# COL334 Assignment – 3

Ishaan Watts
2019PH10629

# Task 1

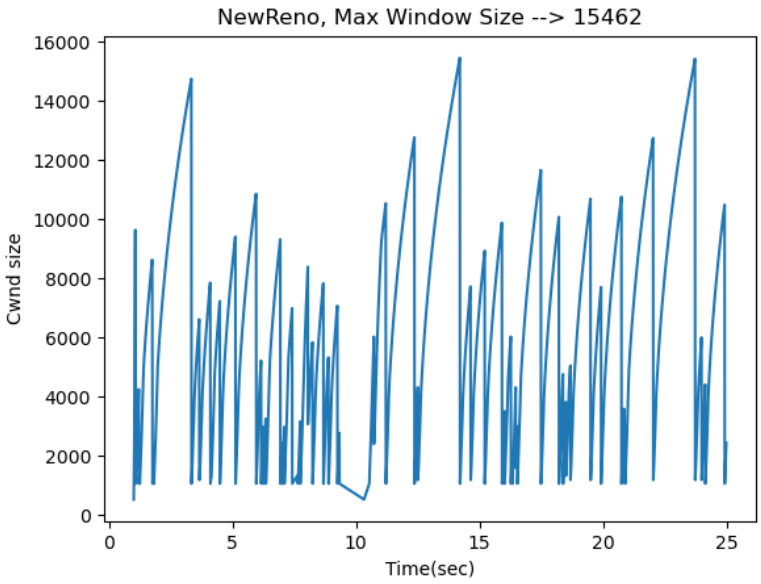
## 1. Congestion Protocol Table

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

We can see that packets dropped are similar for all protocols. This is because e had the same *ReceiveErrorModel* to drop packets at the sink and all other parameters like channel data rate, application data rate, packet size and channel delay are also same. Moreover, only Vegas has a significantly different max. window size.

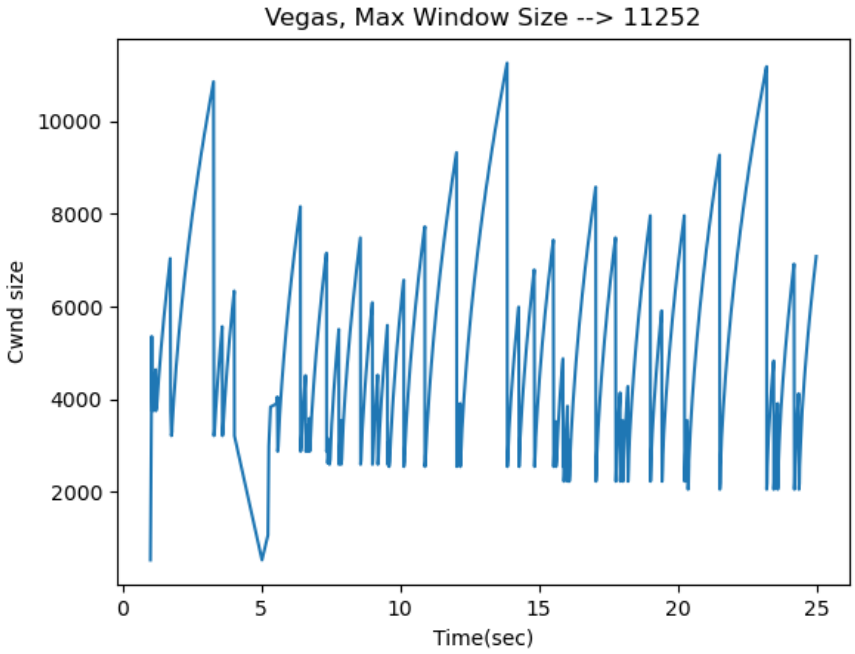
## 2. Time v/s Congestion Window Size plots

