COL334 Assignment 3

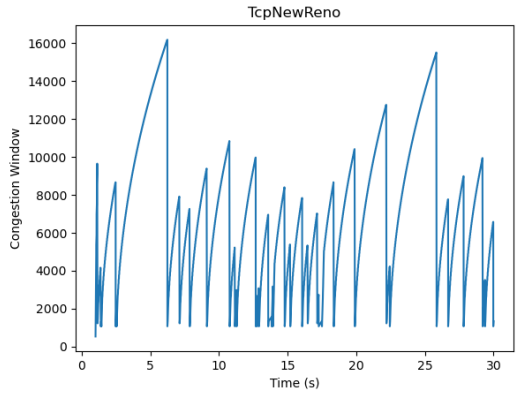
Sayam Sethi

October 2021

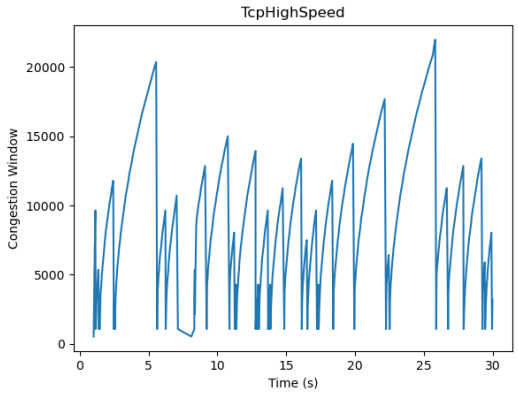
Contents

1 Part 1

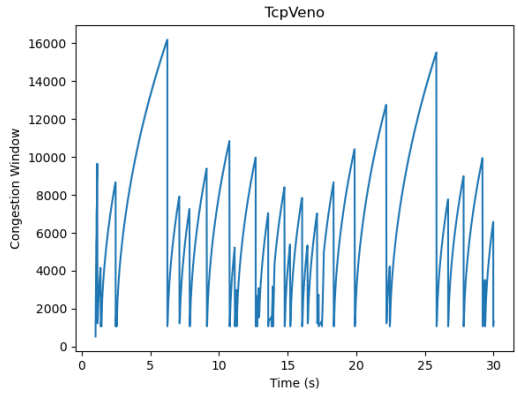
1.1 Plots



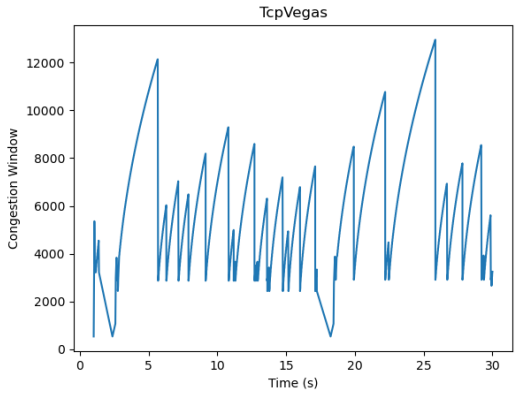
(a) TcpNewReno



(b) TcpHighSpeed



(a) TcpVeno



(b) TcpVegas

1.2 Dropped Packets

1. **TcpNewReno:** There are 38 dropped packets
2. **TcpHighSpeed:** This also has 38 dropped packets
3. **TcpVeno:** This also has 38 dropped packets
4. **TcpVegas:** This has 39 dropped packets

1.3 Observations

1.3.1 TcpNewReno

- There is linear increase in the congestion window in the slow start phase.

- This changes to slow increase in the congestion avoidance phase, which leads to lesser chances of packet drop.
- The slow start threshold is halved with each drop which leads to the decrease in the maximum congestion window after each drop until it start increasing again after it becomes small.

1.3.2 TcpHighSpeed

- This protocol does not visibly have any slow start phase, which leads to the large increase in the congestion window initially and that stabilises once the congestion window is relatively higher.
- In the case of packet drop, the congestion window decreases as a function of the congestion window. Therefore, there is lesser decrease in the congestion window in case of packet drop when the congestion window is smaller.
- Therefore, this model will be suitable for links which have high data link capacity since it handles packet drops efficiently when the congestion window is larger.

