Congestion Control: Part 2

Jonathan George

1 Description

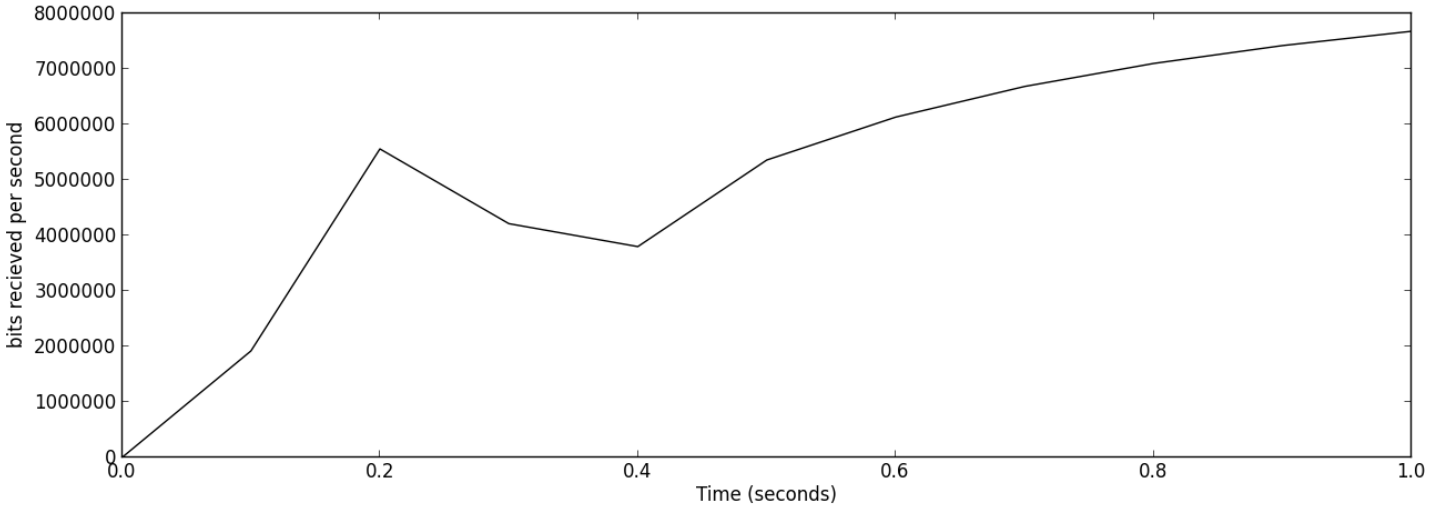
This lab continues to explore the implementation of congestion control we implemented in the last lab. This lab mainly consists of experiments which show how TCP congestion control reacts to different circumstances.

2 Basic Experiments

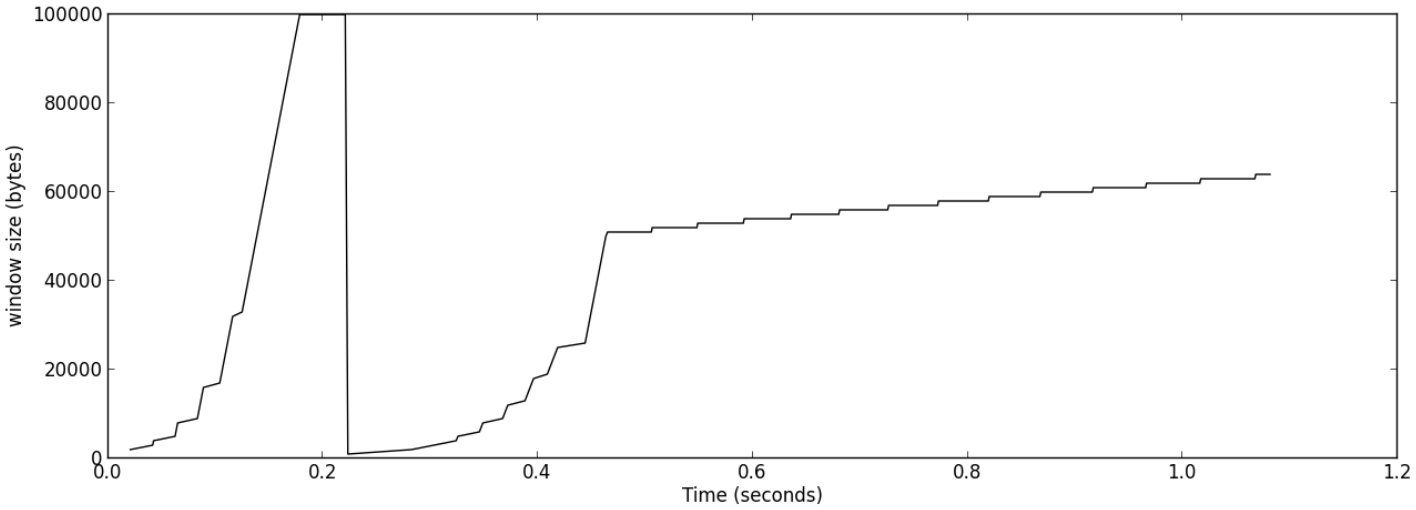
This first section shows how TCP reacts to different flow amounts: 1,2, and 5. It uses a simple two node network. Each link has a bandwidth of 10 Mbps, propagation delay of 10 ms and 100 packet queue size. A 1 Mb file is transferred from n1 to n2. Three graphs are generated in each case: bps vs time, window size vs time, and queue size vs time. The queue size vs time graph also includes 'X' marks where packets are dropped due to queue overflow.

