ECE466 Computer Networks II
Lab 2b Traffic Shaping (Part 2)
March 3rd, 2014

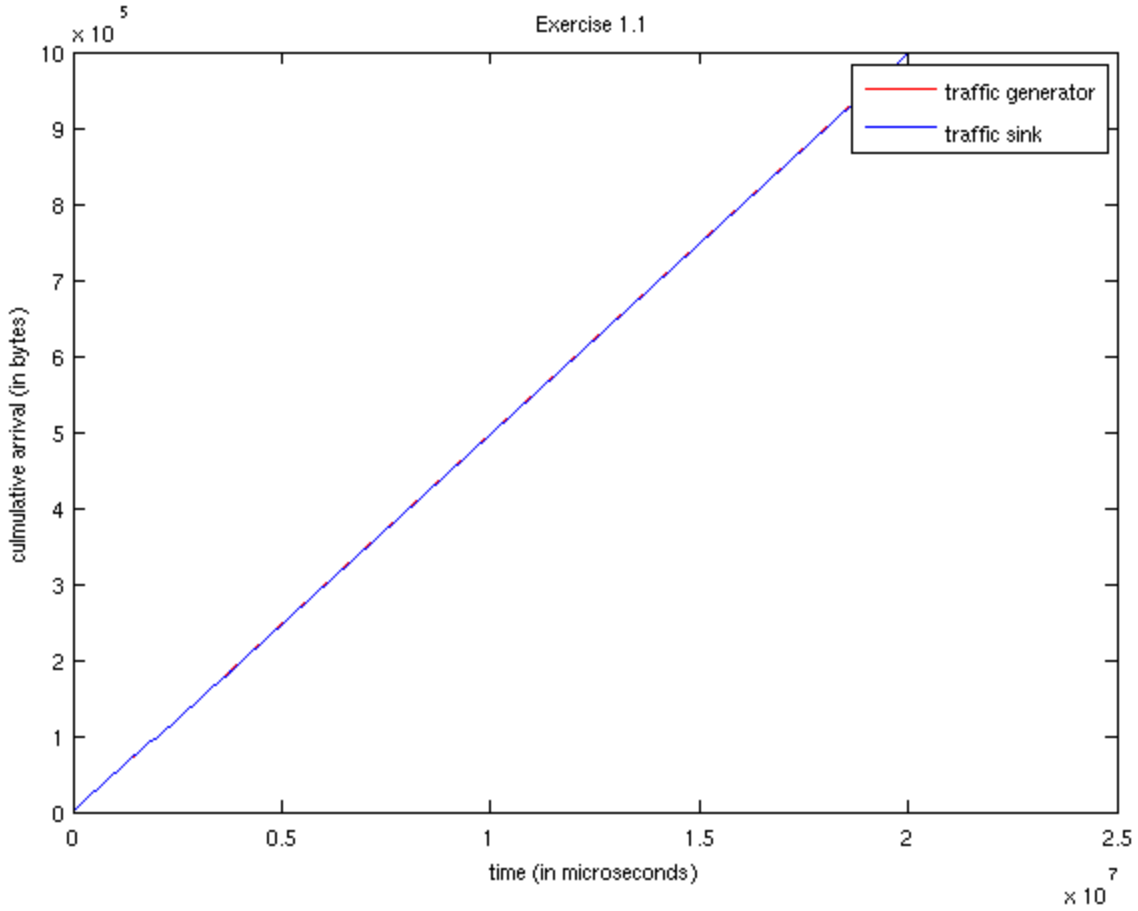
Michael Law
997376343

Part 1. A Reference Traffic Generator

Exercise 1.1 Create and Test Reference Traffic Generator

- Prepare a plot that shows the difference of trace file and the output file. For example, you may create two functions that show the cumulative arrivals of the trace file and the output file, respectively, and plot them as a function of time.

Difference between the trace file (traffic generator) and output file (traffic sink)



The traffic generator in part1/TrafficGeneratorReference.java was provided values of T = 1ms, N = 5 packets, and L = 10 bytes. A sample of 100,000 was plotted, in part1/matlabplot_ex11.m, to compare the difference as we can see contains a very small to no delay.

Part 2. Token Bucket Implementation

Exercise 2.1 Complete the implementation of the Token Bucket

- Describe the design of your implementation for the Token Buffer. Include your source code in the lab report.

  The completed implementation of the Token Bucket can be found in part2/TokenBucket, specifically functionality has been implemented in Bucket.java. The number of tokens is only

calculated when it is needed to avoid continually generating tokens and holding them in the bucket.

Exercise 2.2 Validate your implementation

The following measurement experiments were conducted:

1. $Rate_{Source}$ < rate_TB, N=1, size_TB = 100 Bytes
   $Rate_{Source}$ = NL8/T bps with N=1, L = 100, T = 2ms
   $Rate_{Source}$ = 0.4 Mbps
   rate_TB = 250,000 tokens/sec = 250,000 bytes/s = 2 Mbps

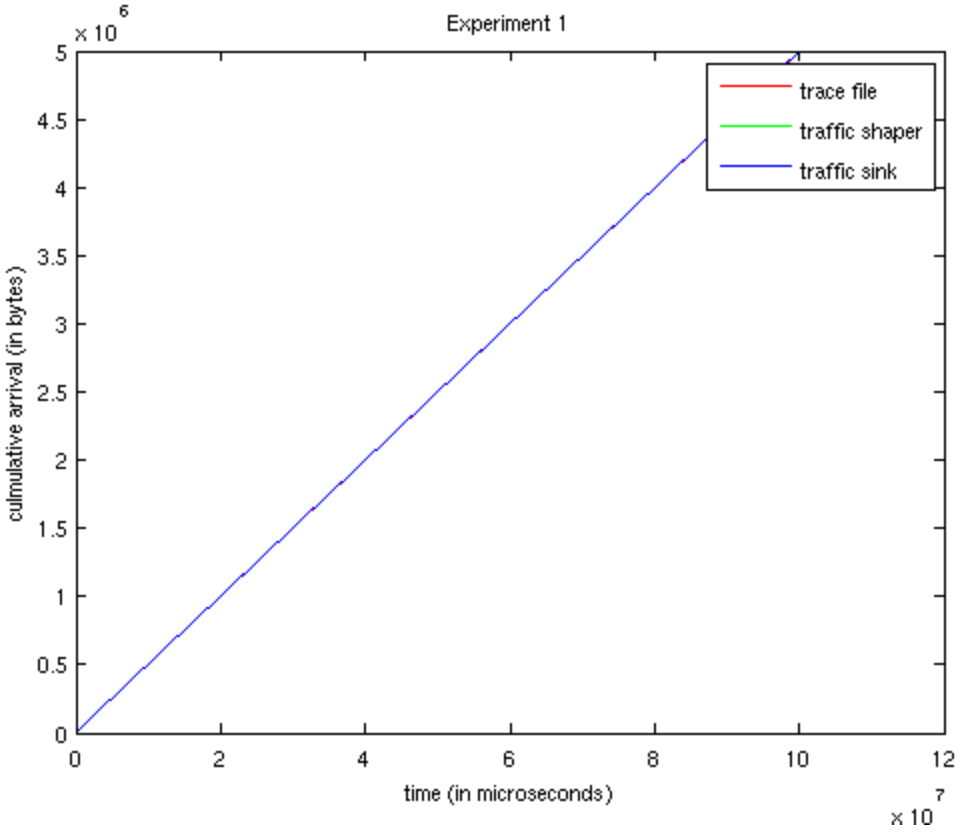
2. $Rate_{Source}$ < rate_TB, N = 10, size_TB = 500 Bytes
   $Rate_{Source}$ = NL8/T bps with N=10, L = 100, T = 10ms
   $Rate_{Source}$ = 0.8 Mbps
   rate_TB = 125,000 tokens/sec = 125,000 bytes/s = 1 Mbps

