment 3

Task 1

Explanation of Code

I have used code from wifi-tcp.cc and seventh.cc.
For counting the number of packets droped I have declared a global variable (int packets_drop = 0).
The MyApp class is from seventh.cc and is self explanatory. I have added (packets_drop++;) in the RxDrop void
For max Window size also I have declared a global variable (max_window_size = 0)
In CwndChange void I have updated the max window as max(current max window , new window size)

Using the code from wifi-tcp.cc I have added a string tcpVariant which can be set to any legal TCP Variant (TcpNewReno , TcpVegas , etc)

The below command is used to change the Tcp variant

```
Config::SetDefault ("ns3::TcpL4Protocol::SocketType", TypeIdValue
(TypeId::LookupByName (tcpVariant)));
```

I have created 2 nodes
I have set PointtoPoint Datarate = 5Mbps and Delay as 3ms
I have set the ErrorRate = 0.00001
I have ran the sink from 1 sec to 25 sec and the source app from 1 sec to 25 sec
I have used Gnu plot helper(from seventh.cc) to plot the congestion window which is also generated with the existing class in seventh.cc
In the end I have output the max size and packets dropped and the tcp variant on the terminal itself

To Run the code use the following command

```
./waf --run task1
```

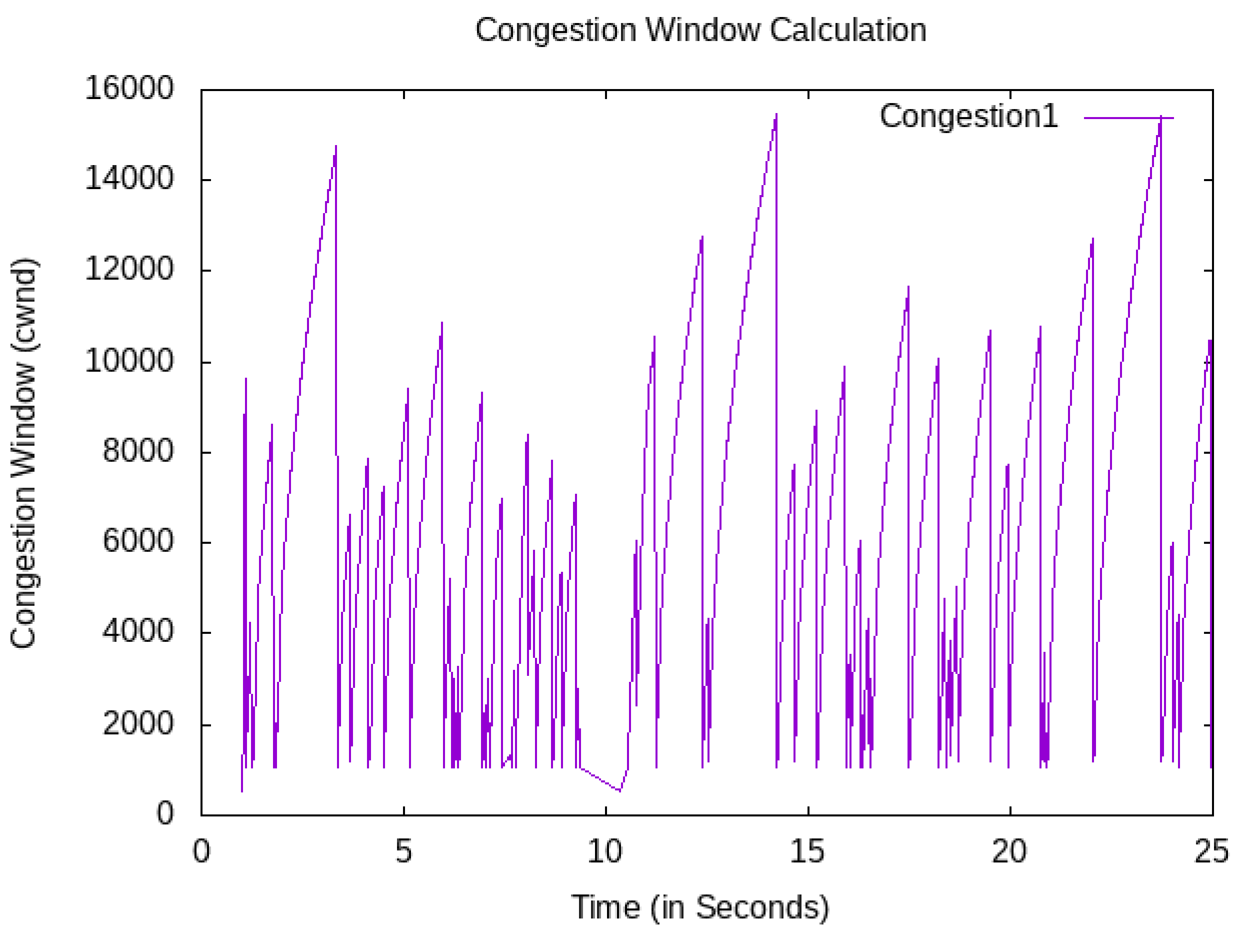
You can change the variable (TcpVariant) in the code itself to run NewReno or Westwood etc

Part 1

Protocol Name	Max. window size	No. of Packets drop
NewReno	15462	58
Vegas	11252	58
Veno	15462	59
WestWood	15471	60

