ECE466 Computer Networks II
Lab 2a Traffic Shaping (Part 1)
February 10th, 2014

Michael Law
997376343

Part 1. Programming with Datagram Sockets and with Files

Exercise 1.1 Programming with datagram sockets

- Repeat this exercise, with the difference, that you run the sender and receiver on two different hosts.

  From this exercise, host of the machines have been determined to be 128.100.13.<ug machine>. For example, the hostname of the receiver on ug166 is 128.100.13.166

Exercise 1.2 Reading and Writing data from a file

- Modify the program so that it computes and displays the average size of the following frame types:
  - I frames: 183776
  - P frames: 111412
  - B frames: 36093

Part 2. Traffic generators

Exercise 2.1 Traffic Generator for Poisson traffic

The traffic generator will send to port 4444 and can be found in part2/TrafficGenerator.java
usage: java TrafficGenerator <hostname>

Exercise 2.2 Build the Traffic Sink

The traffic sink can be found in part2/TrafficSink.java
usage: java TrafficSink <port>

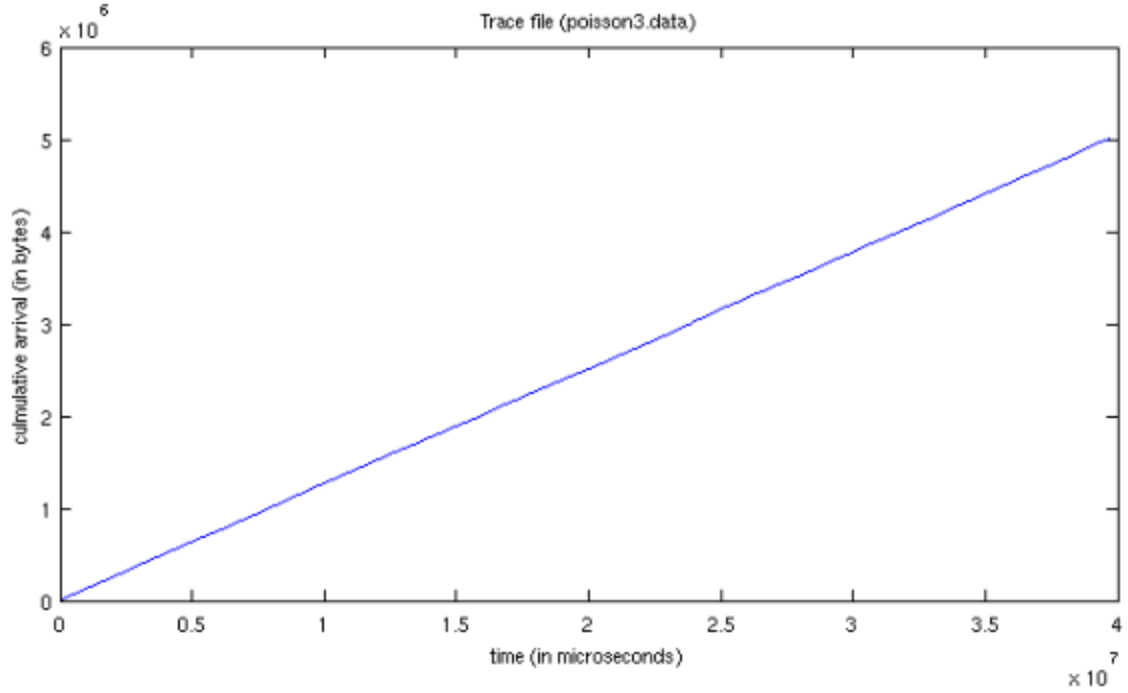
- Test the traffic sink with the traffic generator from Exercise 2.1.

  The output file can be found in part2/TrafficSinkOutput.txt and has the format:
  SeqNo Size (in bytes) arrival time since last packet (in microsec)