1.3.3 TcpVeno

- The plot is exactly the same as the plot of `TcpNewReno`.
- On examining the implementation of `TcpVeno` further, we notice that the implementation calls corresponding functions of `TcpNewReno` with a difference being in the handling of packet drop.
- The different implementation in handling drops is not visible for the duration of the simulation, therefore we can conclude that `TcpVeno` is almost the same as `TcpNewReno` in practical scenarios for small connection durations.

1.3.4 TcpVegas

- This protocol works on the basis of delay instead of the packet drops.
- Therefore, the model is *conservative* in increasing the congestion window and when the delay increases, it avoids it by decreasing the congestion window.

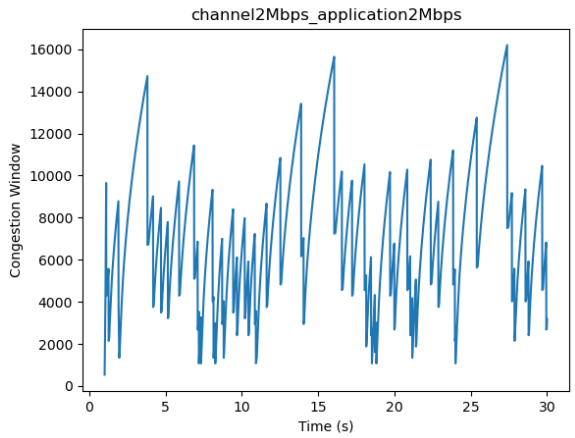
2 Part 2

2.1 Varying Channel Rates

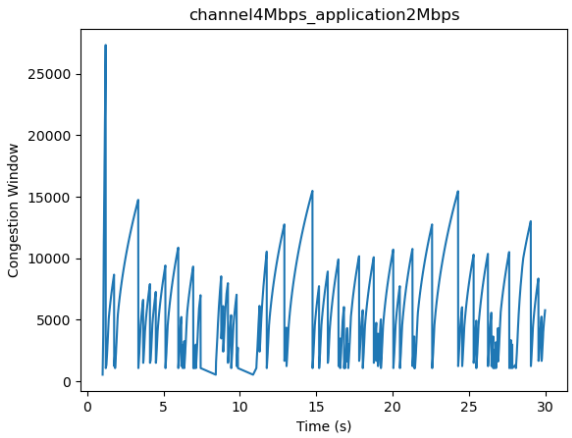
2.1.1 Observation

From the plots given in Section 2.1.2, we notice that the amount of variation in the congestion window per second increases. This is because it becomes possible to transmit more number of packets over the link. The only limitation is the *application rate* which is set at 2Mbps. This is also consistent with the last three graphs being almost identical. Additionally, we state that the large variation in congestion window for larger channel rates is because of larger frequency of packet drops. This happens since even though the error model is unchanged, the number of packets being transmitted each second increases.