2.1 One Flow

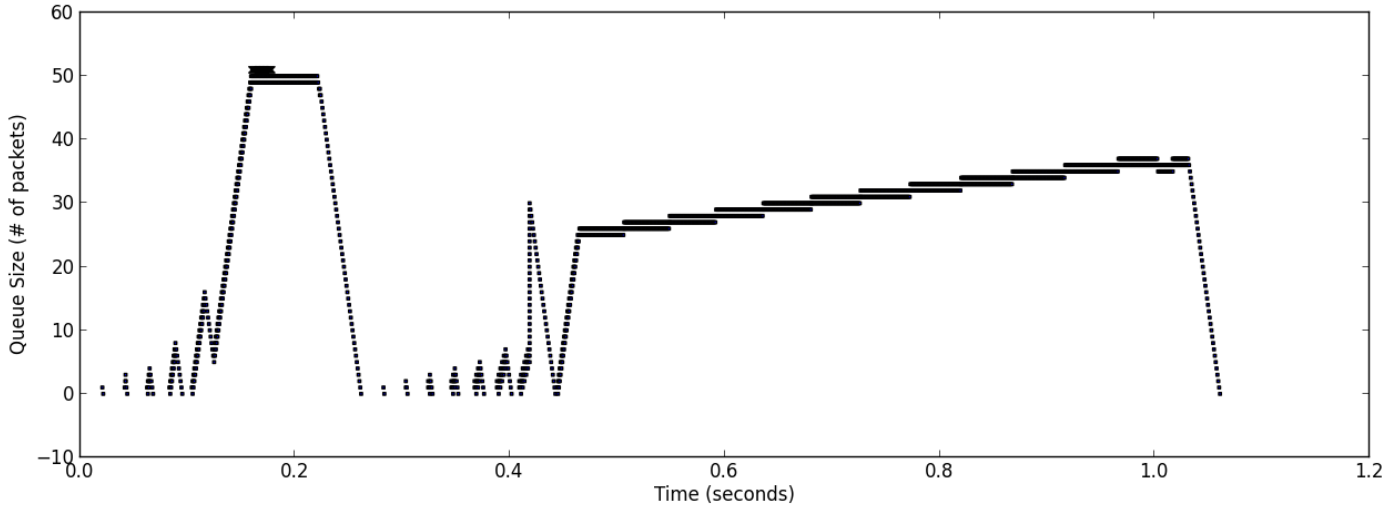
Rate



Window



Queue

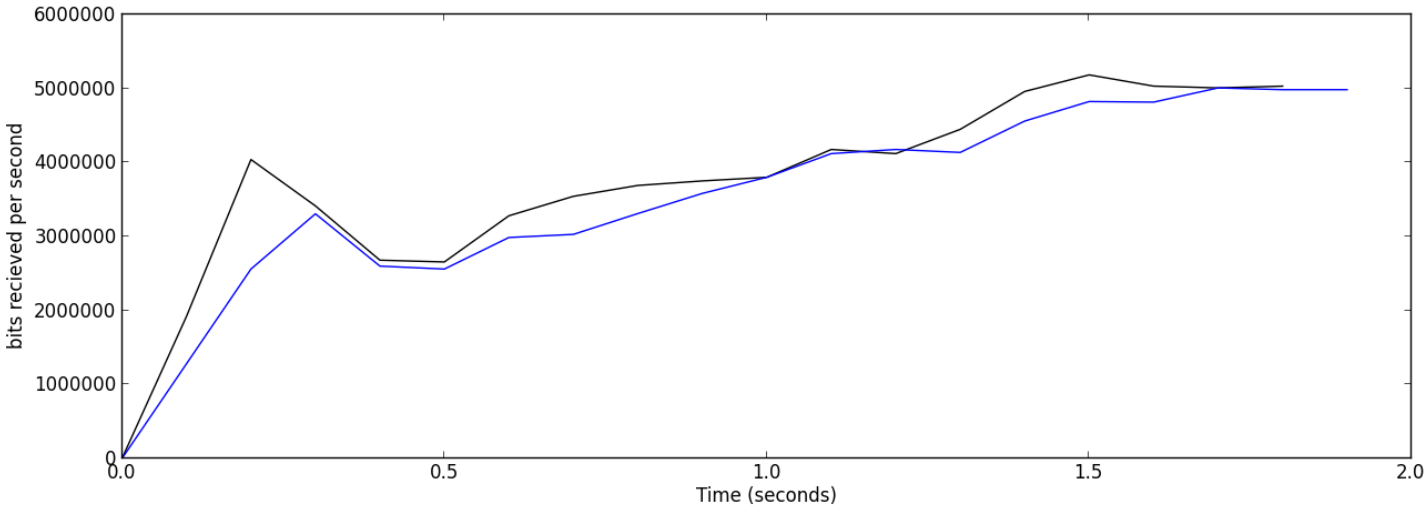


As you can see from the three graphs above, the window size increased exponentially until reaching the queue filled up. It then experienced a loss event and restarted in slow start until it reached the threshold and then continued to increase slowly until the whole file was sent. It used most of the bandwidth towards the end.

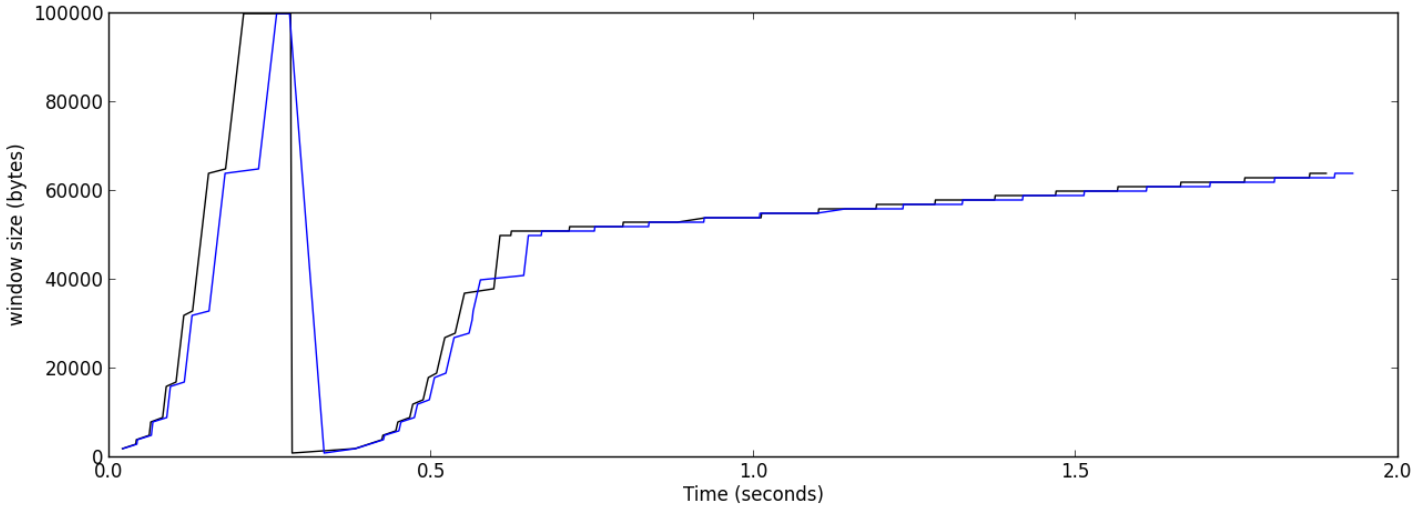
2.2 Two Flow

In this section, the two flows were graphed together.