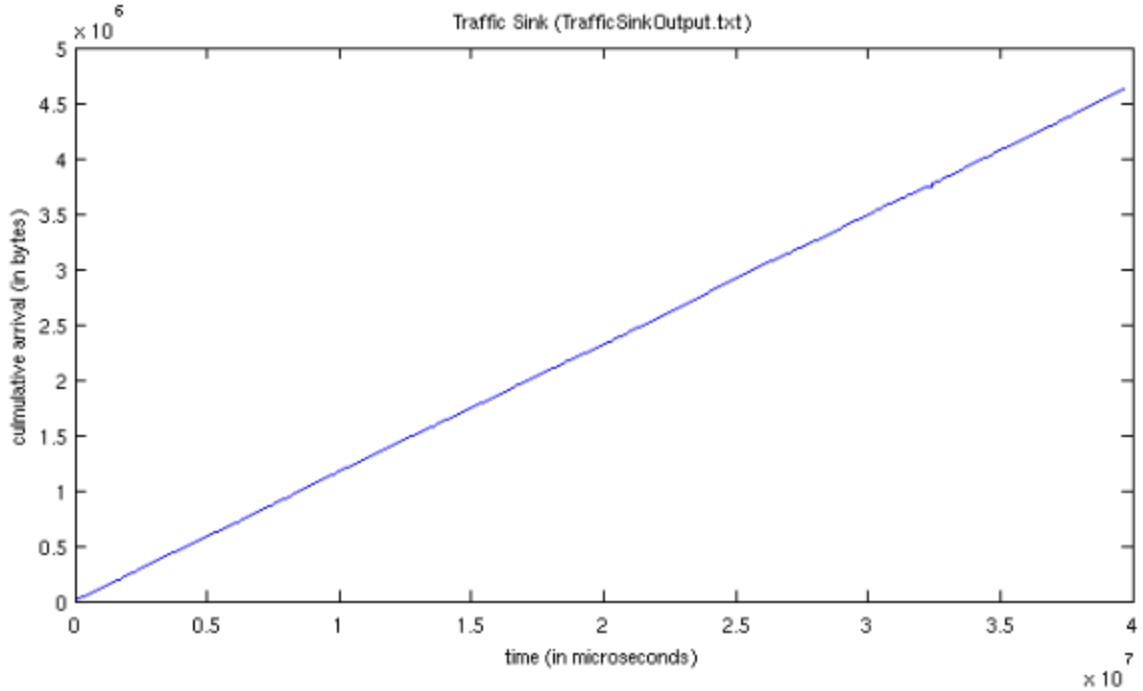
Exercise 2.3 Evaluation

- 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.

Trace file (poisson3.data)



TrafficSink (TrafficSinkOutput.txt)