3. $Rate_{Source}$ ~= rate_TB, N = 1, size_TB = 100 Bytes
   $Rate_{Source}$ = NL8/T bps with N=1, L = 100, T = 1ms
   $Rate_{Source}$ = 0.8 Mbps
   rate_TB = 100,000 tokens/sec = 100,000 bytes/s = 0.8 Mbps

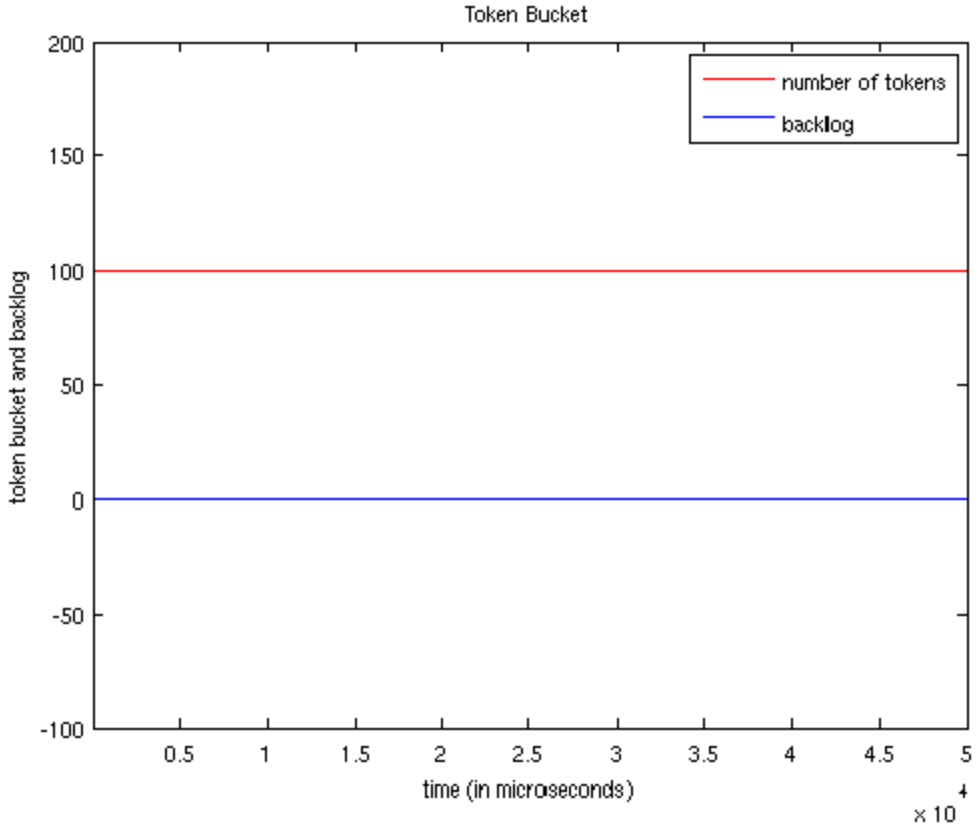
Exercise 2.3 Generate plots for the experiments

For the experiments conducted in Exercise 2.2, the outputs have been plotted in two figures. The first shows the culminative arrivals of all three outputs and the second shows the backlog and tokens in the token bucket as a function of time.

1. $Rate_{Source}$ < rate_TB, N=1, size_TB = 100 Bytes

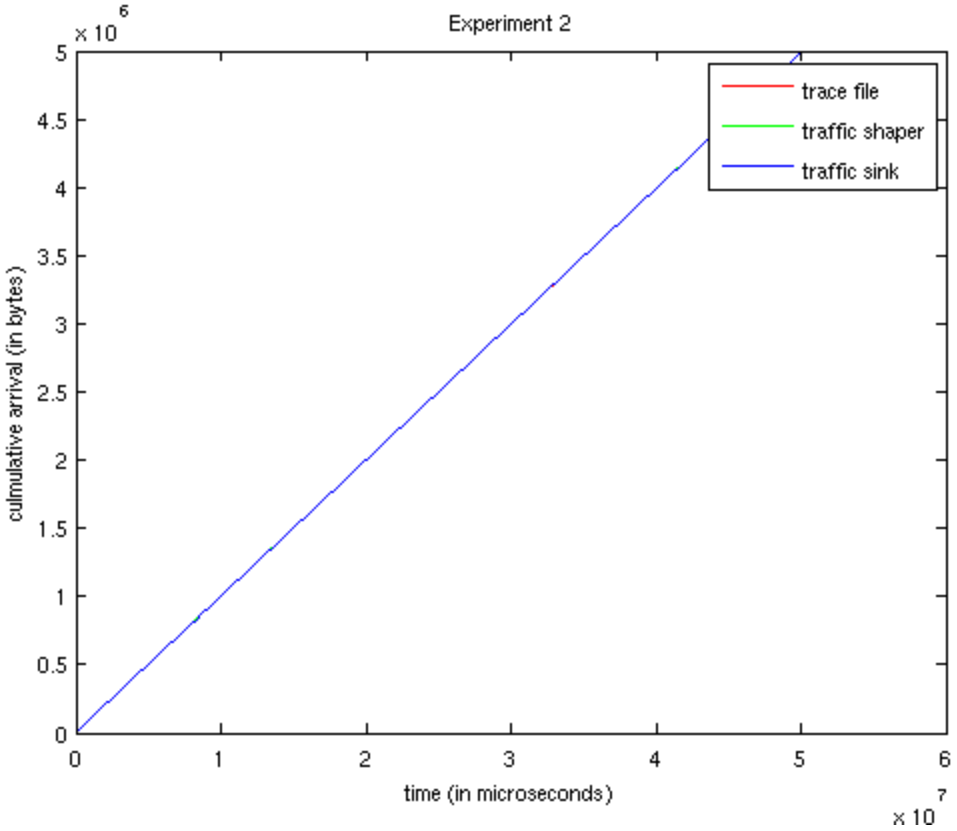


As we can see, all three outputs are on top of each other and have little to no delay.

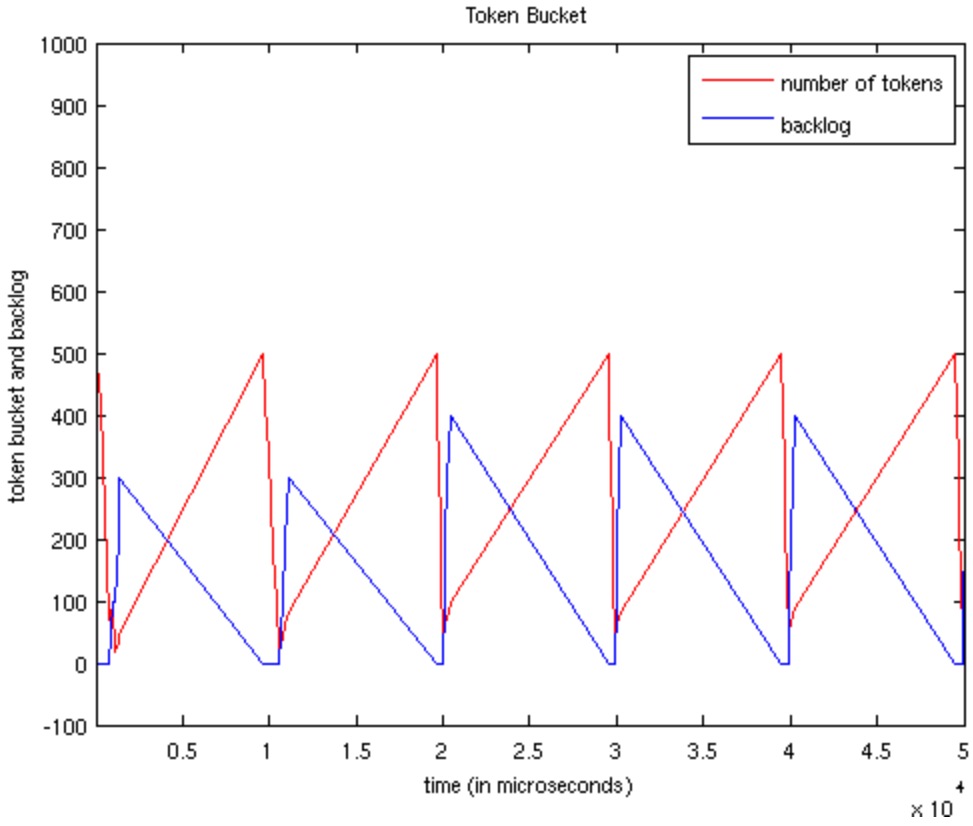


The Token Bucket has always sufficient tokens and there is never a backlog.

