CS3205: Assignment 3

Objective

Simulation of the TCP Congestion Control Algorithm - The objective of this project is to emulate the TCP congestion control algorithm. The assumptions and variables are as follows:

- Receiver Window Size (RWS) is set to 1MB and does not change during the entire duration of the emulation.
- The Sender always has data to send to the receiver.
- Sender's MSS is 1KB. Each segment has a fixed length of one MSS.
- Go-back-N is used, but cumulative acknowledgments are not considered. For each segment, an individual timeout timer and ACK are used.
- The congestion window is always interpreted as a multiple of MSS (1 KB).
- The congestion threshold is always set to 50% of the current CW value. Assume the initial threshold to be RWS/2.

Parameters to observe

- **Ki** , $1 \leq Ki \leq 4$ denotes the initial congestion window (CW). Default value is 1. The initial CW is given by:

  ```
  CWnew = Ki∗ MSS
  ```

- **Km** , $0.5 \leq Km \leq 2$ denotes the multiplier of the Congestion Window, during the exponential growth phase. Default value is 1. When a segment's ACK is successfully received,

  ```
  CWnew = min(CWold + Km*MSS, RWS)
  ```

- **Kn** , $0.5 \leq Kn \leq 2$ denotes the multiplier of the Congestion Window, during the linear growth phase. Default value is 1. When a segment's ACK is successfully received,

  ```
  CWnew = min(CWold + Kn∗(MSS*MSS/CWold) , RWS)
  ```

- **Kf** , $0.1 \leq Kf \leq 0.5$ denotes the multiplier when a timeout occurs:

```
CWnew = max(1, Kf * CWold)
```

- **Ps** , 0 < Ps < 1, denotes the probability of receiving the ACK packet for a given segment before its timeout occurs.

Running the Code

```
python mytcp.py -i <double> -m <double> -n <double> -f <double> -s <double> -T <int> -o outfile
```

The congestion value size after each update is printed to an output file

---

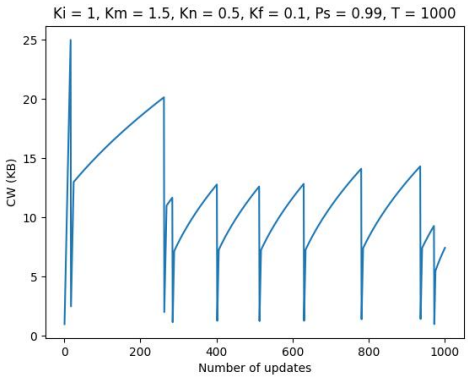
Experiment

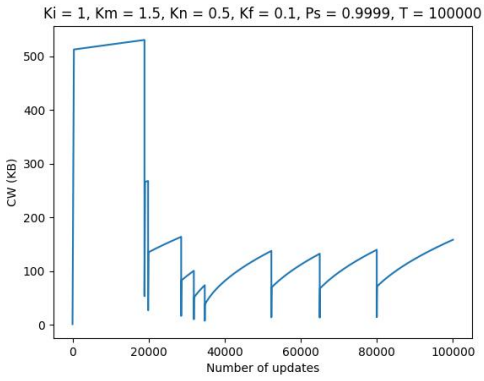
The TCP congestion control is simulated on the following parameter combinations:

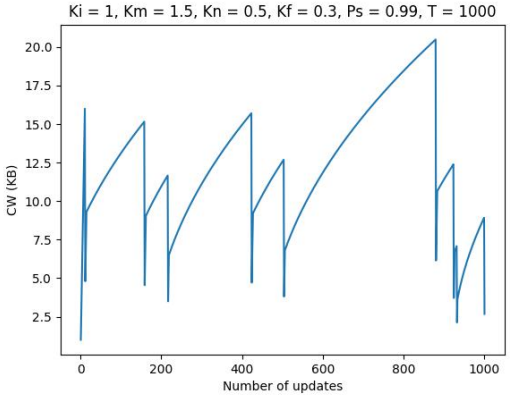
```
Ki ∈ {1, 4}, Km ∈ {1, 1.5}, Kn ∈ {0.5, 1}, Kf ∈ {0.1, 0.3}, Ps ∈ {0.99, 0.9999}
```