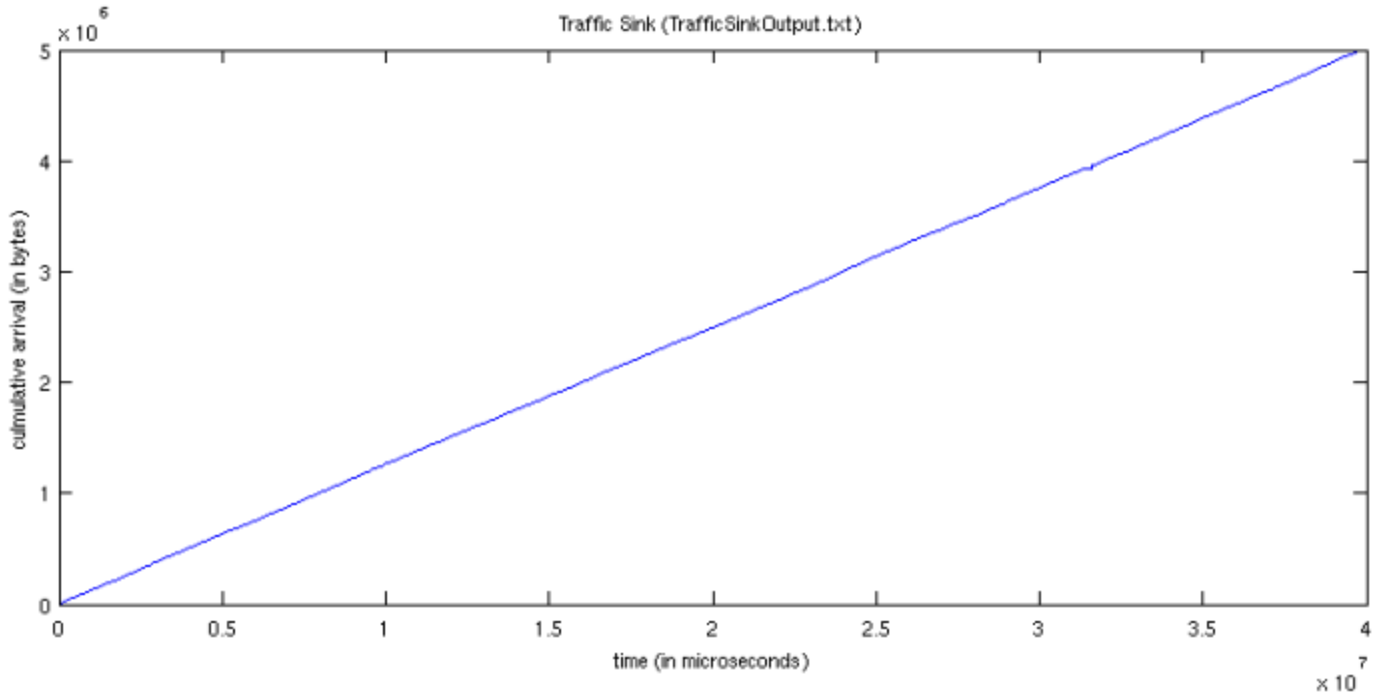


The matlab code for the plots can be found in part2/matlabplot.m. Using 50000 data points, we can see that the Traffic Sink at the 50000'th point has less data cumulated than the poisson3 data. Poisson3.data shows a cumulative arrival of around 5 MB, while the traffic sink shows to have only captured pass 4.5 MB of data. The curves, however, looks to be very similar.

- Try to improve the accuracy of the traffic generator. Evaluate and graph your improvements by comparing them to the initial plot.

  The TrafficSink originally only allocated a 256 bytes array to store the packet coming in, by increasing this to 2048, we can see the improved accuracy of the traffic generator. 2048 was arbitrary chosen, as long as it is bigger than the largest packet in possion3.data of 1469 bytes.

  Improvements on TrafficSink (TrafficSinkOutput.txt)



  We can now see that improved traffic sink collecting around 5MB of data.

Exercise 2.4 Account for packet losses.

The number of packets lost when running the TrafficGenerator and TrafficSink over different machines is about 41. This number was determined by looking at the sequence numbers that were written to the output file. Running both the sender and receiver on the same machine shows that it had no packet losses.

Part 3. Token Bucket Traffic Shaper

Exercise 3.1 Running the reference implementation of the Token Bucket

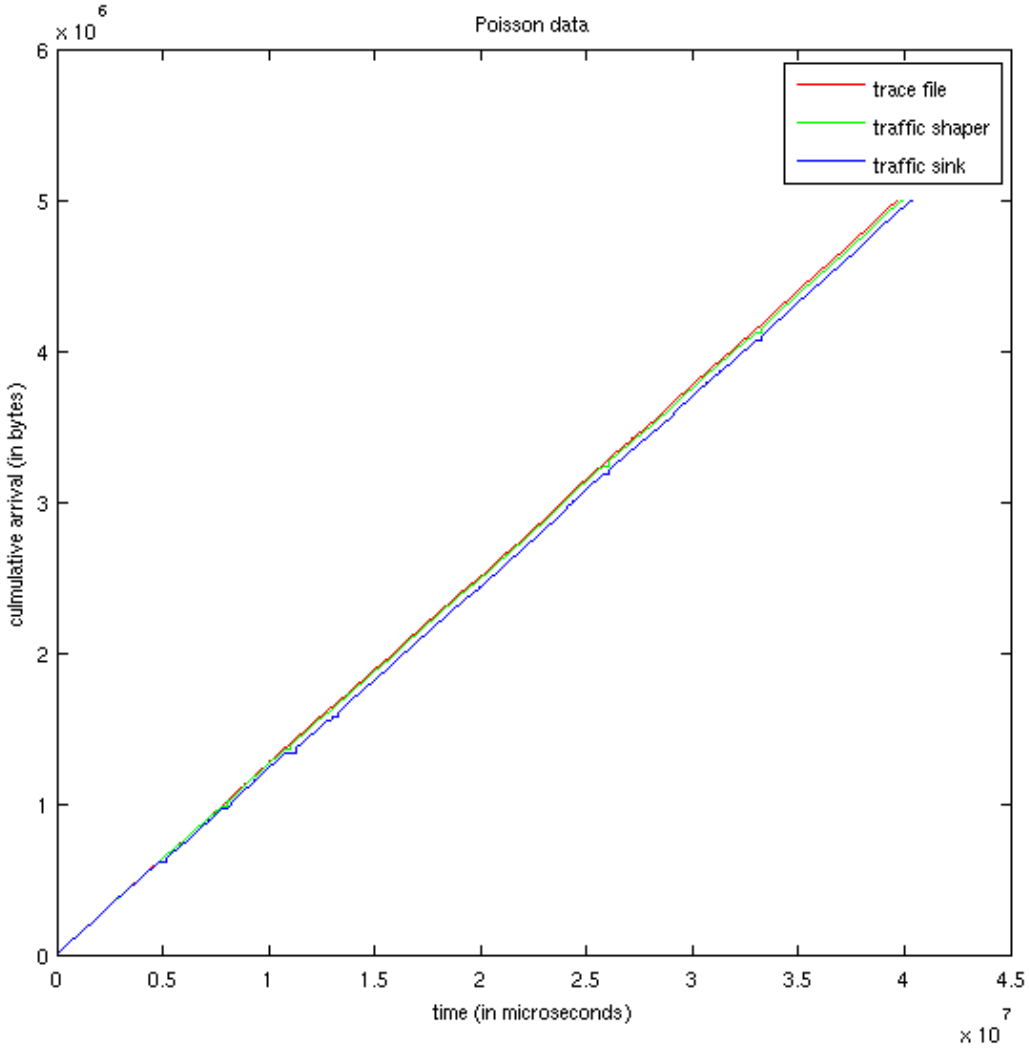
- Provide the source code for your source code for Exercise 3.1. Do not include the sources for the traffic generator and traffic sink. Also, do not include source files of the reference implementation.