2. $Rate_{Source}$ < rate_TB, N = 10, size_TB = 500 Bytes

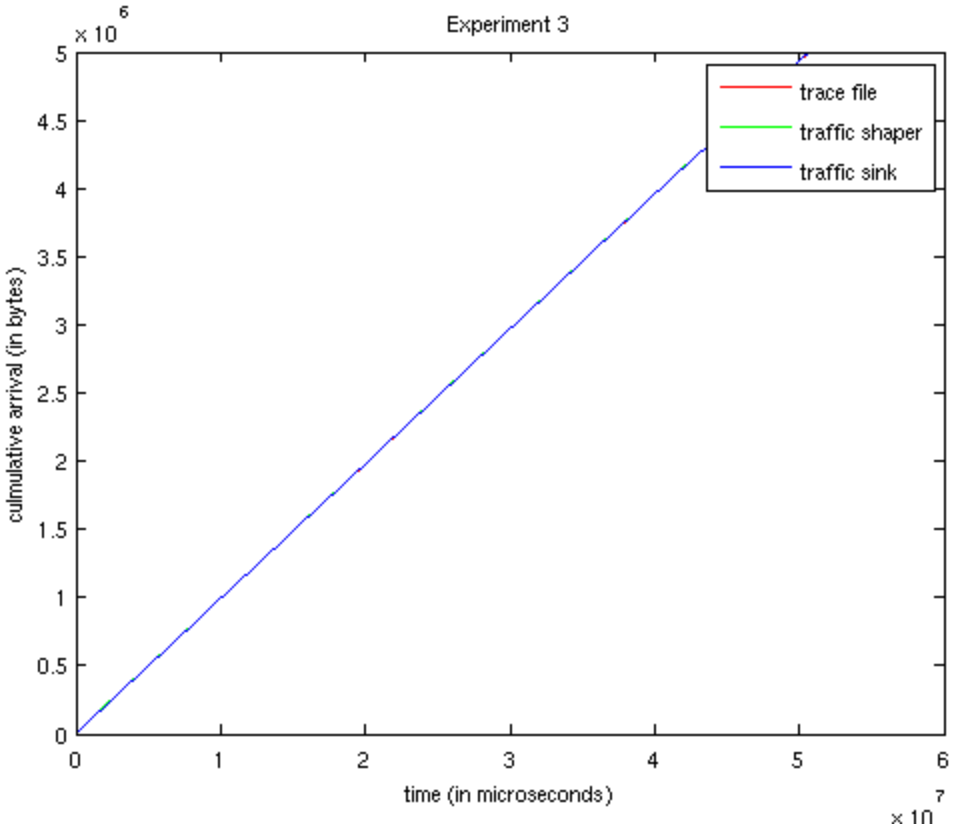


As expected as before, the traffic shaper and traffic sink is able to keep up with the traffic generator since the rate at which we are sending traffic to the shaper is less than the rate at which it can serve the traffic.

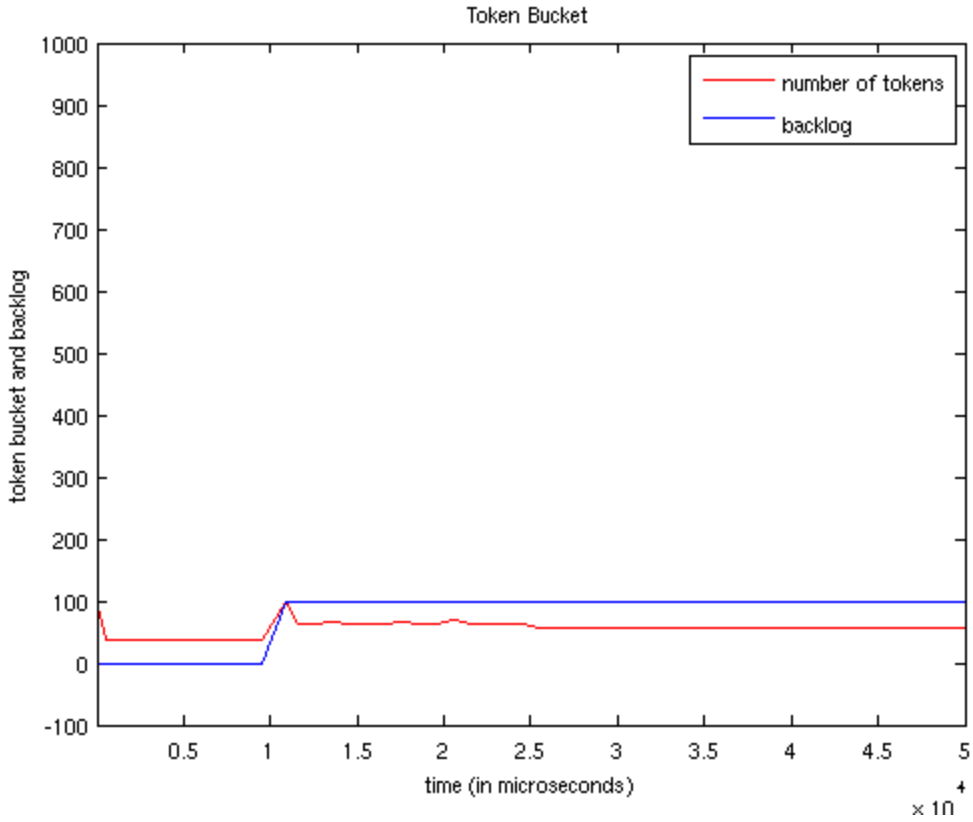


From the output, we can see that the tokens start at the number of tokens we have in the bucket, at 500, quickly reduces to 0 since the buffer is receiving 10 consecutive 100 byte packets. The expected observations show that the rate at which the token bucket allows is greater than the rate of the arrivals as it reaches a maximum backlog and falls back to zero.

3. $Rate_{Source}$ ~= rate_TB, N = 1, size_TB = 100 Bytes



As expected, the three outputs are similar as our rate of arrival is close to our rate of our token bucket.