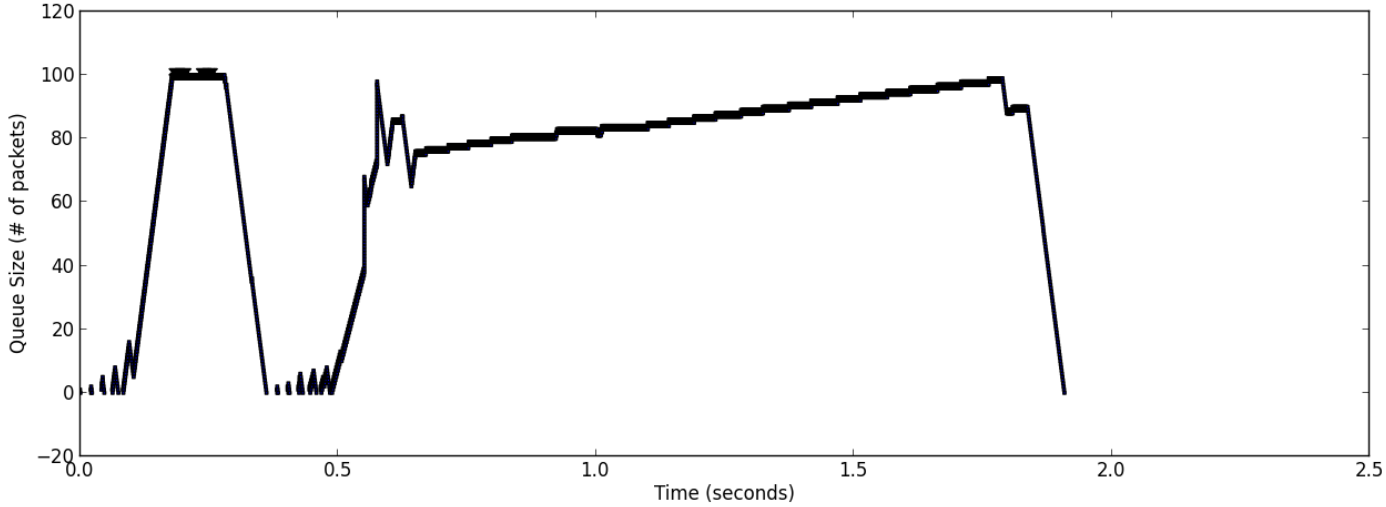
Rate



Window



Queue

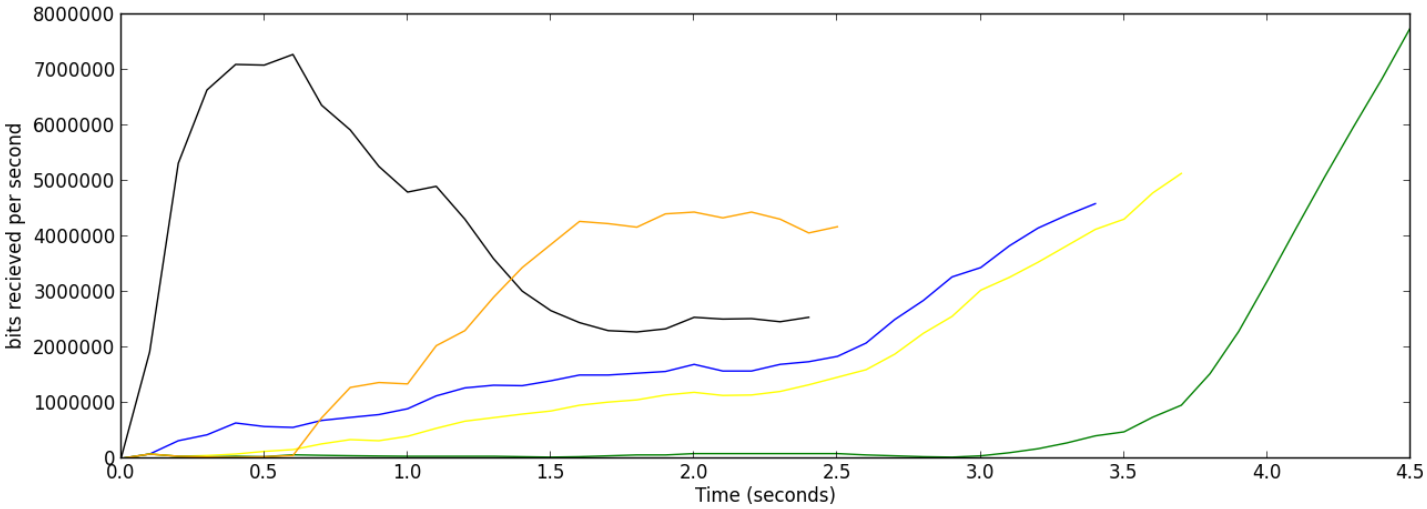


As you can see in these graphs, the queue overflows pretty quickly and both flows have to back off. Over time they even out to both use about 50% of the bandwidth.

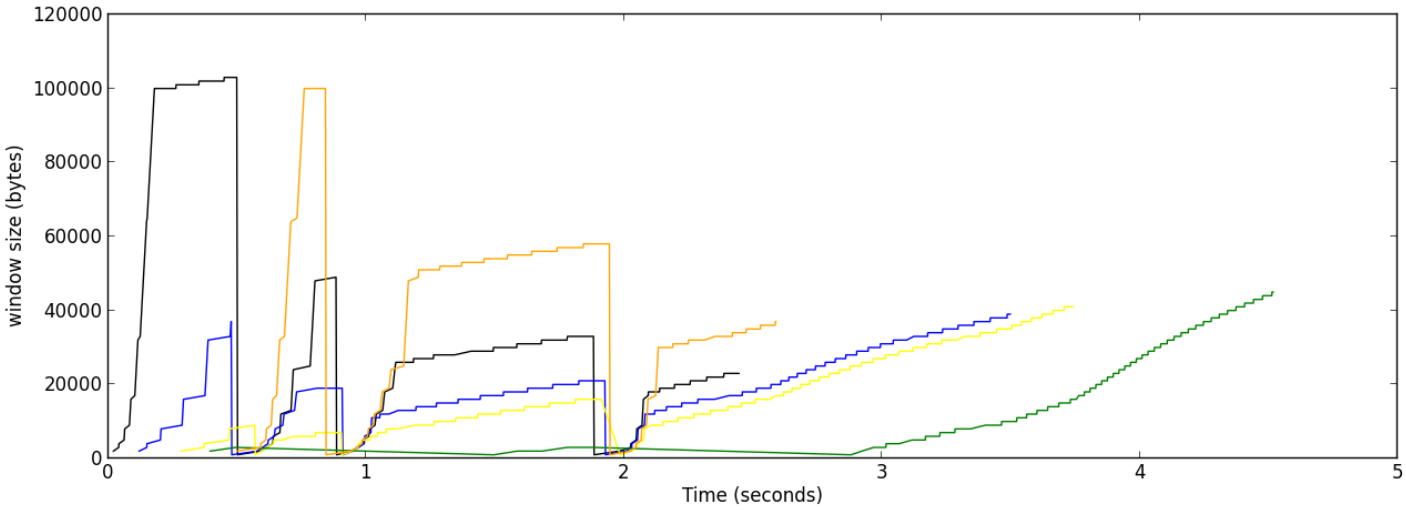
2.3 Five Flow

In this experiment, we stagger the different flows in order to see how they will interact as one flow enters and another leaves.