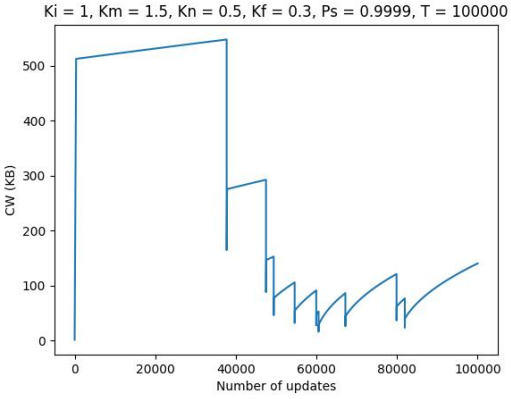
For each simulation, a graph of congestion window size vs update number is plotted.

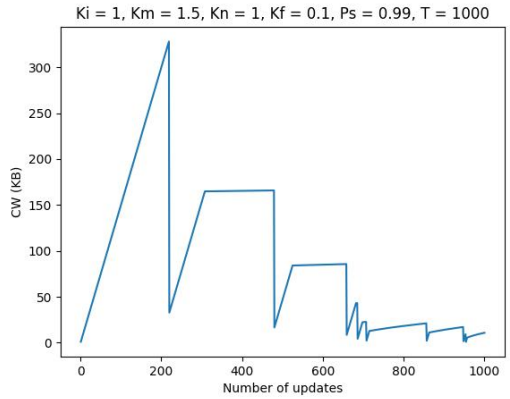
Graphs

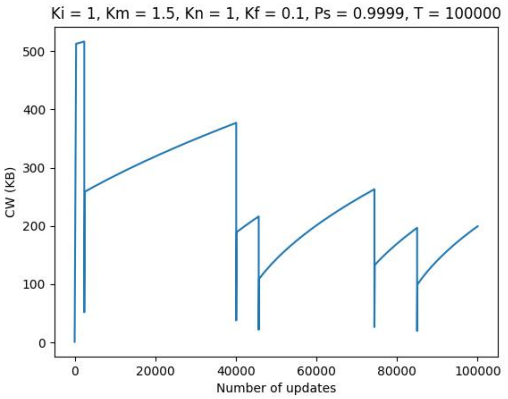


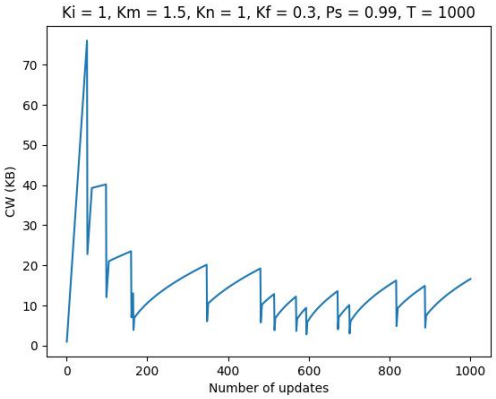


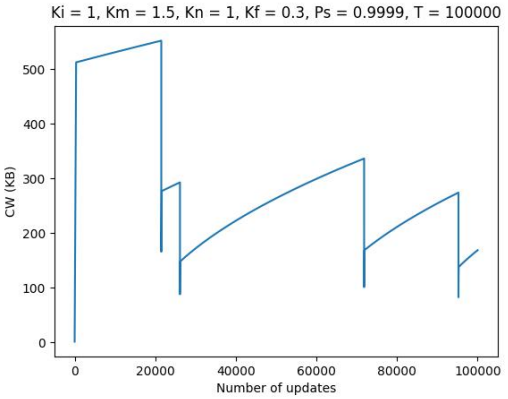

Ki = 1, Km = 1.5, Kn = 0.5, Kf = 0.3, Ps = 0.99, T = 1000
CW (KB)
Number of updates

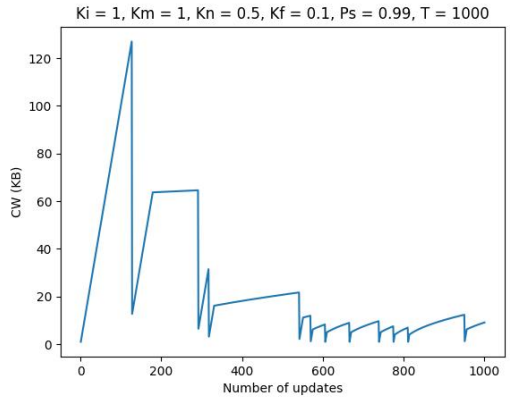

Ki = 1, Km = 1.5, Kn = 0.5, Kf = 0.3, Ps = 0.9999, T = 100000
CW (KB)
Number of updates