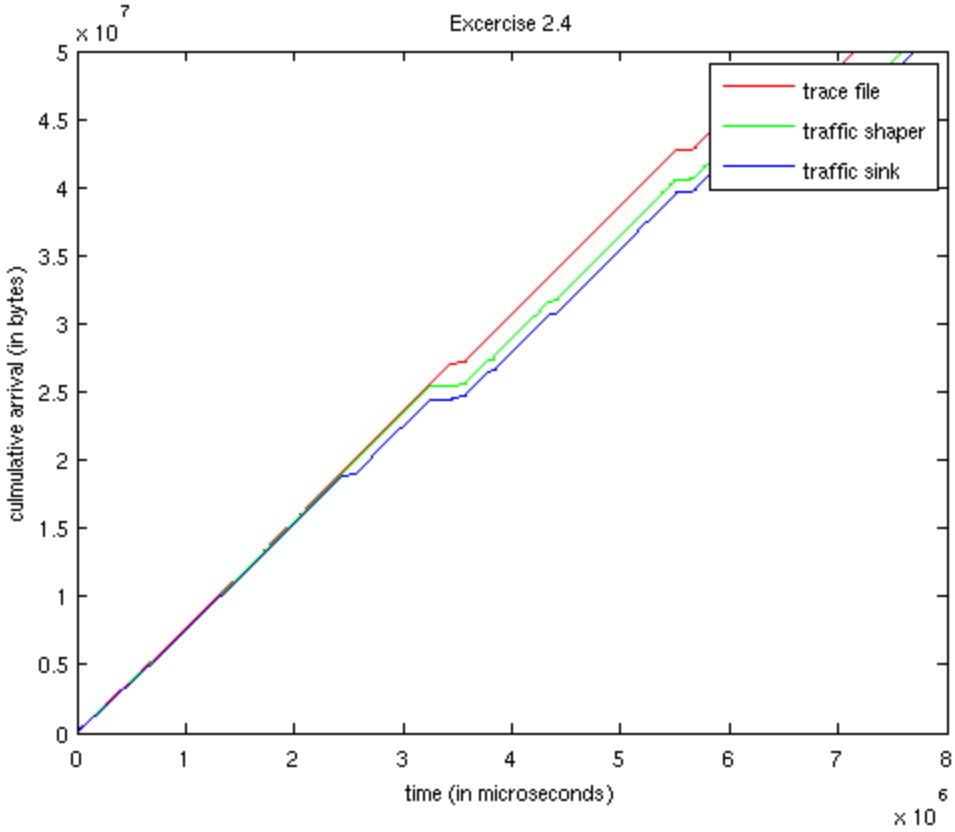
The parallel lines show that the number of tokens and backlog will continue indefinitely even though there is a backlog, it will not build up eventually since the rate of the generator is very close to the rate of the token bucket but not exceeding it.

Exercise 2.4 Maximum rate of Token Bucket

To determine the maximum supported arrival rate, lets first discuss what we represent the maximum supported arrival rate is. The maximum arrival rate that can be supported by the Token Bucket is when we have $Rate_{Source}$ much less than rate_TB, and the traffic sink receives traffic with a delay.

From using binary search with trial and error, a ballpark of the maximum was determined to be around 7.2 Mbps arrival rate. This was produced with N=8 packets, L=900 bytes, T=1ms. The token bucket was given a size_TB of 10000, and a rate of 64 Mbps (essentially rate_TB is much larger than $Rate_{Source}$). The graph below shows the plot of the three outputs.

Maximum rate of arrival supported by the token bucket ~= 7.2 Mbps.



We can see from the graph that the sink continues to have an increased in delay, as time goes on, this means that the delay will not be bounded, thus we have reached a maximum rate of arrival that the token bucket can support.

Part 3. Engineering Token Bucket Parameters

- Discuss your method for selecting the token bucket parameters.

  To select the initial estimate of the token bucket parameters, lets use the maximum packet size of the trace file as the burst and the average arrival rate of the trace files as the rate of the token bucket. This will allow us to determine the actual token bucket parameters by either increasing or decreasing the rate or the size of the token bucket. This will allow us to meet the constraint of having the token bucket's rate at least the average rate of the trace file.

  The following values were determined from the trace files and will be used as the initial values in Exercise 3.1 and 3.2.

  Poisson

  size_TB = 985 bytes
  rate_TB = 126050 bytes/s

  Video

  size_TB = 634344 bytes
  rate_TB = 1.473 Mbytes/s

Ethernet

size_TB = 1518 bytes

rate_TB = 175060 bytes/s

Exercise 3.1 (Long-term) Bandwidth is cheap

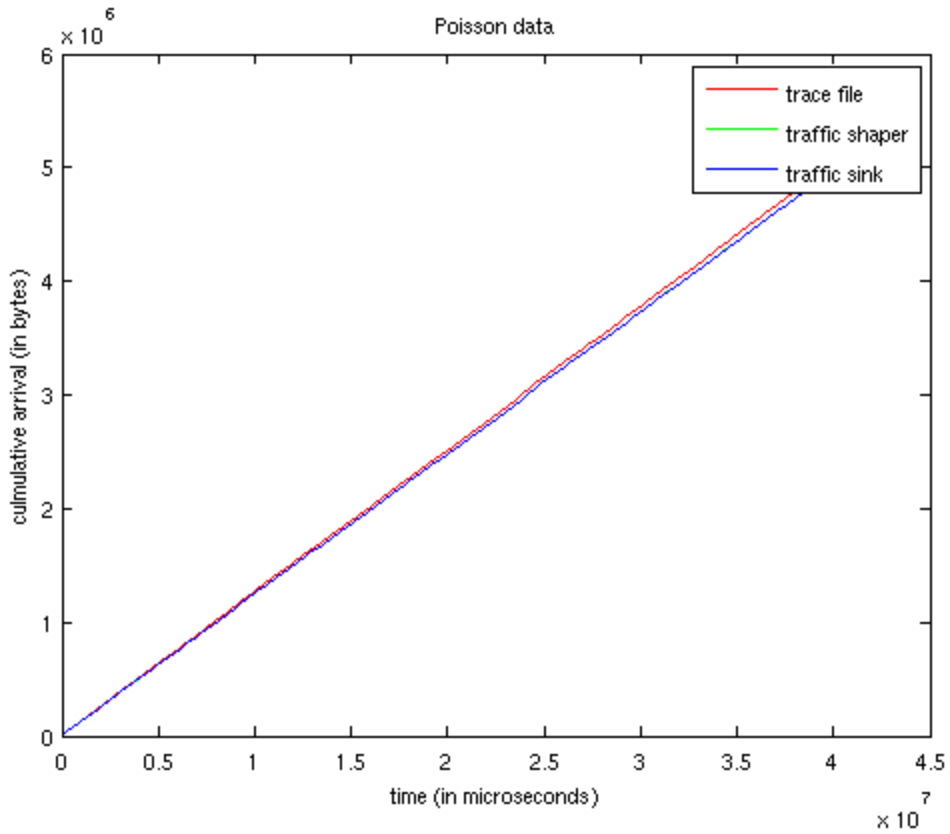
Increase rate_TB until the token bucket satisfies the constraints for the traffic source.

