CS3205 Computer Networks Lab
TCP Congestion Control
Jan-May2021

Ettamsetty Sai Dheeraj CS18B055

March 29, 2021

Dates Performed: March 6, 2021 - March 29, 2021
Instructor: Professor Siva Ram Murthy

1 Objective

To emulate the Modified TCP congestion control algorithm.

2 Introduction

Modification This is an emulation of a modified TCP Congestion control, the modification is that after a timeout the congestion window value does not become equal to its initial value, but changes according to the formula $CW_{new} = max(1, k_f * CW_{old})$.

This simulation does not involve any sockets or any actual connection between two sockets. A Probability (P_s) associated with getting a time-out is given and we can have a random number generator in our program and associate this generated random number with probability and finally decide a timeout has occurred or not.

The value T, the number of updates of the congestion window value is given and we will be sending a single segment for every update.

3 Experimental Details

3.1 Simulation Setup

RWS - Receiver window Size is 1MB
MSS - Maximum Sender Segment size is 1KB
k_i - $\{1, 4\}$
k_m - $\{1.0, 1.5\}$
k_n - $\{0.5, 1.0\}$
k_f - $\{0.1, 0.3\}$
P_s - $\{0.01, 0.0001\}$

Congestion threshold is always set to 50% of the current Congestion window (cw) value.
Initial cw value is given by

$$CW_{new} = K_i * MSS$$

In Exponential growth phase, the cw value changes as,

$$CW_{new} = min(CW_{old} + k_m * MSS, RWS)$$

where k_m is multiplier of Congestion window.
In Linear growth phase, the cw value changes as,

$$CW_{new} = min(CW_{old} + k_n * MSS * \frac{MSS}{CW_{old}}, RWS)$$

where k_n is multiplier of congestion window.
When a timeout occurs, the cw value changes as,

$$CW_{new} = max(1, K_f * CW_{old})$$

where k_f is also a multiplier.

3.2 Entities involved and functions in each entity

I have a written 3 programs in total :

3.2.1 CPP program

A $c++$ program which takes $k_i, k_m, k_n, k_f, p_s, T, outputfile$ as command line arguments and prints the cw value T times in the file with name $outputfile$ in T lines, each line corresponding to one update of cw value.

We can use a random number generator to generate a number and associate it with probability and then decide whether a timeout is occurring or not for that segment.
I generated a random number r under 10,000 and compared it with

$M = p_s * 10000$, if r is less than M then the timeout occurs, else there is no timeout and ACK packet is received.
Every time a timeout is occurred the value of threshold changes to 50% of cw value.
Finally if there is no timeout, we will decide whether we are currently in exponential growth phase or linear growth phase by comparing the cw value with threshold.

3.2.2 PYTHON program

A *python* program which takes the $outputfile$ generated in the previous step as input and plots a graph with $Y - axis$ as cw values and $X - axis$ as number count (starting from 1).

I imported 3 modules sys, matplotlib, numpy. sys for taking command line arguments, numpy for arrays, matplotlib for plots.
For K in $[1, 32]$ every $outputK.txt$ is taken as input for this program and generates the plot and saves it as $figK.png$.

3.2.3 BASH script

A *bashscript* which runs the above 2 programs 32 times with all combinations of the input. This script makes sure that every run of $c++$ program with different input prints its output into different output files and each output file gets a corresponding plot.

I wrote 5 nested for-loops (each loop of length 2) inorder to generate $2^5 = 32$ combinations of input. All the required command line arguments are given in proper order for both $c++$ and *python* programs.

3.3 Extra Details

I recorded a session of my program run with the help of script command and stored it in the file *shell_record.md*. A replay of this session can be done using the command scriptreplay.

I am printing all the 32 combinations of input in the terminal as well.

