15-441: Computer Networks
Fall 2013
School of Computer Science
Carnegie Mellon University

Project 3: Video CDN
Write Up

Team 11
:Lauren Milisits (lvm)
Eric Telmer (eit)

Table of Contents

1. Implementation Details of Proxy

This section will describe the implementation details of our proxy.

When the proxy is run, it first opens up a proxy log file for writing the proxy output in the correct format. The proxy then sets up a browser listener socket and waits for action, or to request a connection. When the proxy requests a new connection, if the connection is valid, then a browser-server connection is created to allow for streaming. The proxy then determines if the connection is from a browser or a server.

If the connection is from a browser, the proxy behavior depends upon what the GET request is for. If the request is for a chunk of the form "/vod/###SegX-FragY," then the proxy starts a new chunk, sets chunk->time_started to the current time, and adapts the bitrate based on the best available bitrate that can be supported by the current throughput. The new chunk is started and the GET request is modified with the adapted bitrate and the response is sent.

If the connection is from a server, the proxy behavior depends again upon the GET request and the header details. For a header with Content-Type "text/xml" request, the current throughput is set to the lowest bitrate available because the stream is only beginning.

If the header request Content-Type is of the form "video/f4f", then the proxy measures the number of bytes read, the length of the content, and continues to send and receive until all bytes of the chunk have been read. When the number of bytes remaining is 0, the proxy finishes the chunk.

When finishing a chunk, chunk->time_finished is set to the current time. The throughput for this specific chunk can then be calculated. The current, or average, throughput is also updated using an exponentially-weighted moving average calculation based on Equation 1 and Equation 2 in Section 2 of this report. Finally, the time, duration, throughput, average throughput, bitrate, server-ip and modified chunk name are recorded in the proxy log.

The proxy continues to stream video, adapting the bitrate and logging the information to the proxy log until the connection is lost.

2. Exploring the Behavior of the Proxy

In this section, we describe in detail the behavior of the proxy and bitrate adaptation based on varying alpha values.

Given a new chunk, the throughput of that chunk is calculated using the following equation:

$$T = \frac{B}{t_f - t_s} * 8/1000 \qquad \text{(Eq. 1)}$$

where B is the size of the chunk in bytes, t_f is the finish time of the chunk, t_s is the start time of the chunk and T is the throughput in Kbps.

The throughput for each new chunk is then used to update our average throughput for the proxy. This is done by using an exponentially-weighted moving average calculation. The equation to update the current throughput estimate is the following:

$$T_{current} = \alpha T_{new} + (1 - \alpha) T_{current} \qquad \text{(Eq. 2)}$$

where $T_{current}$ is the current EWMA calculated throughput and α is the constant that controls the tradeoff between a smooth throughput estimate or one that reacts quickly to changes.

Given the above equations, we can run the proxy with varying values for alpha, ranging between 0.0 and 1.0 and then graph the log output using the grapher.py application. The graphs for each value of alpha can be found in the Appendix.

As we increase alpha from 0.1 to 0.5 and to 0.9, it appears as though the fairness increases. In Figure 1a we notice a large dip for about 20 seconds in the fairness. However, as alpha increases to 0.5, we notice that the dip in fairness is for a shorter time, approximately 10 seconds in Figure 2a. Finally, in Figure 3a, any drop in fairness is almost instantaneous. This leads us to believe that a larger value of alpha results in a better overall fairness.

Note that our plots are for time periods longer than 60 seconds. This leads to some of the information of the plots being harder to read, and had the plots generated been for only a 60-second time period, certain details would be easier to distinguish. The fact that our generated plots are for a longer time period make distinguishing the % link utilization between varying alphas a little more difficult. However, we can still discuss and interpret the current graphs.

Overall, it appears as though the % link utilization widely varies over time for all values of alpha. These results can be seen in Figure 1b, 2b, and 3b. It also appears as though the %link utilization, in general, decreases as alpha increases. Notice that in Figure 1c for alpha = 0.1, the %link utilization is often around 60-80% and occasionally reaches higher 100%, but overall does not fluctuate as much as the other plots. In Figure 2c for alpha = 0.5, the %link utilization seems to decrease a little towards 60-80% range. However, in Figure 3c for alpha = 0.9, the plot is very

widely distributed over time. Based on these results combined with those in Figure Xa, we conclude that increasing utilization decreases fairness.

