

COL334 - Assignment 3

Ayush Verma
2019MT6190749

October 2022

Contents

1 Task I	1
1.1 Notes	1
1.2 NewReno	3
1.3 Vegas	4
1.4 Veno	5
1.5 Westwood	6
1.6 Observation	6
1.7 Bottleneck Bandwidth and Round-trip propagation time (BBR) .	7
2 Task II	7
3 Notes	7
3.1 Part(a) - Application data rate is 5Mbps on NewReno	7
3.2 Part(b) - Channel data rate is 4Mbps on NewReno	10
4 Task III	14
4.1 Notes	14
4.2 Questions	15

1 Task I

1.1 Notes

:

- I have taken help from the site [NS3 Tutorial](#).
- For this task I have submitted three files in Task I folder. *cwnd.cc* file is the file that I edited from the *examples/tutorial* folder of ns3(particularly *fifth.cc* and renamed it) and *plot_graph.py* files will plot the graph for the outputs generated by *cwnd.cc* file.

- Apart from all the configurations that I made in `cwnd.cc` as mentioned in Assignment, I also changed the number of packets to sufficiently large enough for the simulation to last enough (here in task I, it is $t = 25s$)
- Put the `cwnd.cc` and `plot_graph.py` the files in the `./ns - allinone - 3.29/ns - 3.29/scratch` folder and `run.sh` file in `./ns - allinone - 3.29/ns - 3.29/` folder and open the terminal in `./ns - allinone - 3.29/ns - 3.29/` and run

```
1 ~/ns-allinone-3.29/ns-3.29 $ bash run.sh <Protocol name (For
   example-Vegas)>
```

- To calculate the number of packets dropped and maximum congestion window, I instantiated a variables

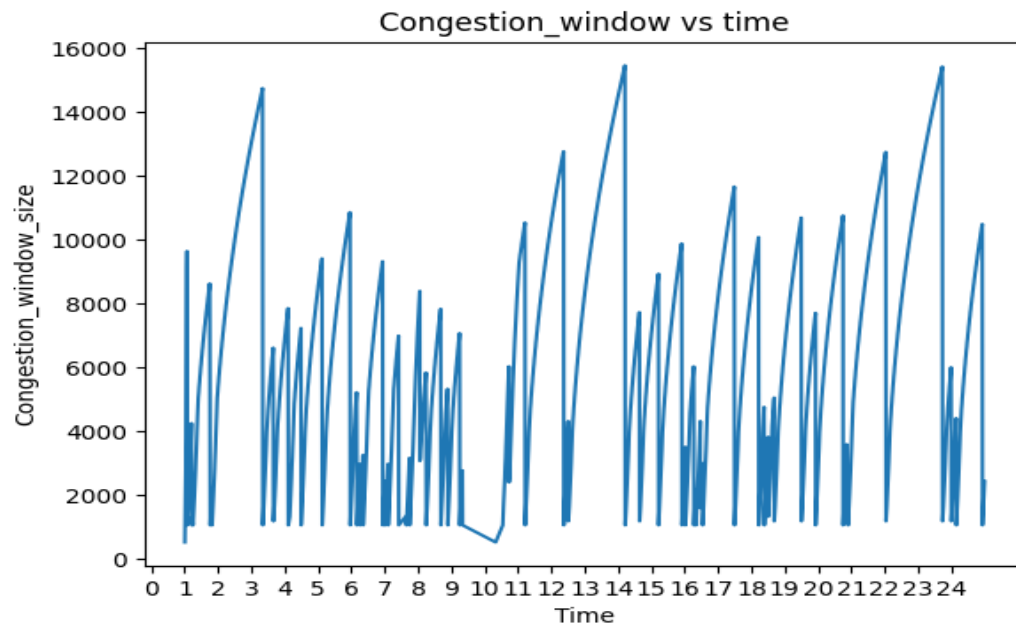
```
1 int drop = 0; //line 12 of cwnd.cc
2 int Max_window = 0;
```

Then I incremented drop variable whenever drop happens and updated Max_window variable whenever congestion changes

```
1 //line 116 of cwnd.cc
2 static void
3 CwndChange (Ptr<OutputStreamWrapper> stream,uint32_t oldCwnd,
4             uint32_t newCwnd)
5 {
6     NS_LOG_UNCOND (Simulator::Now ().GetSeconds () << "\t" <<
7     newCwnd);
8     *stream->GetStream () << Simulator::Now ().GetSeconds () <<
9     " " << newCwnd<<"\\n";
10    Max_window = max(Max_window,(int)newCwnd);
11 }
12
13 static void
14 RxDrop (Ptr<const Packet> p)
15 {
16     // NS_LOG_UNCOND ("RxDrop at " << Simulator::Now ().
17     GetSeconds ());
18     drop++;
19 }
```