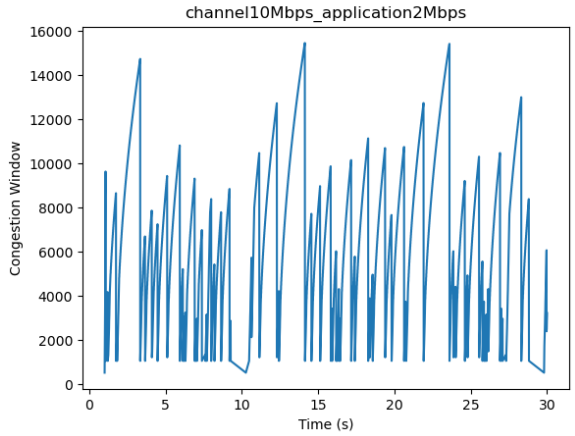
2.1.2 Plots



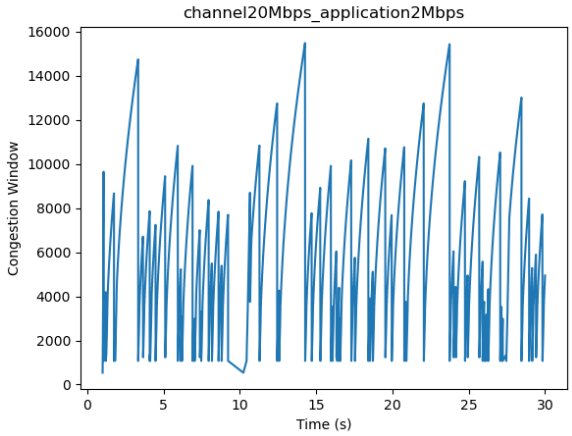
(a) Link rate = 2Mbps



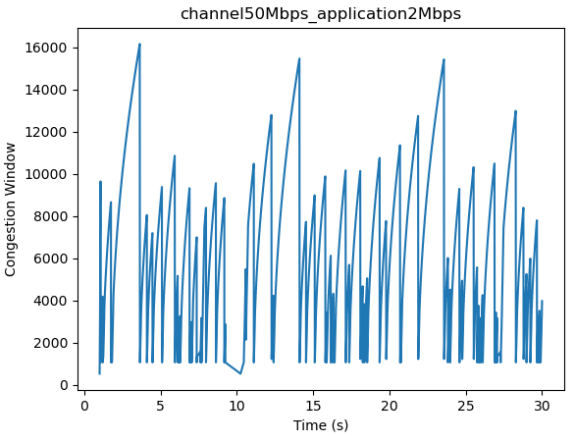
(b) Link rate = 4Mbps



(c) Link rate = 10Mbps



(d) Link rate = 20Mbps



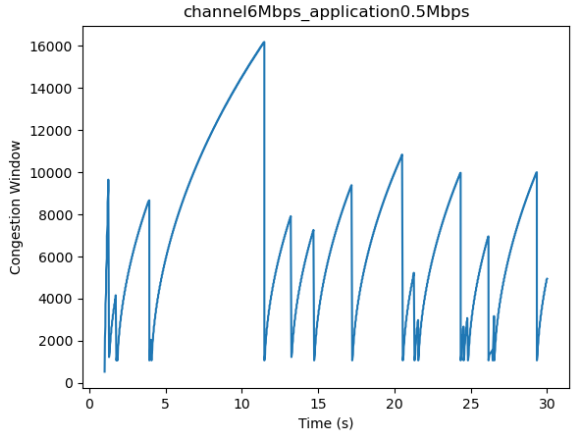
(e) Link rate = 50Mbps

2.2 Varying Application Rates

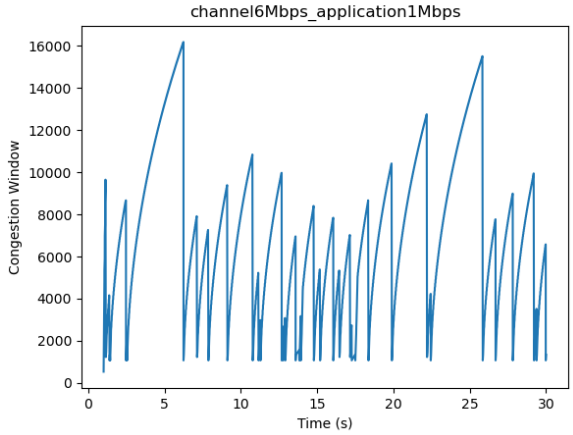
2.2.1 Observation

From the plots in Section 2.2.2, we observe that for smaller application rates, the congestion avoidance phase stays for a longer time. However, as the application rate increases, the shift happens towards the slow start phase. From this, we infer that the drop rate increases for larger application rates. The source of congestion in this case becomes the link as the application rate rises. For the largest application rate, there is a region of considerably low congestion window. This implies that the application has to slow down its functioning to accomodate the congestion in the network.