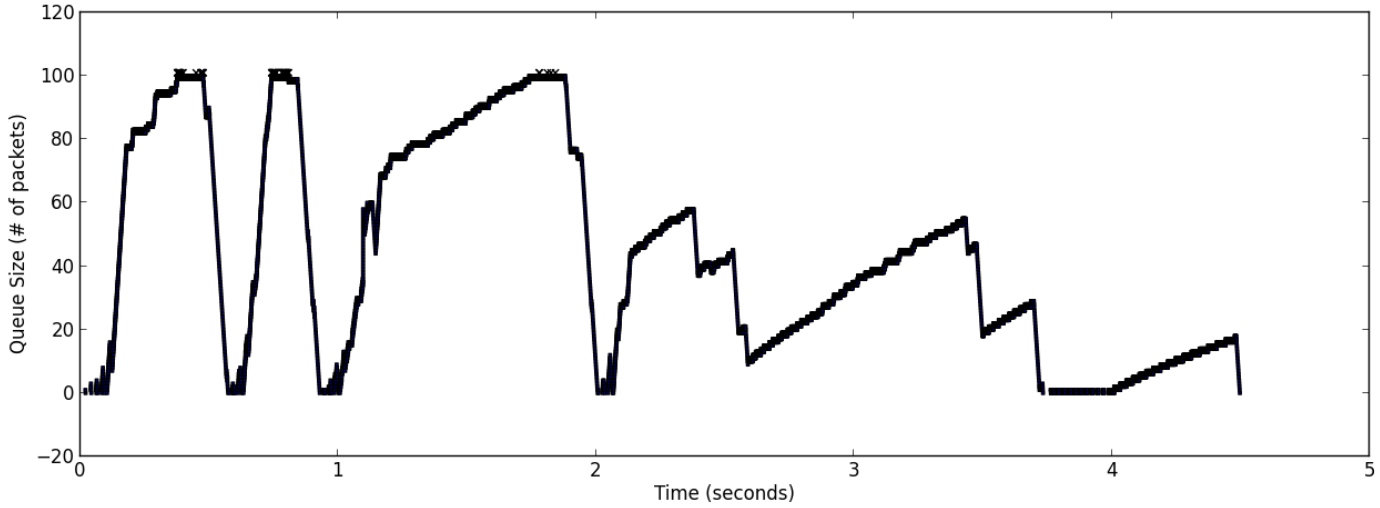
Rate



Window



Queue



As new flows start, they quickly grow larger than the other flows. However, once the queue overflows they drop down closer to the other flows. Over time, they find a good equilibrium. As a flow finishes, the other flows continue to grow and consume the bandwidth. This consumption of free bandwidth is pretty slow due to the additive increase of the window size.

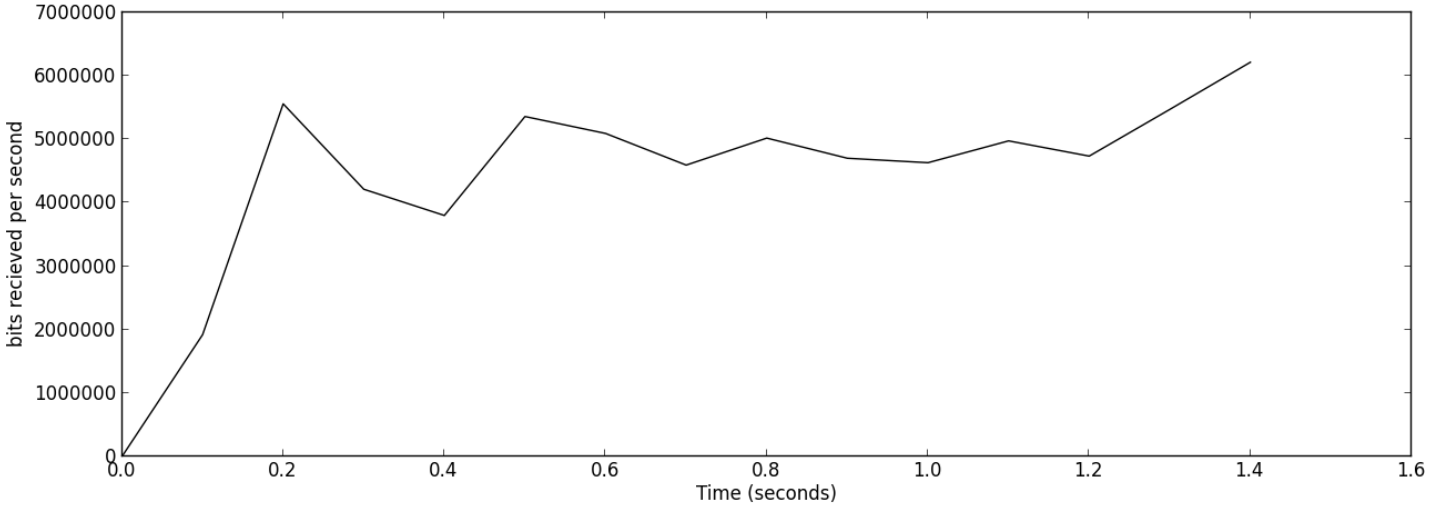
3 Advanced Experiments

In this section, we will examine different method of implementing fairness between flows.