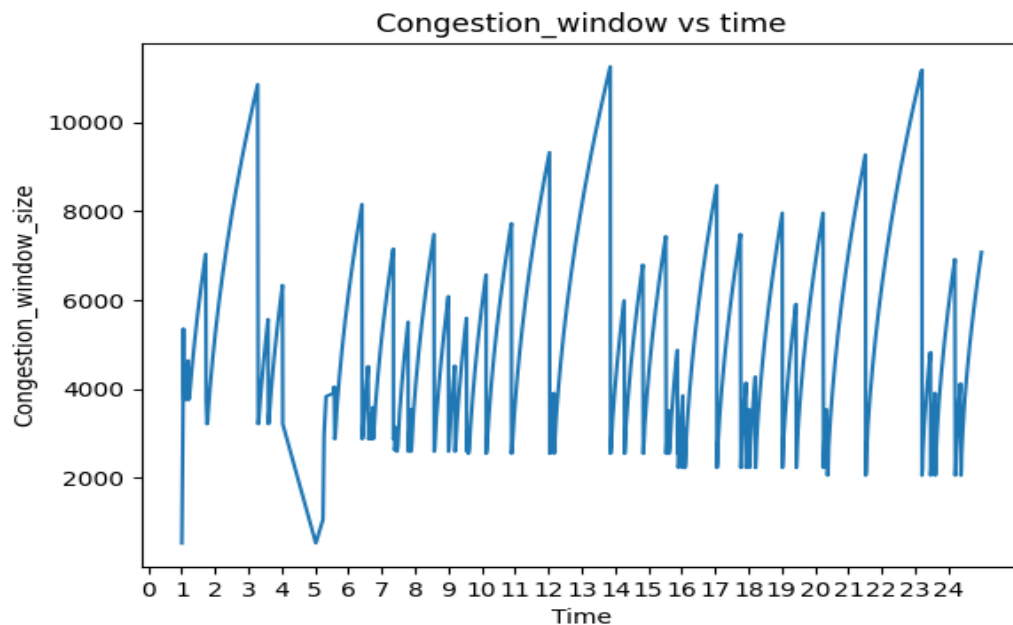
Protocol Name	Max. window size	No. of Packets drop
NewReno	15462	58
Vegas	11252	58
Veno	15462	59
Westwood	15471	60

1.2 NewReno



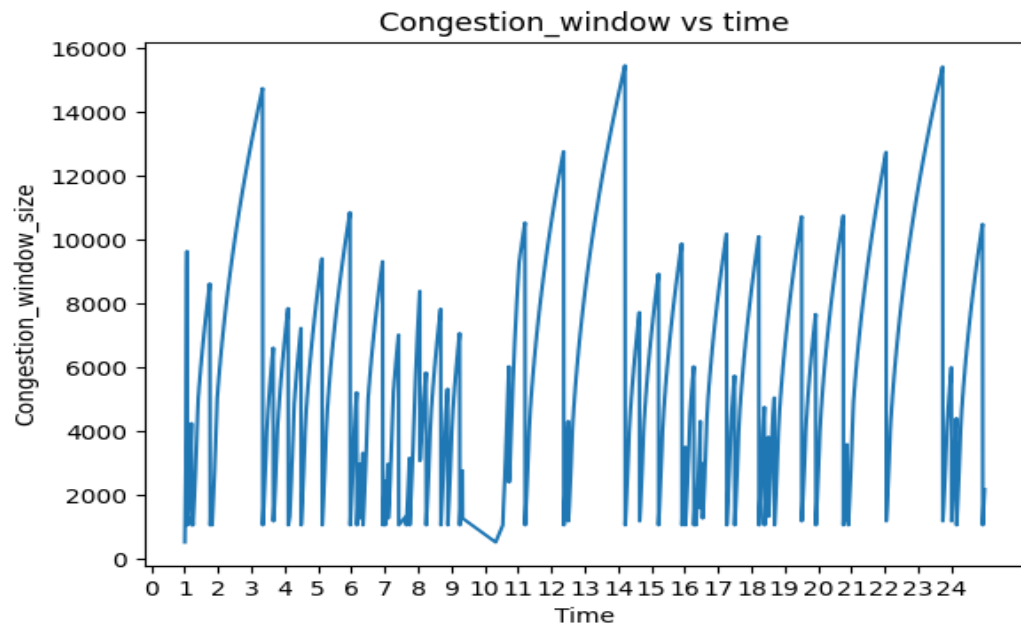
It is the modified form of Reno. NewReno differs from Reno in that it does not instantly halve ssthresh, which may shrink the window too much if numerous packet losses occur. It does not quit fast-recovery and reset ssthresh until all data has been acknowledged.