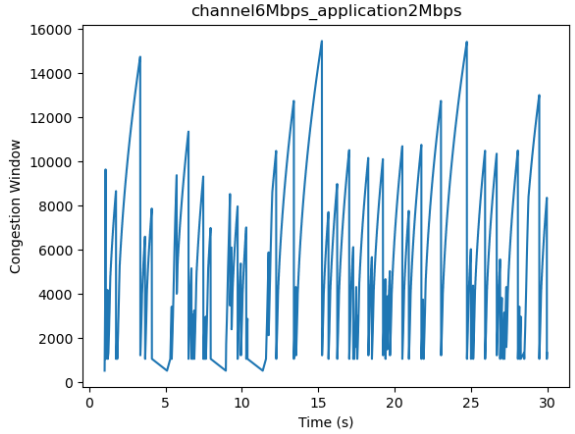
2.2.2 Plots



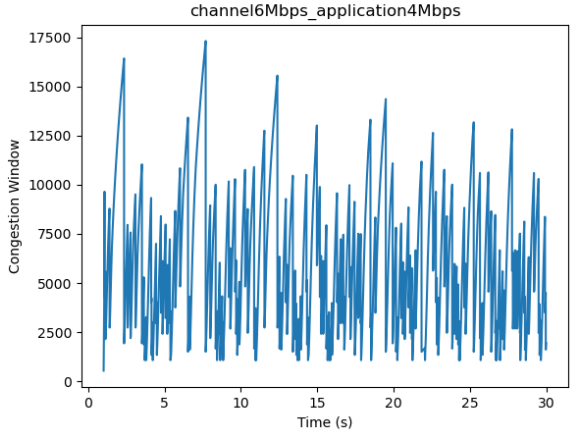
(a) Link rate = 2Mbps



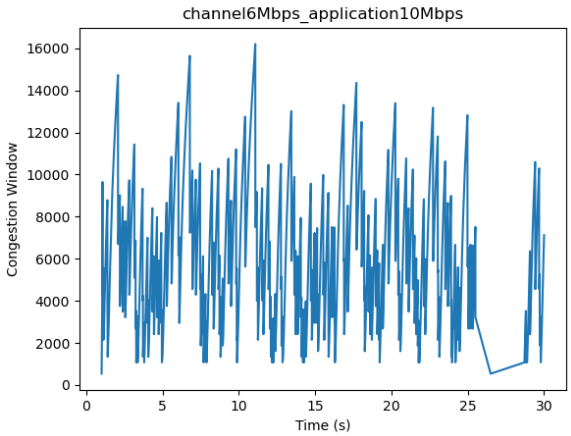
(b) Link rate = 4Mbps



(c) Link rate = 10Mbps



(d) Link rate = 20Mbps

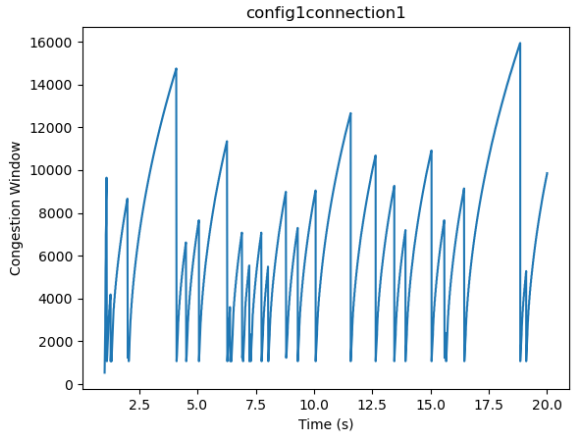


(e) Link rate = 50Mbps

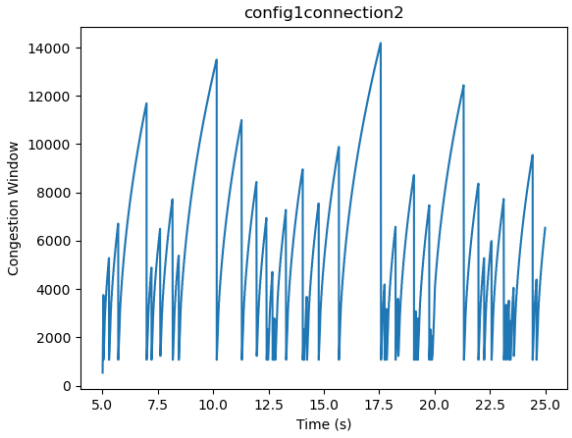