Poisson

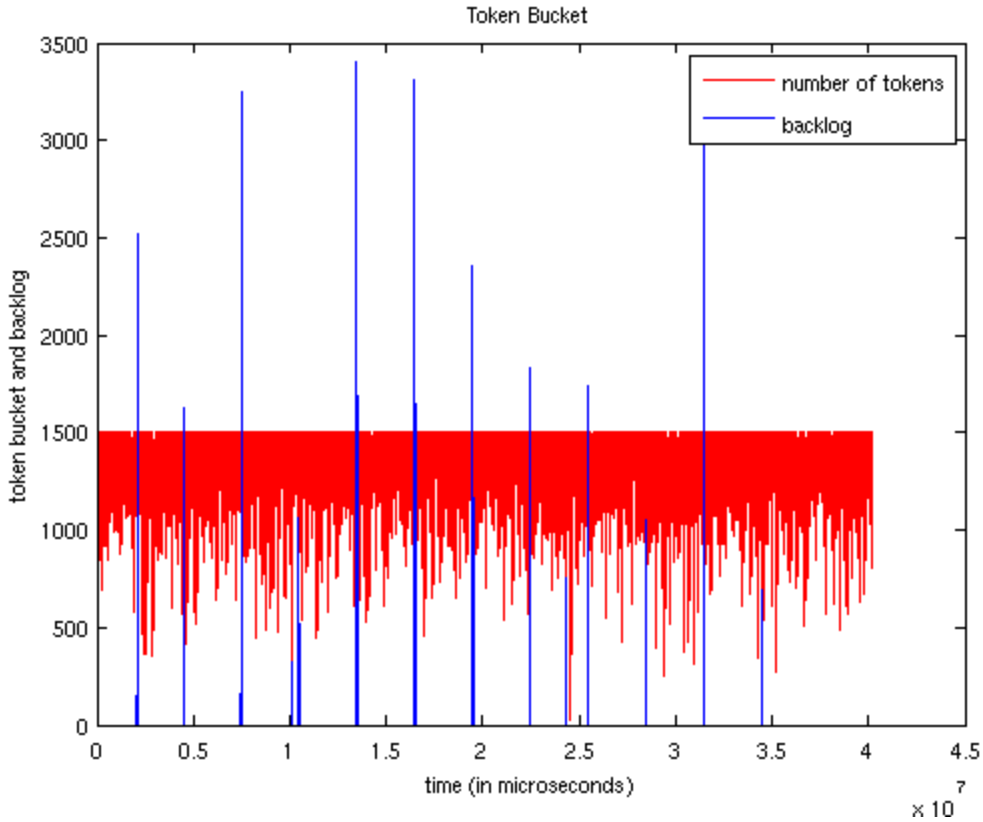
size_TB = 985 bytes → 1500 bytes

rate_TB = 126050 bytes/s → 300000 bytes/s

The token bucket was having troubles maintaining the bucket given a size_TB of 985 bytes and was increase to 1500 bytes to allow the experiment to continue. Then of which the rate_TB was increased to 300000 bytes/s.



The output of the files are as expected to be overlapping each other.



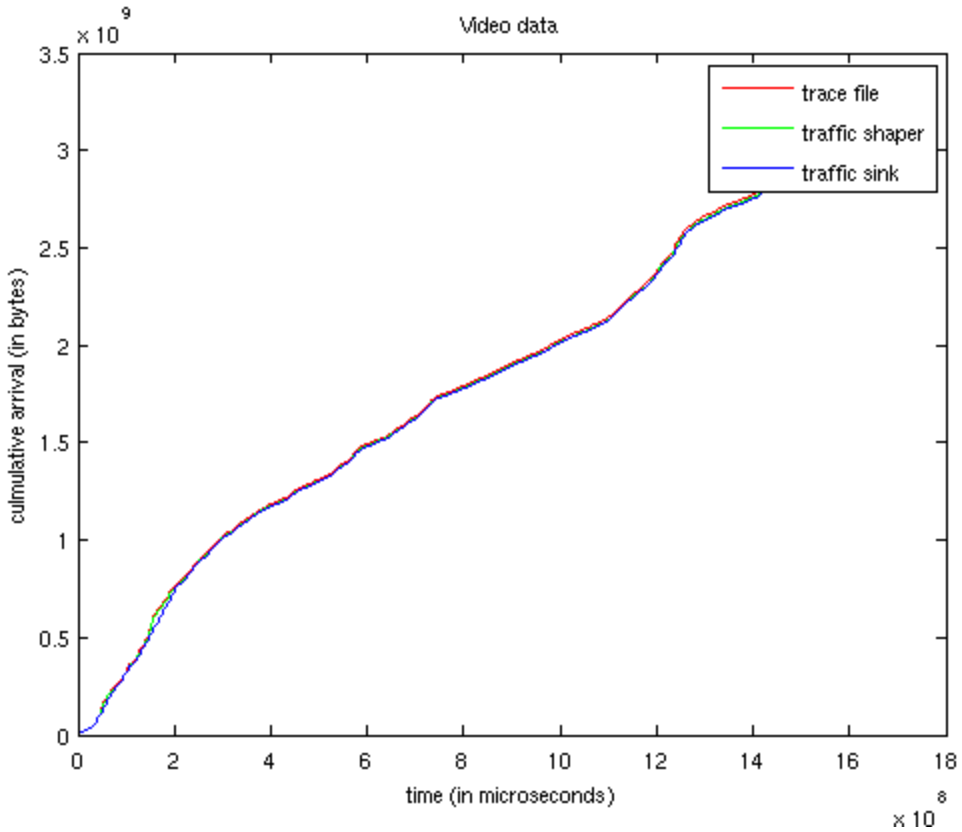
The pattern of the poisson data conforms with the graph of the tokens and backlog.

Video

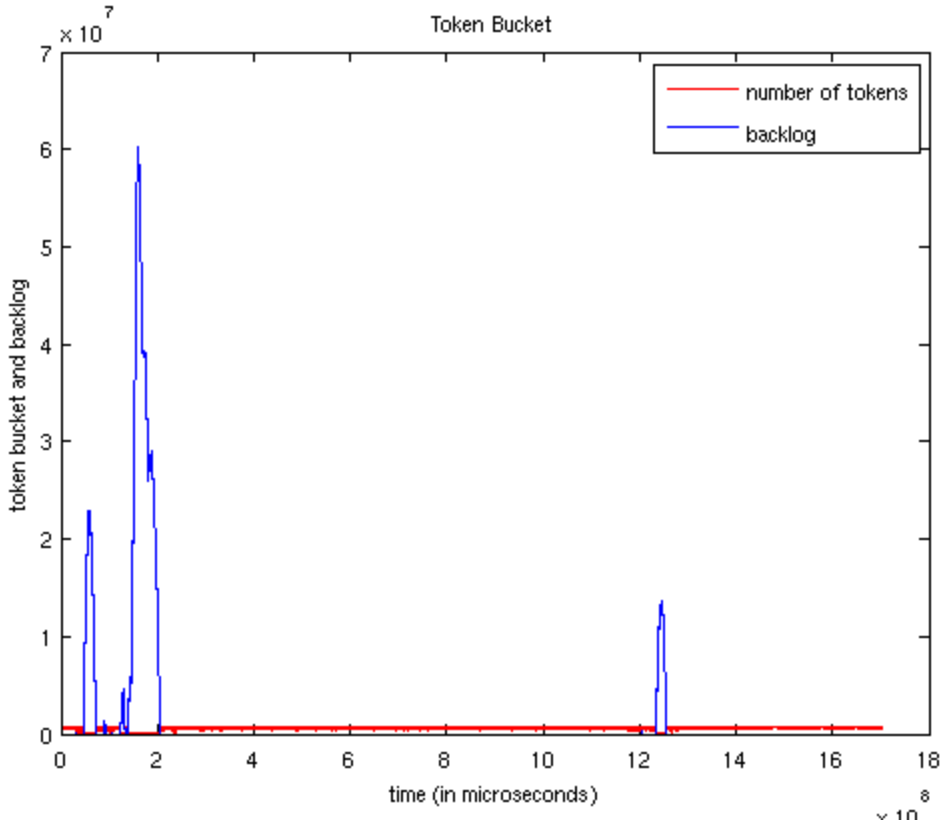
size_TB = 634344 bytes

rate_TB = 1.473 Mbytes/s → 4.473 Mbytes/s

Increasing rate_TB at 1MBps until we no longer get packet drops resulted in the final rate_TB of 4.473 MBps. This is less than the peak rate of 19 MBps which satisfies the constraint.



We can see a corresponding delay at time unit of 2 on the time axis in the above graph with the huge backlog in the graph below. Other than that, the three outputs look very similar.



