Assignment 2

Meesa Shivaram Prasad (CS18B056)

Aim : To emulate the TCP congestion control algorithm.

Introduction :

TCP/IP model has following layers:

1) Link Layer
2) Internet Layer
3) Transport Layer
4) Application Layer

Congestion control comes under Transport Layer, basically TCP uses sliding window protocol for flow control,Congestion is a state when the message traffic is so heavy the response time of the network ,so we have to adjust congestion window size to avoid a lot of timeouts.

Experimental details:

- Experimental/Simulation setup :
    1. Using a random engine generator with bernoulli distribution , which outputs true with input probability.
    2. Running a simple cpp program where we assume that we have sent all packets of the current congestion window and receive the acknowledgements of each packet one by one.
    3. This receiving of acknowledgement is done with the help of random engine generator mentioned in the step-1
    4. It's interpreted as time out if the output of the generator is 1.
    5. Based on the acknowledgement we adjust the congestion window size as per the algorithm
    6. Here the Assumptions are receiver window size is fixed and sender has infinite number of packets to send.
- Entities involved and functions in each entity:
    1. Random engine generator with bernoulli distribution :
       outputs true with input probability .
    2. A main program which takes input from the user , the set of hyper parameters as required for the algorithm and simulates the congestion control algorithm ,uses generator to simulate acknowledgments received from packets.
    3. A python script for running of this algo with 32 different hyper parameter configurations and generate CW values vs num of updates plots.

- ## Additional details:

The congestion control algorithm that was used is :

```
   Initially CW_threshold = INFINITY , and system in exponential phase
 if  timeout
   CW_threshold = CW/2
   CW = max(1.0,Kf*CW)
else
   if(CW<=CW_threshold)       # exponential phase
      CW = min(CW+Km*MSS,RWS)
   else           #Linear phase
      CW = min(CW+Kn*MSS*(MSS/CW),RWS)
```

K_i = {1,4}

K_m = {1,1.5}

K_n = {0.5,1}

K_f = {0.1,0.3}

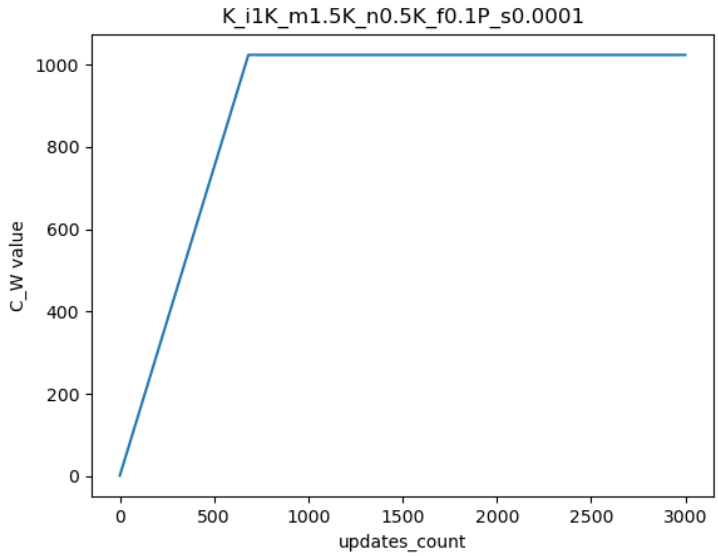
P_s = {1e-2,1e-4}

MSS = 1KB

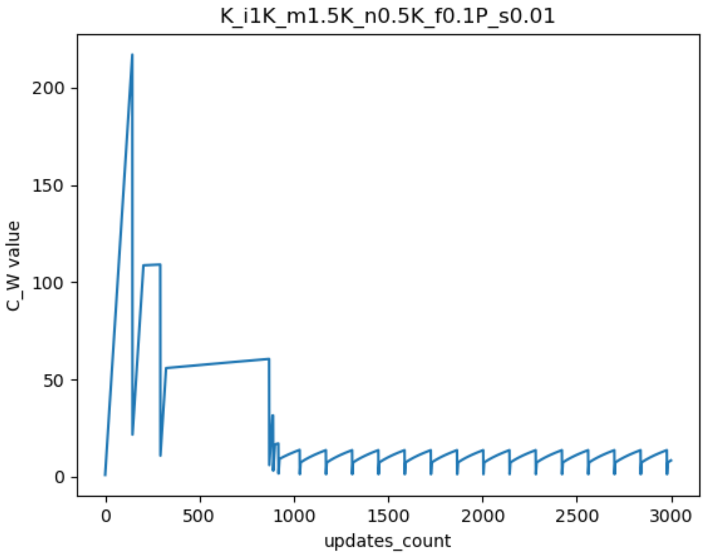
RWS = 1MB

Results and Observations:

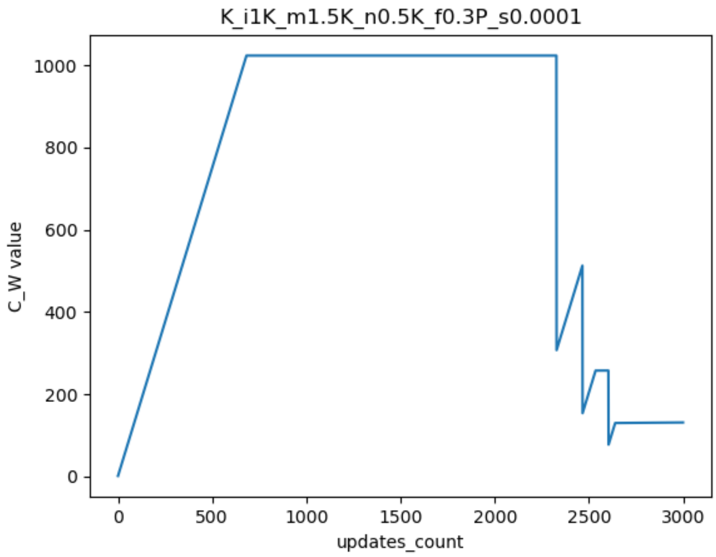
Note: C_W values are in Kilobytes



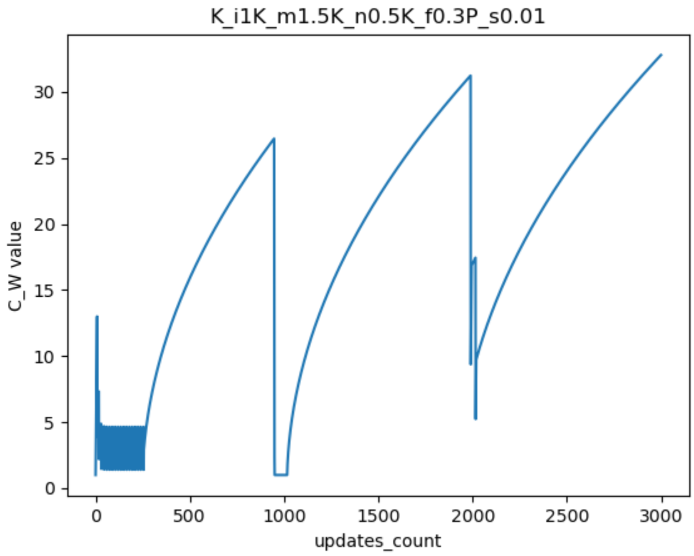
Observation: As the probability of timeout is very low ,time out didn't occur,so we are in exponential phase,it increased exponentially but finally capped by RWS which is 1MB.



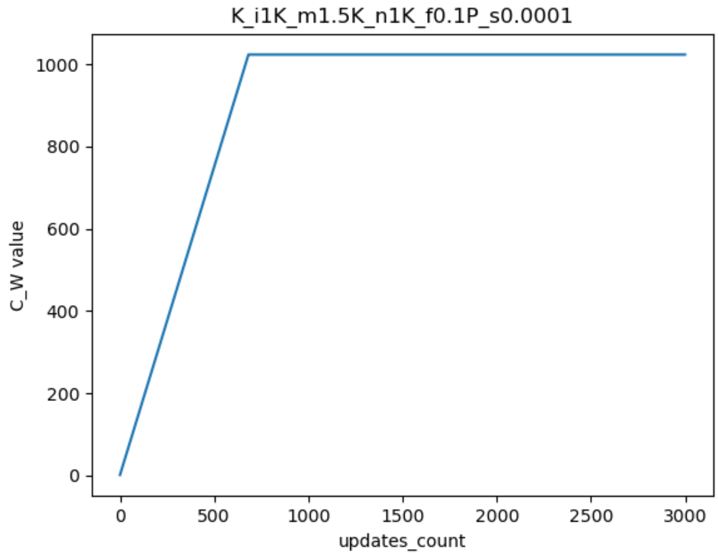
Observation: As K_m is high we can see the slope of exp phase is high and as K_f is low we can see there is a huge drop in C_W value after timeout , and with multiple number of timeouts the threshold value got decreased , because threshold is CW/2,because timeouts happened in exponential phase(at ~800).small K_n is also a reason for low values of C_W ,because CW is not increasing much.



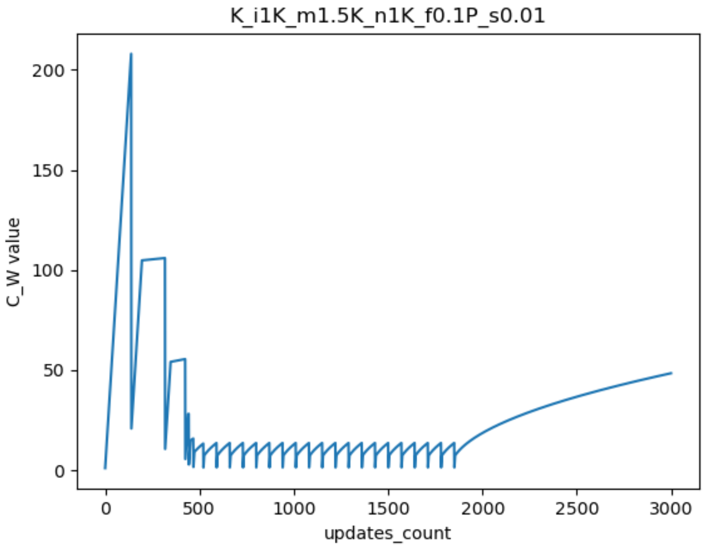
Observation: here similar to above threshold decreased a lot because of timeout in exponential phase(at ~2300) , as P_s is small we didn’t see much time outs as compared to the previous figure. As K_n is small we can see a very small slope in linear phase.small K_n is also a reason for low values of C_W ,because CW is not increasing much.



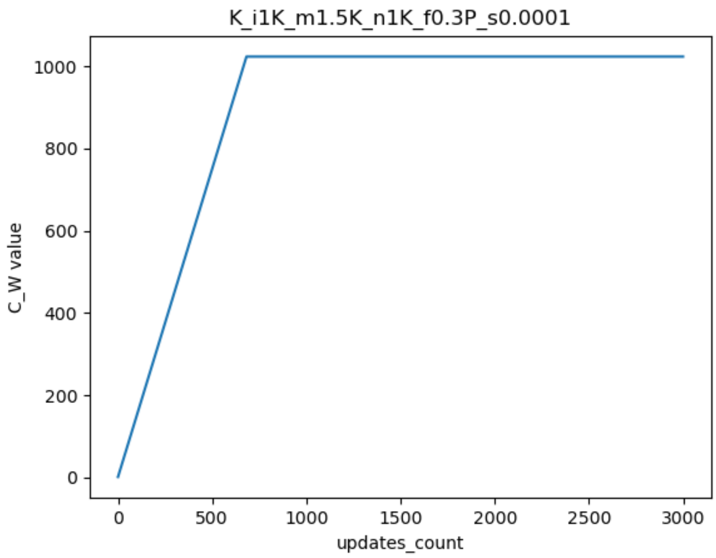
Observation: those multiple timeouts happened initially it's a completely random event but yeah these are much in number because P_s is much higher compared to previous images.