The plots for overall smoothness are shown in Figure Xb. In Figure 1b, there is only 1-2 peaks for each host. As alpha increases, though, the number of peaks increases. These graphs allow for us to conclude that increasing alpha decreases smoothness. As alpha increases, the throughput estimate reacts much quicker to change. However, a smaller alpha value gives a smoother throughput estimate. This could explain the reason why the %link utilization graphs tend to fluctuate much more as alpha increases. Higher alpha results in a faster change of the throughput estimation, and therefore the bitrate can adapt much quicker. Thus, as the bitrate adapts constantly to a fluctuating throughput, the link utilization also fluctuates with this behavior.

Overall, higher alpha results in fluctuating link utilization and a less-smooth throughput, but more fairness. Lower alpha results in less fairness, a smoother throughput and a more consistent link utilization percentage.

3. Vulnerabilities and Additional Notes

Currently, our proxy is vulnerable to Autolab. Each submission results in a failed proxy grade, yet it has proven to work through testing on our local machines.

Our proxy is also vulnerable to pipe-lining.

Overall, our project is at the following stage:

- The proxy can stream video, adapt bitrates, and log in the correct format.
- Our DNS server can implement the round robin protocol.

We need to continue to look into fixing our proxy to pass on the Autolab grading script. We're also still working on completing the BFS and will need to create the proper log calls for the DNS server.

4. Appendix

Fairness plots for all values of alpha:

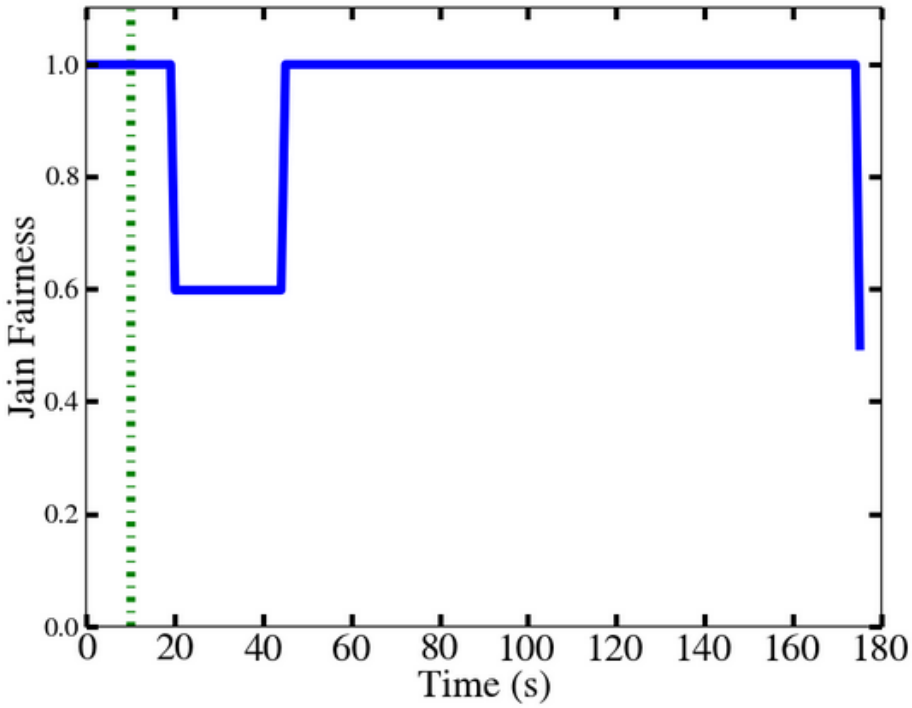

Figure 1a: Fairness plot for alpha = 0.1

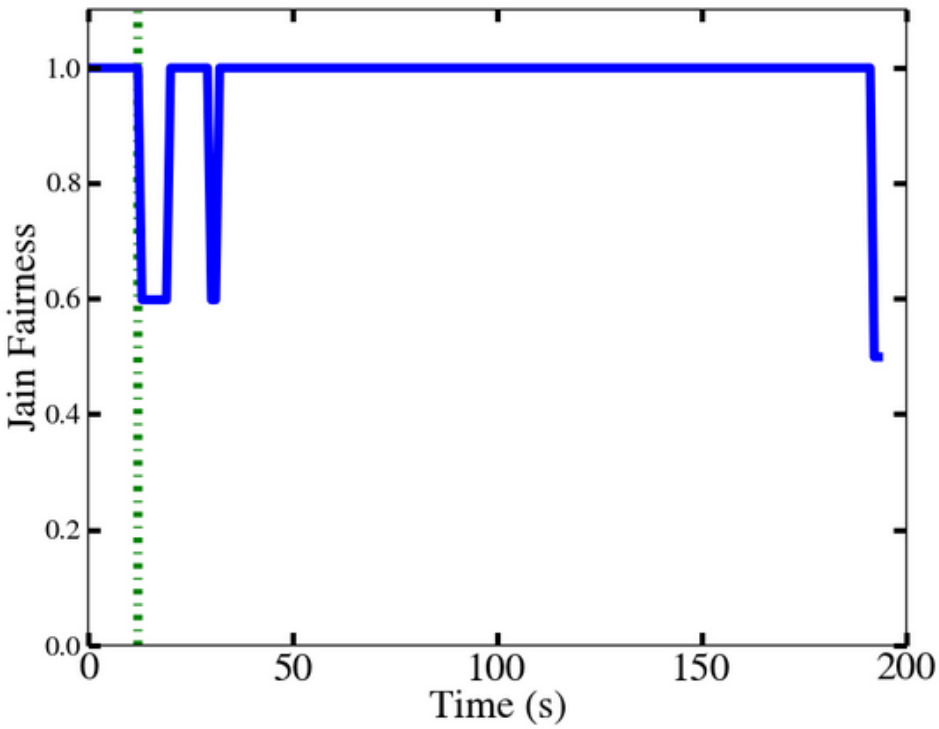

Figure 2a: Fairness plot for alpha = 0.5

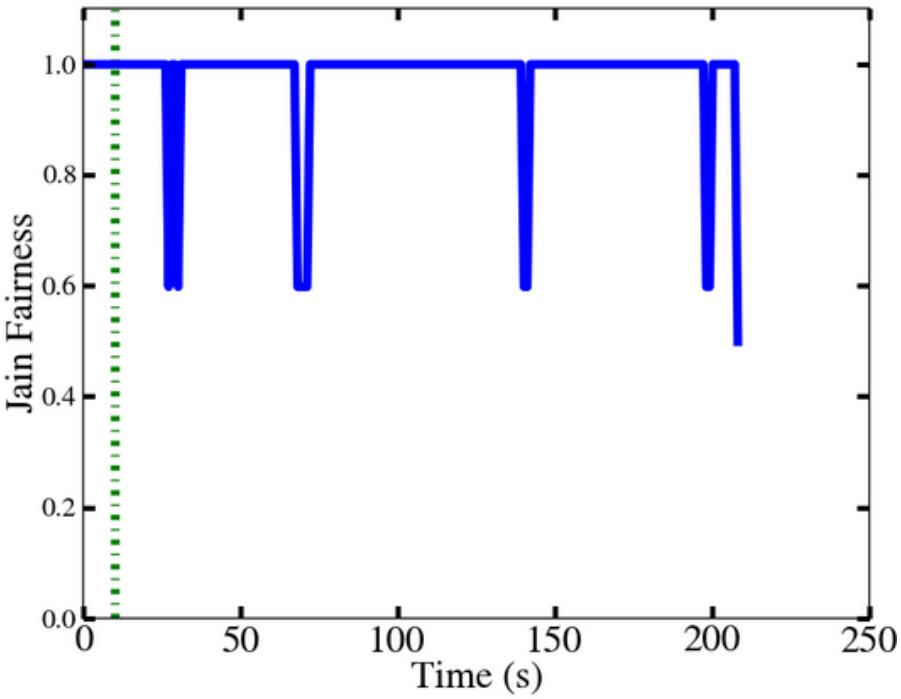

Figure 3a: Fairness plot for alpha = 0.9

Smoothness plots for all values of alpha:

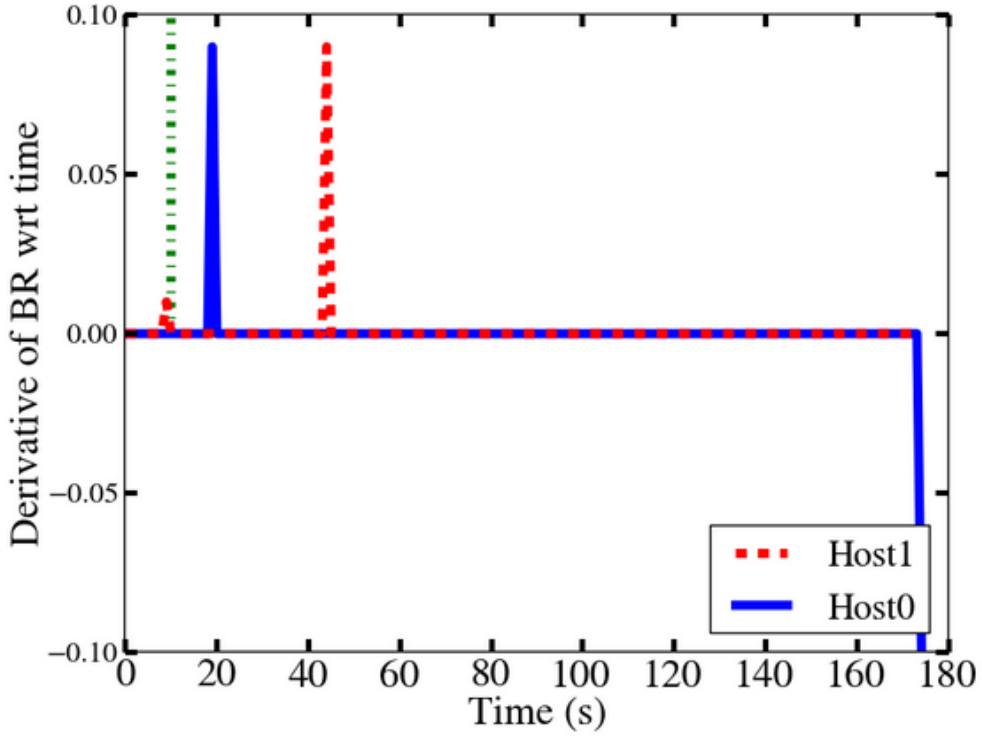

Figure 1b: Smoothness plot for alpha = 0.1

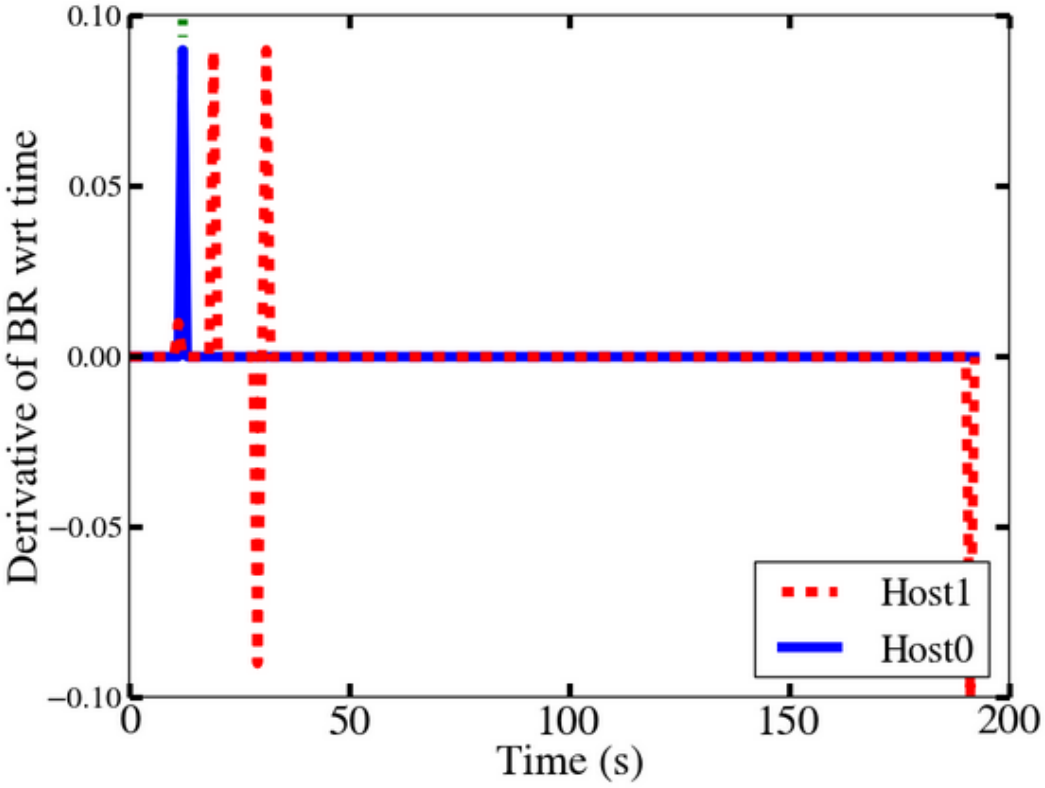

Figure 2b: Smoothness plot for alpha = 0.5