  The source code can be found in part3/TrafficShaper.java

Exercise 3.2 Evaluate the reference implementation for the Poisson traffic file

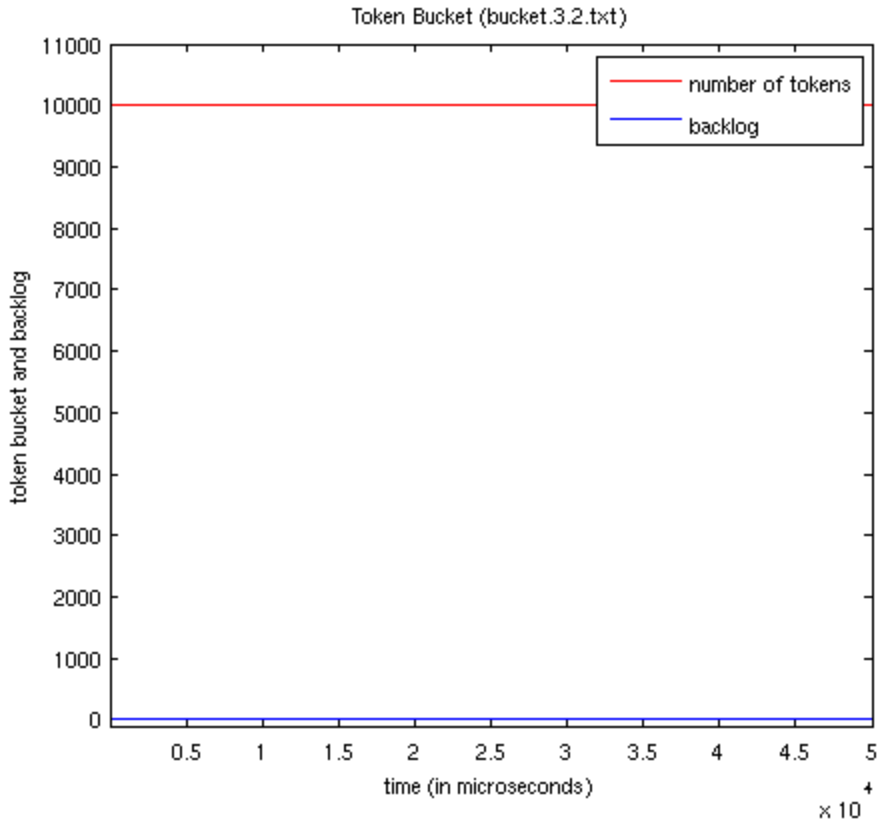
Prepare a single plot that shows the cumulative arrival function as a function of time of:

- The data of the trace file (as read by the traffic generator)
- The arrivals at the token bucket
- The arrivals at the traffic sink



The code for the plots can be found in part3/matlabplot_ex32.m. We can see that for all three plots, we have almost zero packet lost and some propagation delay. The plot can also be found in part3/ex3.2_plot_single.png.