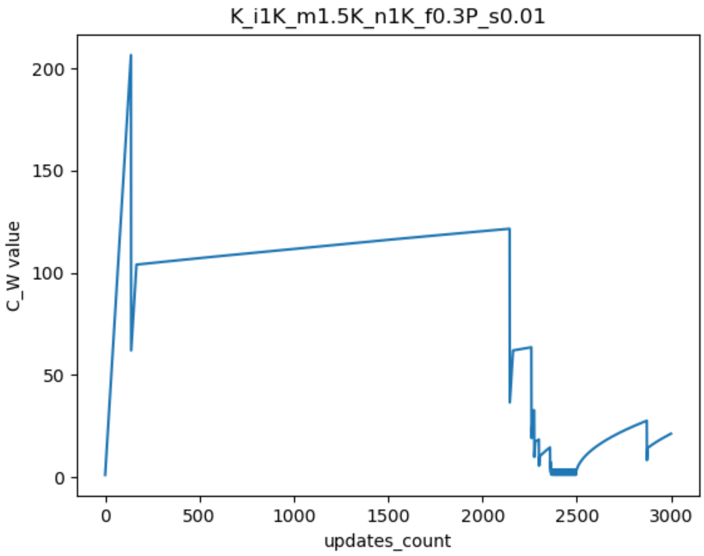
Observation: As the probability of timeout is very low ,time out didn't occur,so we are in exponential phase,it increased exponentially but finally capped by RWS which is 1MB.



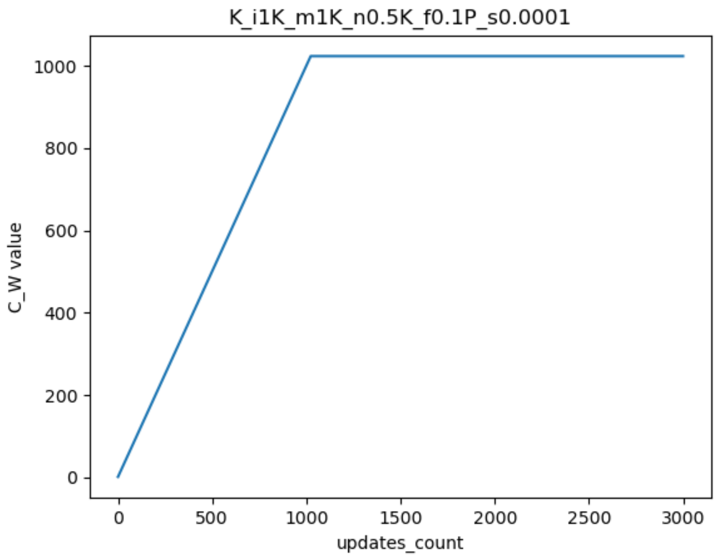
Observation: Because of low K_f the C_W value reduced a lot after time out , and as P_s is high there are a lot of timeouts in between and as K_n value is higher K_n is high and C_W is low we can see a curve with high slope from (~1800 - 3000) .



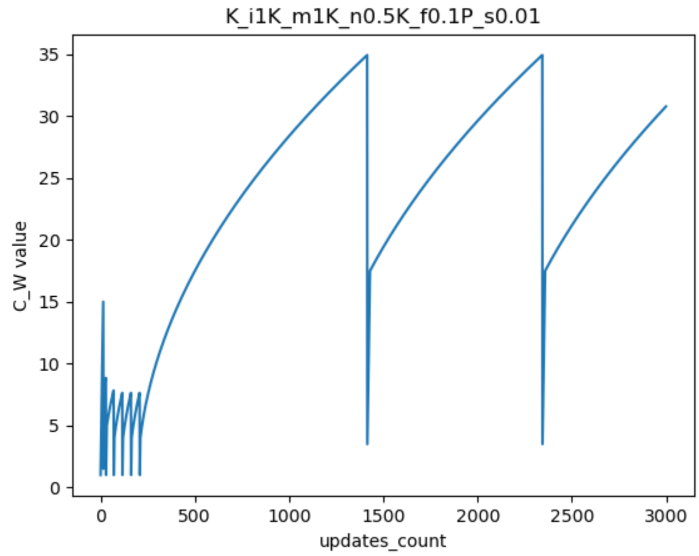
Observation: As the probability of timeout is very low ,time out didn't occur,so we are in exponential phase,it increased exponentially but finally capped by RWS which is 1MB.



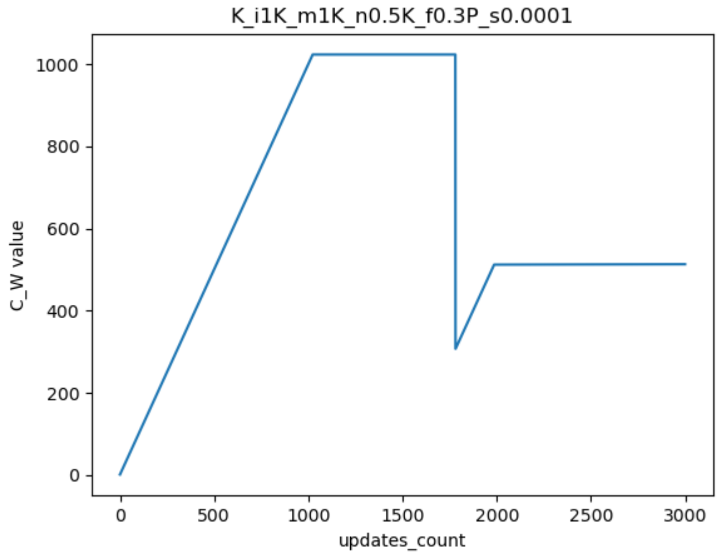
Observation: As K_m is high we can see a high growth rate in exponential phase and similar to previous images the sudden decrease in C_w value is because of adjacent timeouts (as P_s is high) & time outs in growth phase.as K_n is high and C_W is low at (~2500) we can see a nice rise in that linear phase.



