Networks assignment 2: TCP Congestion Control

Rishika Varma K CS18B045

March 2021

1 Aim

The aim of this assignment is to implement an emulation of the TCP congestion control using the Go back N sliding window protocol. This implementation does not involve cumulative acks but rather independent acks. Through this experiment we observe the effect of various parameters on the congestion window value. The receiving of packets is dictated by a probability that is given. We also do not consider re-transmission due to 3 duplicate acks. We only consider timeouts.

2 Introduction

Network congestion in data networking and queuing theory is the reduced quality of service that occurs when a network node or link is carrying more data than it can handle. Typical effects include queuing delay, packet loss or the blocking of new connections. A consequence of congestion is that an incremental increase in offered load leads either only to a small increase or even a decrease in network throughput.

Networks use congestion control and congestion avoidance techniques to try to avoid collapse. These include: exponential backoff in protocols such as CSMA/CA in 802.11 and the similar CSMA/CD in the original Ethernet, window reduction in TCP, and fair queueing in devices such as routers and network switches.

Transmission Control Protocol (TCP) uses a network congestion-avoidance algorithm that includes various aspects of an additive increase/multiplicative decrease (AIMD) scheme, along with other schemes including slow start and congestion window, to achieve congestion avoidance. The TCP congestion-avoidance algorithm is the primary basis for congestion control in the Internet.

In TCP, the congestion window is one of the factors that determines the number of bytes that can be sent out at any time. The congestion window is maintained by the sender and is a means of stopping a link between the sender and the receiver from becoming overloaded with too much traffic.

To avoid congestive collapse, TCP uses a multi-faceted congestion-control strategy. For each connection, TCP maintains a congestion window, limiting the total number of unacknowledged packets that may be in transit end-to-end. This is somewhat analogous to TCP's sliding window used for flow control. TCP uses a mechanism called slow start[1] to increase the congestion window after a connection is initialized or after a timeout. It starts with a window, a small multiple of the maximum segment size (MSS) in size. Although the initial rate is low, the rate of increase is very rapid; for every packet acknowledged, the congestion window increases by 1 MSS so that the congestion window effectively doubles for every round-trip time (RTT).

When the congestion window exceeds the slow-start threshold, ssthresh,[a] the algorithm enters a new state, called congestion avoidance. In congestion avoidance state, as long as non-duplicate ACKs are received[b] the congestion window is additively increased by one MSS every round-trip time. This corresponds to the AIMD algorithm described below.

3 Experiment Details

Here the congestion window size is calculated each time a packet is received or a timeout occurs. The receiving of the packet is simulated based on a probability that is input by us as part of the experiment. Each time there is a change in the cw size we print it to an output file and then plot these values to observe the variation over a series of values.

3.1 Simulation Setup

Here, to simulate receiving of packet or timing out, I used the Bernoulli's distribution generator in $c++$ with the given probability. The algorithm is implemented in $c++$ and compiled using $g++$. For testing, we plot the values obtained and for this a couple python scripts were written in which the matplotlib, sys, os libraries were used. These were compiled and run using python3. There is also a make file which is responsible for the compilation and cleaning for the corresponding files.

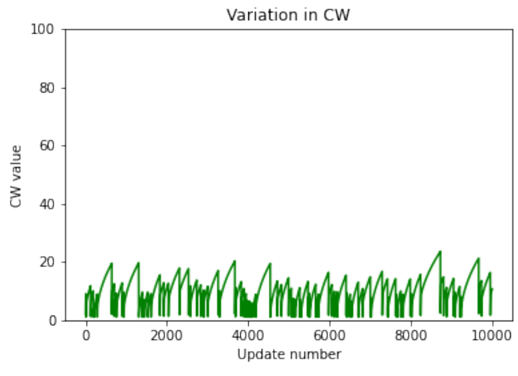
3.2 Entities involved and functions in each entity

There is only 1 entity is involved which is the cw.cpp code and it gives the cw value at each update. It only has the main function which takes care of predicting whether or not the packet was received, and correspondingly take action to change value of the cw size based on whether or not it is greater than the threshold value.