Part 2

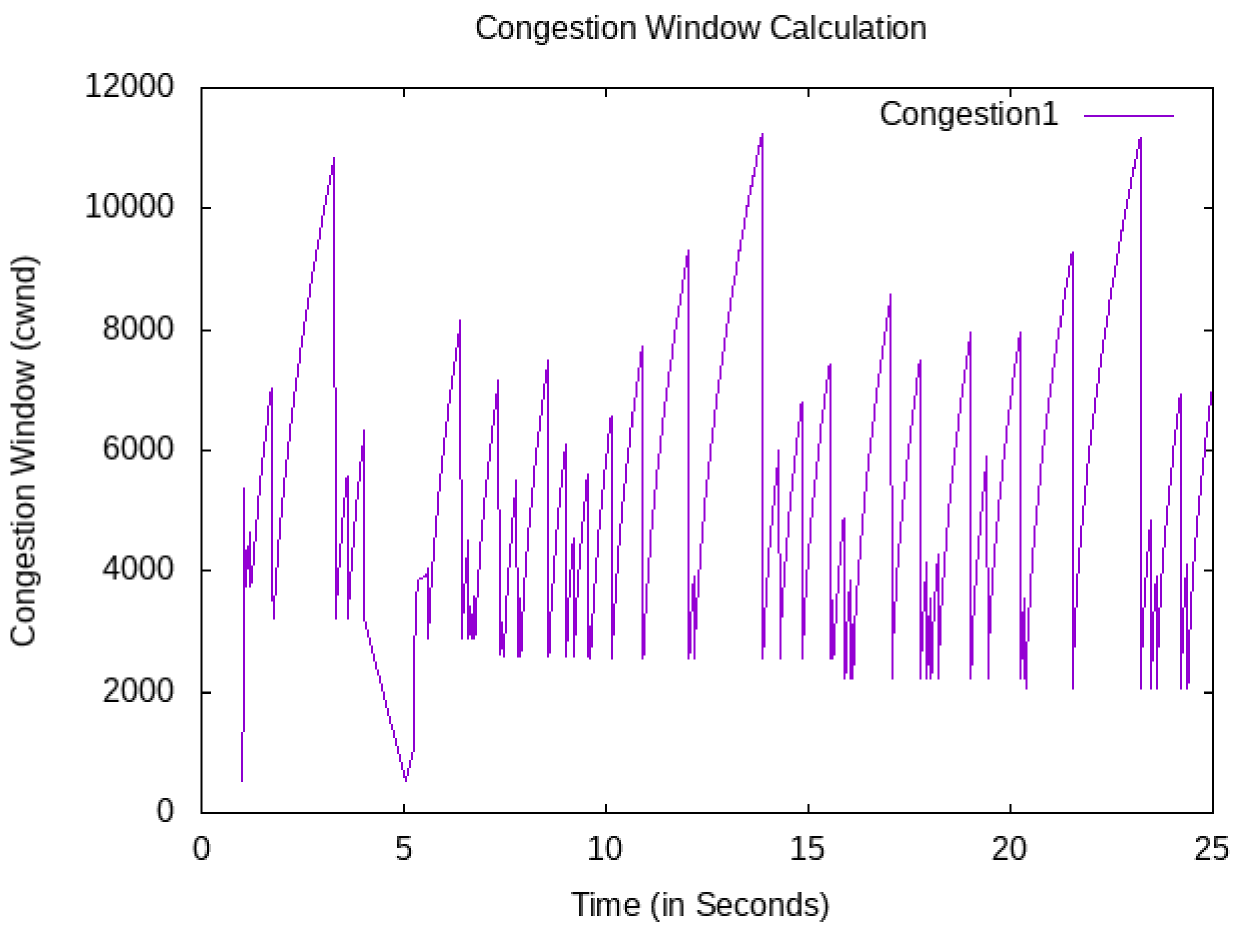
New Reno



The maximum window size is 15462 and the number of dropped packets is 58. The minima(ssthresh) is amost same in all the cases.
TCP NewReno includes partial acks. Initially the congestion window is 1 segement. Each time an ack is recieved, it increases the congestion window size by one. This leads to a multiplicative increase until a threshold size is reached. Then it enters a slow phase to avoid congestion and thus resets the its congestion window to half of the ssthreshold (slow start threshold) and then increases the congestion window by 1