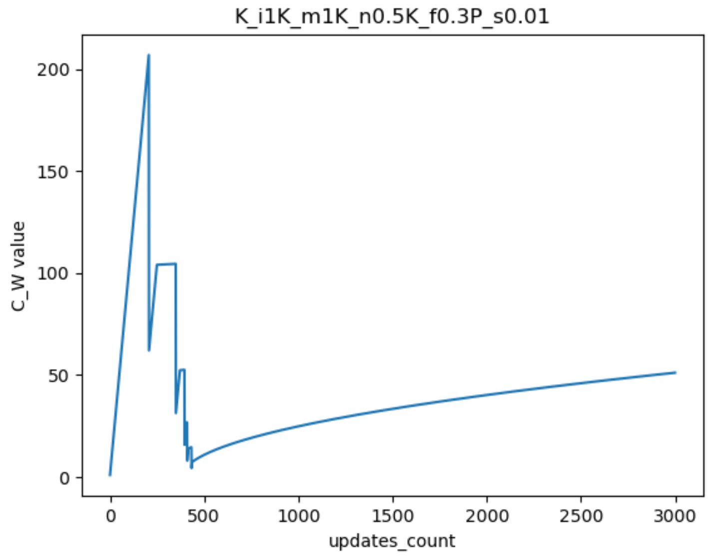
Observation: As the probability of timeout is very low ,time out didn’t occur,so we are in exponential phase,it increased exponentially but finally capped by RWS which is 1MB.



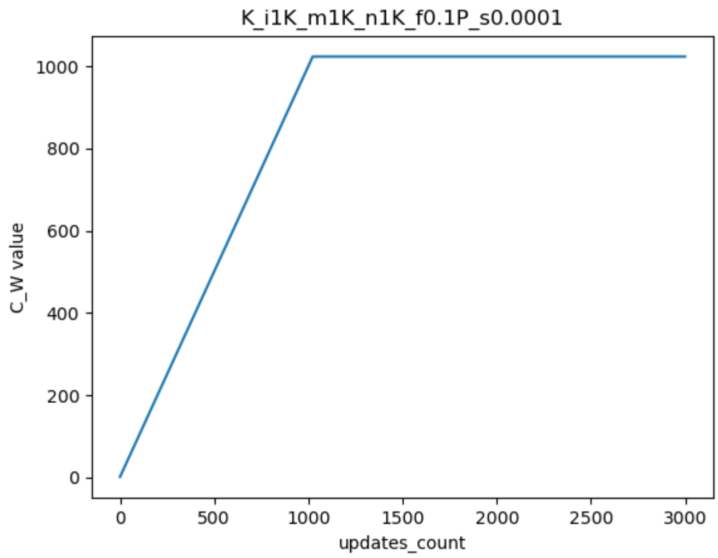
Observation: as K_f is very low we can see C_W value dropped by a lot when there is a time out,because of high P_s there are lot of timeouts.



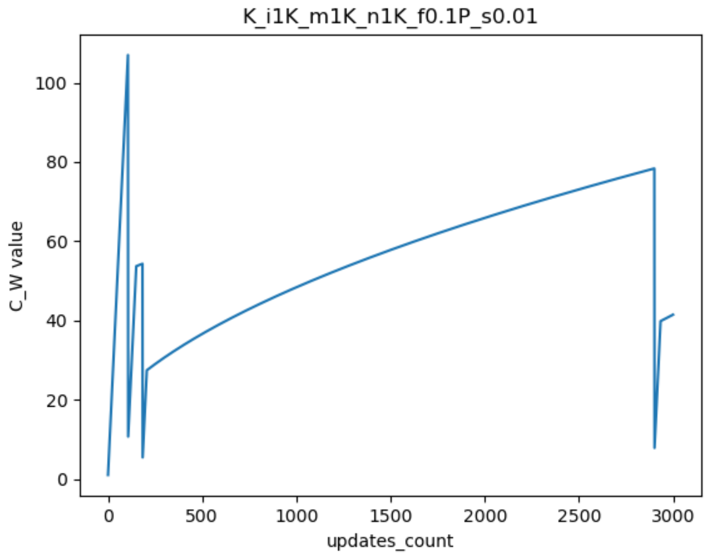
Observation: As the probability of timeout is very low ,time out occurred only once, because of low K_f we can see nice drop in CW after time out , as CW value is high & K_n is low we can see very small slope in (~2000-3000)



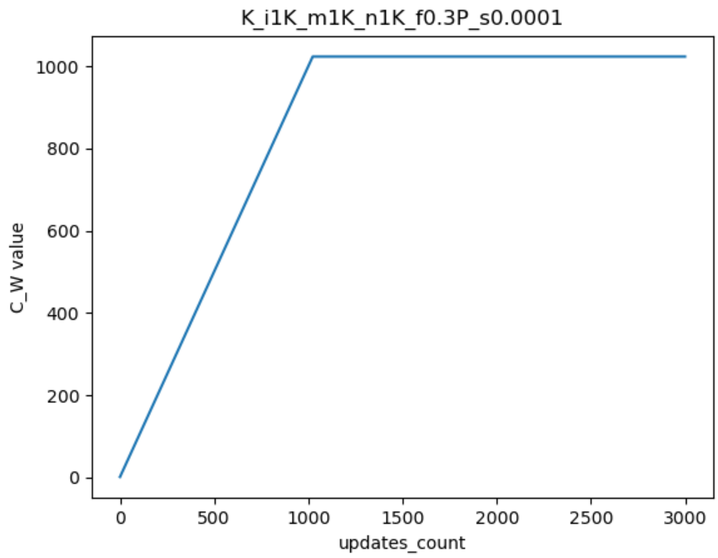
Observation: because of low C_W values despite low K_n we have (high)good slope of linear phase from range(~400-3000).