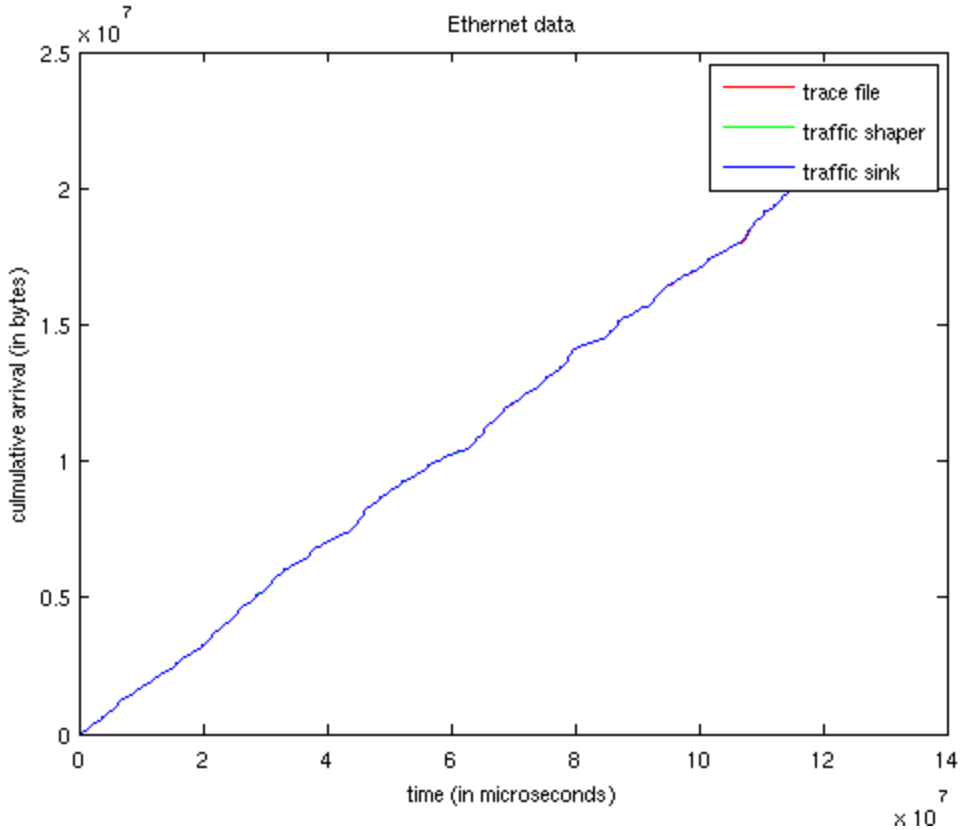
We can see that with the token bucket receives a lot of data and then the rate of the token bucket will bring the backlog back to zero.

Ethernet

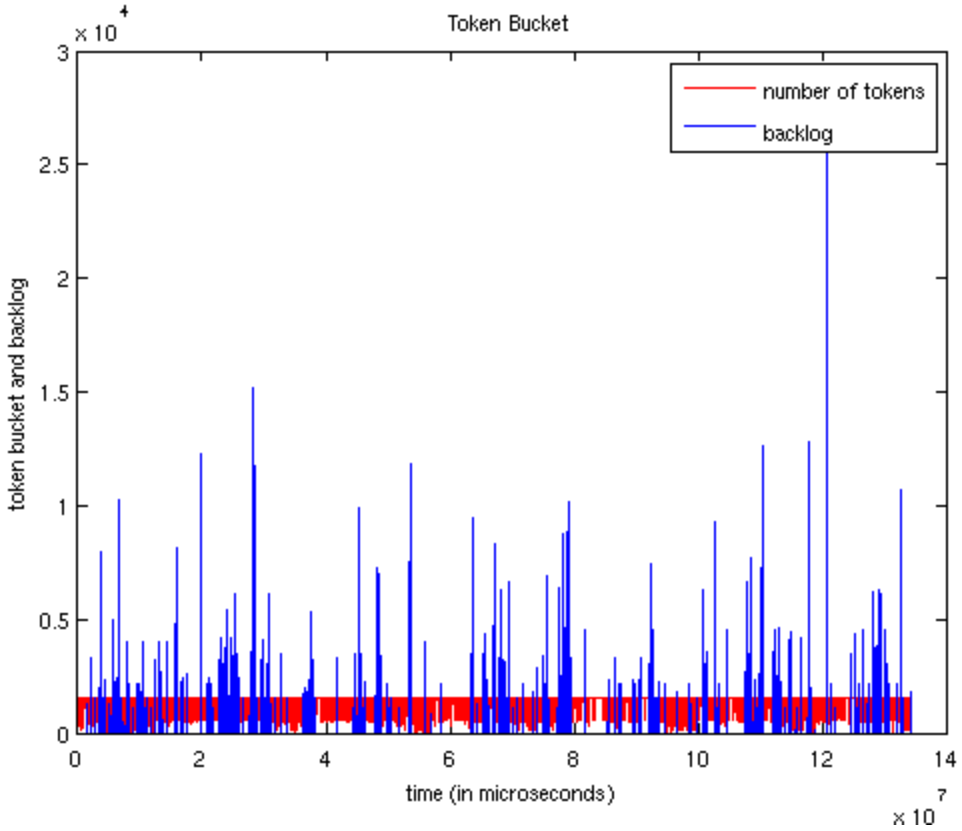
size_TB = 1518 bytes

rate_TB = 175060 bytes/s → 1MB/s

The rate of the increased to 1MB/s to conform to the arrival of the ethernet data.



With the provided token bucket rate, the outputs were able to have very little delay.



We can observe that the rate of the token bucket quickly brings the backlog back down as fast as it accumulates and we have a bounded graph.

Exercise 3.2 (Long-term) Bandwidth is expensive

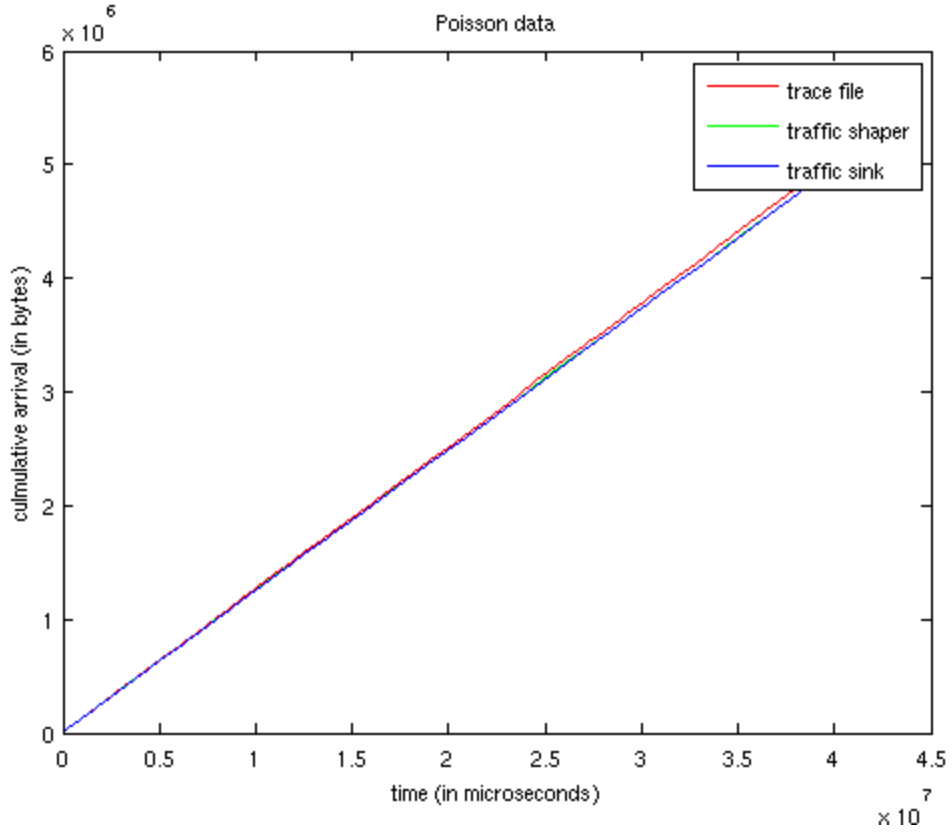
Increase size_TB until the token bucket satisfies the constraints for the traffic source.