1.3 Vegas



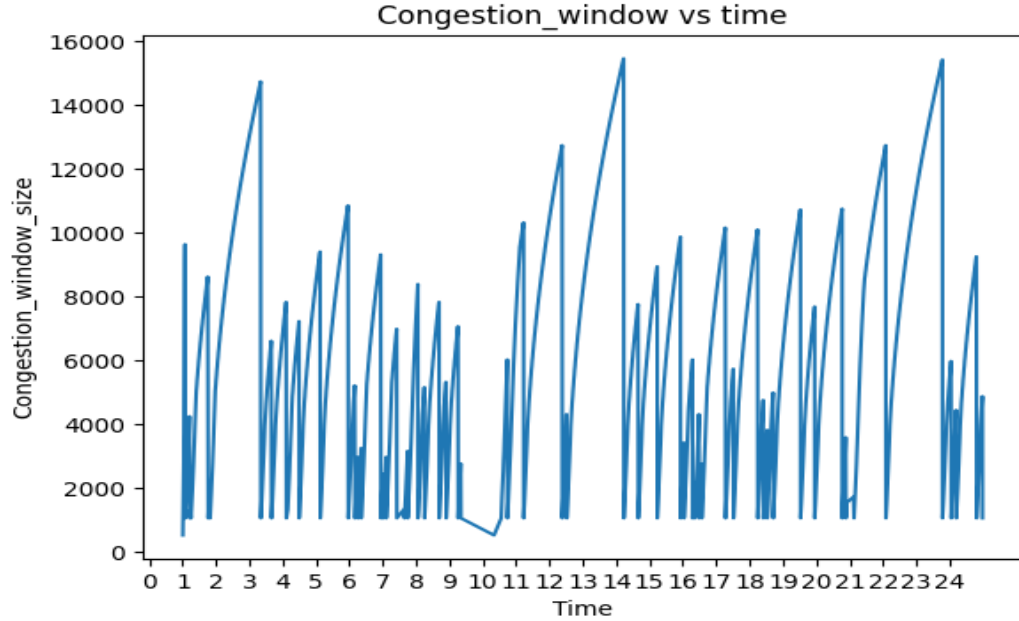
TCP Vegas regulates its window size by watching the RTTs (round-trip times) of packets previously transmitted by the sender host. If measured RTTs grow excessive, TCP Vegas detects that the network is becoming crowded and reduces the window size accordingly. Alternatively, if RTTs become modest, the sender host of TCP Vegas detects that the network is no longer congested and expands the window size. In an ideal circumstance, the window size should thus converge to an optimal value.

1.4 Veno



This protocol tries to refine the additive increase and multiplicative decrease. VenO and Reno behave very similarly

1.5 Westwood



TCP Westwood (TCPW) is a sender-only modification to TCP New Reno designed to better manage large bandwidth-delay product routes, the possibility of packet loss due to transmission or other problems. It is based on end-to-end bandwidth estimate. This estimation is obtained by filtering the stream of returned ACK packets and is used to adaptively set the control windows when network congestion is encountered.

1.6 Observation

- The maximum congestion window or the Vegas is minimum among all. This can be because of Vegas being different from all the other implementations in its behavior during congestion avoidance. It does not use the loss of segment as a congestion indicator. It identifies congestion based on a drop in sending rate relative to the expected rate, caused by the accumulation of big queues in the router.
- The number of dropped packets is similar across all protocols. This demonstrates that all protocols utilise network resources in an almost equivalent manner.

1.7 Bottleneck Bandwidth and Round-trip propagation time (BBR)

This protocol differs from all preceding protocols in the sense that it determines the transmission rate based on latency. It does not rely solely on lost packets to reduce the transmitting rate. BBR simulates the network to transmit data as quickly as the available bandwidth allows. BBR really performs very well under moderate packet losses comparison to other above protocols

This protocol has the disadvantage of consuming all available bandwidth and displacing other TCP streams that use.

2 Task II

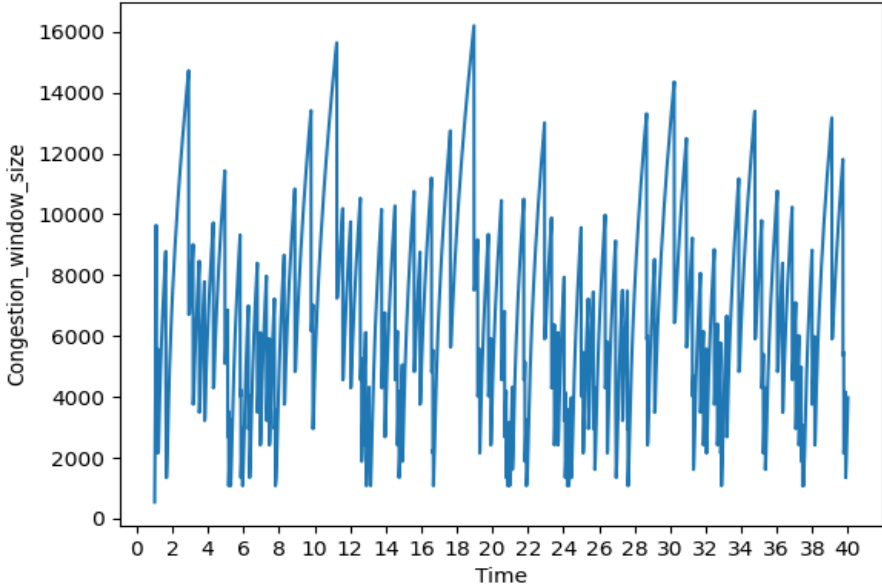
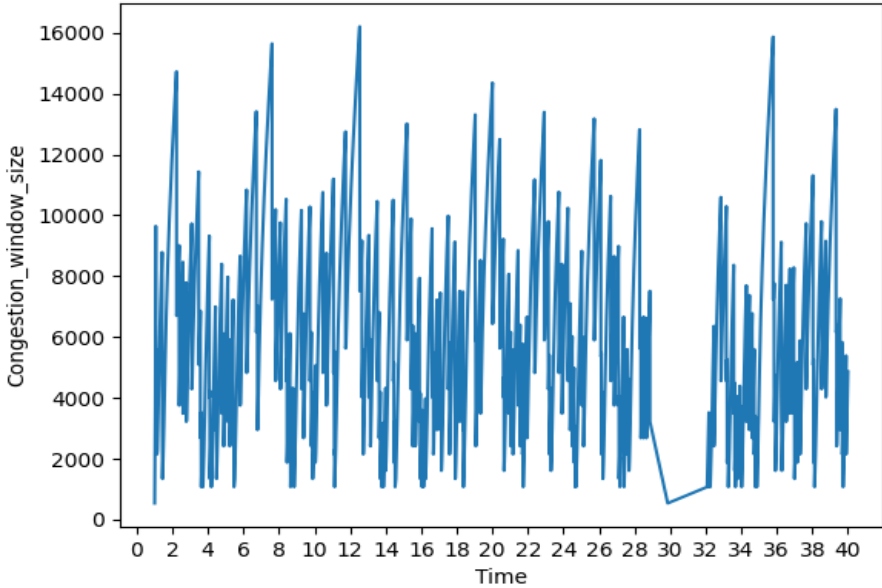
3 Notes