### a. NewReno protocol



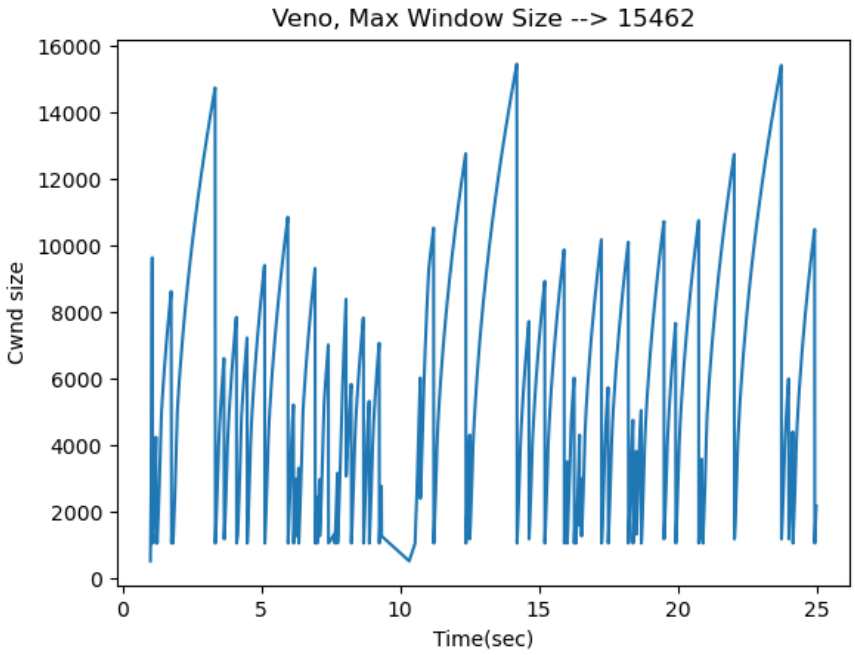

We can see congestion in the network as window size drops to a low value.

## b. Vegas protocol



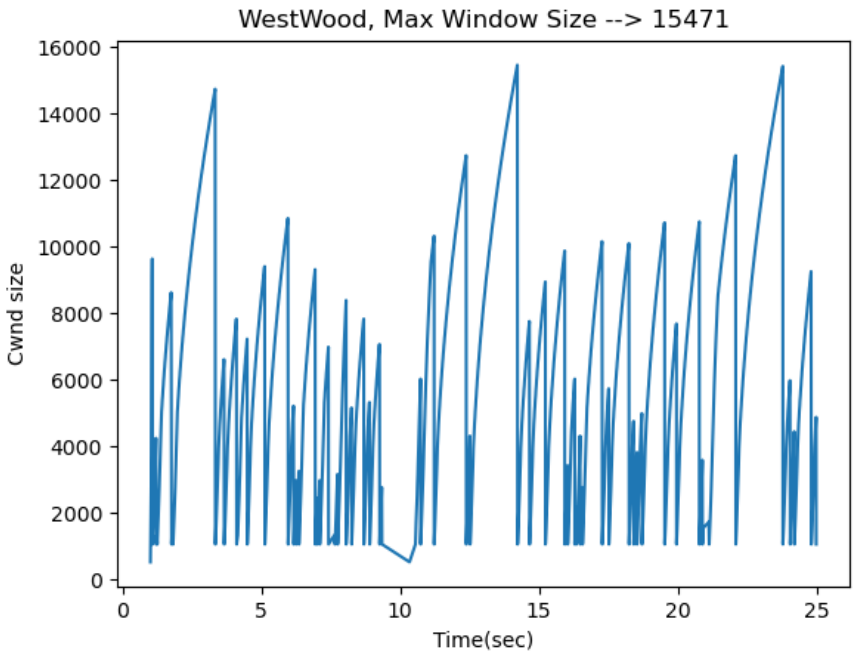

We can again observe congestion in the network but with a decrease in maximum window size. We also observe that the window size drops to a higher value here.

## c. Veno protocol




This plot is similar to TCP NewReno. Similar max window size is observed.

### d. WestWood protocol




This plot is again similar to TCP NewReno. Similar max window size is observed.

## 3. Comparison between Protocols

**a. TCP NewReno:** It is an extension of TCP Reno. Packet loss is measured to alter the window size using *AIMD (additive increase and multiplicative decrease).* During slow start, the cwnd increases by MSS for each ACK received and it ends when cwnd exceeds ssthresh. In congestion avoidance there is a linear increase and ssthresh is halved when congestion occurs. It improves over TCP Reno to deal with multiple packet loss by using *partial ACKs*.