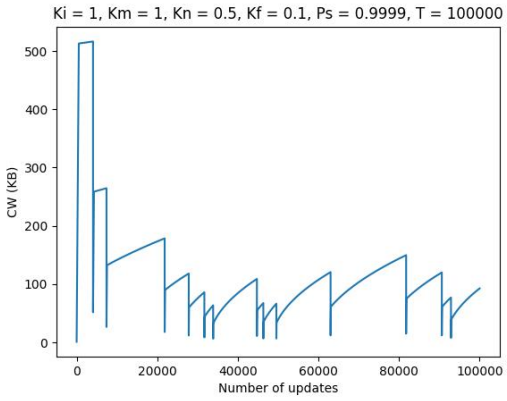

Ki = 1, Km = 1.5, Kn = 1, Kf = 0.1, Ps = 0.99, T = 1000
CW (KB)
Number of updates

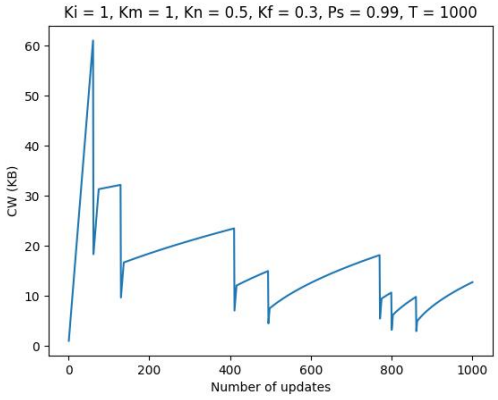

Ki = 1, Km = 1.5, Kn = 1, Kf = 0.1, Ps = 0.9999, T = 100000
CW (KB)
Number of updates

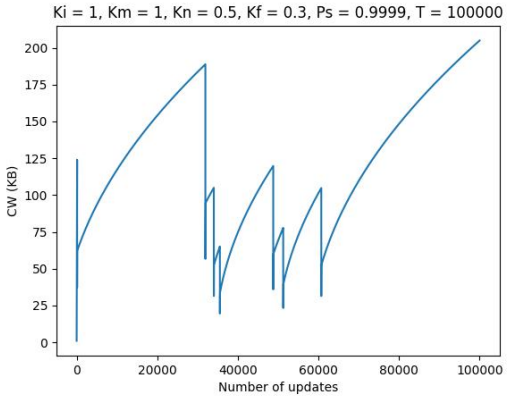

Ki = 1, Km = 1.5, Kn = 1, Kf = 0.3, Ps = 0.99, T = 1000
CW (KB)
Number of updates

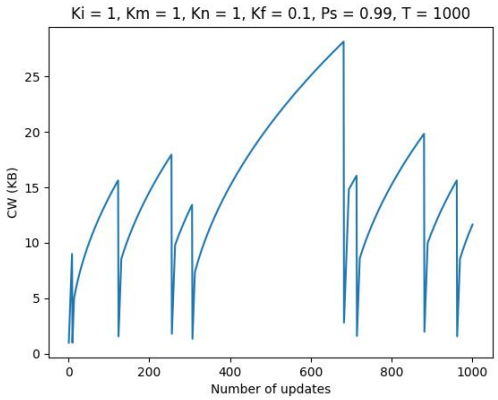

Ki = 1, Km = 1.5, Kn = 1, Kf = 0.3, Ps = 0.9999, T = 100000
CW (KB)
Number of updates

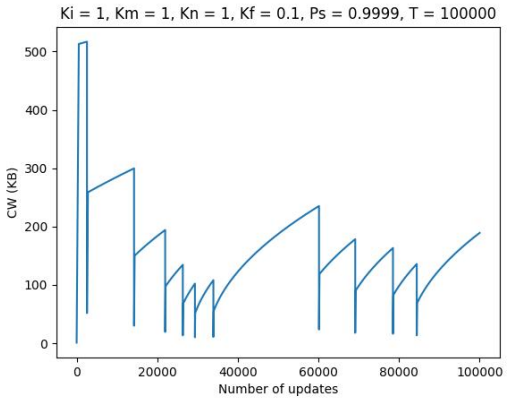

Ki = 1, Km = 1, Kn = 0.5, Kf = 0.1, Ps = 0.99, T = 1000
CW (KB)
Number of updates