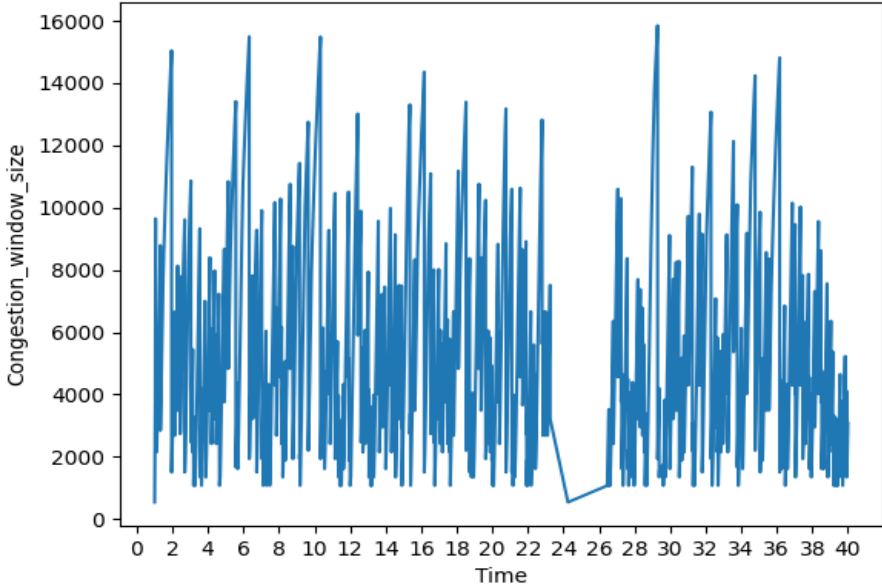
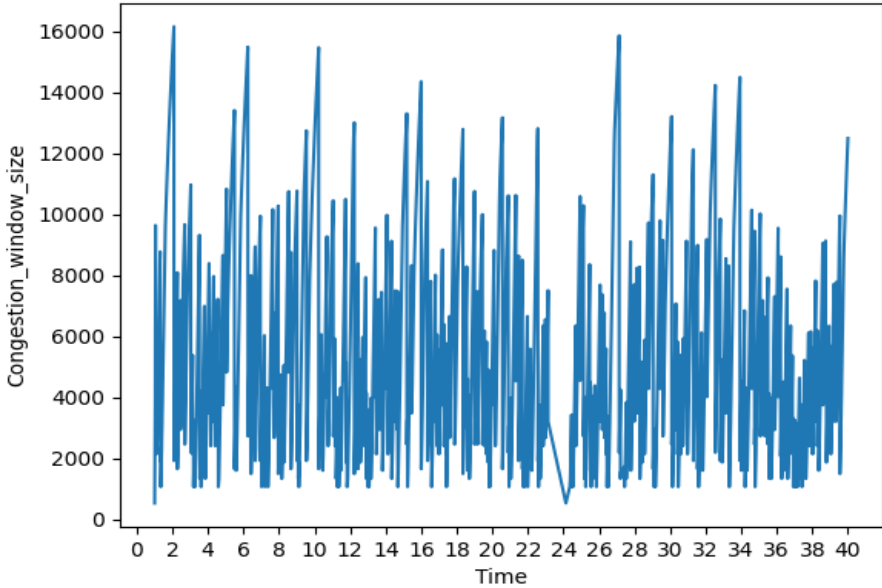
- For this task , I have submitted three files in Task II folder. *cwnd.cc* is edited from Task I *cwnd.cc* file and *plot_graph.py* files will plot the graph for the outputs generated by *cwnd.cc* file.
- To verify the plots and data, Put the *cwnd.cc* and *plot_graph.py* the files in the *./ns - allinone - 3.29/ns - 3.29/scratch* folder and *run.sh* file in *./ns - allinone - 3.29/ns - 3.29/* folder and open the terminal in *./ns - allinone - 3.29/ns - 3.29/* and run