3 Part 3

3.1 Configuration 1

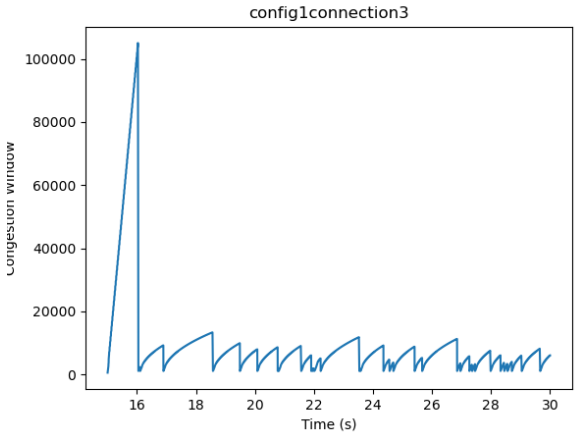
There are 113 dropped packets.



(a) Connection 1



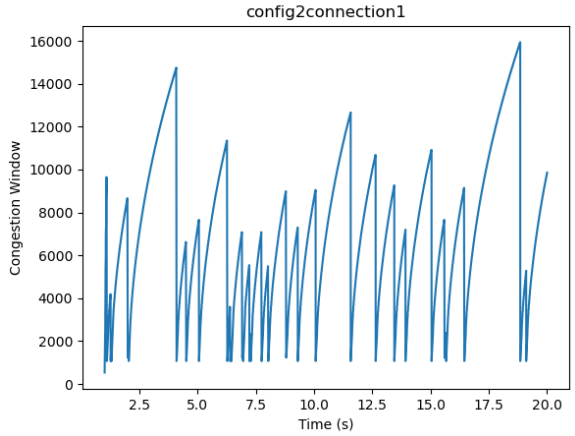
(b) Connection 2



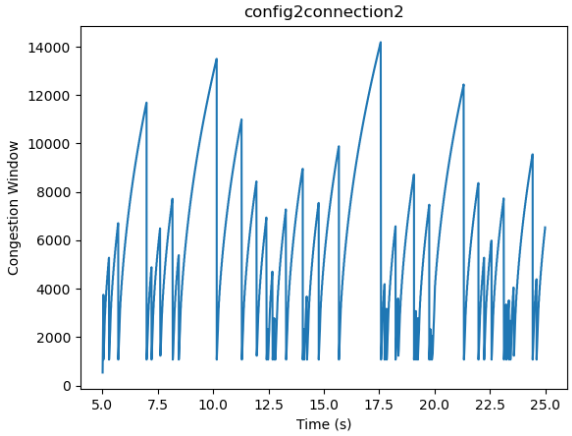
(c) Connection 3

3.2 Configuration 2

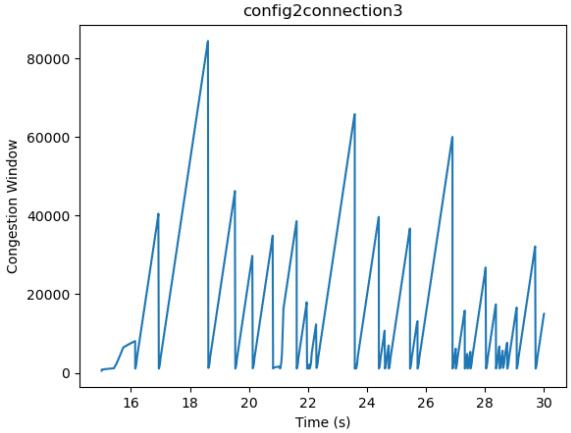
There are 112 dropped packets in this case.