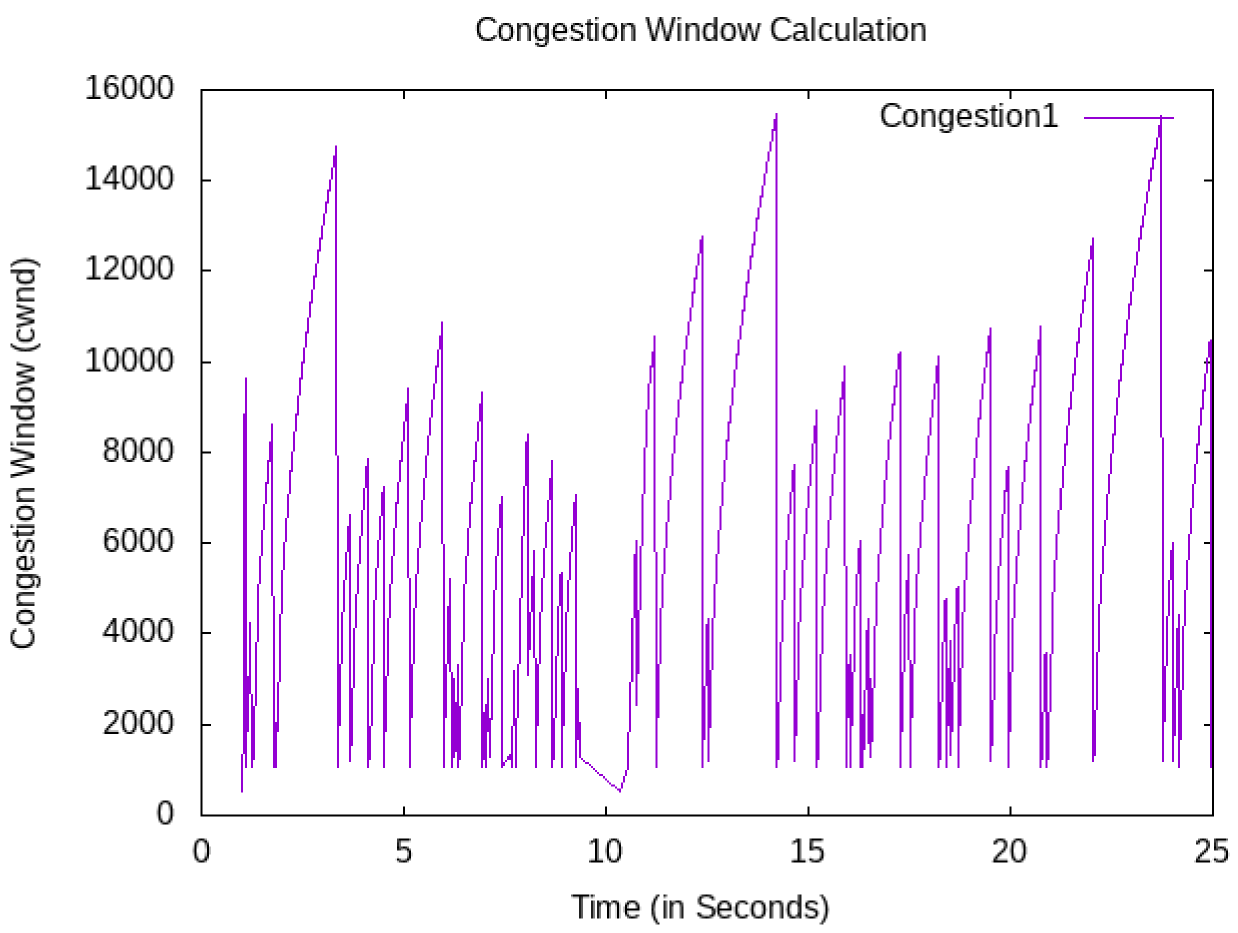
each RTT.
Unlike tcp reno congestion window doesnot half its size at every packet drop. It reduces its size by 50% if and only if all the packets in the congestion window are dropped.
From the graph it can be seen there there are a few incidents where the size is dropped to 1/2 of the previous size

Vegas



When ever a packet loss is encountered the system enters a packet loss prevention phase. It can sense a packet loss even before it has occured. It doesnot follow any concept of ssthresh. It contains to values α and β. First extra data flow is calcualted via the formula
$extradata = (expectedOutput - actualOutput) * baseRTT$
now if this $extradata$ is less then α then window size is increased by 1. Else if it is greater then beta then the window size is decreased by 1. Else there is no change in the window size.
We can see from the graph, there is a phase when the cwnd keeps on increaseing and it reaches a peak, here $extradata > \beta$ therefore the untill the extradata becomes less than beta , it keeps on decreasing the cwnd