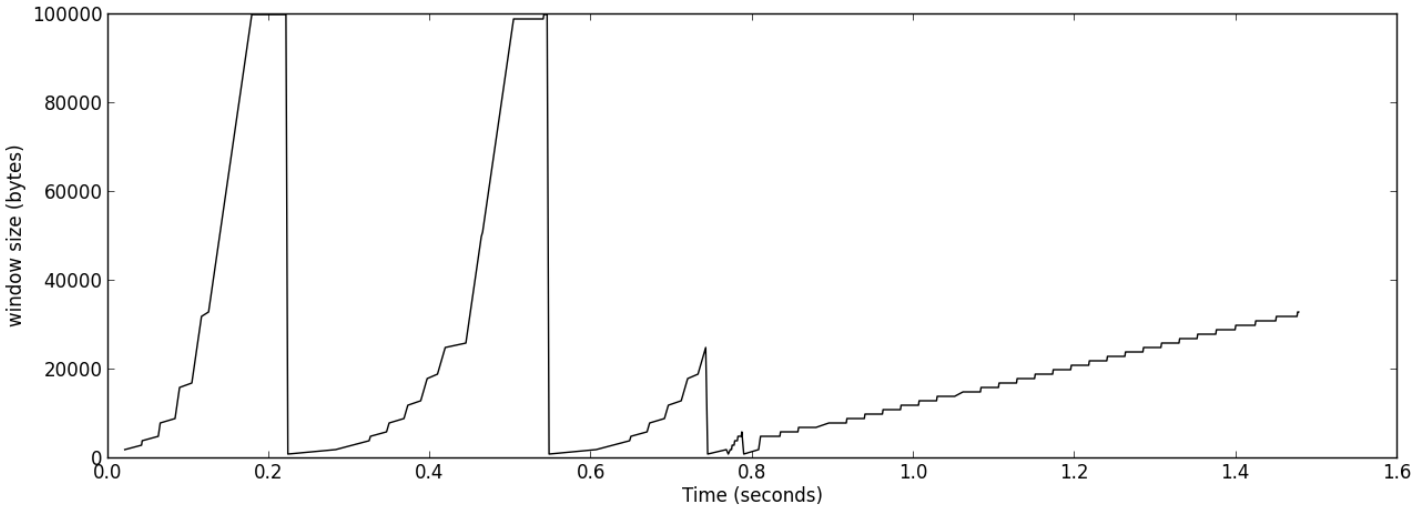
3.1 Additive Decrease

This first section attempts to use additive decrease instead of multiplicative decrease to back off in the event of a loss.

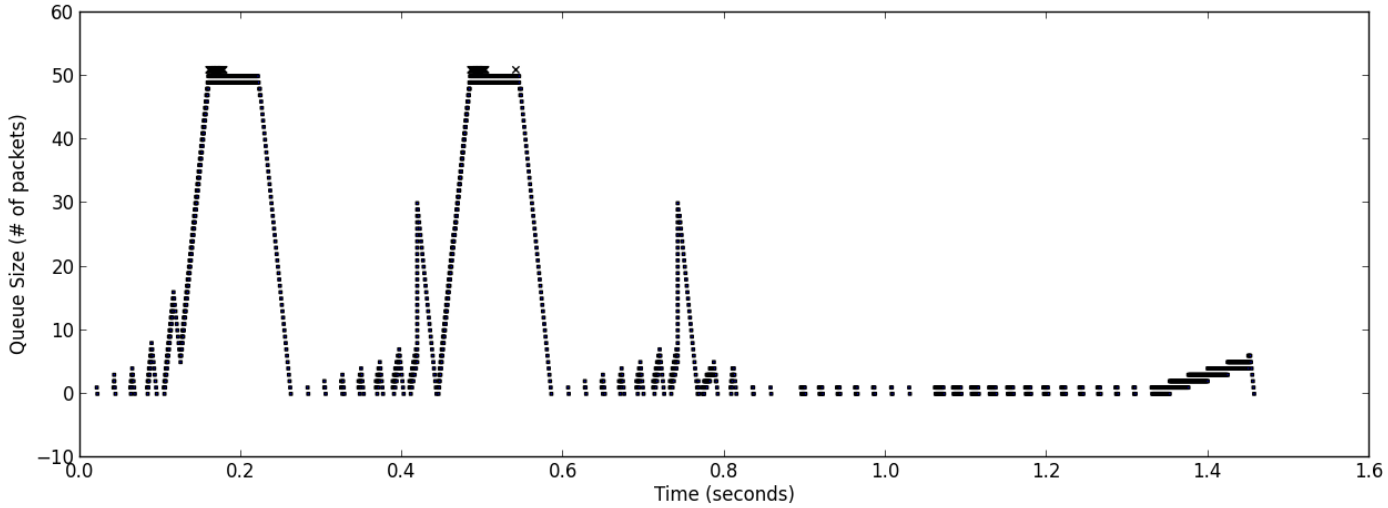
Rate



Window



Queue

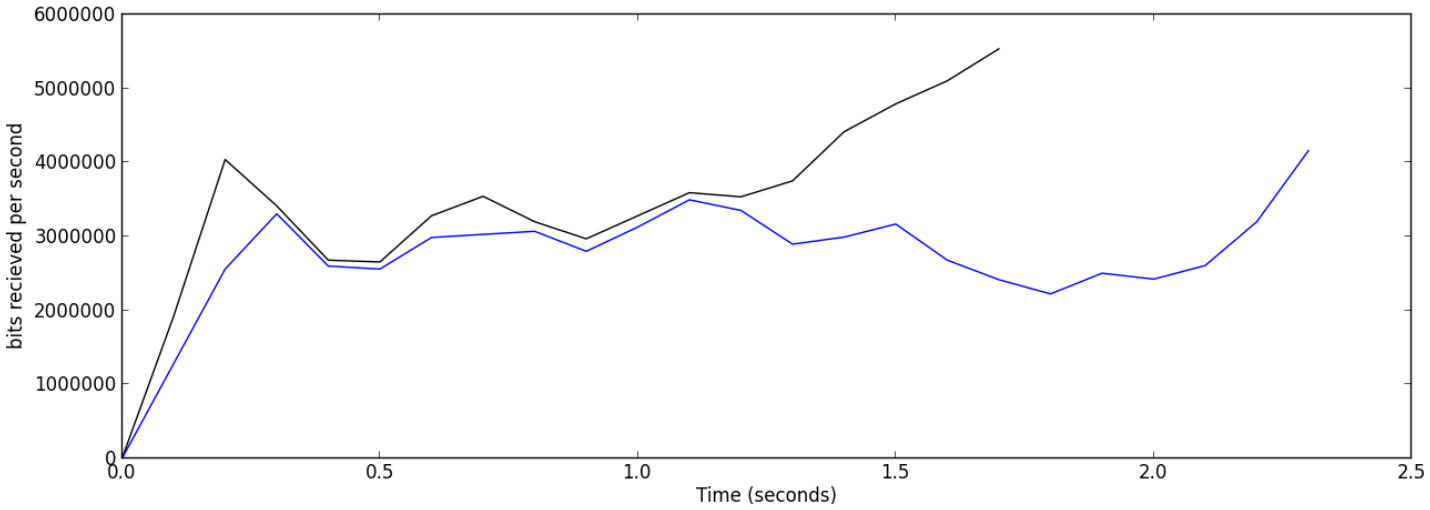


As can be seen the window size graph, additive decrease has a hard time finding a stable threshold level. This causes it to jump up and down quickly at first and eventually the threshold is sufficiently decreased that it calms down. This stabilizes eventually, but isn't as good as before.

