6.829 Problem Set 2

Alan Ho, Zach Kabelac

We started testing the link by using a constant window size. From our tests, we found that the score peaked around a constant window size of 20. Both the delay and throughput increased for each window size, but the score eventually peaked. We also found that the measurements were fairly consistent with a variation of .05 for a given test. In the plot, the window size is decreasing with the delay on the x-axis.

As instructed in the assignment, we started by implementing an AIMD (additive increase, multiplicative decrease) scheme similar to that of TCP. We used similar constants. In particular, the additive increase of the window size was by one packet per round trip time, implemented by accumulating fractional window size increases for each ACK. The constant for multiplicative decrease was 2, or halving our windows size whenever we detected congestion.

This scheme alone does not work at all if you strictly adhere to the TCP definition of congestion for a client, which is whenever there is a dropped packet. Because of the reliability guarantees of cellular networks (99.99% packet delivery), the system never sees any packet drops and therefore never adjusts the window.

The assignment recommends a simple delay-triggered scheme. We measured the average RTT over the past 10 ACKs to sense the current delay of the network. This delay is correlated with the current capacity of the network. We set a threshold of 70 ms. If the average RTT exceeds the threshold, we assume that the network is congested and do multiplicative decrease. We noticed that this alone would cause the congestion window to collapse to 0 whenever there was some congestion, because it would be visible over a few ACKs. So we also enforced a “cooldown” period to ensure that the congestion window did not get halved too many times if there was a temporary intermittent drop in capacity.