(a) Connection 1



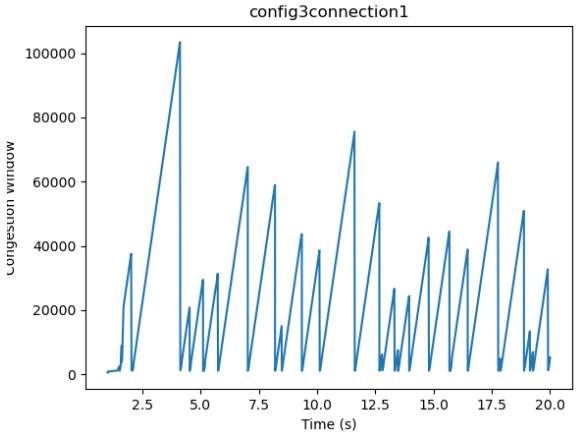
(b) Connection 2



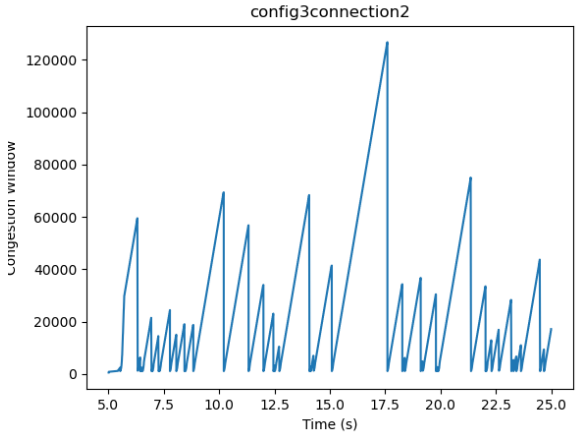
(c) Connection 3

3.3 Configuration 3

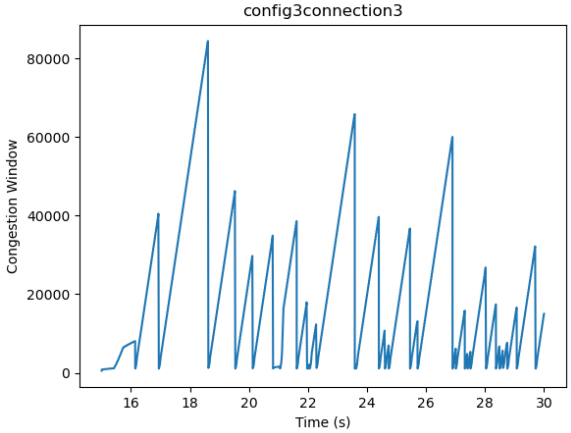
There are 110 dropped packets.



(a) Connection 1



(b) Connection 2



(c) Connection 3

3.4 Observations

Note: Two error models (with the same paramters) have been used, one for each link

1. The plots for connection $1, 2$ is the same for configurations 1 and 2 since the error models and links are independent.

2. Looking at the plots, we notice that the average congestion window size is higher whenever we use `TcpNewRenoCSE` instead of `TcpNewReno` for a single connection

3. Also, considering connection $1, 2$ for configurations 1 and 3, we notice that congestion window size reached much higher values in configuration 3 as compared to configuration 2.

4. Because of large number of packet drops on the first link, for both models, both the slow start as well as congestion avoidance states are achieved.

5. In the case of configuration 3, for connection 3, the slow start phase increases the congestion window slightly and then it shifts to congestion avoidance, thus the large linear increase.

6. However, in the case of configuration 1, for connection 3, the protocol largely remains in slow start phase which leads to relatively lower congestion window sizes.

4 Code Explanation

The code for all parts has been modified from `examples/tutorial/sixth.cc` provided in the `ns3` tutorial. Command line arguments have been taken and a shell script has been written for each part. To run the code, each folder needs to be copied to the `scratch` directory of the `ns3` folder after which entering the directory of corresponding part and running the shell script suffices. The plots have been made using `plot.py` file present in the root submission directory.