4 Results and Observations

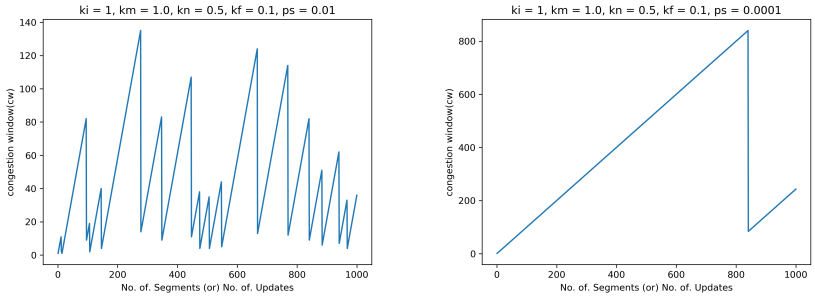

Figure 1: Figure for point (i) containing plot 1(L) and 2(R)

(i) P_s is changing and remaining all values remain constant,
If $P_s = 0.01$, the chances of getting timeout is high, we can see lots of ups and downs in cw graph.

If $P_s = 0.0001$, the chances of getting timeout is very low, In majority plots, there is a timeout roughly after 800 updates.

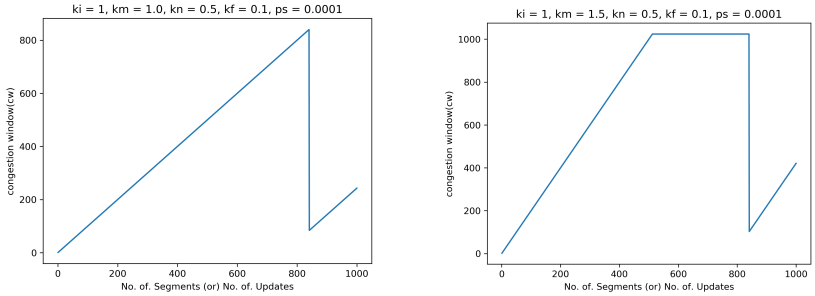

Figure 2: Figure for point (ii) containing plot 2(L) and 10(R)

(ii) When $k_m = 1.5$ and $p_s = 0.0001$, the cw value reaches maximum value of 1024, but in the case of $k_m = 1.0$ and $P_s = 0.0001$, the cw value does not reach maximum value at any point. The flat line in the right fig. indicates the reaching of maximum value.

(iii) The maximum value of cw, when $p_s = 0.01$ is roughly around 150 - 160 irrespective of all other values.

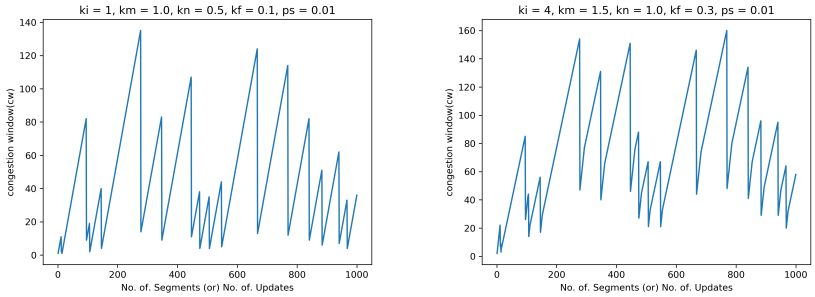

Figure 3: Figure for point (iii) 1(L) and 31(R)

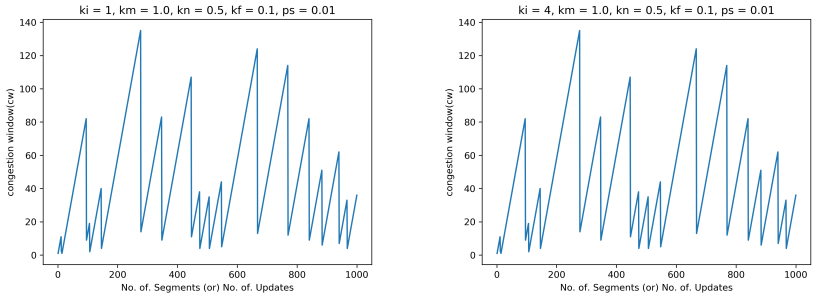

Figure 4: Figure for point (iv) 1(L) and 17(R)

(iv) There is no much difference between the plots, when k_i is changed keeping all others constant as the reason might be that the initial value of 1KB or 4KB is relatively small when compared to maximum value cw takes in the plot. The no. of. updates being 1000 also decreases the effect initial values can have in a plot.

