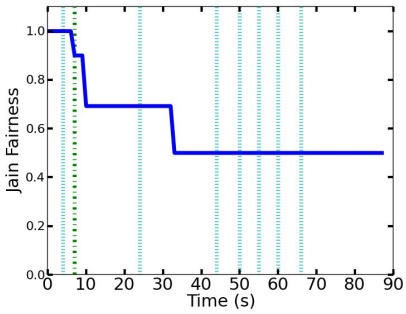
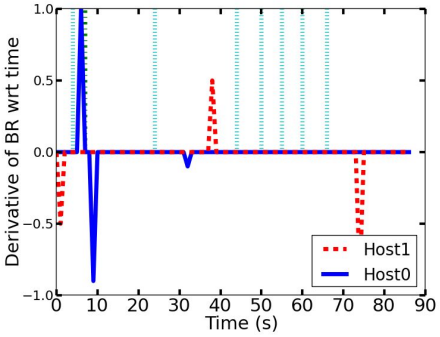
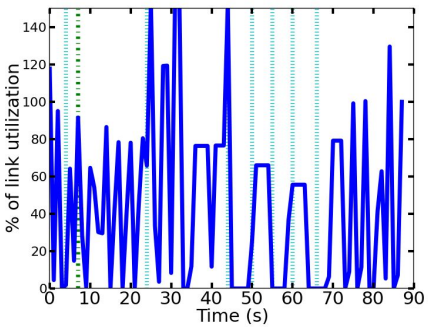
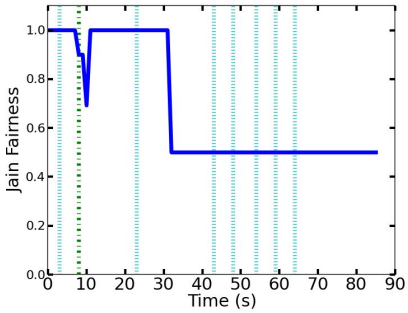
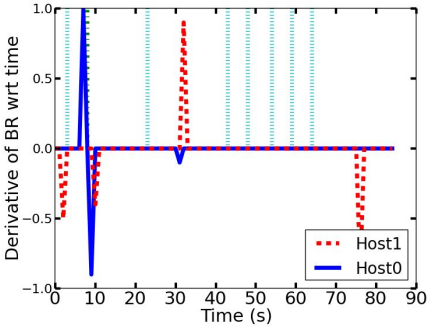
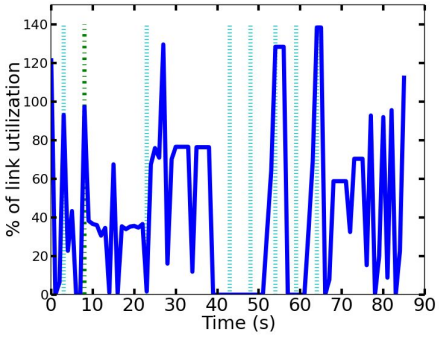
Jackson Chen (jc4697)
Computer Networks Project 1 Write up

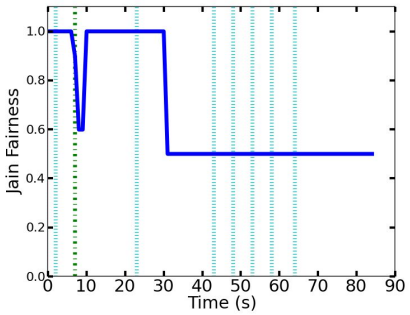
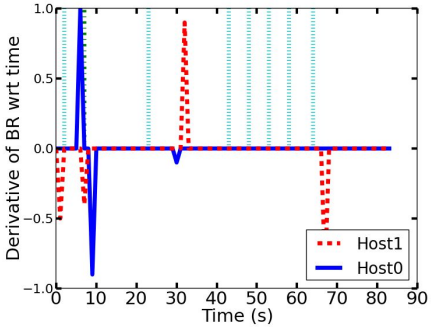
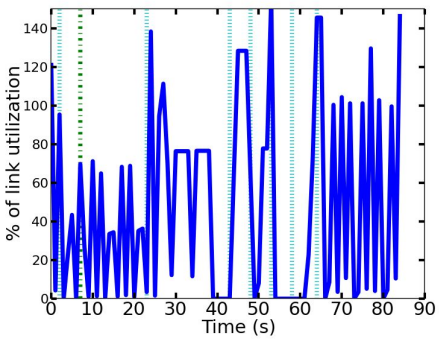
Alpha = 0.1

Fairness	Smoothness	Utilization
		

Alpha = 0.5

Fairness	Smoothness	Utilization
		

Alpha = 0.9

Fairness	Smoothness	Utilization
		

Writeup

In terms of fairness, all of the graphs drop to a Jain Fairness score of 0.5 after 30 seconds. This means the primary differentiator in fairness will happen in the first 30 seconds. In the first 30 seconds, it appears that larger alpha values generally have higher fairness values with both the 0.5 and 0.9 alpha values performing significantly better than an alpha of 0.1. The alpha of 0.1 permanently drops once the second client starts to stream from the other proxy, while 0.5 and 0.9 revert back to a score of 1 shortly after the event occurs. From a numeric perspective, a higher alpha value biases the average throughput towards the most recent chunk's throughput, which means as the two clients begin to compete for throughput, the throughput of their most recent chunk suffers (for both clients). This will significantly lower the requested bitrate for the next chunk for both clients, establishing fairness in resource usage.

In terms of smoothness, this is the one metric where all three alpha values are very similar. Host0's spikes are all the same magnitudes at the same times. The only difference in this metric is alpha 0.1, which has fewer spikes and lower magnitudes for Host1. This makes sense because a lower alpha 0.1 biases the weighted average in favor of the previous weighted average rather than the most recent chunk throughput. This means that the throughput estimate will be generally smoother with smaller variation as it new values are unable to significantly change the average throughput value.

In terms of utilization, all of the alpha values lead to vary spiky graphs. As sometimes both clients will be requesting chunk data, while other times, the connection may be overwhelmed and disconnected. The most chaotic period in the graphs (i.e longest periods at 0% utilization) occur between 40 seconds and 70 seconds. Here alpha of 0.9 spends the shortest amount of time at 0% utilization compared to the other values, which translates to higher efficiency. From above, a high alpha value biases the requested bitrate towards the most

recent chunk's throughput. When the throughput gets significantly lowered because the other client begins to request information, it will cause dramatically reduce the request bitrate much more quickly to prevent the connection from being overwhelmed. One interesting note is that alpha of 0.5 is the least spiky graph of them all. This can likely be due to the fact that an alpha of 0.5 dramatically reduces the bitrate when a lowered throughput is detected, much like an alpha of 0.9, but does not overwhelmingly increase the bitrate when a higher throughput is detected, unlike an alpha of 0.9.