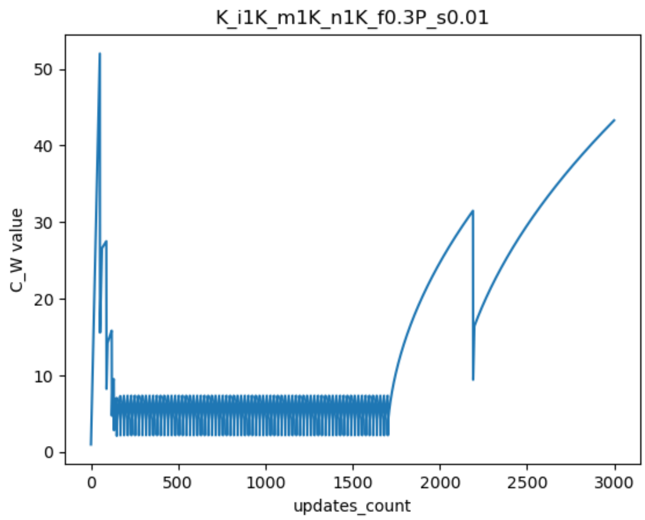
As the probability of timeout is very low ,time out didn’t occur,so we are in exponential phase,it increased exponentially but finally capped by RWS which is 1MB.



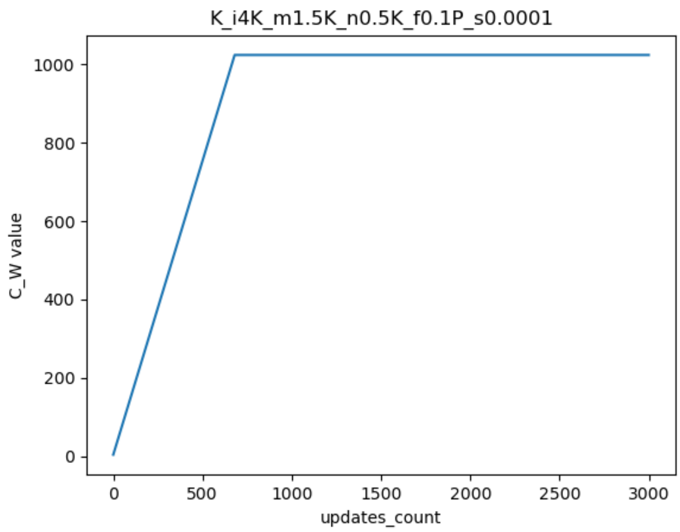
Observation: because of high K_n we have (high)good slope of linear phase from range(~400-3000),and because of low K_f the CW values dropped a lot.



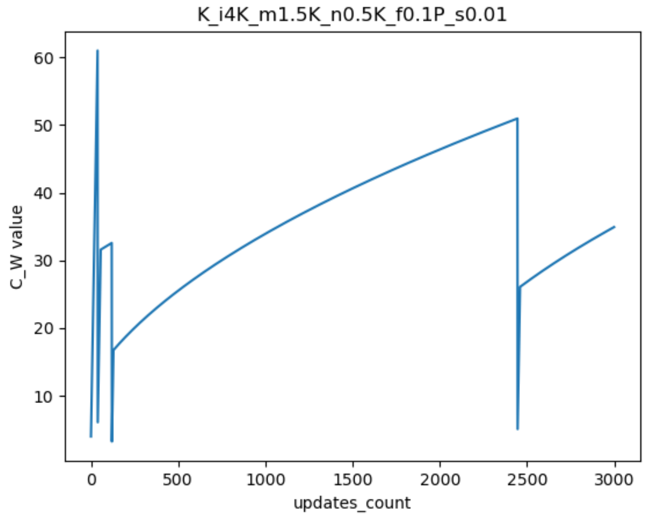
Observations: As the probability of timeout is very low ,time out didn't occur,so we are in exponential phase,it increased exponentially but finally capped by RWS which is 1MB.



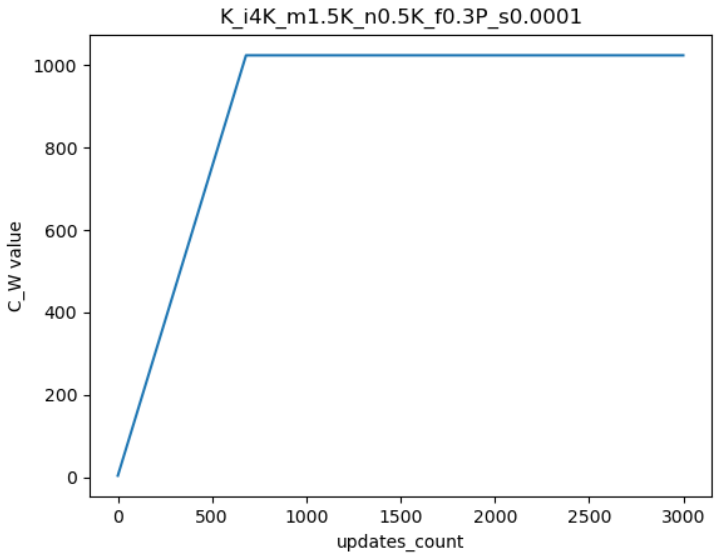
Observations: because of high K_n we have (high)good slope of linear phase,and because of low K_f the CW values dropped a lot,we are seeing these many time outs because of high P_s value.



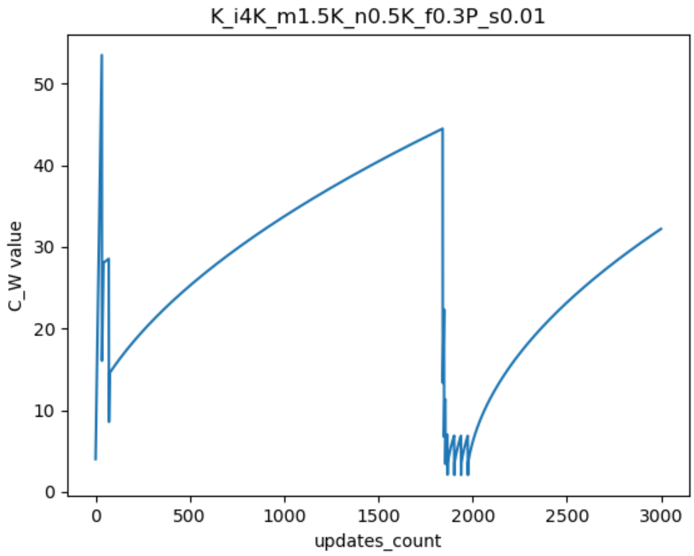
Observation: As the probability of timeout is very low ,time out didn’t occur,so we are in exponential phase,it increased exponentially but finally capped by RWS which is 1MB