(v) The effect of k_n on the cw value is so less because the term $k_n * MSS * \frac{MSS}{cw}$ is dominated by the term $\frac{MSS}{cw}$. Reason being 1/cw value is very much lesser on an average and hence multiplication of k_n to this very less value has very small effect.

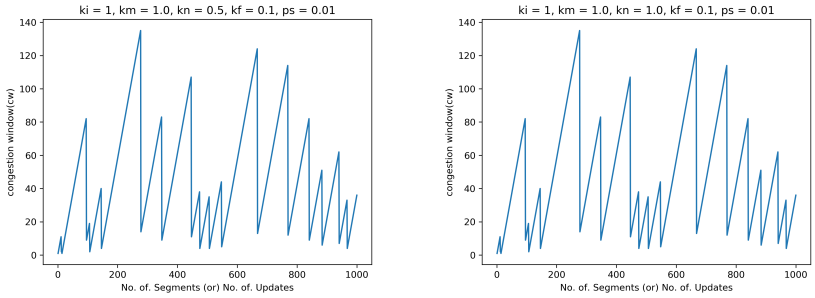

Figure 5: Figure for point (v) 1(L) and 5(R)

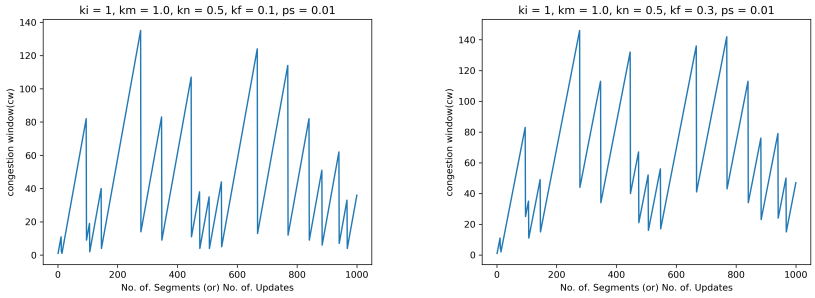

Figure 6: Figure for point (vi) 1(L) and 3(R)

(vi) The effect of k_f can be seen whenever there is a timeout, when $k_f = 0.1$, cw value goes down to smaller value when compared to that of $k_f = 0.3$. All the cw values after a timeout are affected due to this multiplier term k_f.

5 Learnings

1. Working and importance of TCP congestion control algorithm (Go-back-N algorithm in this case).

2. Drawing plots in python using built-in matplotlib library.

3. Writing bash scripts which helps us run the programs will different inputs many times one by one.

4. *script* command in linux terminal which will help us record the testing

session of the program.

5. File handling in $c++$ and *python* like opening files, reading and writing into them.

6 Conclusion

1. When P_s is high, there will be lots of ups and downs for congestion window value as there is high chances of getting a timeout.

2. The given values for k_i is so close in our case and hence the plots have not much differences.

3. For higher values of k_m, we can reach maximum quickly compared to lower values of k_m.

4. The given values of k_n in our case when multiplied by a number very much less than 1 gives almost equal values and hence there is not much difference in terms of k_n in our case.

5. The lower values of k_f will make cw go to a lesser value when a timeout occurs compared to a higher value of k_f.

All these Experimental Observations match with the Theoretical observations drawn from the Formulas.

7 References

a. https://linoxide.com/script-command-recorder/

b. https://www.taniarascia.com/how-to-create-and-use-bash-scripts/