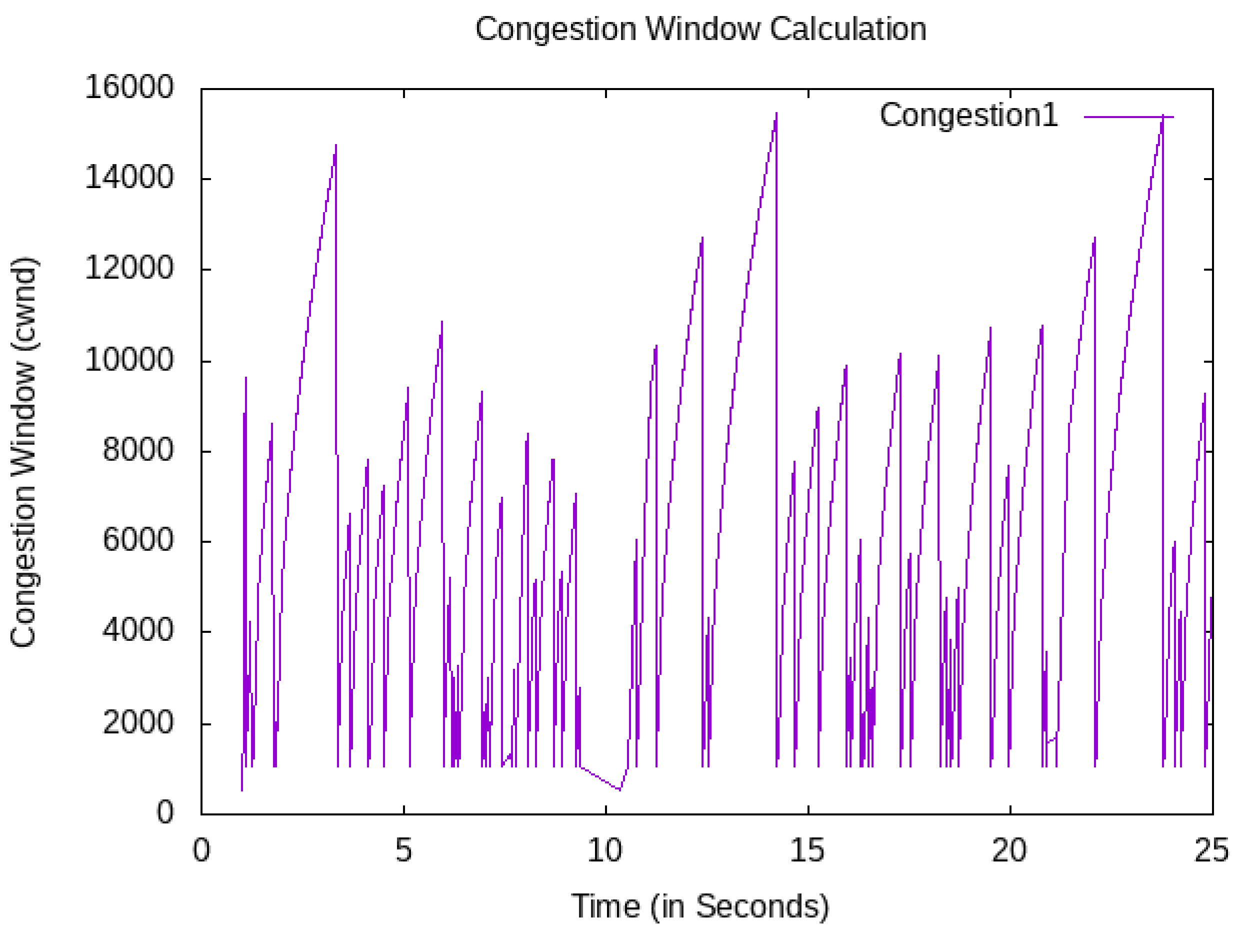
Veno



It works almost similar to vegas as well as reno. The only major change is when packet drop happens.
If the packet drop happens when the $extradata < \beta$ then veno assumes loss is random and it increases its cwnd by 1/cwnd at every new RTT also it makes the ssthresh = 4/5 times the cwnd. else the loss is congestion and the cwnd is increased by 1/cwnd by every 2 RTTs and makes the ssthres = 1/2 of the cwnd.

The main difference between veno and reno's graph is that veno stays in large window region longer because it does not increases its congestion window by much during the congestion period it.

Westwood



It maintains a ssthresh. A Bandwidth Estimate (BWE) is maintained, it is equal to the data delivered to The TCP reciever. The sender will reset the ssthresh and cwnd based on BWE on reception of 3 duplicate ACKs $cwin = BWE \times RTTmin$
more pecisely,

```
If 3 Dupacks are recieved then
----ssthresh = (BWE $\times$ RTTmin)/seg_size
----If cwin > ssthresh ..... congestion avoidance
--------cwin = ssthres
----endif
endif
```

As we can also see from the graph there is a ssthresh(fixed as BEW is same) maintained just like tcp newReno.

Part 3

The graphs of newReno, Veno, Westwood are almost similar where as vegas is different to a greater extent.

The main difference between veno and reno's graph is that veno stays in large window region longer because it does not increases its congestion window by much during the congestion period it.

West wood also works similar to reno. In these settings there is no noticable difference between the graph of reno, veno, westwood.

Also I have already explained the working of all the TCP variants along with the graphs.

Part 4

BBR (Bottleneck Bandwidth and Round-trip propagation time)

Unlike other protocols, it uses latency rather than lot packets to determine sending rate, thus generating better throughput.
Highest throughput and lowest delay is acheived when amount of data in flight is equal to BDP(bandwidth delay product = max BW $\times$ min RTT). BBR dinamically estimates max BW and min RTT on each ACK. Therefore trying to achieve ideal conditions. It then probes the max BW and min RTT and tries to keep the rate near BDP. It also maintains a small queue independent of buffer depth. Unlike other protocols, the cwnd size in BBR doesn't fall much below the maximum thus achieving high throughput
It can work much better than the existing protocols.