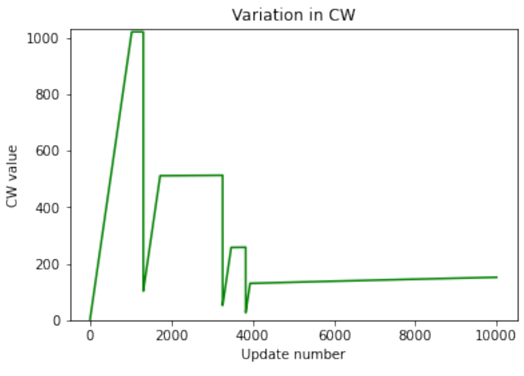
4 Results

The experiment is conducted for all combinations of the given values and plotted. The obtained plots are shown below.

Figure 1:

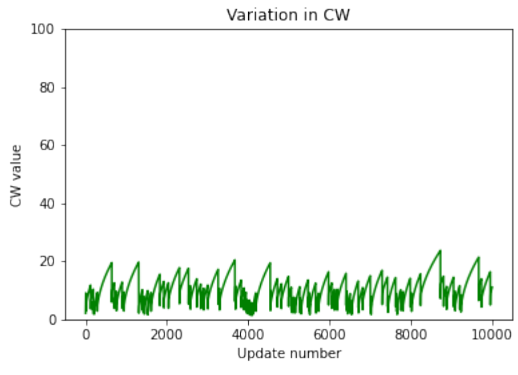


Ki= 1 Km= 1 Kn= 0.5 Kf= 0.1 Ps= 0.01

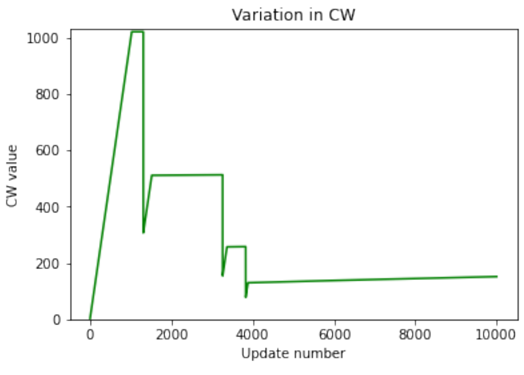


Ki= 1 Km= 1 Kn= 0.5 Kf= 0.1 Ps= 0.0001

Figure 2:

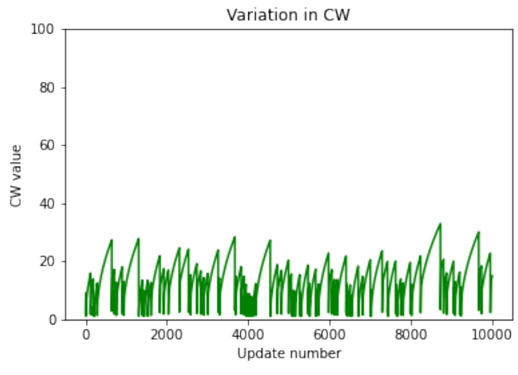


Ki= 1 Km= 1 Kn= 0.5 Kf= 0.3 Ps= 0.01

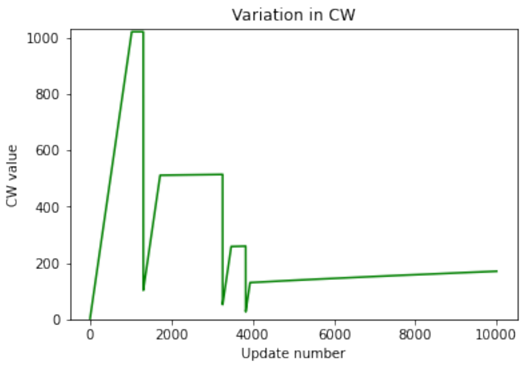


Ki= 1 Km= 1 Kn= 0.5 Kf= 0.3 Ps= 0.0001

Figure 3:



Ki= 1 Km= 1 Kn= 1 Kf= 0.1 Ps= 0.01



Ki= 1 Km= 1 Kn= 1 Kf= 0.1 Ps= 0.0001

Figure 4:

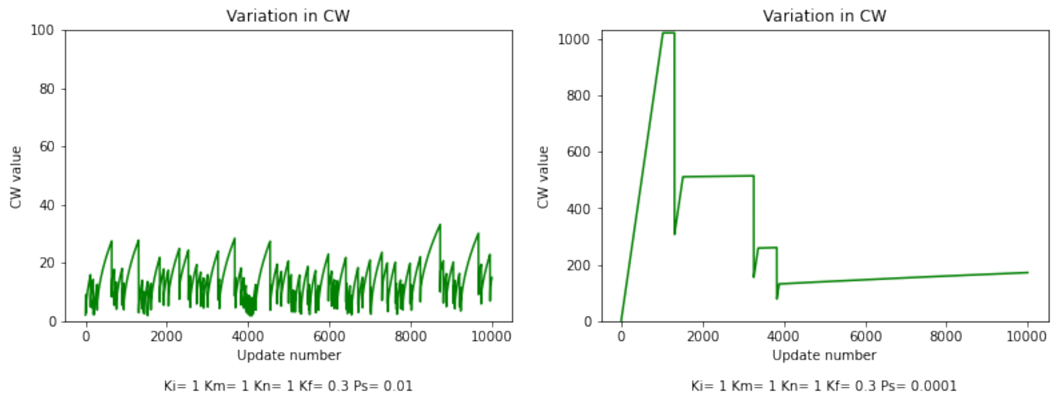


Figure 5:

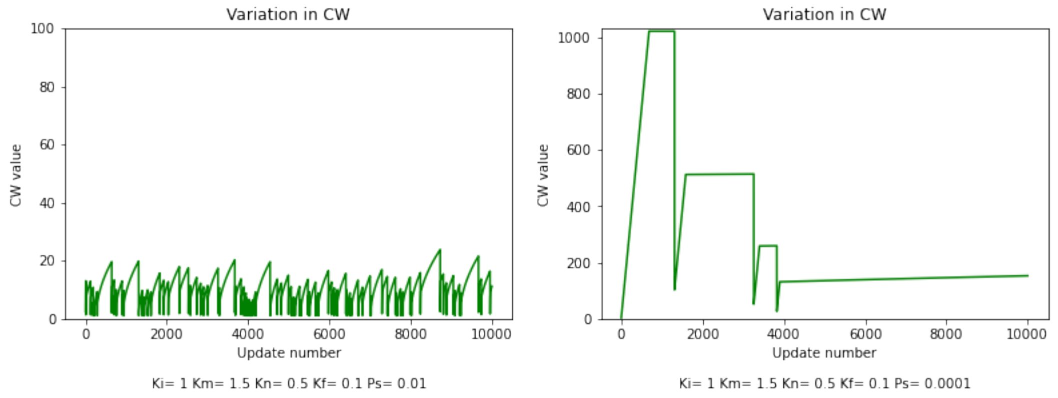


Figure 6:

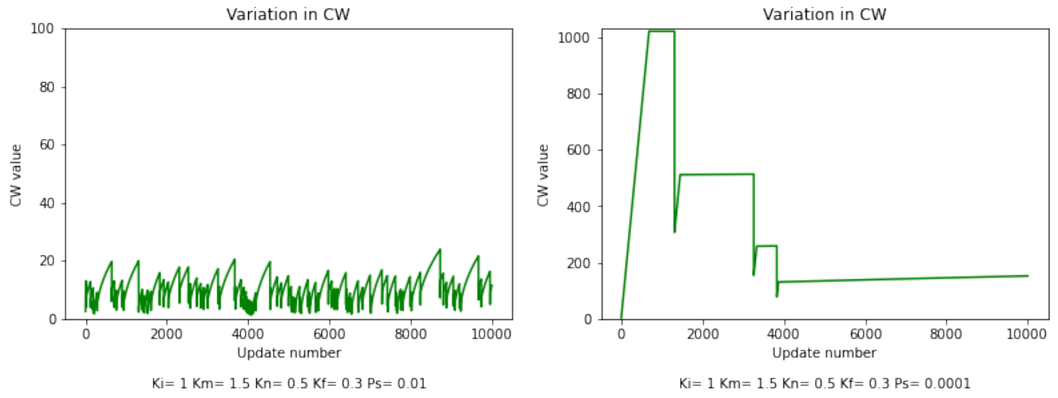


Figure 7:

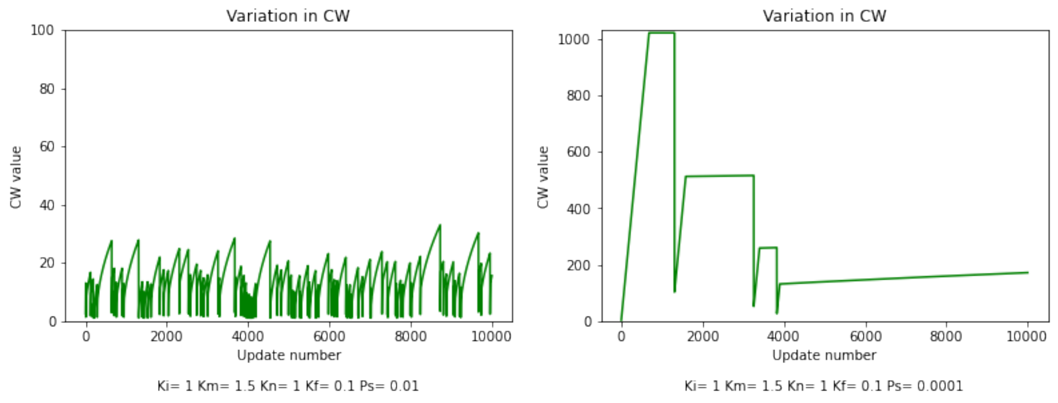


Figure 8:

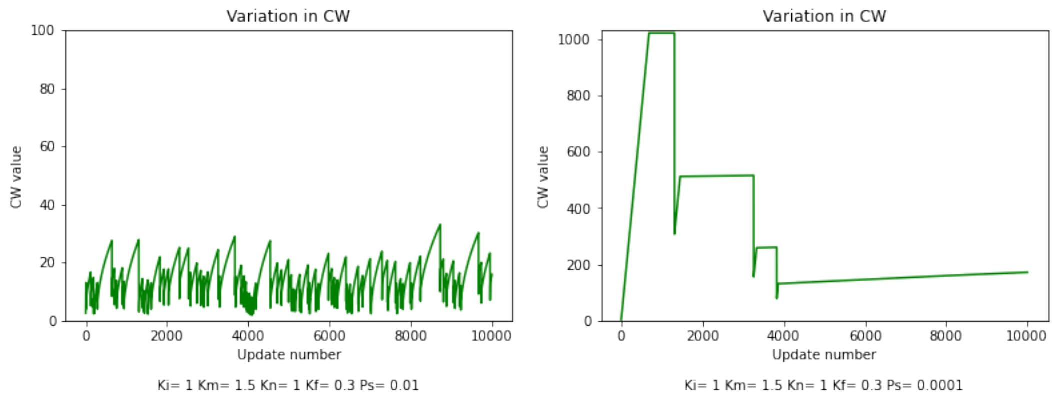


Figure 9:

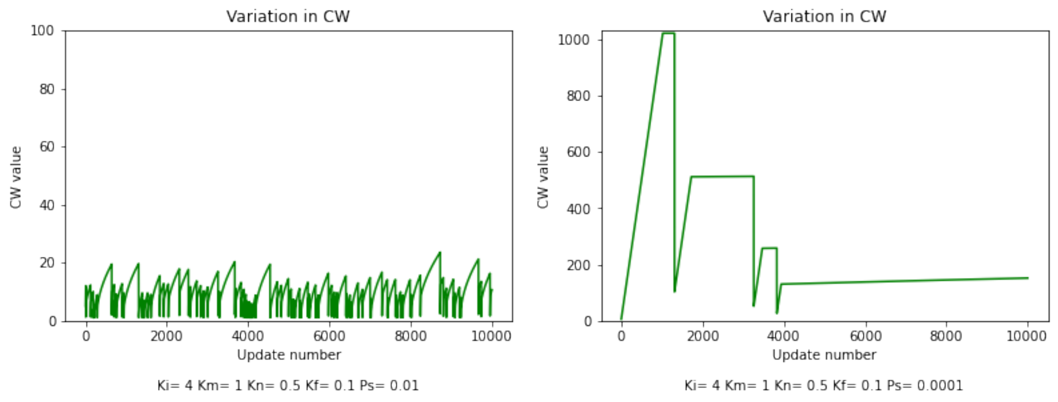


Figure 10:

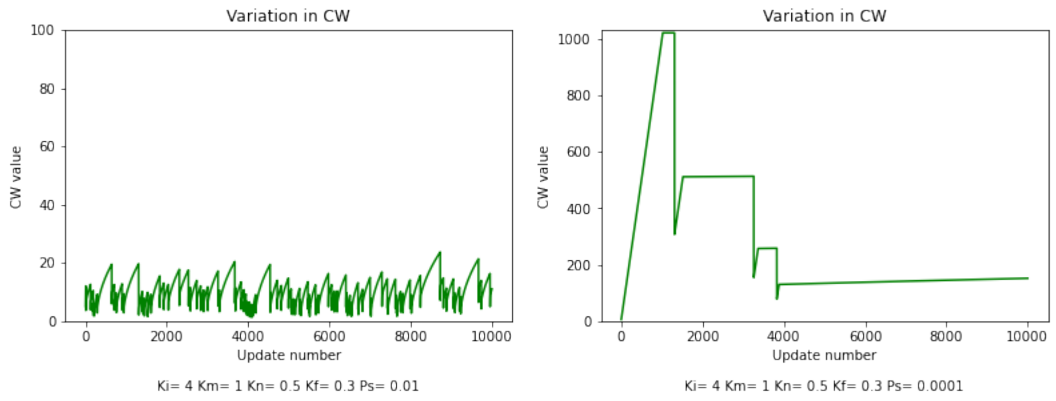


Figure 11:

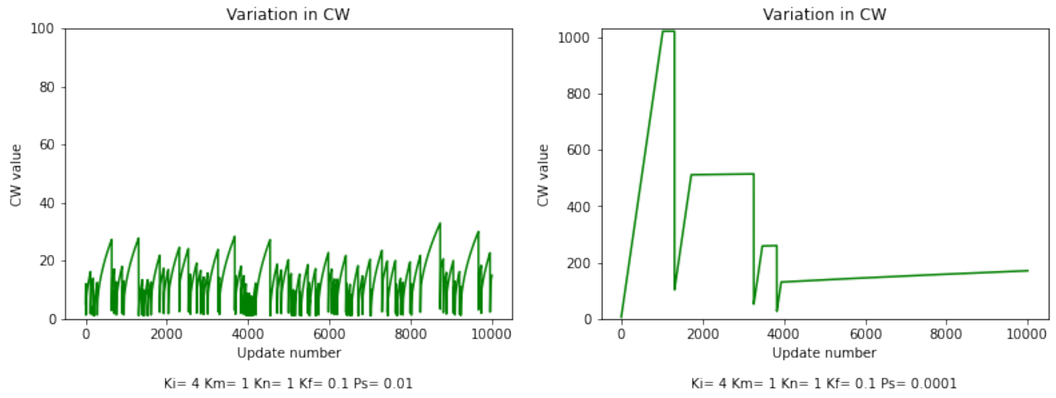


Figure 12:

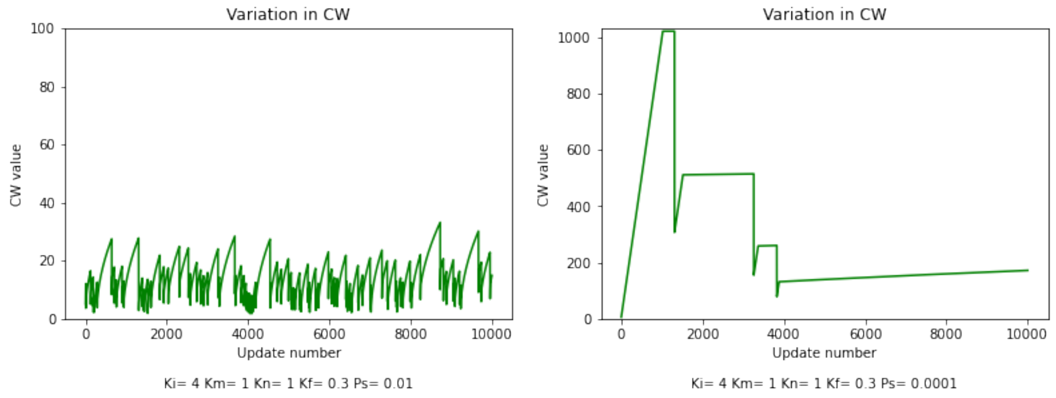


Figure 13:

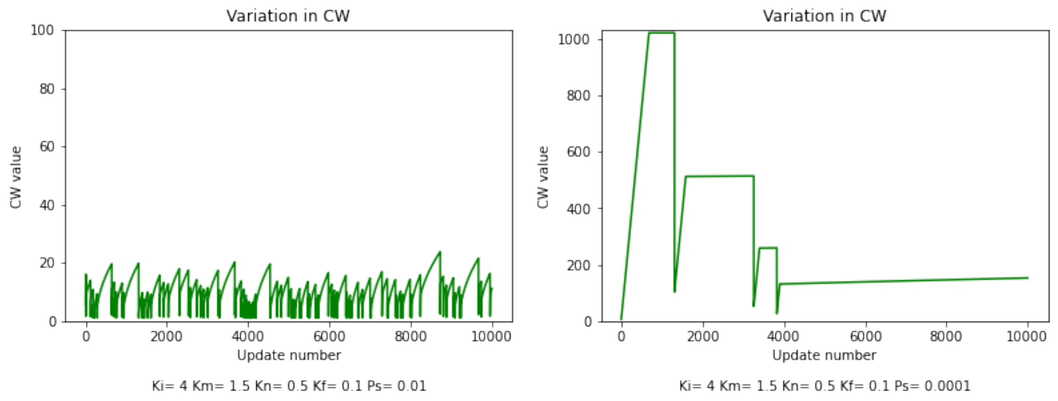


Figure 14:

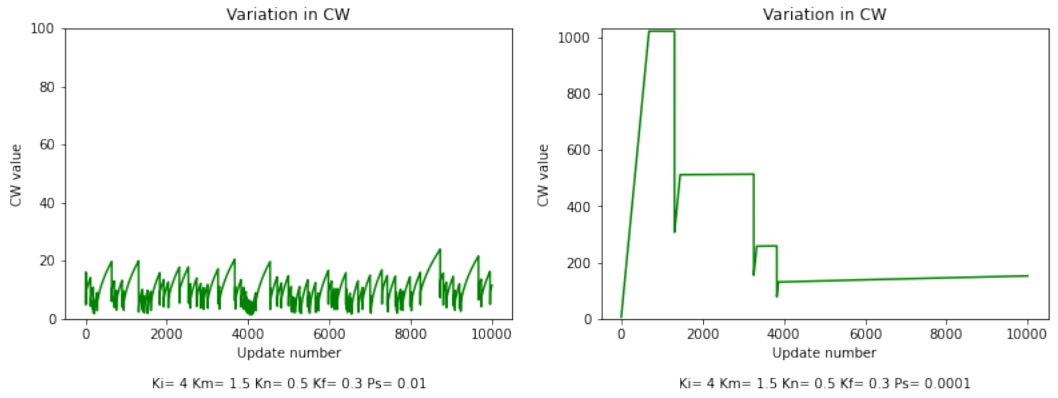


Figure 15:

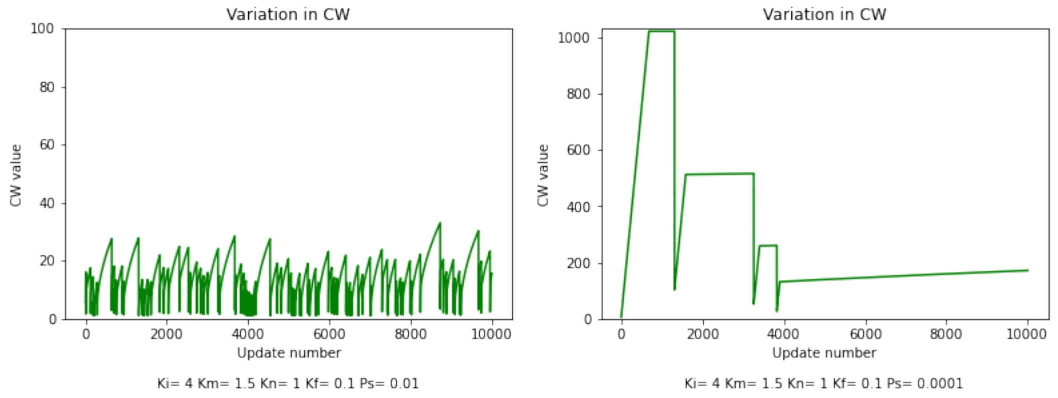
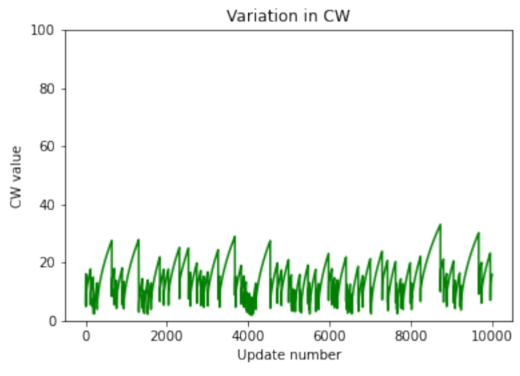
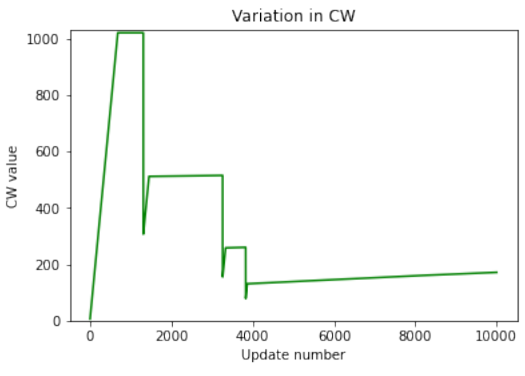


Figure 16:



Ki= 4 Km= 1.5 Kn= 1 Kf= 0.3 Ps= 0.01



Ki= 4 Km= 1.5 Kn= 1 Kf= 0.3 Ps= 0.0001

5 Observations

On comparing all the plots on the left to those on the right the first thing we notice is that with 10 fold decrease in probability of getting a timeout, the variation in cw size is much more continuous and less noisy. The maximum value of the cw size is also much higher. Thus we can say that if our equipment is more reliable in sending and receiving the packets (in which case probability of timeout is lower) it is clear that more amount can be sent without congestion.

It is also observed that the maximum value of cw size appears earlier, especially when timeout probability value is lower.

It can be seen that when probability of timeout is lower the cw size settles to a linearly increase trend over time.

When the probability of timeout is higher, the congestion window value oscillates at quite low values effectively making the receiver wait to fill its window. Thus most of the analysis of effect of parameters is done on the less probability of timeout case.

The odd numbered figures correspond to lesser Kf value and the even numbered figures correspond to greater Kf value. On observing the right plots (lesser timeout probability), it is clear that the linear growth phase is greater for the plots where Kf is greater (even numbered plots) as compared to the others. Thus it can be seen that greater the value of multiplier value used when timeout occurs, greater are the linear growth phases.

There is no major change due to Ki value. This can be observed from comparing figures 1,9 or 2,10 etc. This seems valid as the trend is not much affected by the initial value of CW.

With increase in Km value the exponential growth is much steeper which also can be explained from the nature of the algorithm. This can be seen from comparision of figures 1,5 or 2,6 etc.

Same argument as above can be made for Kn in case of the linear growth phase as observed in fig 1,3 or 2,5 etc.

6 Learning Outcomes

Through this experiment I learnt all about how TCP congestion works and how it varies the value of CW size based on whether or not there is a time-out. I learnt all about the underlying algorithm and how there are various parameters affecting the variation of CW in different ways. I also learnt how to observe the data and conclude how each parameter affected the variation in its own way and also correlating it with theory as to how it can be explained. I also learnt about how acks work in ensuring that the packets have been delivered or not and the consequences of the same.

7 Additional thoughts

It also shows that if our setup is not good and results in higher loss of packets then this will lead to system not working efficiently and the receiver made to wait for longer periods.

8 Conclusion

This experiment has given an insight about how congestion control takes place in real life albeit at a much bigger level. It was observed how different parameters cause variations in the way the algorithm works. It shows how important it is to calibrate and choose the parameters properly so that neither is there a high loss of packets nor that the receiver is made to wait unnecessarily due to low CW size. Thus, through this simulation different conclusions are drawn and explained.

9 References

https://en.wikipedia.org/wiki/TCP_congestion_control
https://en.wikipedia.org/wiki/Network_congestion
https://en.cppreference.com/w/cpp/numeric/random/bernoulli_distribution