**b. TCP Vegas:** This protocol uses *packet delay* instead of packet loss to detect congestion. Since it is able to anticipate congestion based on RTT values, it is able to detect congestion earlier and we observe a *lesser max. congestion window size*. It is characterized by extra data (expected output – actual output)*baseRTT, alpha and beta values. If extra data is greater than beta then cwnd is

linearly increased while if it is less than alpha than cwnd is linearly increased.

c. **TCP Veno:** It is also a modification of TCP Reno and refines AIMD. It is designed to *improve performance of TCP in wireless links*. Since we are using a point-to-point topology with the RecieveErrorModel at sink, we observe a similar plot to TCP NewReno.

d. **TCP WestWood:** It is again a sender-side modification to TCP Reno to better handle large bandwidth-delay product due to transmission errors and dynamic loads. Since here we have a constant load, we observe a similar plot to TCP NewReno.

4. **BBR**

BBR is a *TCP delay-controlled TCP flow control algorithm* from Google. It is very similar to TCP Vegas which already differentiates it from the rest of the three protocols described above.

BBR determines the underlying bottleneck available bandwidth and path RTT by a number of factors in addition to the data being passed through the network for this particular flow.

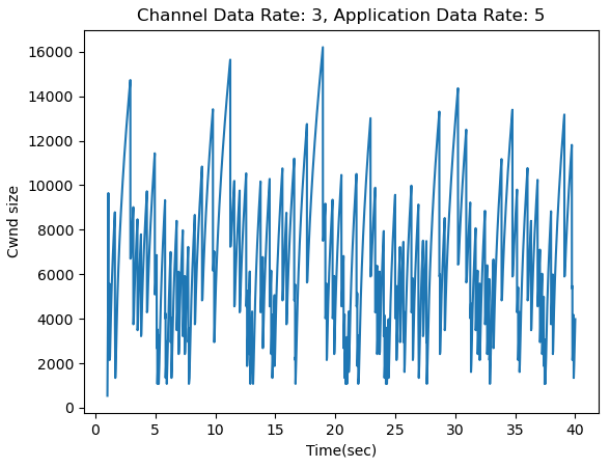
BBR's overall adaption to increased bandwidth on the path is *exponential* rather than the linear adaptation used by TCP Vegas. It regularly probes the path.

## Additional Info:

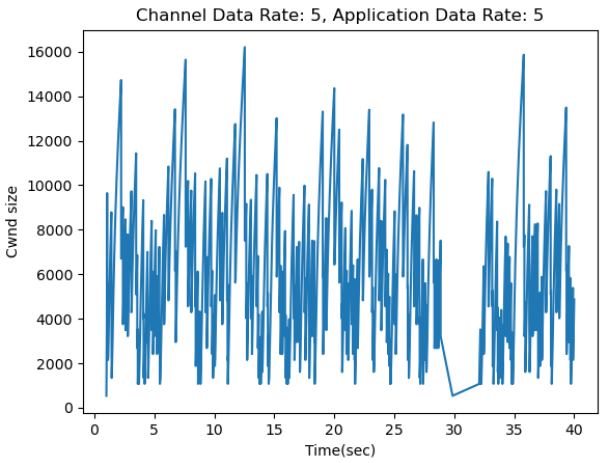
1. Code used as reference was *sixth.cc example file* in ns-3.
2. To count packet loss an int type variable was initialized and it was incremented inside the *RxDrop function* which logs the packet drop instances.