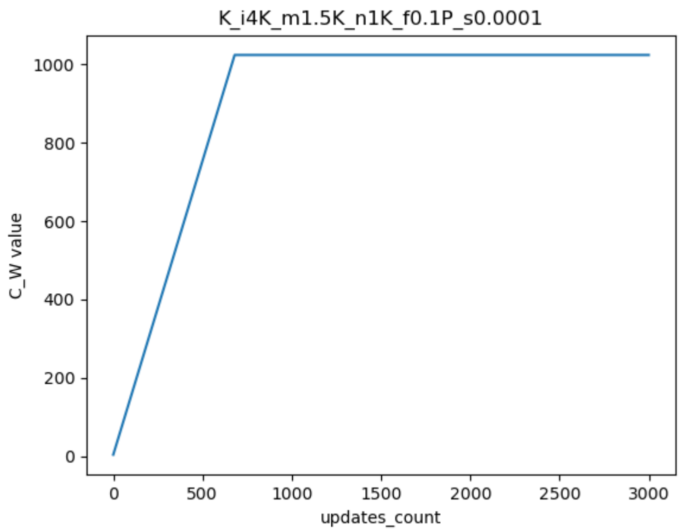
Observation: Because of high K_m we can see good slope on exponential phase , and those huge drop in C_W's are due to low K_f



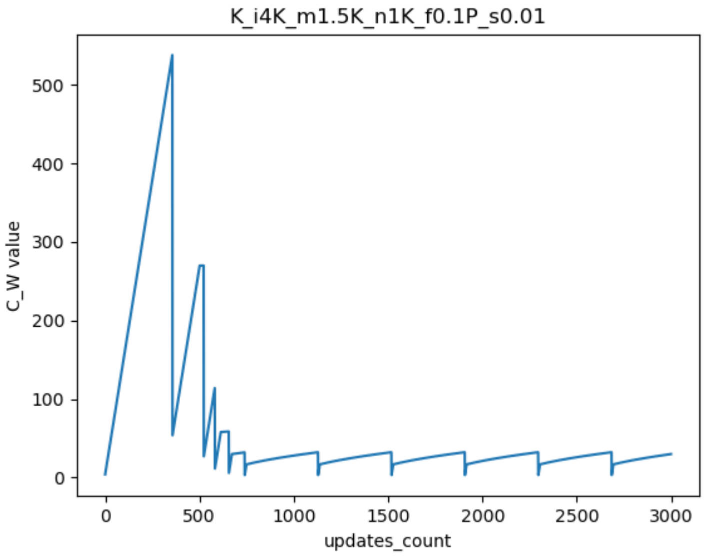
Observation: As the probability of timeout is very low ,time out didn't occur,so we are in exponential phase,it increased exponentially but finally capped by RWS which is 1MB.



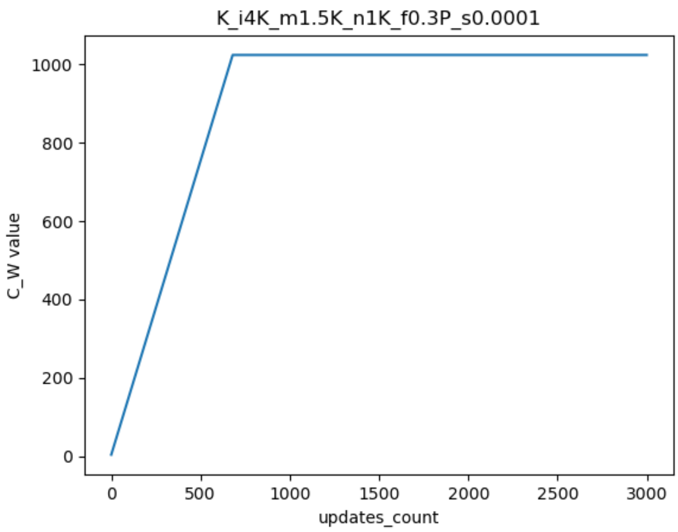
Observation: Because of high K_m we can see good slope on exponential phase , and those huge drops in C_W's are due to low K_f.



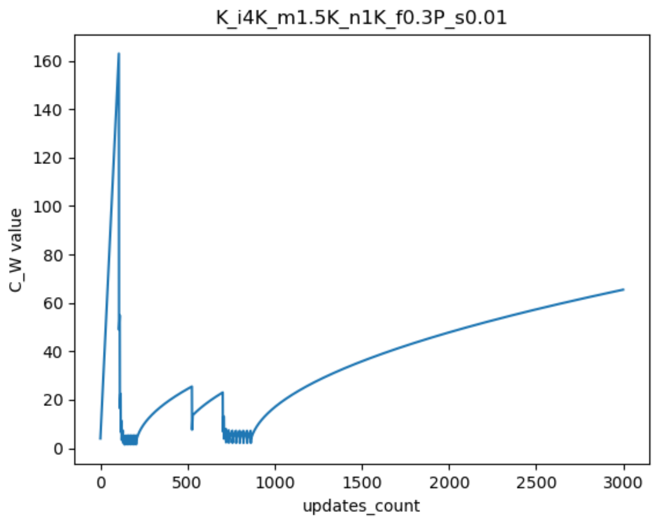
Observation: As the probability of timeout is very low ,time out didn't occur,so we are in exponential phase,it increased exponentially but finally capped by RWS which is 1MB.