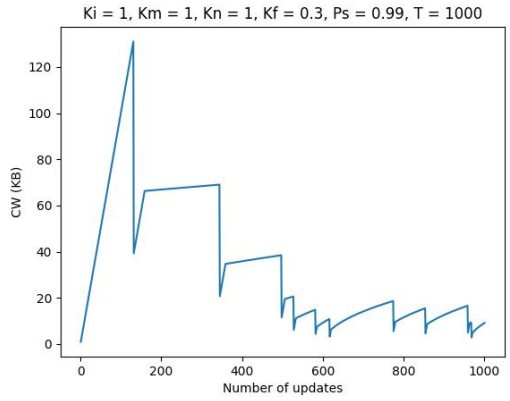

Ki = 1, Km = 1, Kn = 0.5, Kf = 0.1, Ps = 0.9999, T = 100000
CW (KB)
Number of updates

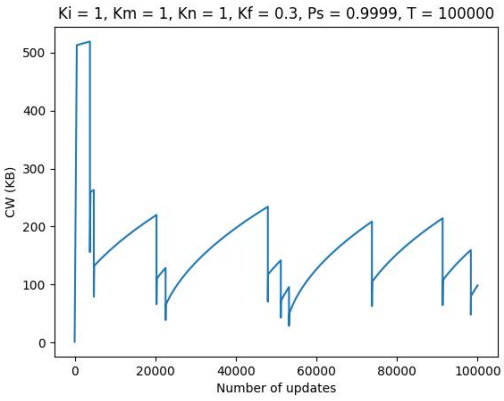

Ki = 1, Km = 1, Kn = 0.5, Kf = 0.3, Ps = 0.99, T = 1000
CW (KB)
Number of updates

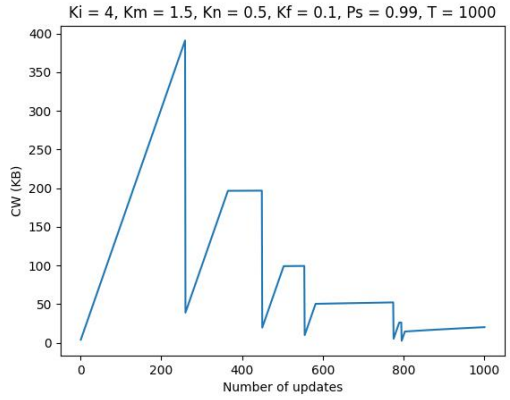

Ki = 1, Km = 1, Kn = 0.5, Kf = 0.3, Ps = 0.9999, T = 100000
CW (KB)
Number of updates

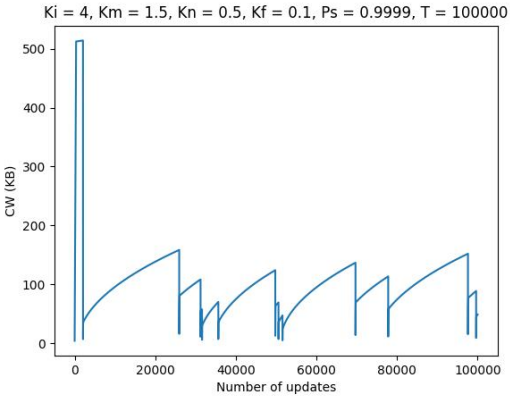

Ki = 1, Km = 1, Kn = 1, Kf = 0.1, Ps = 0.99, T = 1000
CW (KB)
Number of updates

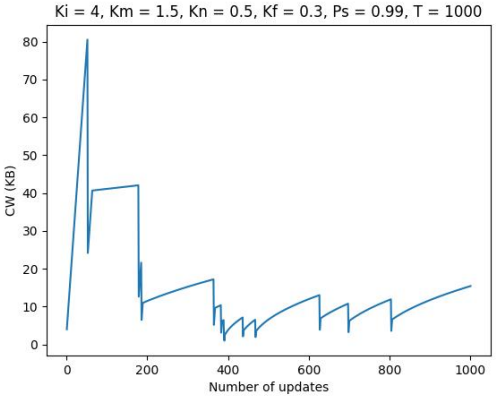

Ki = 1, Km = 1, Kn = 1, Kf = 0.1, Ps = 0.9999, T = 100000
CW (KB)
Number of updates