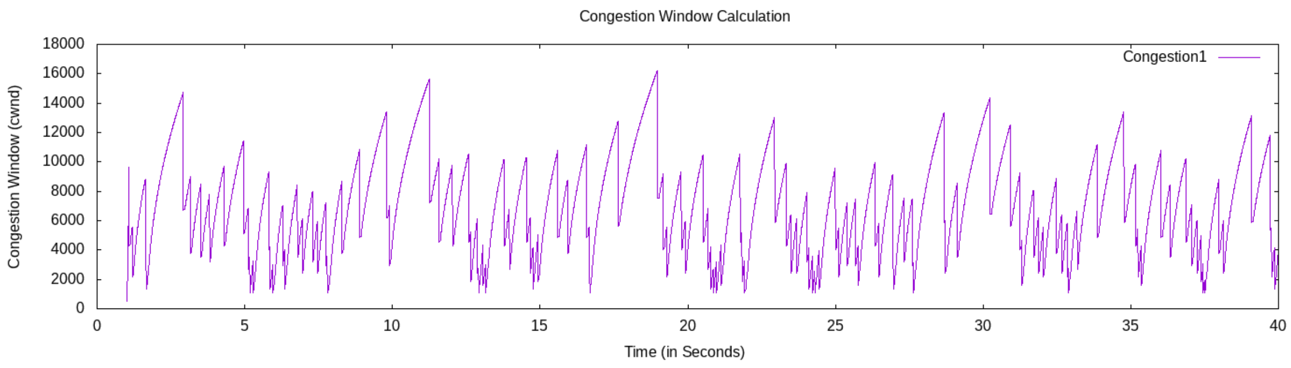
Task 2

Explanation

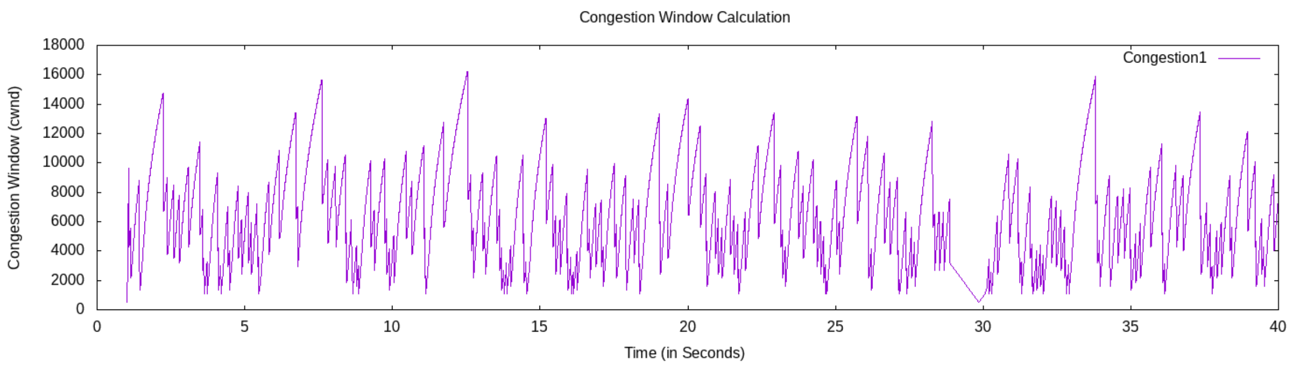
This code is just the copy of the code 1, I have set the variant to NewReno and set the packet size to 3000bytes.
I have varied the applicaition rate and the channel rate as told in the part a and part b