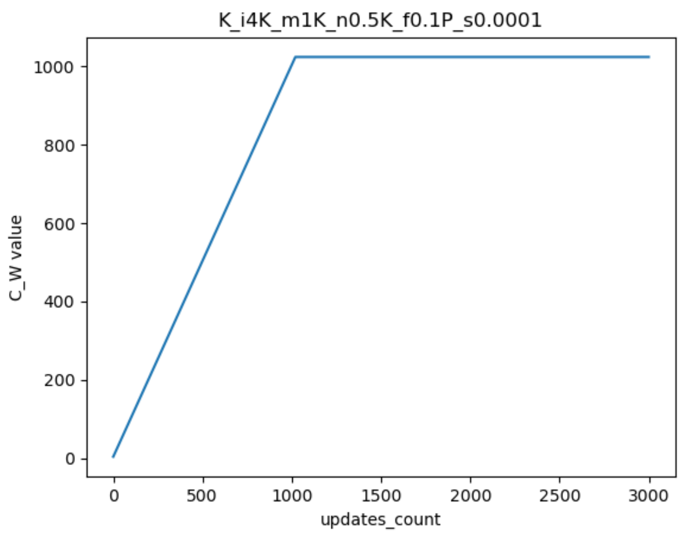
Observation: Because of high K_m we can see (high)good slope on exponential phase , and those huge drops in C_W's are due to low K_f. Because of high K_n we can see a decent slope in linear phase.



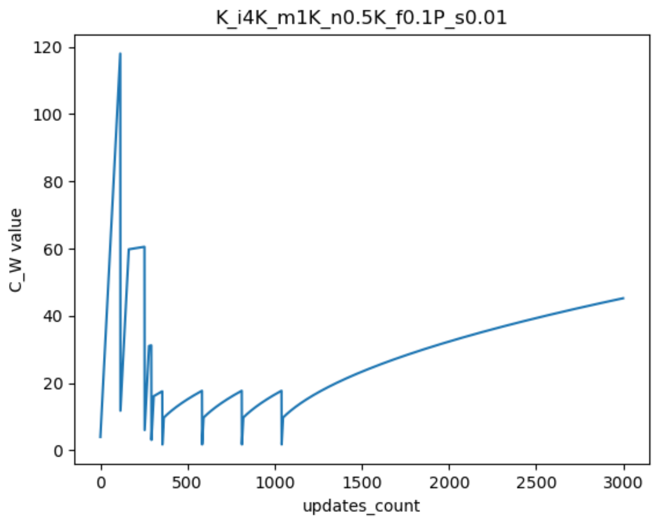
Observation: As the probability of timeout is very low ,time out didn't occur,so we are in exponential phase,it increased exponentially but finally capped by RWS which is 1MB.



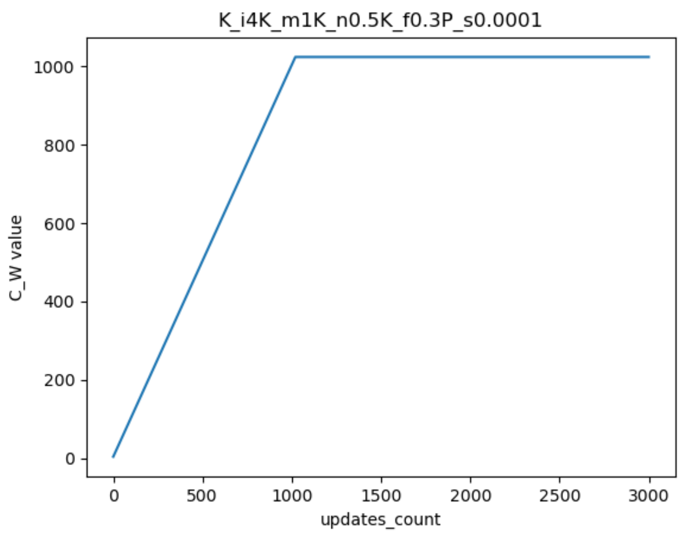
Observation: High slope in exponential phase because of high K_m ,and as K_n is high the linear phase slope is also good.



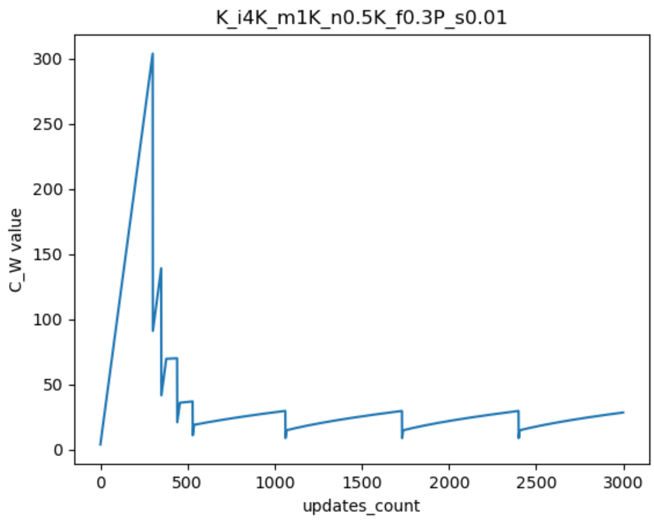
Observation: As the probability of timeout is very low ,time out didn't occur,so we are in exponential phase,it increased exponentially but finally capped by RWS which is 1MB.