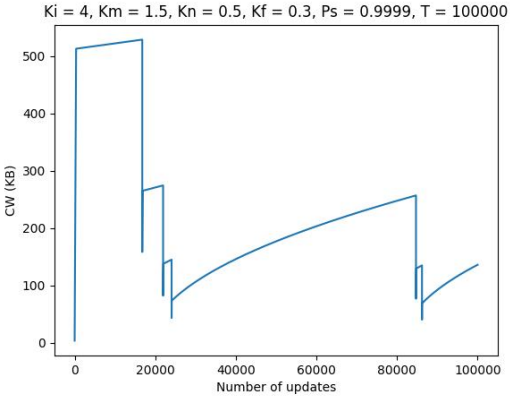

Ki = 1, Km = 1, Kn = 1, Kf = 0.3, Ps = 0.99, T = 1000
CW (KB)
Number of updates

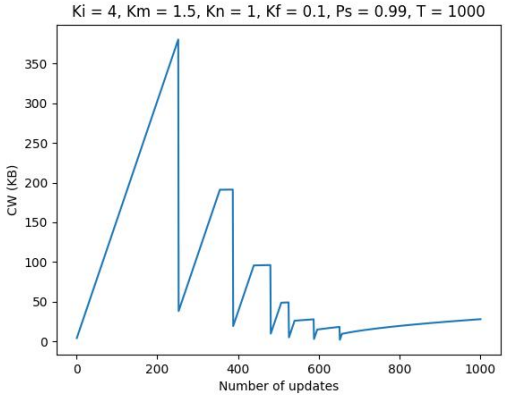

Ki = 1, Km = 1, Kn = 1, Kf = 0.3, Ps = 0.9999, T = 100000
CW (KB)
Number of updates

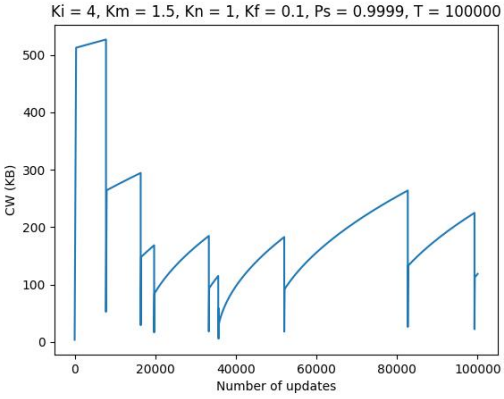

Ki = 4, Km = 1.5, Kn = 0.5, Kf = 0.1, Ps = 0.99, T = 1000
CW (KB)
Number of updates

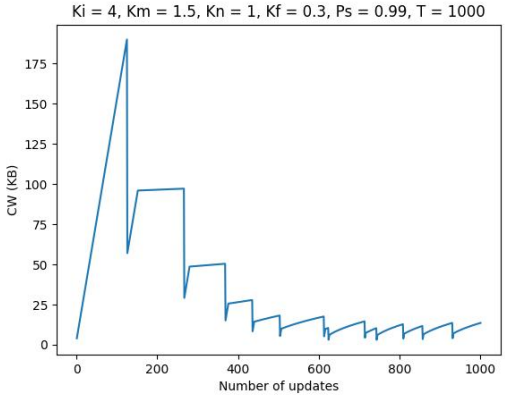

Ki = 4, Km = 1.5, Kn = 0.5, Kf = 0.1, Ps = 0.9999, T = 100000
CW (KB)
Number of updates

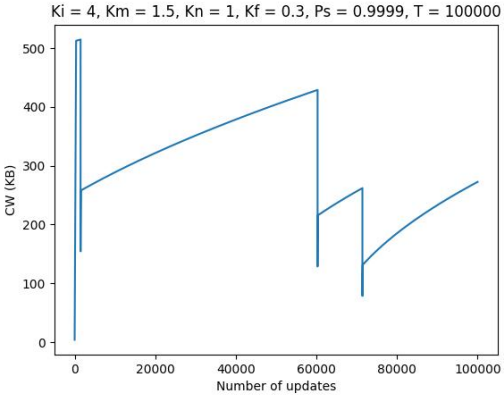

Ki = 4, Km = 1.5, Kn = 0.5, Kf = 0.3, Ps = 0.99, T = 1000
CW (KB)
Number of updates