Part A

AppRate = 5Mbps
ChannelRate = 3Mbps

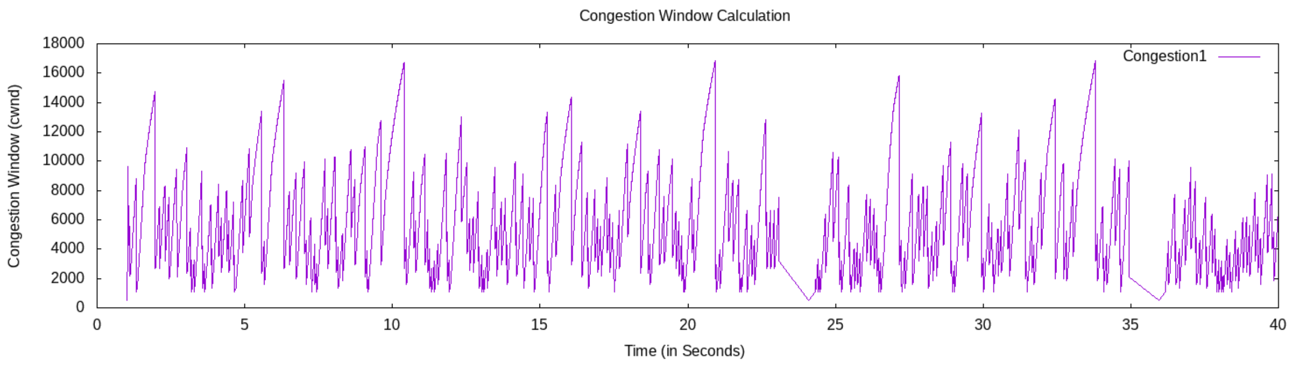


AppRate = 5Mbps
ChannelRate = 5Mbps



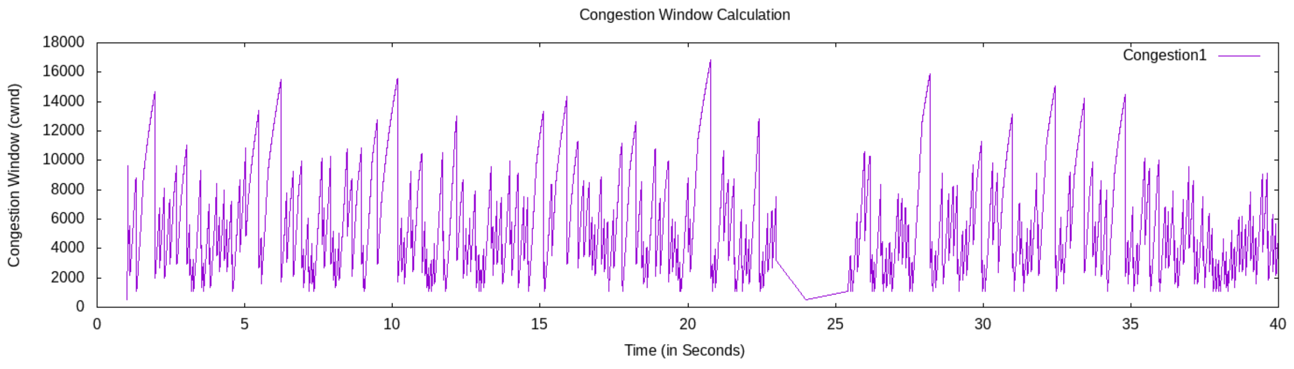
AppRate = 5Mbps

ChannelRate = 10Mbps



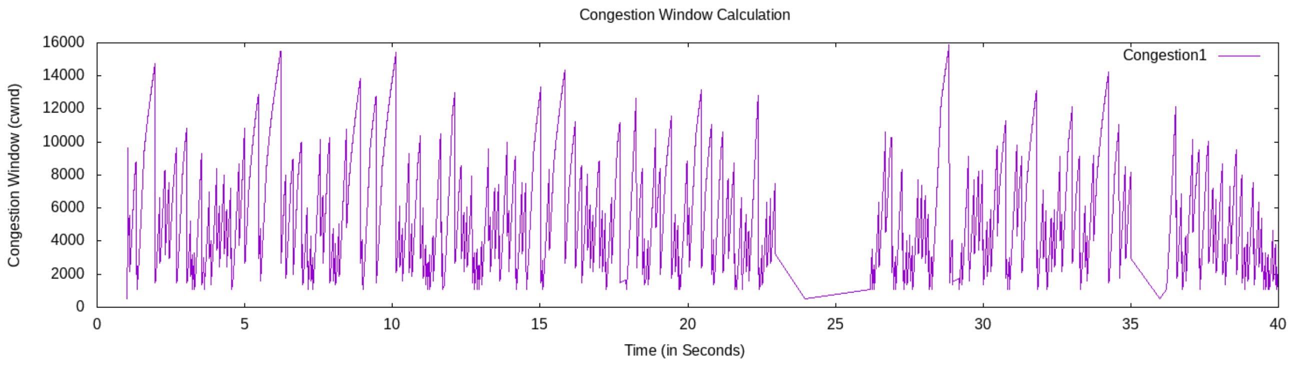
AppRate = 5Mbps

ChannelRate = 15Mbps



AppRate = 5Mbps

ChannelRate = 30Mbps

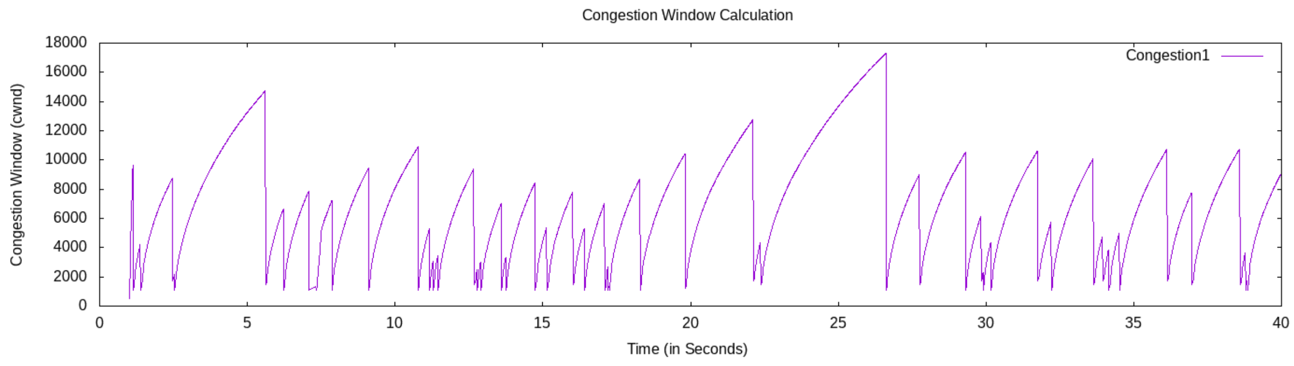


How does channel data rate affect congestion window size?

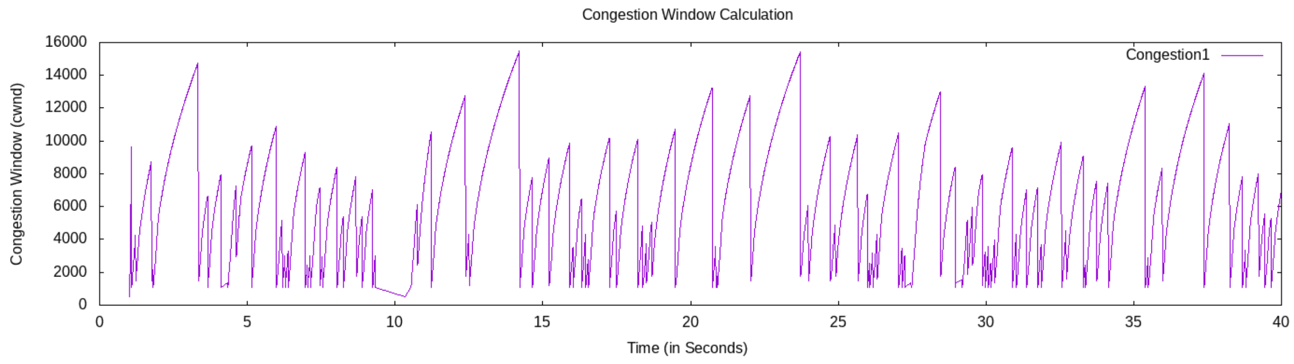
It was expected that the Max Window size should increase but there seems to be a different trend according to my outputs. However a reasonable justification, explaining the above result is as follow.
On increasing the datarate of the channel, we could reach more rate of flow more frequently thus the dropping of packet happens more frequently. Now since the packet drop happens more frequently, the new reno protocol starts the congestion control more often thus not letting the max window size to increase more.
Therefore on increasing channel rate, the number of dropped packets increase and we also get a better average thoughput.