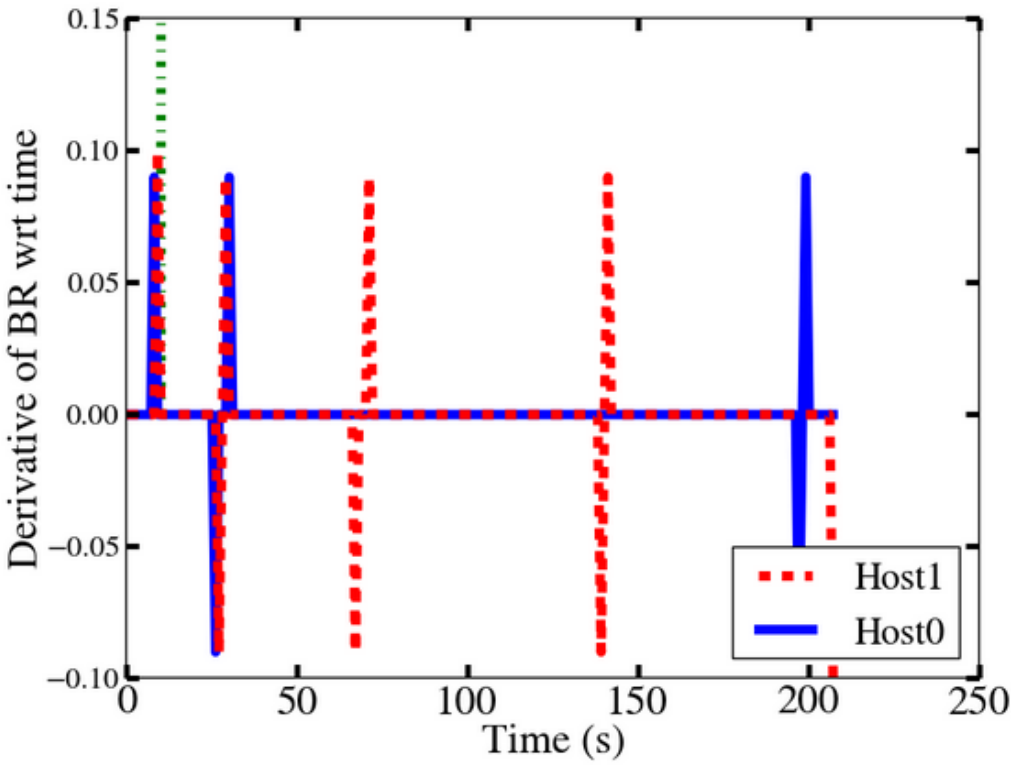

Figure 3b: Smoothness plot for alpha = 0.9

Utilization plots for all values of alpha:

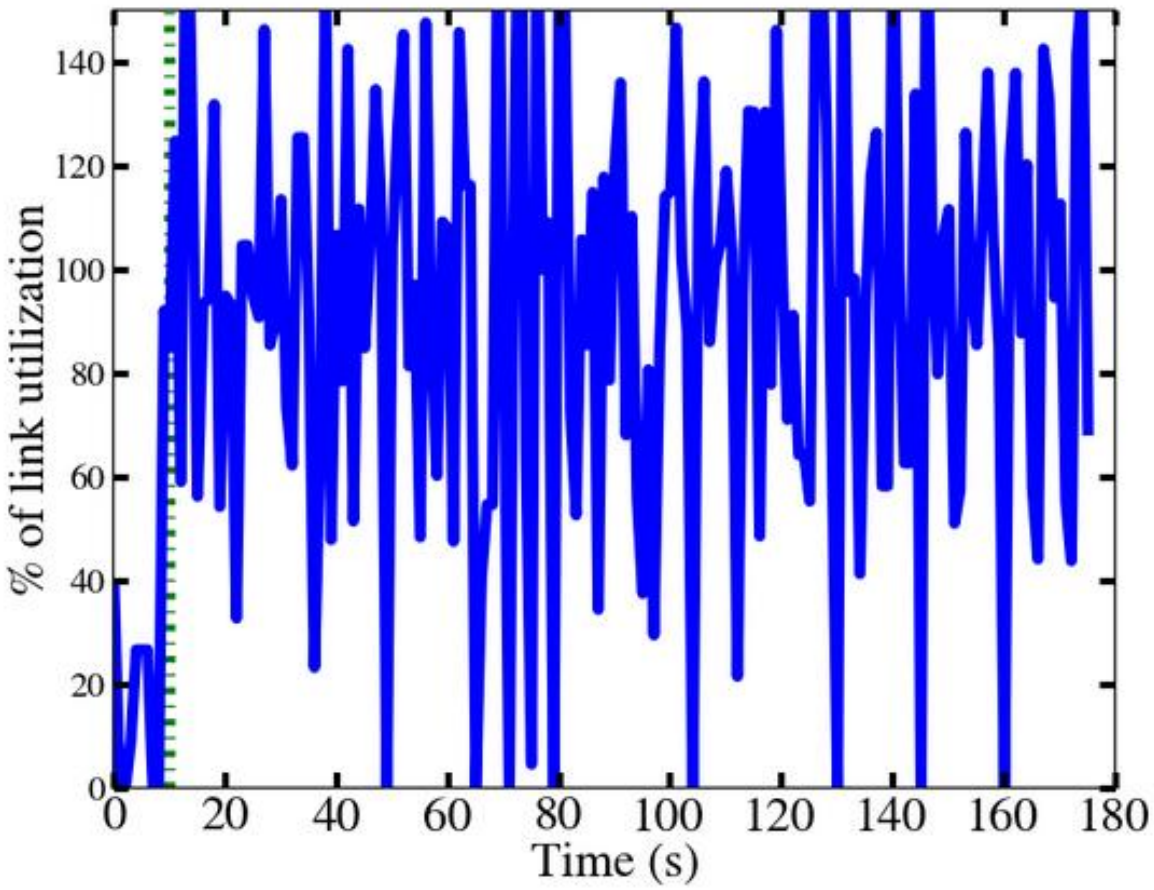

Figure 1c: Utilization plot for alpha = 0.1

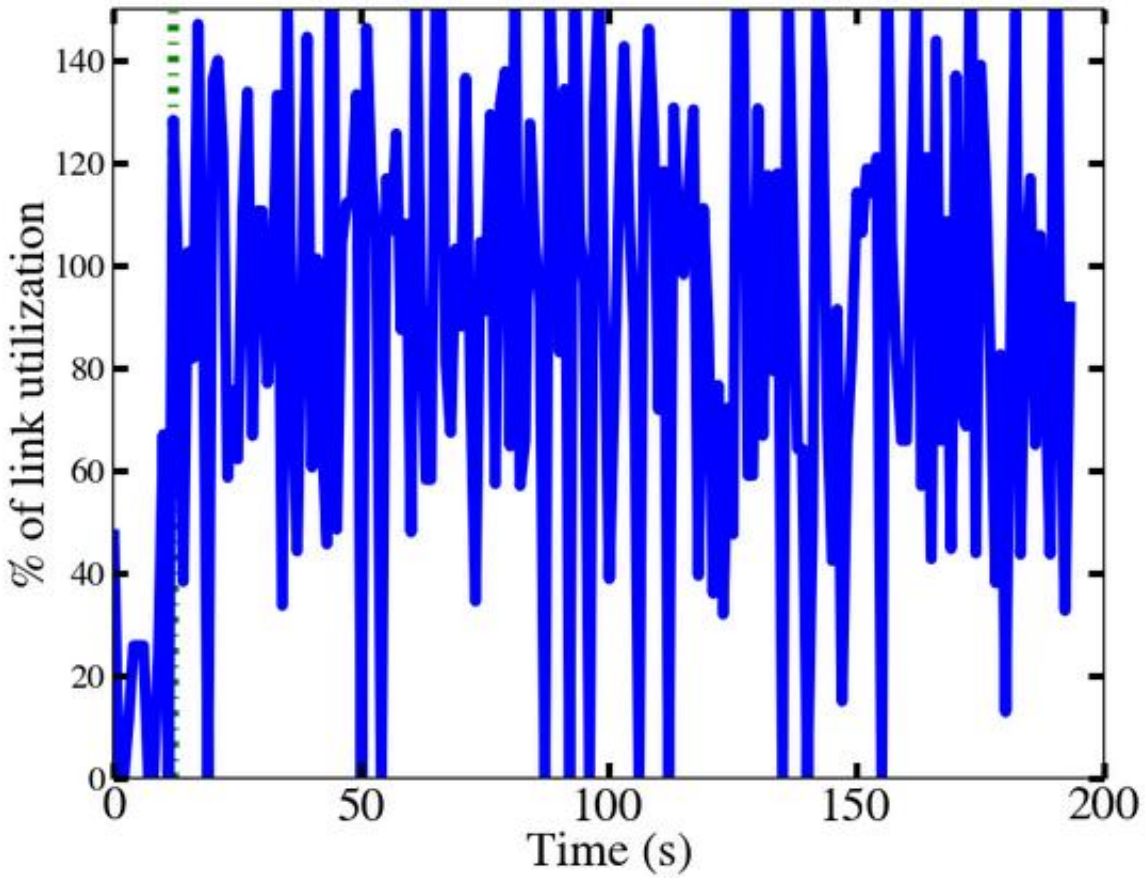