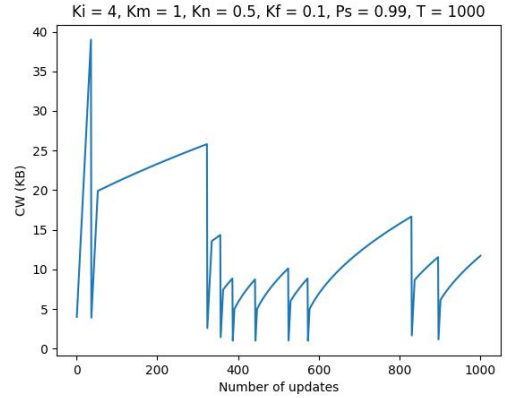

Ki = 4, Km = 1.5, Kn = 0.5, Kf = 0.3, Ps = 0.9999, T = 100000
CW (KB)
Number of updates

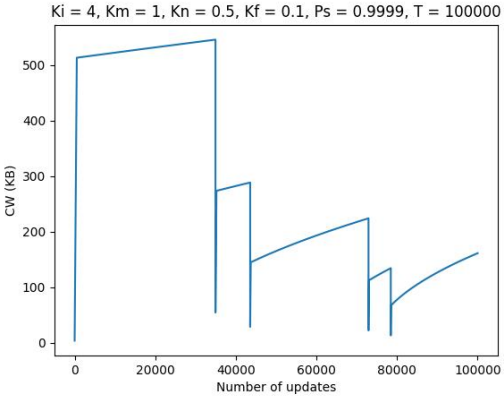

Ki = 4, Km = 1.5, Kn = 1, Kf = 0.1, Ps = 0.99, T = 1000
CW (KB)
Number of updates

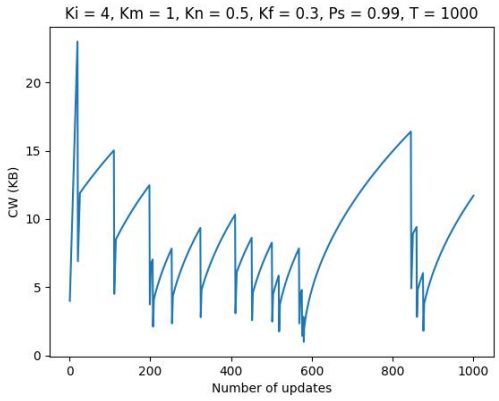

Ki = 4, Km = 1.5, Kn = 1, Kf = 0.1, Ps = 0.9999, T = 100000
CW (KB)
Number of updates

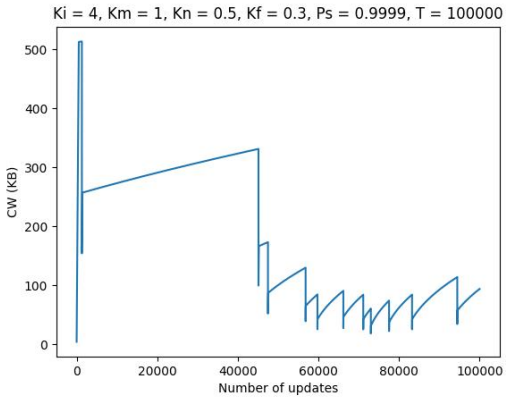

Ki = 4, Km = 1.5, Kn = 1, Kf = 0.3, Ps = 0.99, T = 1000
CW (KB)
Number of updates

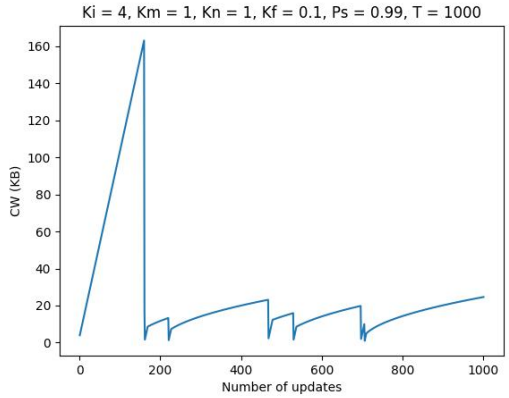

Ki = 4, Km = 1.5, Kn = 1, Kf = 0.3, Ps = 0.9999, T = 100000
CW (KB)
Number of updates