# Task 2

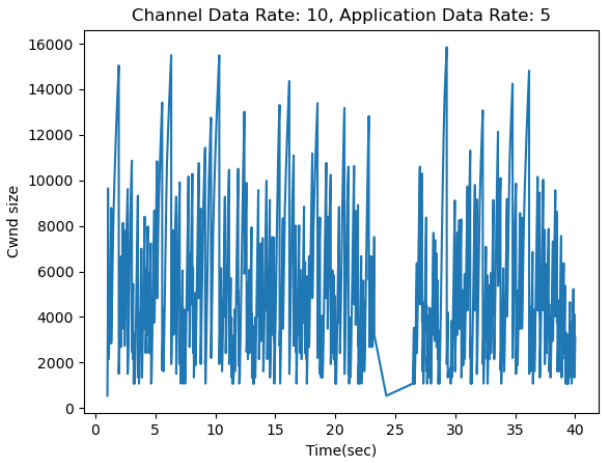
## 1. Constant Application Data Rate (5Mbps)



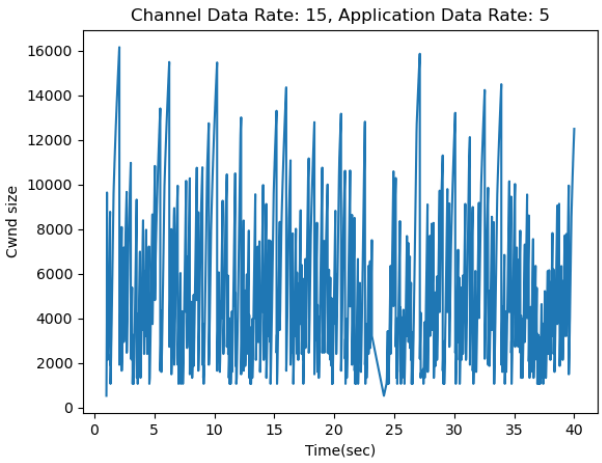

a. 3Mbps Channel Data Rate



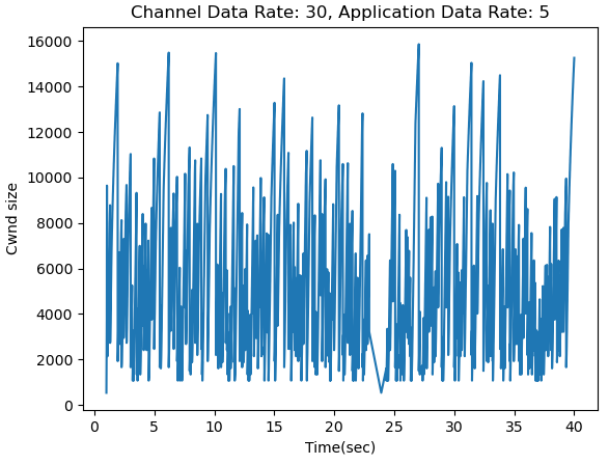

b. 5Mbps Channel Data Rate




c. 10Mbps Channel Data Rate

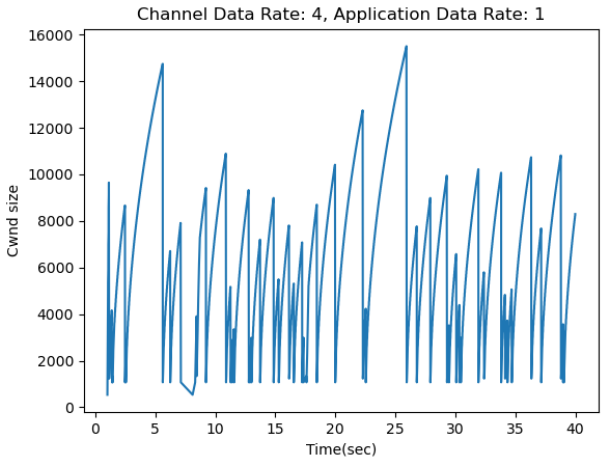



d. 15Mbps Channel Data Rate

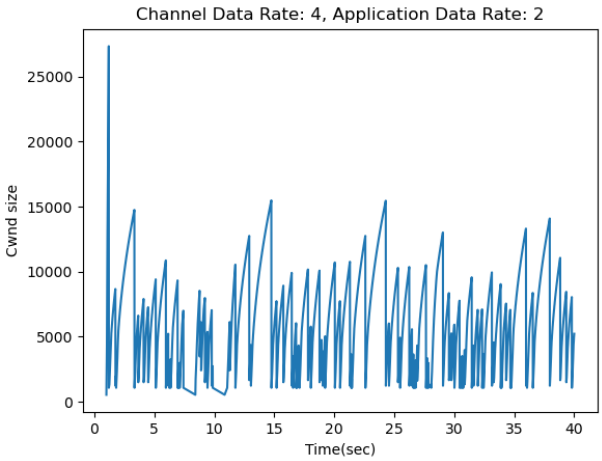



e. 30Mbps Channel Data Rate

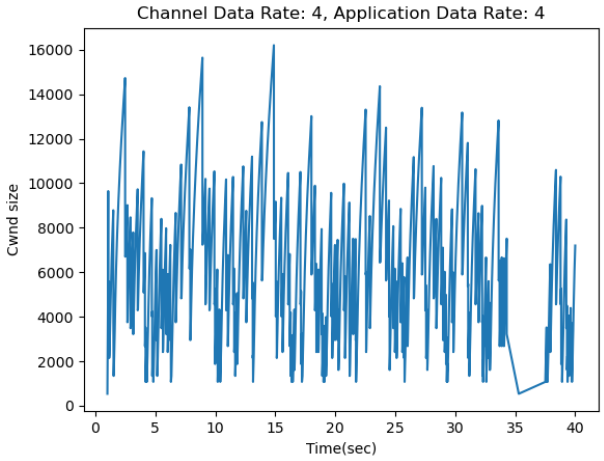
## 2. Constant Channel Data Rate (4Mbps)



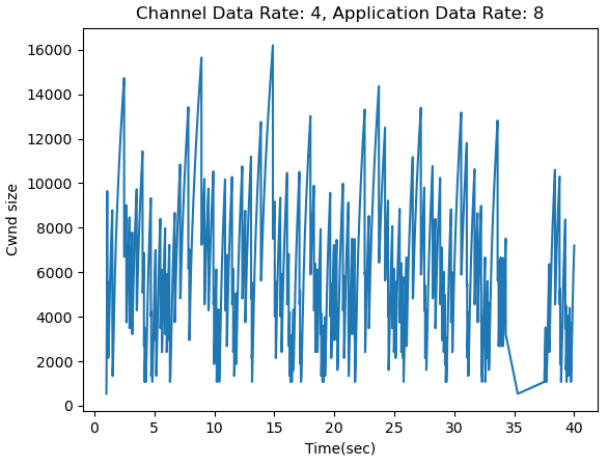

a. 1Mbps Application Data Rate



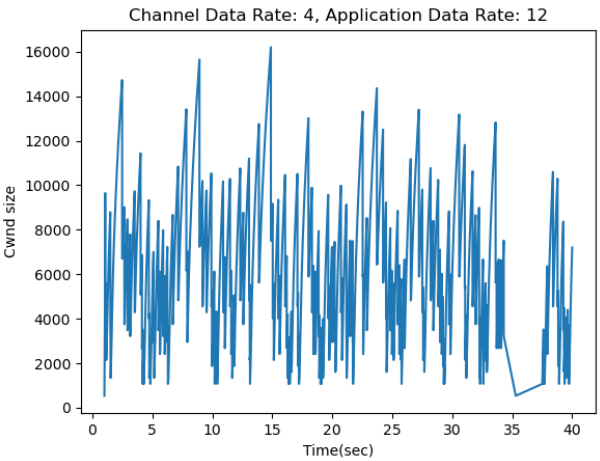

b. 2Mbps Application Data Rate




c. 4Mbps Application Data Rate




c. 8Mbps Application Data Rate




e. 12Mbps Application Data Rate

- How does application data rate affect congestion window size?

We have fixed the channel data rate at 4Mbps. When application data rate is increased from 1Mbps to 12Mbps, we can see that the frequency of packet drop also increases. The graph remains similar when the application data rate is beyond the channel data rate. The max congestion window size is same but frequency of change in window size increases.

- How does channel data rate affect congestion window size.