Part B

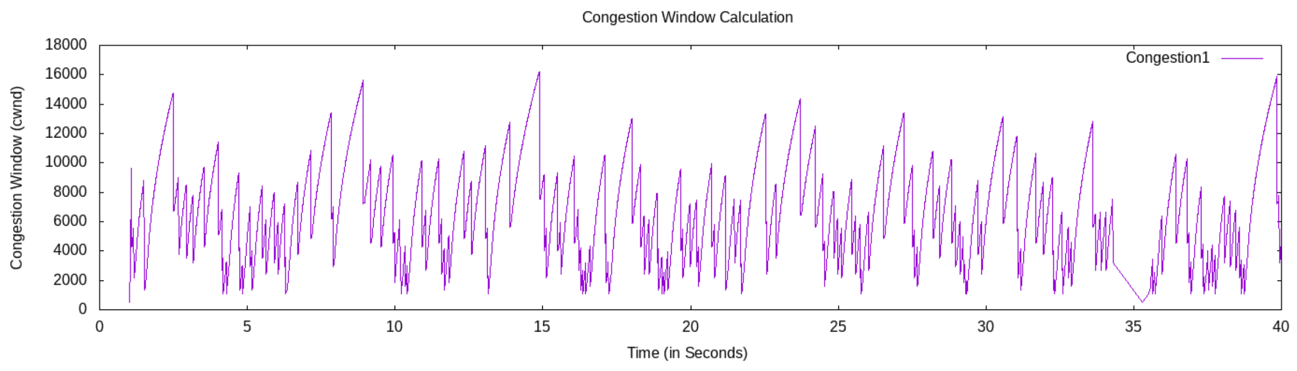
AppRate = 1Mbps
ChannelRate = 4Mbps



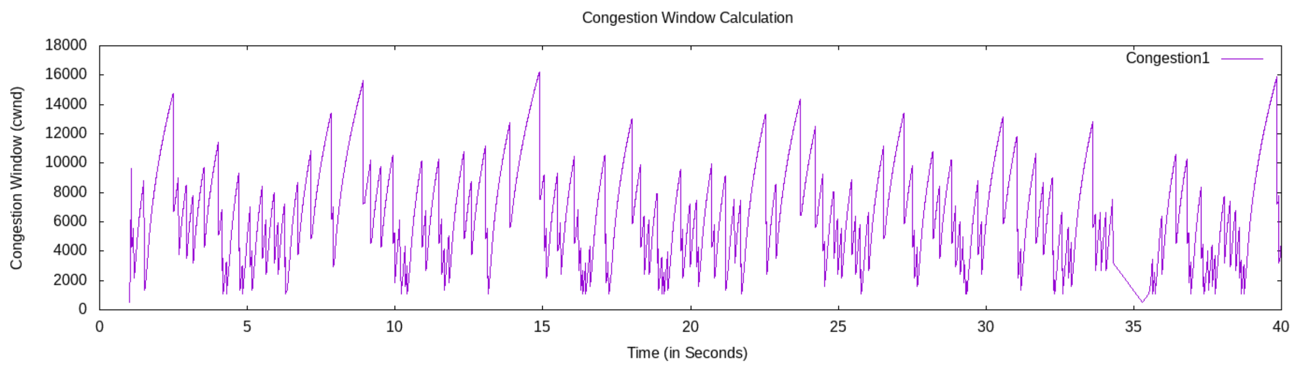
AppRate = 2Mbps
ChannelRate = 4Mbps



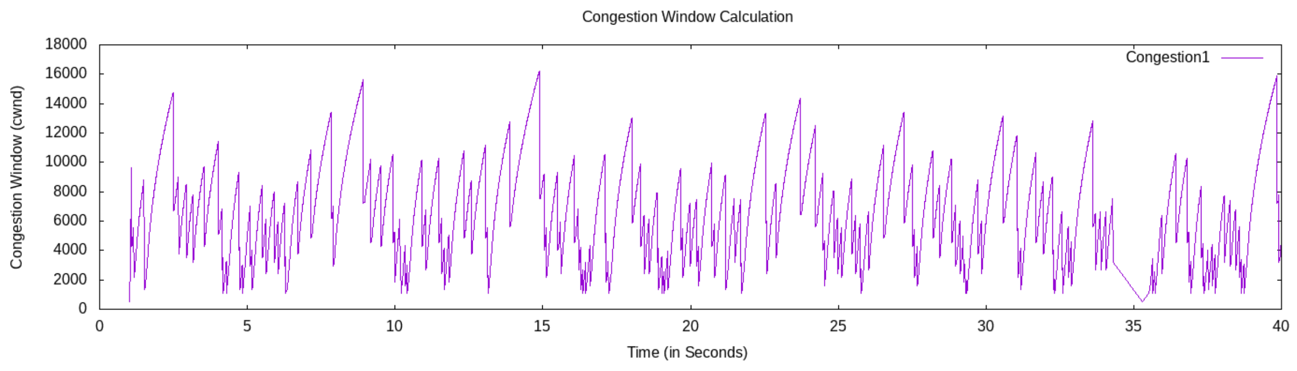
AppRate = 4Mbps
ChannelRate = 4Mbps



AppRate = 8Mbps
ChannelRate = 4Mbps



AppRate = 12Mbps
ChannelRate = 4Mbps



How does application data rate affect congestion window size.

On increasing the applicaiton rate from 1Mbps to 2Mbps, we can reach more size of congestion window more quickly also the packet drops happen more frequently due to the same reason. On increasing application rate from 4Mbps to further, there is no noticable change which explains that the rate of data transfer is upper bounded by the rate of link here and the pakcet drop and max window size is depends upon the rate of data transfer only thus it explains the trend of the graphs.

What relation do you observe between channel data rate and application data rate?