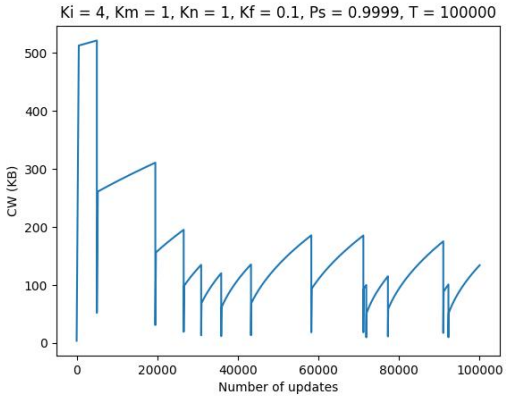

Ki = 4, Km = 1, Kn = 0.5, Kf = 0.1, Ps = 0.99, T = 1000
CW (KB)
Number of updates

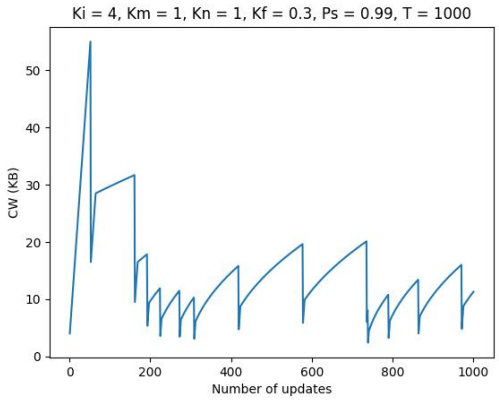

Ki = 4, Km = 1, Kn = 0.5, Kf = 0.1, Ps = 0.9999, T = 100000
CW (KB)
Number of updates

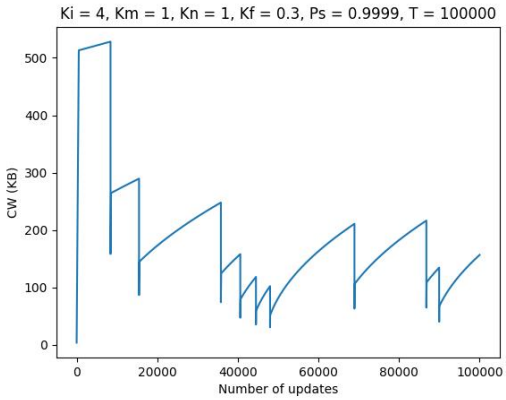












Observations

Influence of each parameter on the congestion control:

- **Ki -** Determines initial window size. Doesnt have much influence on congestion control.

- **Km -** Exponential growth phase multiplier. Higher the value of Km implies faster growth in the congestion window size during Exponential growth phase. Hence assuming that there is lower network traffic. We can observe steeper graphs during exponential growth phase for higher value of Km
- **Kn -** Linear growth phase multiplier. Higher the value of Kn implies faster growth in the congestion window size during linear growth phase. Hence assuming that there is lower network traffic. We can observe steeper graphs during linear growth phase for higher value of Kn.
- **Kf -** Timeout phase multiplier. Lower value of Kf means we are assuming that network traffic might be very high and hence starting from a much lower value. We can observe that for Kf = 0.1, the drop in congestion window size is more than that for Kf = 0.3.
- **Ps -** Success probability. This parameter affects the observations most. Network congestion is inversely proportional to Ps. We can see that we get less timeouts for Ps = 0.9999 than Ps = 0.99 for same interval. So we use higher number of changes in graph for Ps = 0.9999 (100000) and less for Ps = 0.99 (1000).

Conclusion

The exercise of implementing a simulator for TCP congestion control for this assignment provided in-depth understanding of the Additive Increase Multiplicative Decrease (AIMD) algorithm. We were able to get the insight of working of the TCP congestion control that is used in current day networks. We also observed the working of congestion control on different circumstances and parameters.