Poisson

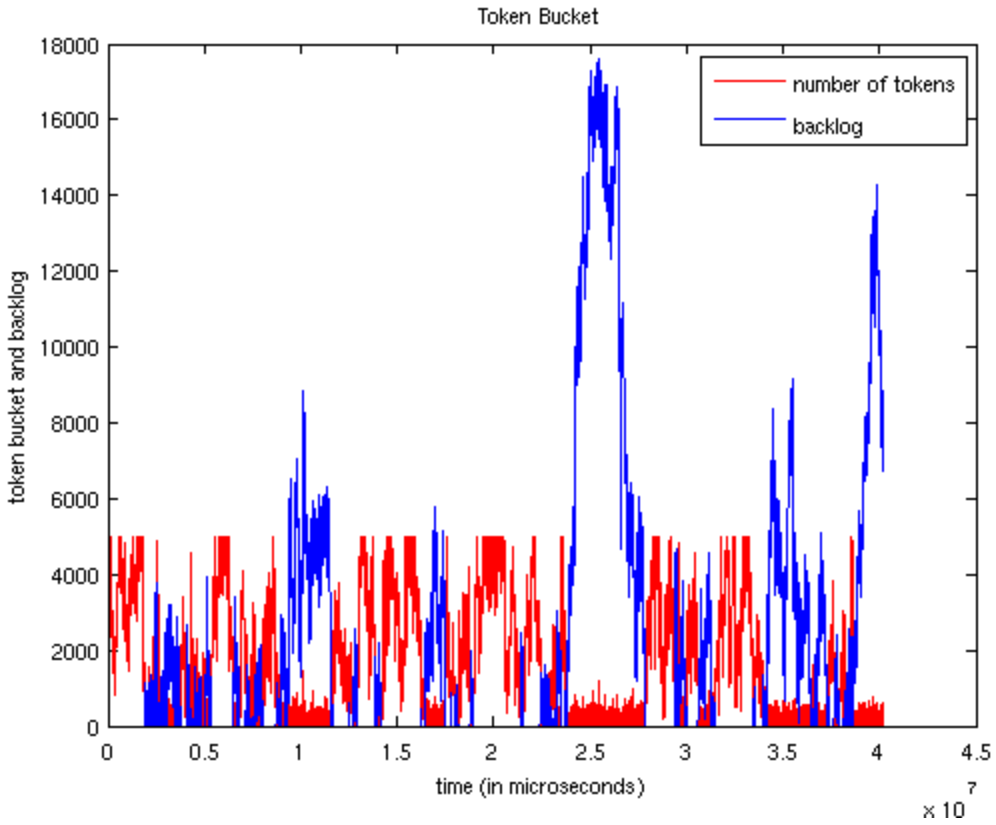
size_TB = 985 bytes → 5000 bytes

rate_TB = 126050 bytes/s

size_TB was increased to 5000 to reduce the delay below 100ms.



As expected, the outputs are looking good.



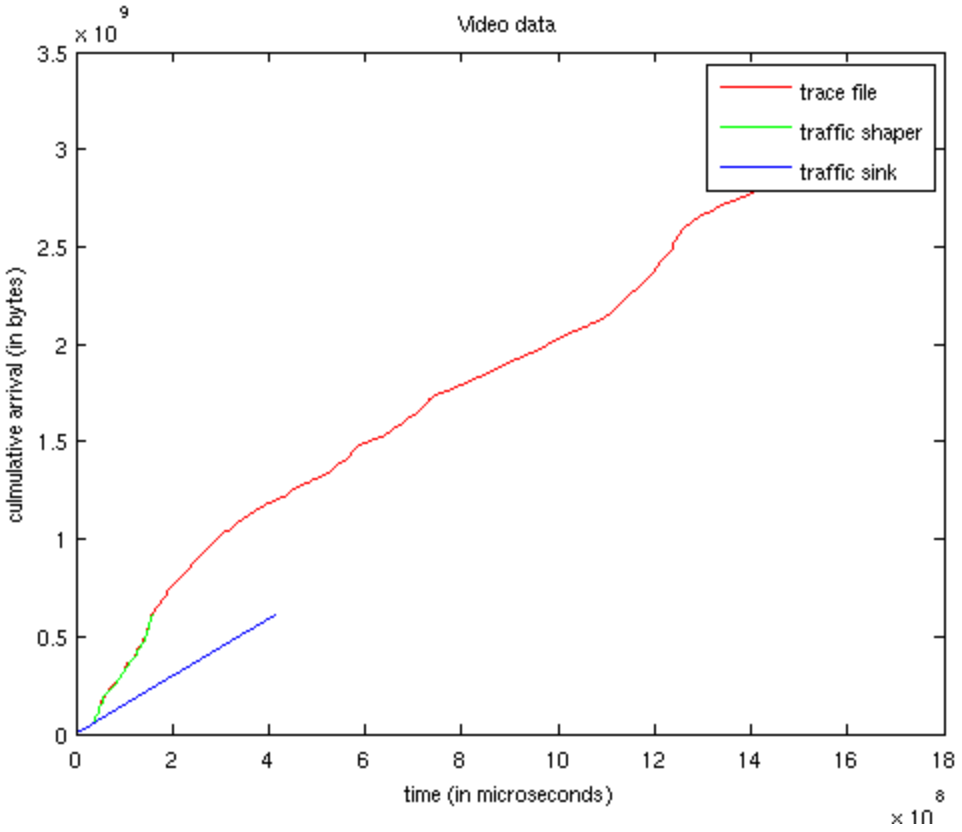
With the high burst, the backlog fills up eventually