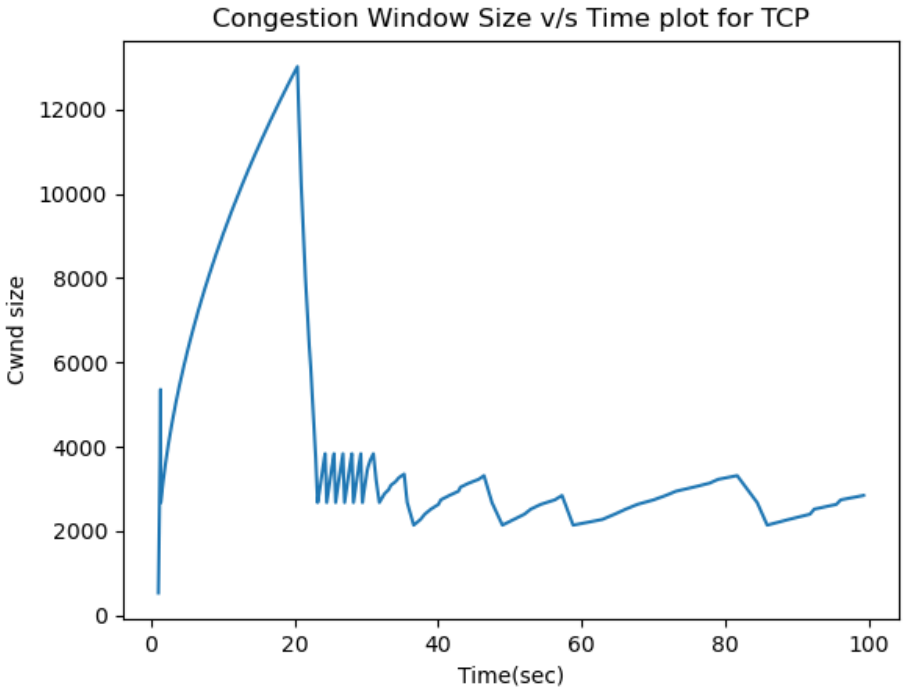
Here we have fixed the application data rate at 5Mbps. We have varied the channel data rate from 3Mbps to 30Mbps. Apart from the 3Mbps case where application rate is greater than channel rate, the graphs are identical showing the size of the congestion window will remain same when there is enough capacity in the link.

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

We can observe that when the application data rate is less than the channel data rate, there is movement of the congestion window with less frequency meaning less packet loss. As the application data rate increases the or comes near to the channel data rate then packet loss increases and there is violent oscillation and the plot becomes similar.

## Task 3

1. Congestion window size vs time graph for TCP connection




2. We can see that the window size sharply decreases after 20secs as soon as the UDP connection is switched on with data rate 250Kbps. There is fast of oscillation of the window size till 30secs. After this when the UDP data rate is increased to 500Kbps, we can see the max window size decreases further and the increase is also slower than before.

3. There are 4 devices in the topology with 2 nodes each. Hence, I have generated 8 pcap files.

## Included Files

1. Task 1 includes 2 files --> task1.cc and part1.py
    a. **task1.cc** : It contains the code to run the simulation and generate a csv file in the home directory containing time vs cwnd size values. It asks to choose a protocol upon running as input.
    b. **part1.py** : It reads all the 4 protocol .csv files and generates a .png plot in the same directory.

2. Task 2 includes 2 files --> task2.cc and part2.py
    a. **task2.cc** : It contains the code to run the simulation and generate a csv file in the home directory containing time vs cwnd size values. It asks to input the channel and application data rates upon running as input.
    b. **part2.py** : It reads all the 10 .csv files with different application and channel data rates and generates a .png plot in the same directory.

3. Task 3 includes 2 files --> task3.cc and part3.py
    a. **task3.cc** : It contains the code to run the simulation and generate a csv file in the home directory containing time vs cwnd size values. It also generates 2 pcap file for each device (total 8).
    b. **part3.py :** It reads the .csv file and generates a .png plot in the same directory.
    c. **task3-{$n_o$}-{$n_1$}.pcap:** 8 .pcap files are generated. 1 for node 0, 2 for node 1, 3 for node 2, 1 for node 4 and 1 for node 5.