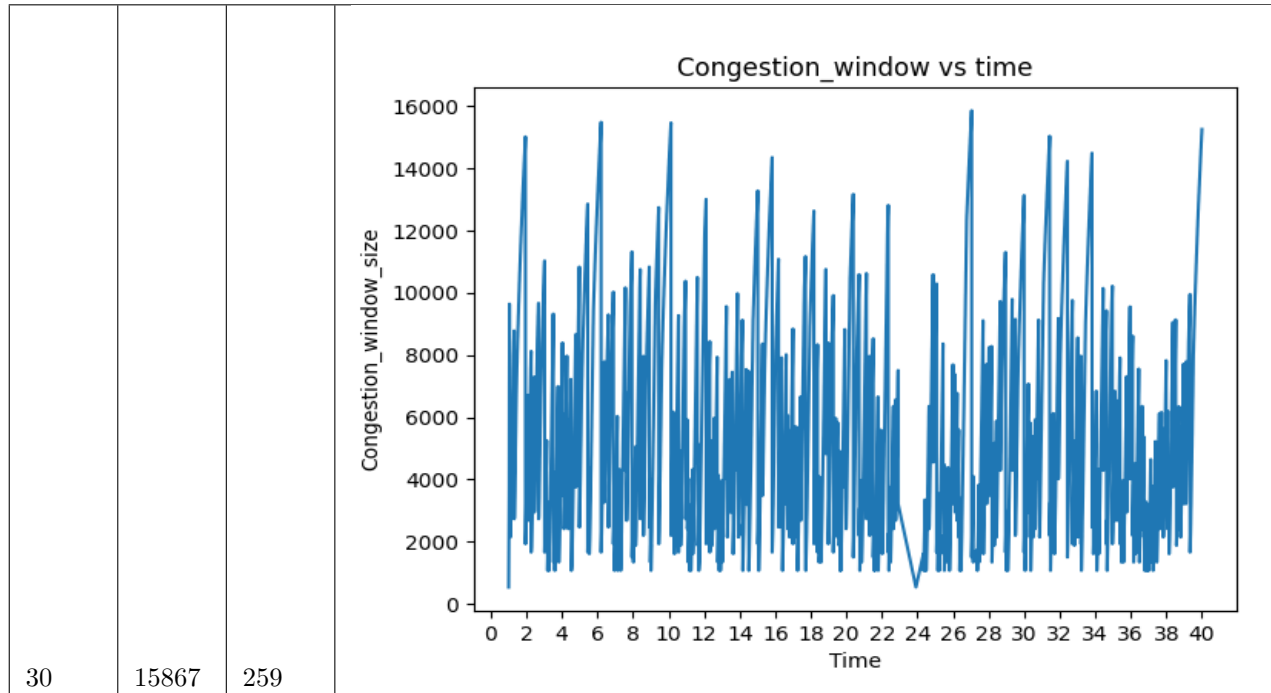
```
1 ~/ns-allinone-3.29/ns-3.29 $ bash run.sh <application data
   rate in Mbps(example - 5)> <Channel data rate in Mbps(
   example - 3)>
2
```

3.1 Part(a) - Application data rate is 5Mbps on NewReno

Channel Rate(in Mbps)	Max. win-dow size	No. of Pack-ets drop	Graph
-----------------------	-------------------	----------------------	-------

3	16206	131	<p>Congestion_window vs time</p> 
5	16206	194	<p>Congestion_window vs time</p> 

10	15849	246	<p>Congestion_window vs time</p> 
15	16162	260	<p>Congestion_window vs time</p> 