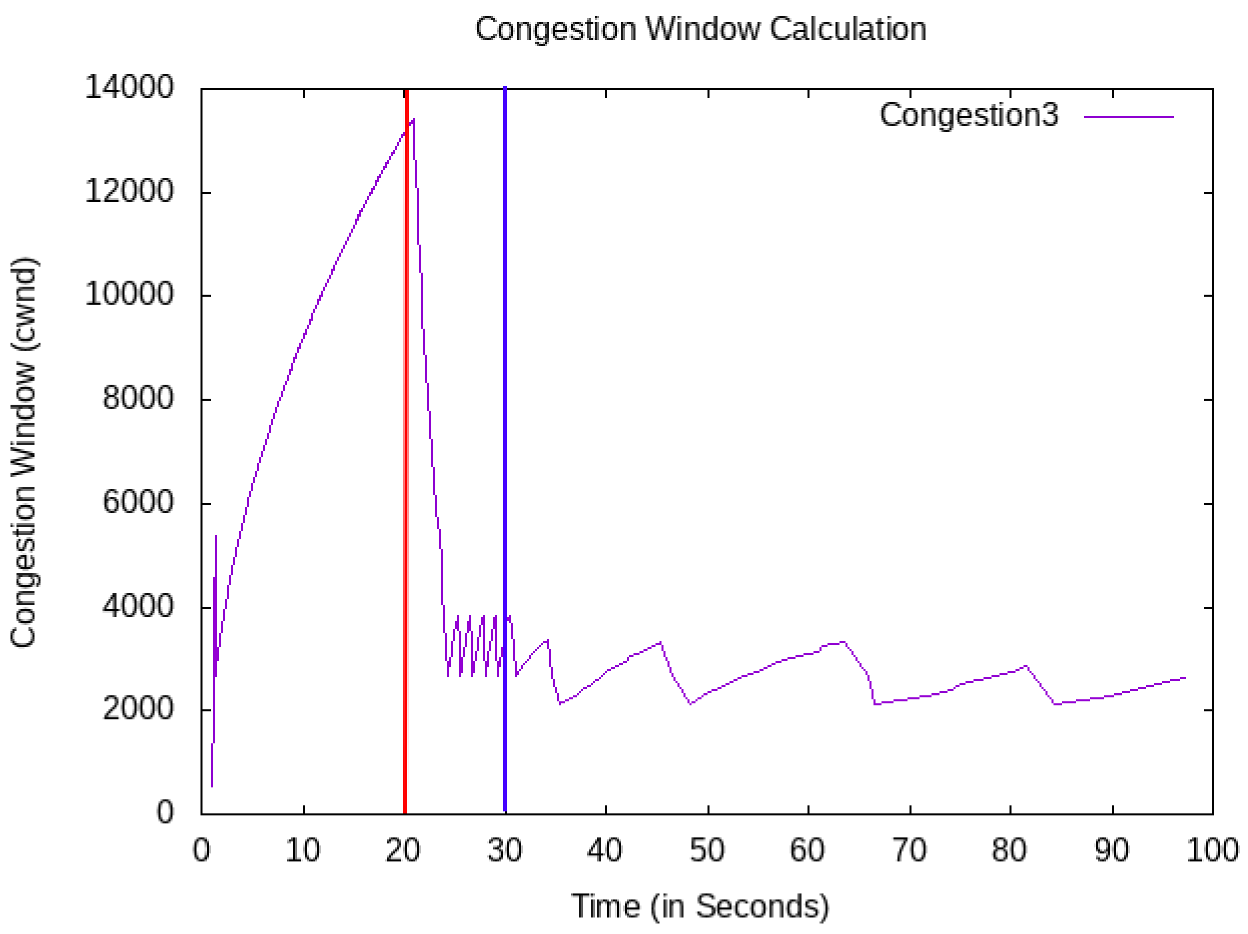
We obseverd that when channel rate is more than the application rate (not significant difference) then there is less packet drop and slow increase in the cwnd size. Thus having almost equal channel rate and application rate is benefitial. If we increase the channel rate alot then also the number of dropped packets increase significantly. NewReno doesnt seem to make much use of the channel rate. On Increaseing the application rate more than the channel data rate, there doesnt occer any noticable difference, explaining that the packet loss that happens is realated to how much data has passed through the link.

Task 3

Explanation of Code

I have used most of the code from task 1.
I have also simulated the network on NetAnim you can view the "animation.xml" file in the NetAnim
I have changed the declaration of the nodes to global so that I can declare 5 nodes.
I have also declared 4 channels betweeen them using PointToPointHelper and NetDeviceContainer. (d1d2 , d2d3 , d3d4, d3d5)
I have assigned the ip address using Ipv4AddressHelper
The IP address of node 1 is 10.1.1.1 , node 2 /interface 1 is 10.1.1.1 node 2 interface 2 is 10.2.2.1, node 3 interface 1 is 10.2.2.2 and so on.
I have also set the tcp variant to TCP vegas.
I have also assinged ipv4 addresses to all the nodes
Then using the ApplicationContainer I have declared 2 UDP applications with data rates as mentioned(I have calcualted the Intervals accordingly by Rate = number of packets / Interval) and a TCP appliction with data rates as given in the assignment
I ran the TCP application from 1sec to 100sec and the 1st UDP application from 20sec to 30sec and the 2nd UDP application from 20sec to 100sec
I have used the gnu plot lib to plot the congestion window size
I have also generated the pcap file

Part 1



Part 2

The red line indicates when we have started the first udp application at the data rate of 250KBps. The blue line indicates when second udp application was started at the rate of 500KBps.
During the initialisation of the UDP app(red line) we can see that there is a small delay before the affect of the UDP can be seen.
Due to congestion of the channel from udp, the congestion window falls down rapidly as the tcp vegas notices extra data exceeding β.
Now when the second udp application is turned on(500KBps) then the increase in the cwnd size of the tcp application becomes much slower due to congestion of the entire link via udp application. Thus we see from the graph that not only the cwnd size has decreased but also the rate of increase of window size has decreased.

Part 3

The Pcap files can be found in the folder named Task3PcapFiles

Files submitted

I have submitted 8 files and 1 folder

1. Task3PcapFiles folder contains the .pcap files for task 3
2. 2020CS50436_Report.pdf contains the report
3. task1.cc contains the code for task - 1
4. task2.cc contains the code for task - 2
5. task3.cc contains the code for task - 3
6. wscript helps to build and run the files
7. congestion1.plt contains the script for plotting the graph for task1
8. congestion2.plt contains the script for plotting the graph for task2
9. congestion3.plt contains the script for plotting the graph for task3