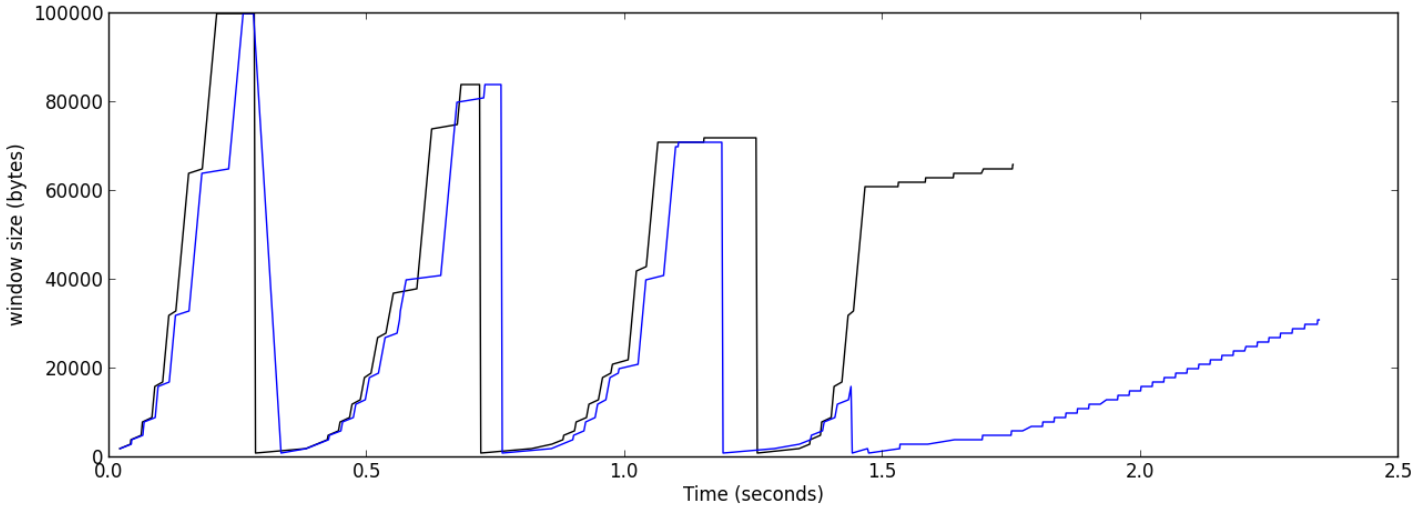
3.2 AIMD - 5/6th factor

In this test, we changes the factor by which multiplicative decrease lowers the threshold by. Instead of setting the threshold to 1/2 of the window size, we set the threshold to 5/6th of the window size.

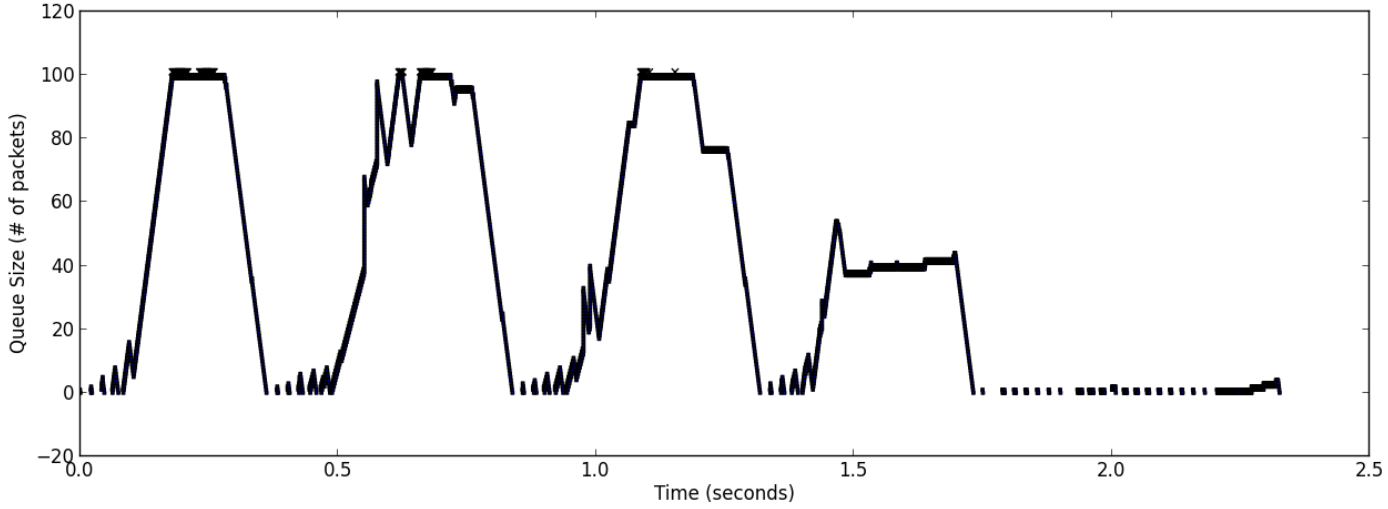
Rate



Window



Queue



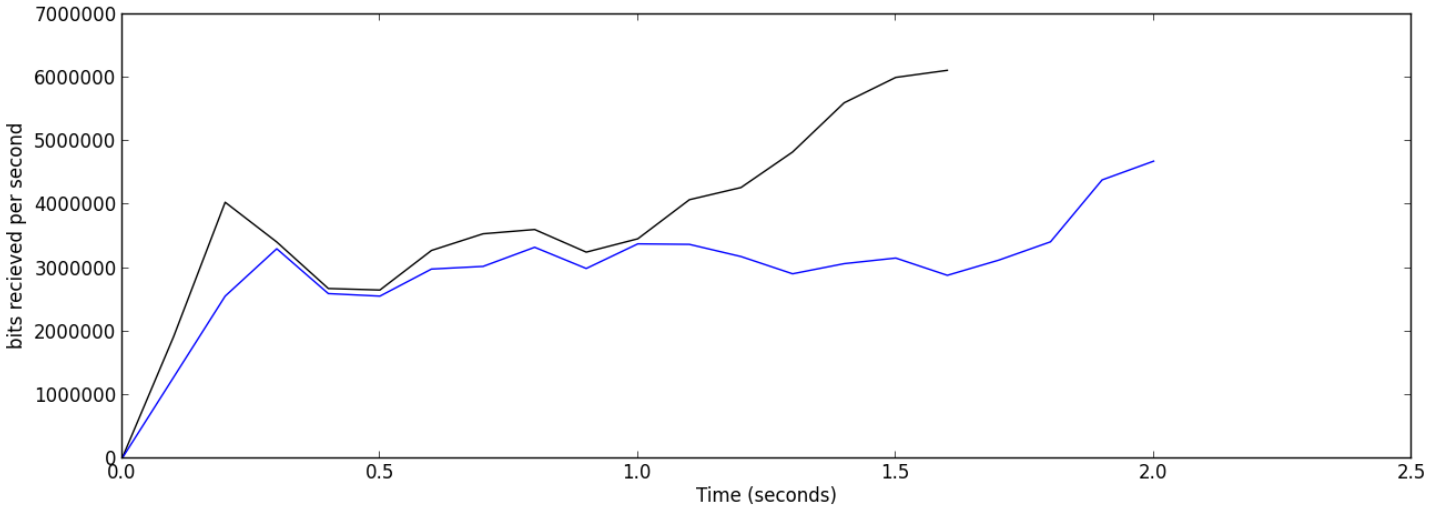
Comparing to the basic two flow graph, it can be seen that the queue size jumps up and down more frequently due to the decreased drop in threshold values. Eventually it does level off, but in this case it

didn't level off evenly. The blue flow sees a erroneous packet loss close to time 1.4 seconds. If needed this loss can be address further, however Justin approved this graph before as being accurate besides this point. Overall I think that this method could create a stable situation, but would require a larger file over a longer period of time to accomplish it.