Question

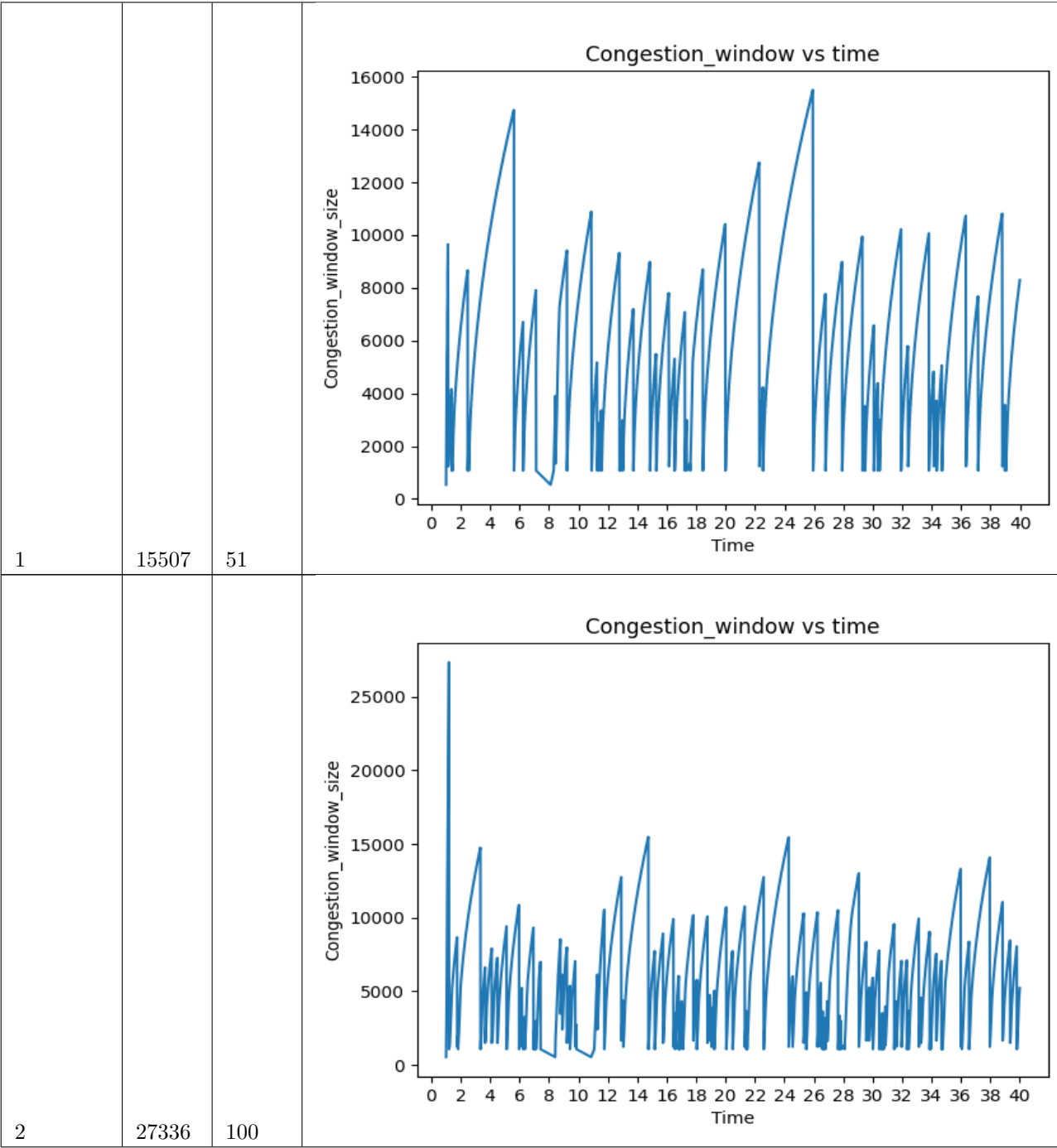
How does channel data rate affect congestion window size.

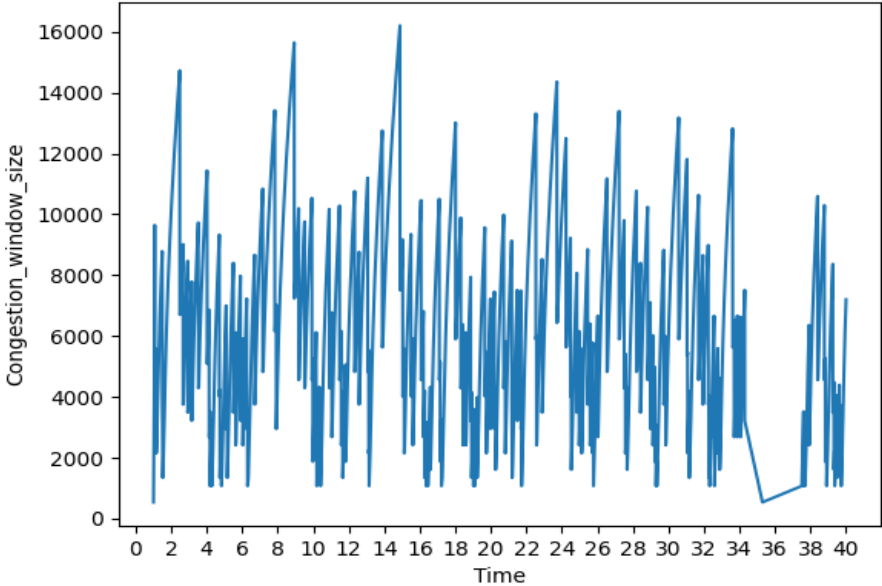
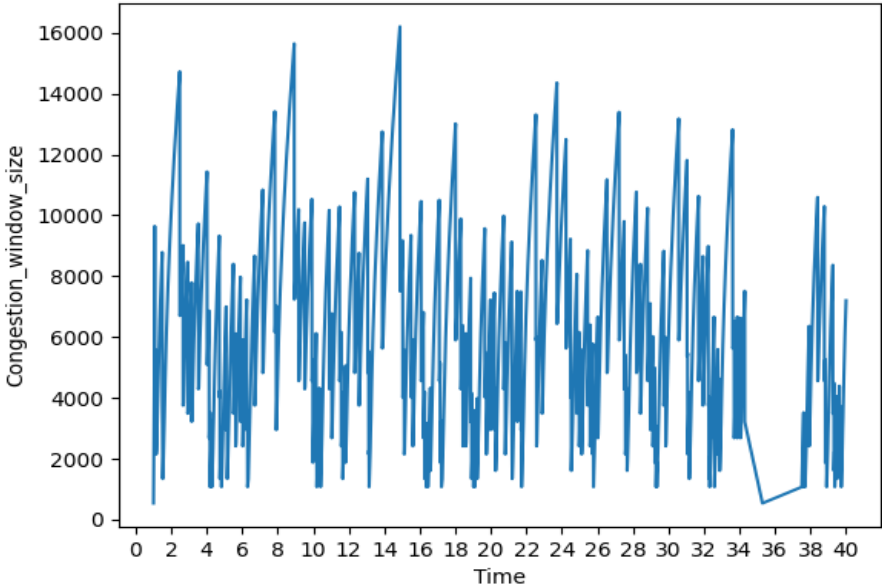
Ans. Firstly, number of drop packets increases as channel rate increases upto 15 and then it decrease when channel rate becomes 50. This is also visible from the graph because width of peaks gets decreases as the channel rate gets increased