Figure 2c: Utilization plot for alpha = 0.5

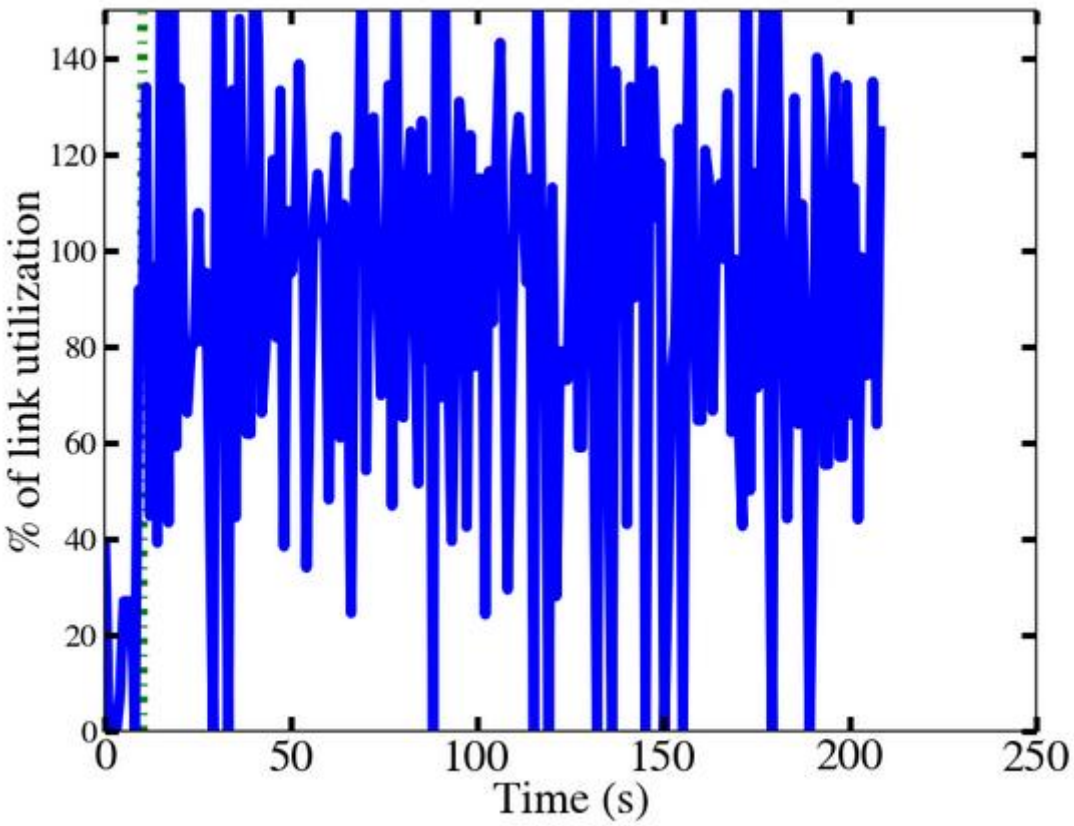

Figure 3c: Utilization plot for alpha = 0.9