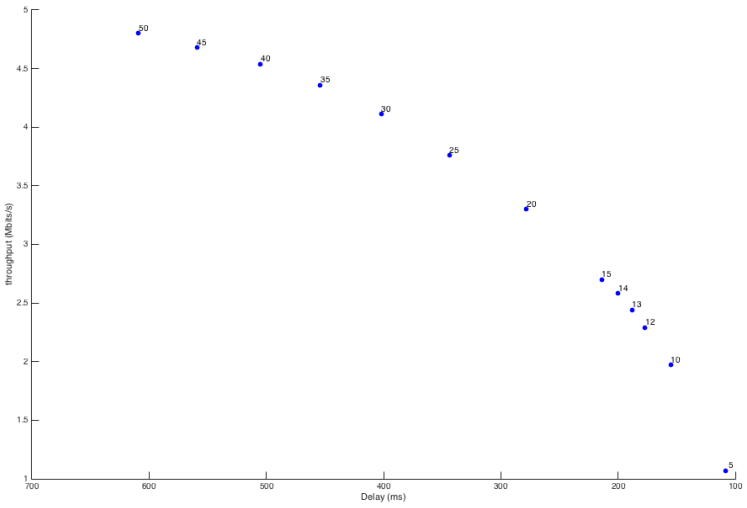
CS 244 Problem Set 2

Winter '16, Sean Choi, Eyal Cidon

The github repo url is https://github.com/yo2seol/cs244
The latest commit hash is
https://github.com/yo2seol/cs244/commit/5c948731da93bf96b98e5233c0c6fac5a6090095

Question 1

Here is the graph of the throughput vs 95-percentile signal delay for varying window sizes, where the window sizes are denoted as the numbers above the dots. Maximum score of 12.98 occurred at



window size 13. Finally, as for the repeatability of this experiment, the value of score often changes by 0.01, but it stays pretty close and constant.

Question 2

As for the AIMD implementation, I used the additive increase constant of 1 and the multiplicative decrease constant of 1/2. Also for other variables, I set the timeout as 70ms as it worked the best. Given this configuration it gave the following statistics.

Throughput: 2.28 Mbits/s
95th percentile per-packet queuing delay: 84ms
95th percentile signal delay: 122ms
Power score: 18.69

The score was much better than the score given by the simple window size implementation that were used for part A.

Question 3

For the best performing metrics, I set the upper threshold to 70ms lower to 50ms, if we are above 70 we decrease window size by 1 if below 50 we increase by 1. Given this, it gave the following statistics.

Average throughput: 2.61 Mbits/s
95th percentile per-packet queueing delay: 52 ms
95th percentile signal delay: 91 ms

Given this configuration, we got the score of 28.68.

Question 4

For the final implementation, I implemented a variant of TCP reno. We basically have three phases, slow start, congestion avoidance and timeout phase. Given this, I adjust the window size accordingly. The important decisions I had to make were following. First, I needed to tweak the timeouts for this algorithm. The initial 70 milliseconds that I have used for part A did not work well. I ended up increasing it to 90ms. I believe that is due to the algorithm performing more close to the threshold. I also had to work with what the initial window size is. I used the window size of 13 that was given from part A. It seemed to be the value with the best output. Finally, I had to choose the number of timeouts until another window decrease occurs. This was a bit tricky, because we were not able to count the duplicate acks. Thus, I looked at the latest window size and the number of packets above the timeout. I decreased the window size only if we have more packets above the timeout than half of the prior window size. This seemed to work the best. The final score was 33.72.

Question 5

Our algorithm name is seaneyal