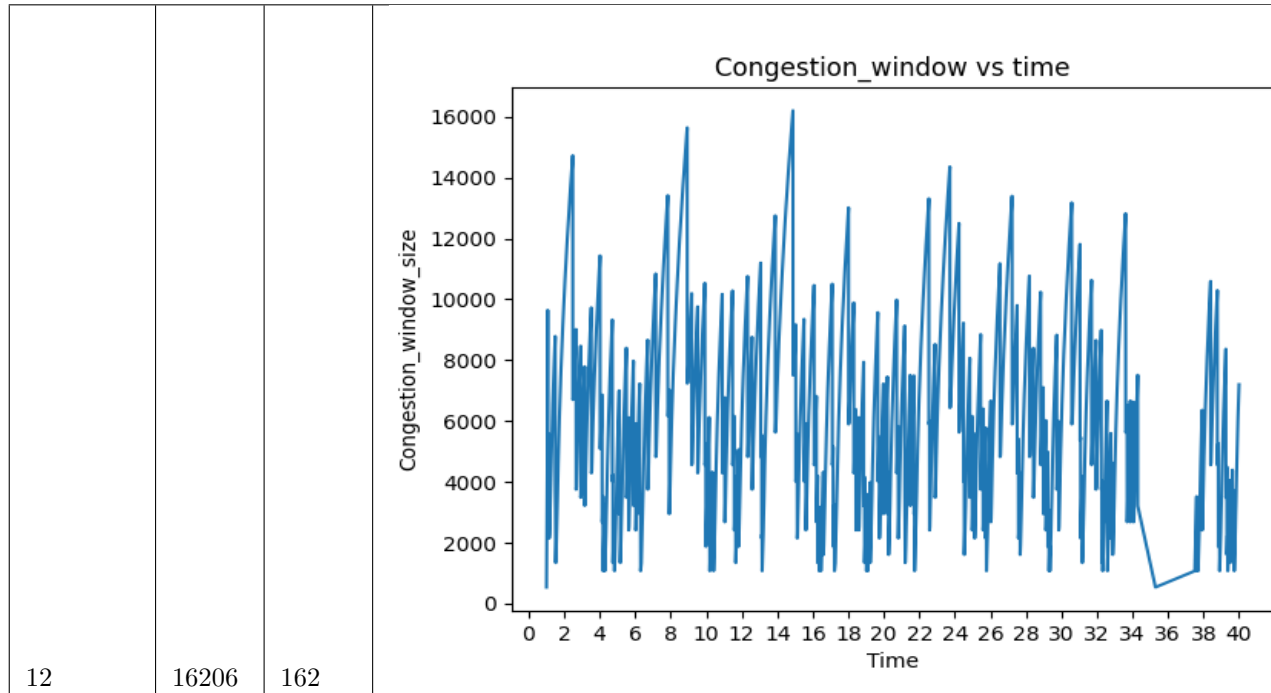
Maximum congestion window is higher and number of peak drops to 1(or minimum) is lesser for lower channel rates . This is because for lower channel rates(i.e 3Mbps), the protocol is in congestion avoidance phase for larger period and it goes to fast recovery phase more often than for higher channel rates.

3.2 Part(b) - Channel data rate is 4Mbps on NewReno

Application Rate(in Mbps)	Max. win-dow size	No. of Pack-ets drop	Graph
---------------------------	-------------------	----------------------	-------



4	16206	162	<p>Congestion_window vs time</p> 
8	16206	162	<p>Congestion_window vs time</p> 



Question

How does application data rate affect congestion window size.