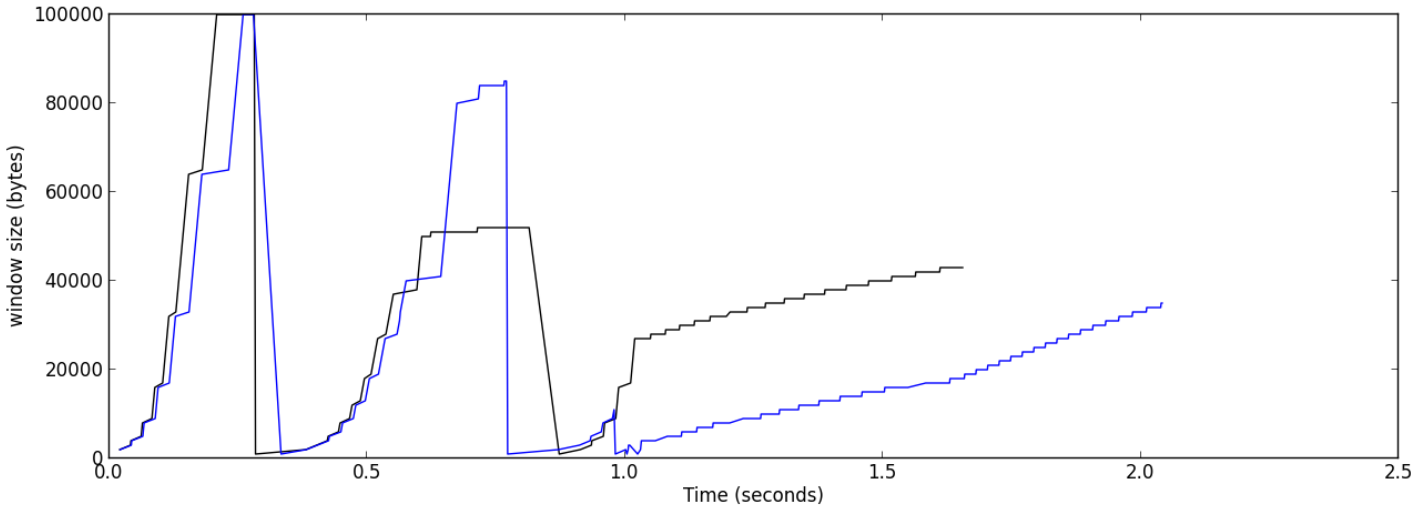
3.3 Competing AIMD

In this section, we set multiplicative decrease factor to 5/6 (black), but left the other flow at 1/2 (blue). This creates an imbalance as can be see in the graphs below.