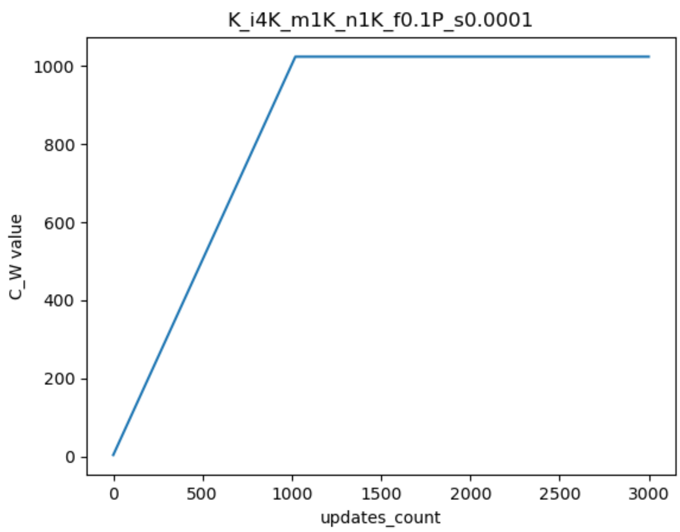
Observation: Because of very low K_f the C_W values dropped by a lot,because of low C_W values at ~(1000-3000) the slope of linear phase is good despite low K_n



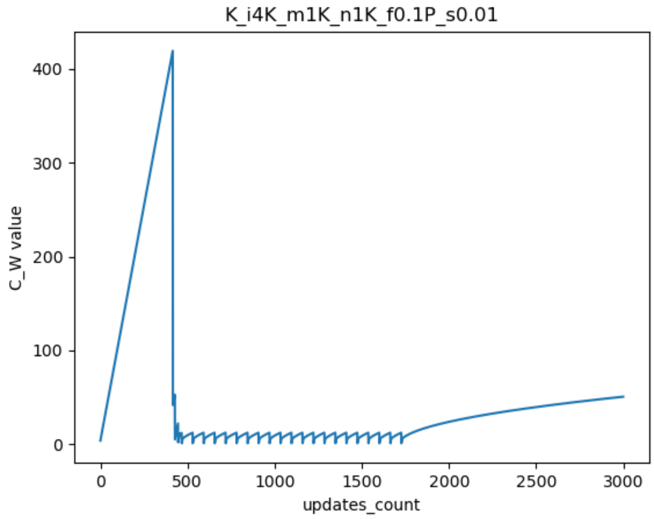
Observation: As the probability of timeout is very low ,time out didn't occur,so we are in exponential phase,it increased exponentially but finally capped by RWS which is 1MB.



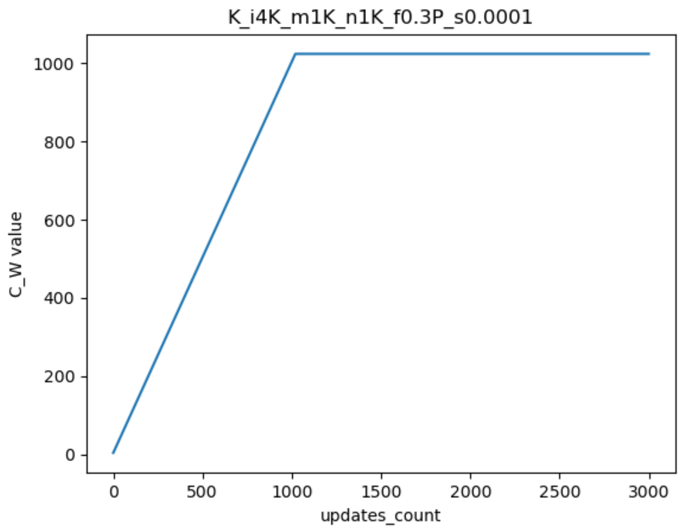
Observation: we have low slope on linear phase because of less K_n



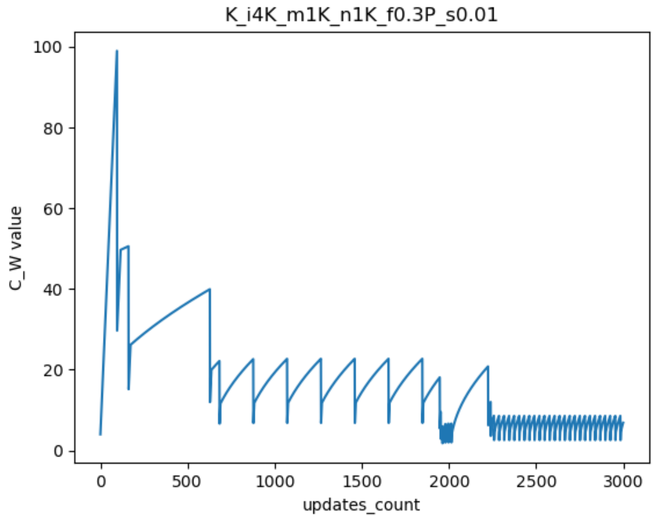
Observation: As the probability of timeout is very low ,time out didn't occur,so we are in exponential phase,it increased exponentially but finally capped by RWS which is 1MB.



Observation: there is huge drop in C_W because of very low K_f



Observation: As the probability of timeout is very low ,time out didn't occur,so we are in exponential phase,it increased exponentially but finally capped by RWS which is 1MB.



Observation: significant drop in C_W because of low K_f .

Summary of Observations:

1) We can see that if P_s is 0.0001 then there are less number of timeouts (on an average 1 time out from the graphs shown below), and it should be true because P_s is too low (which is probability that there is a time_out).
2) The duration of exponential phase depends on the value of K_f ,if K_f is high then we have less exponential phase duration ,because CW will cross the threshold fastly, and it's also evident from the plots.
3) The slope of the exponential phase depends on K_m,if K_m is high then the slope is also high and exponential phase duration is less.
4) The slope of the linear phase depends on K_n,if K_n is high then the slope is also high and linear phase duration is less.
5) Different values of K_i don't have much significance as it's just the setting up of initial C_W.
6) Overall our algorithm does well,if there is a timeout it will reduce the CW there by reducing the network traffic , and again increase till it gets time out.

Learnings :

1) The most important thing is the probability of timeout if it's less. We can increase CW value to high value(meaning we can send many packets in a given time).
2) If exponential phase duration is high then CW value will be high for a lot of time & keep on increasing but it is anyway capped by the threshold.

Additional thoughts:

1) I felt it would have been better if I simulated the probability of acknowledgement based on the message traffic,here I have just used a random generator and probability of time out is independent of every_other thing happening ,so we can't really talk about # of timeouts and traffic control in context of remaining parameters other than P_s.

Conclusion:

We have successfully emulated the TCP congestion control algorithm given in the problem statement and found some interesting and useful observations.

References:

Lab slides.