Ans. Number of drop packets keep increasing as the application rate increase till 4Mbps. Then it stays constant upto 12Mbps. Similar trend is seen for maximum congestion window. Graph is nearly same(maybe exactly same) for application data rate greater than or equal to channel rates(i.e for 4Mbps,8Mbps and 12Mbps)

Maximum congestion window is highest for 2Mbps which happens just at the start of simulation. This can be because of congestion window size being just less than threshold value and then it gets doubles giving the maximum congestion window size and then it shifts to congestion avoidance phase ,where immediate packet drop happens(for example -because of timeout) which results in peak dropping to 1(ie minimum) . Also width of peaks gets decreases as the application rate gets increased

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

Channel Rate: It is the maximum data transmission capacity of channel.

Application data rate: It is the actual data transmission speed.

We observed keeping one fixed and increasing other, increases the number of dropped packets and decreases the width of peaks. We can also see that when application rate and channel rates do not differ much and application rate is lesser than channel rate then there are less drops and protocol has wide peaks, i.e they remain longer in congestion avoidance phase. This can be depicted by first two plots of both the parts (where application rate \neq channel rate).

4 Task III

4.1 Notes

- For this task , I have submitted all the files in Task III folder.
- To verify the plots and data, Put the *cwnd.cc* and *plot_graph.py* the files in the *./ns - allinone - 3.29/ns - 3.29/scratch* folder and *run.sh* file in *./ns - allinone - 3.29/ns - 3.29/* folder and open the terminal in *./ns - allinone - 3.29/ns - 3.29/* and run

```
1 ~/ns-allinone-3.29/ns-3.29 $ bash run.sh
2
```

- I have created two app objects . One for Tcp connection(Node 1 as source and Node 4 as sink) and other for Udp(Node 2 as source as source and Node 5 as sink).
- I used a Schedule() function to change the application data rate for the Udp connection, for which i created a public function in MyApp class named Change_Rate().
- **"Schedule methods allows us to schedule an event in the future by providing the delay between the current simulation time and the expiration date of the target event."**
- I enabled the global static routing as Tcp and Udp connection needs to be routed.

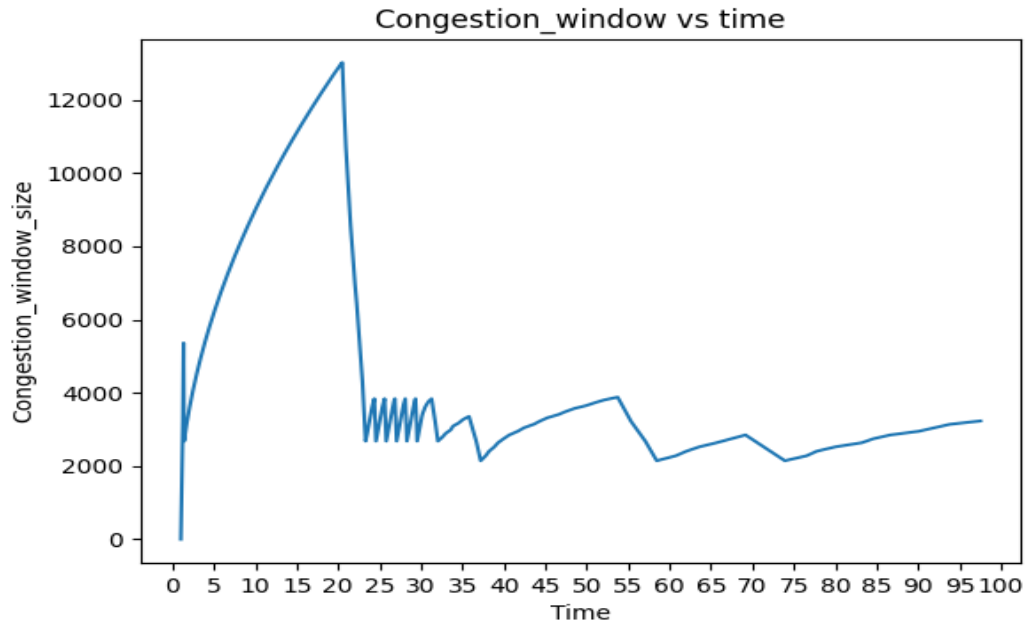
```
1 Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
2
```

- To generate pcap files, I enabled pcap tracing

```
1 pointToPoint.EnablePcapAll ("scratch/PCAP");
2
```

4.2 Questions

- Plot Congestion window size vs time graph for TCP connection



- Indicate the points, where you observe the effect in the congestion window size of TCP connection, when UDP connection is started and when we change the data rate of UDP connection

Max_congestion_window : 13017

We can observe that initially the congestion window size keeps on increasing till the time Udp connection has not started. At around $t=22s$, we can see that there are lots of packet drops in the protocol, this is because Udp connection clogs the half of the bandwidth of N2-N3 link. Later when application rate of Udp connection is further increased, we can see that congestion window size decreases by much from the maximum size. Also peaks are wide i.e protocol remains in congestion avoidance phase for a longer period of time.