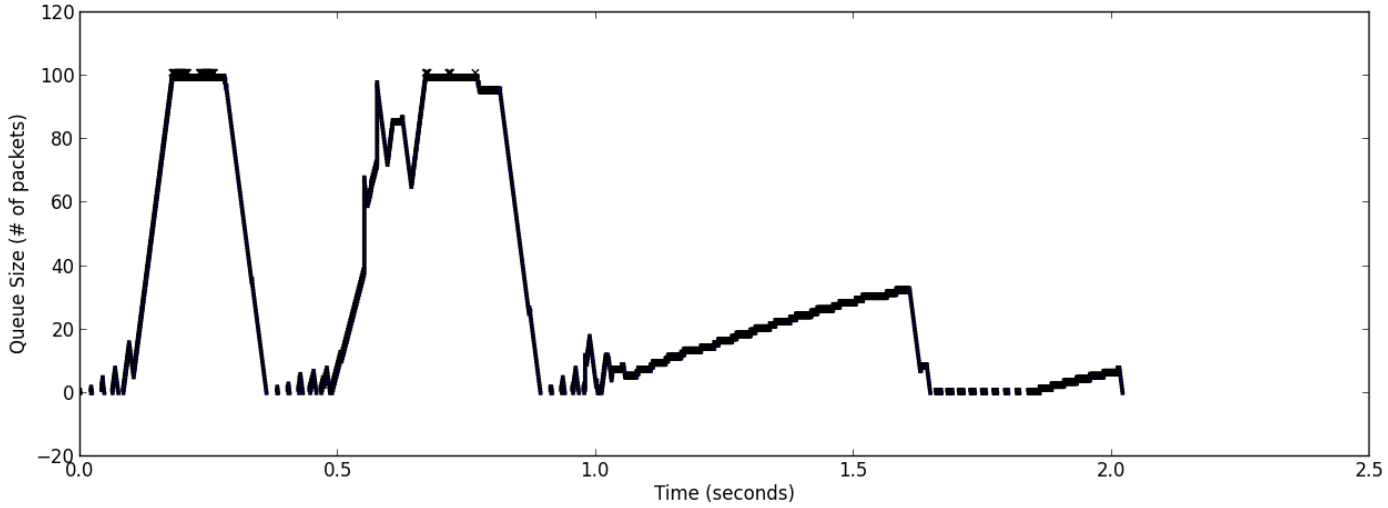
Rate



Window



Queue

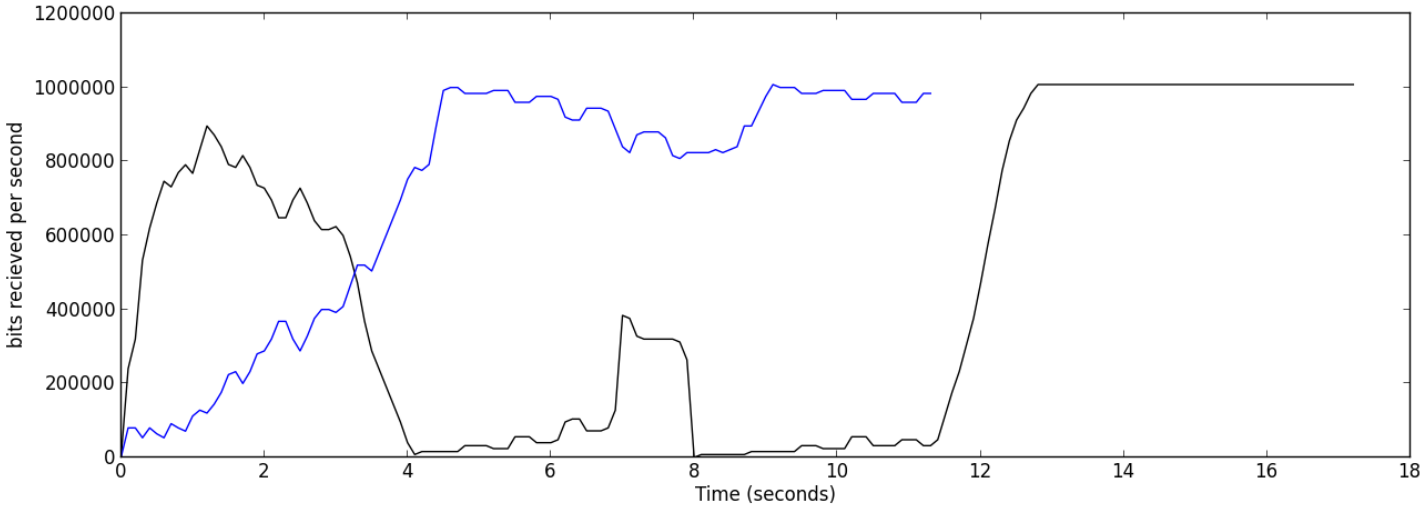


Instead of staying even with each other, the black flow gets ahead of the blue flow. The bps becomes far in the favor of the black flow and it finishes long before the blue.

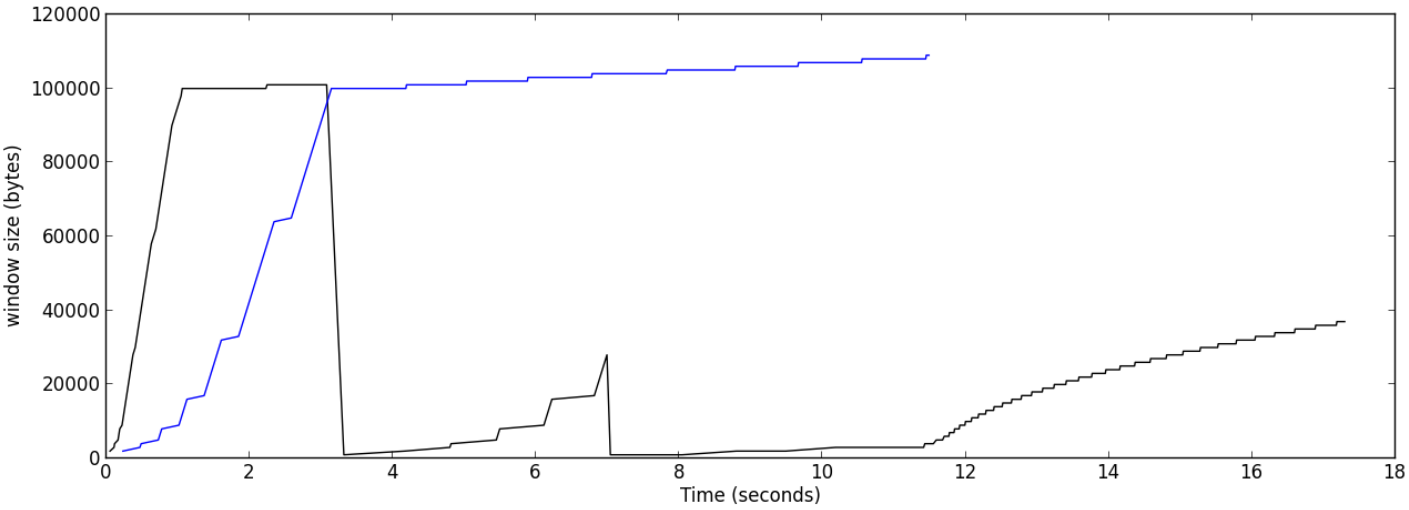
3.4 Competing RTT

In this final section, we created a four node situation where two flows start on different branches but meet at the second node and travel to the same destination. Also, one flow (blue) is has a longer propagation delay.

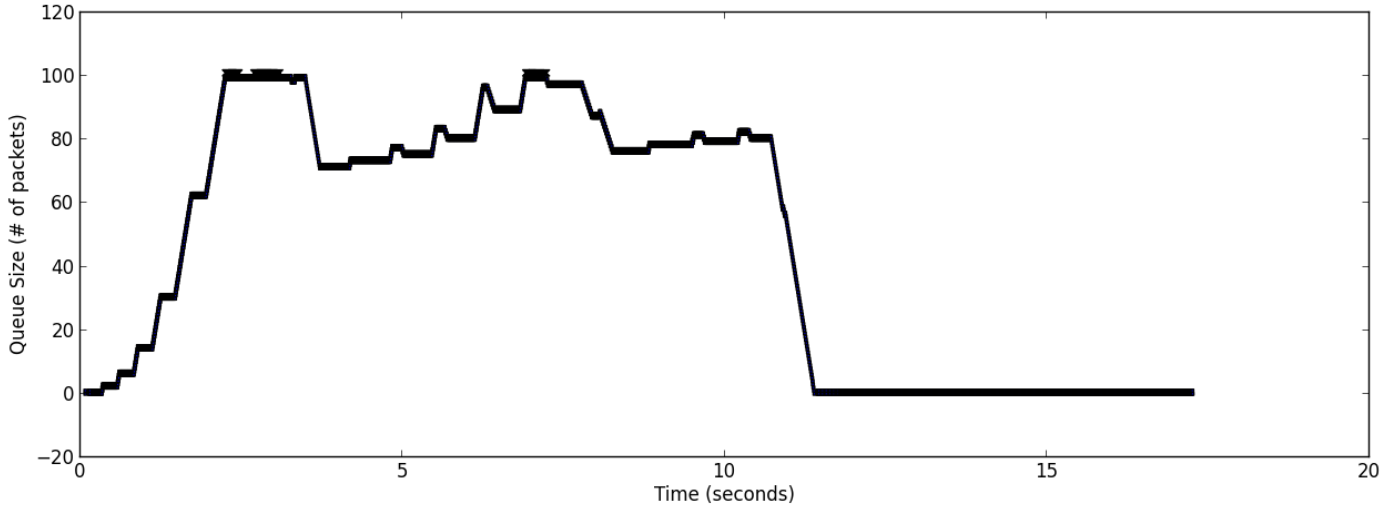
Rate



Window



Queue of Link connecting the center node to destination node



As we can see from the window size graph, the slower blue flow misses the drop events which the other flow runs into. This cause the blue flow to dominate the bandwidth because it runs into its threshold and starts additive increase while the black flow is stuck being very low. Further along it can be see that as the black flow tries to increase back up higher, it again reaches the queue limit and loses more packets. The source of all of the black flows problems are due to the smaller delay it has getting it's packets to the crowded queue and so it also sees the most dropped packets. The blue flow unwittingly takes all of the bandwidth for itself.