Video

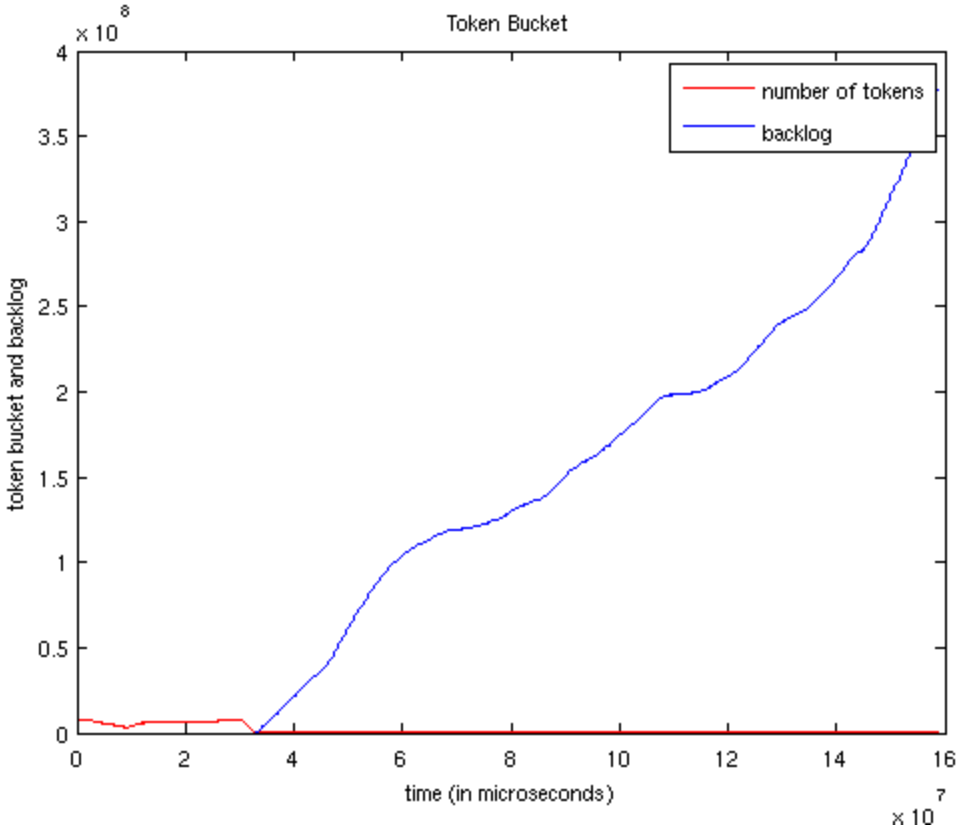
size_TB = 634344 bytes → 7543400 bytes

rate_TB = 1.473 Mbytes/s

Increasing size_TB at 1,000,000 bytes did not result in no packet drops which means our final size_TB of 7,543,440 bytes is not bounded. Our implementation could not support the video traffic with a constant token bucket rate of 1.4MB/s with simply increasing the size of the token bucket. This may be due to the high variance of the traffic arrival rate with the peak rate being very large at 19 MB/s



We can see here that the output of the traffic sink fails at around one quarter of the entire duration of the run.

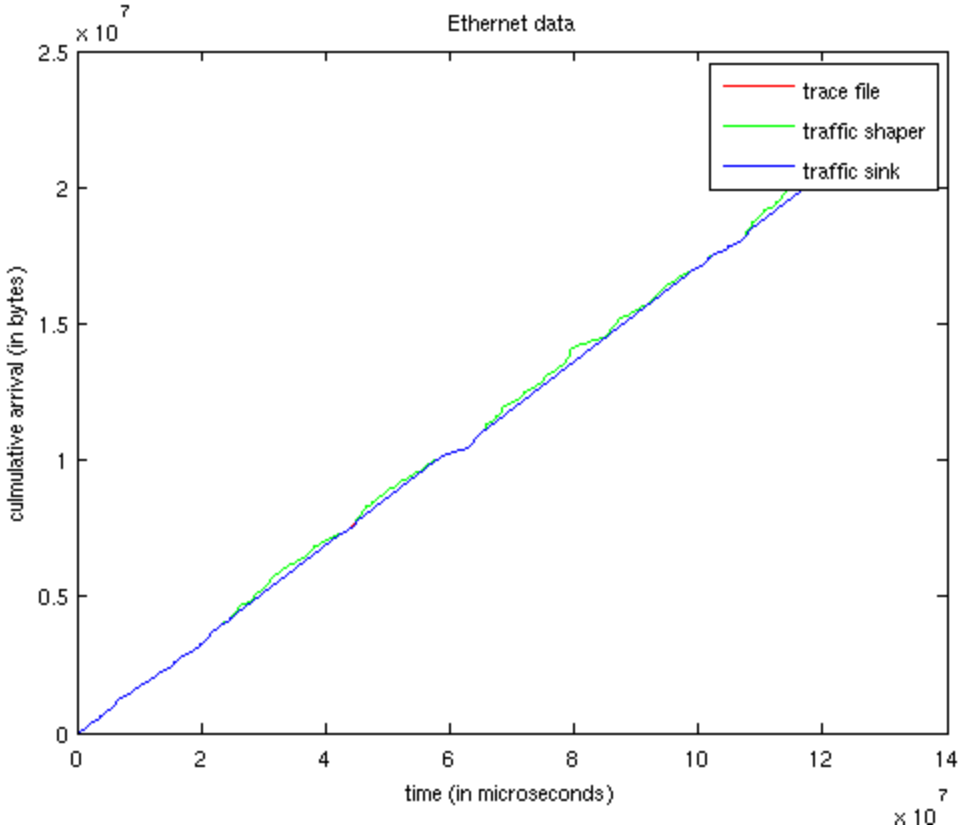


The graph shows that it is not possible to increase the size_TB to have a bounded backlog.

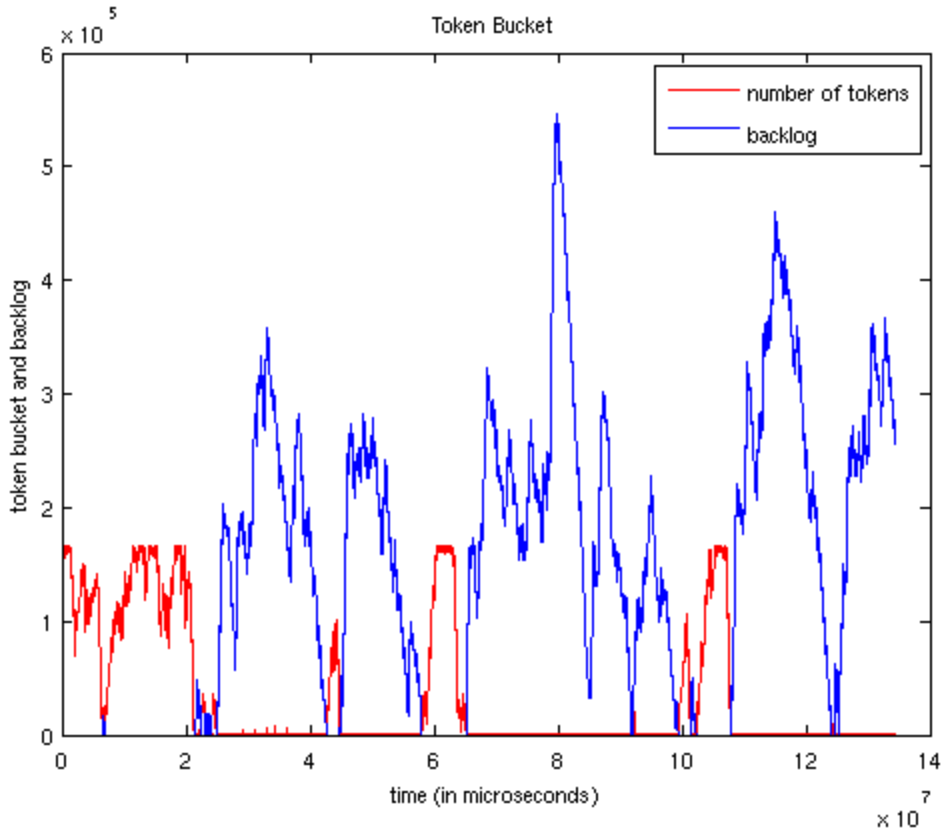
Ethernet

size_TB = 1518 bytes → 170000 bytes

rate_TB = 175060 bytes/s



By increasing size_TB to 170000 bytes, we were able to see the culminative arrivals very close to each other, having a small delay.



The backlog and token bucket looks to be bounded with the given high burst throughout all the arrivals of the ethernet data.

- Compare the results for the cheap bandwidth and expensive bandwidth scenarios.

  It was much easier to increase the rate of the token bucket than increase the size of the token bucket to conform to the incoming traffic. As we saw, with the video data, the traffic had such high variance that even with the average rate of the traffic as the rate of the token bucket and increasing the burst was not enough to lower the delay between the output and the token shaper.

- Compare the results for the three traffic types. For example, which type of traffic is most demanding in terms of bandwidth or memory requirements? Which type of traffic benefits the most from traffic shaping?

  The most demanding traffic in terms of memory requirements is the movie trace since it will have sudden large increases of arrivals and will need a large buffer to store the data. Ethernet is the most demanding in terms of bandwidth since it will contain a consistent stream of packets that is large and small and will require a high rate to serve it. Since we were able to build a shaper for the poisson and ethernet data using the shaper consisting of an envelope that closely resembles the traffic these are both types of traffic that will benefit from the traffic shaper.