- Provide a second plot that shows the content of the token bucket and the backlog in the Buffer as a function of time



Since we are using the reference implementation, the backlog never builds up and the tokens are never used.

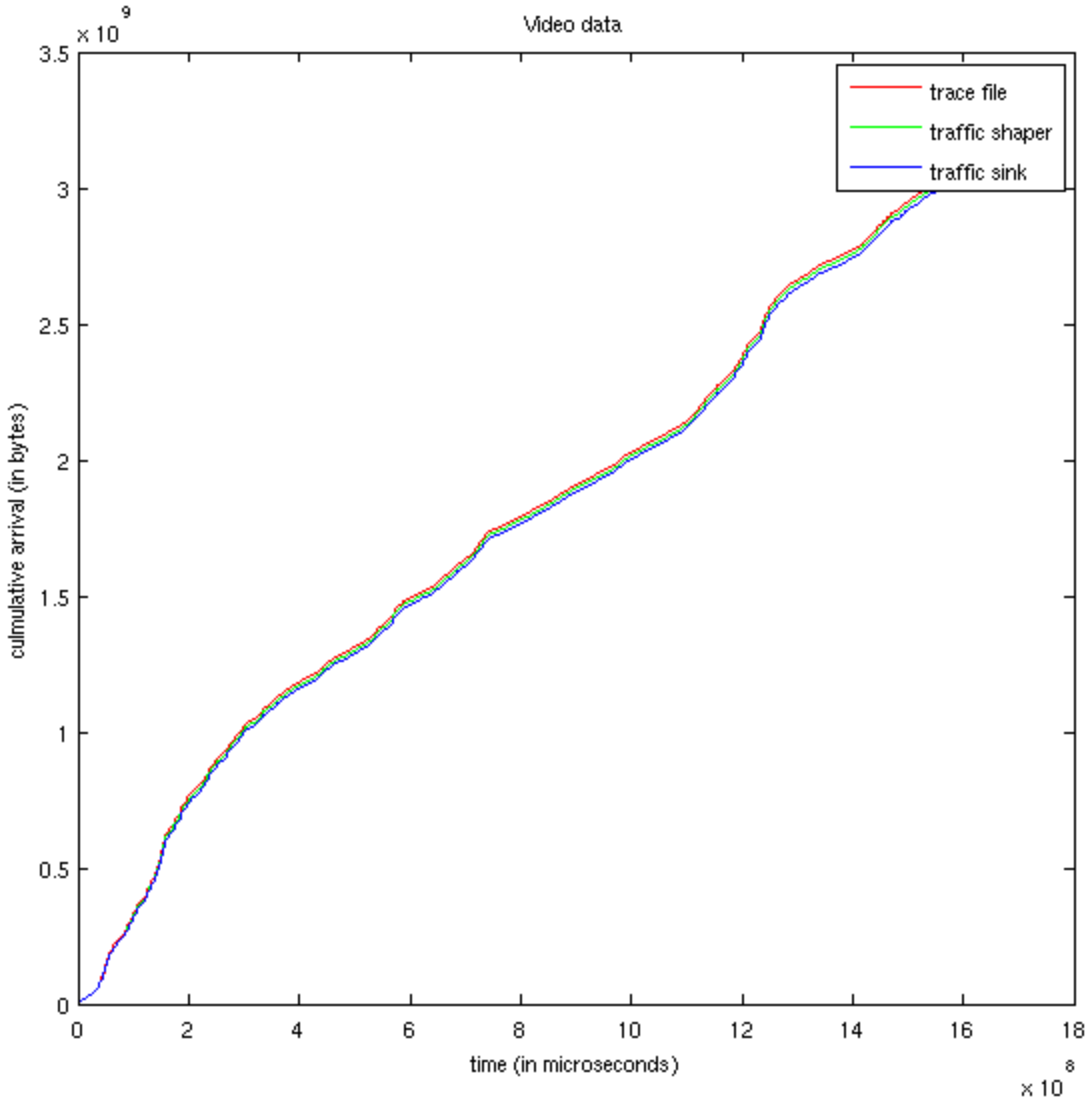
Exercise 3.3 Evaluate the reference implementation for the Ethernet and Video Trace files

Video Trace file (movietrace.data)

The traffic generator can be found in part3/TrafficGeneratorVideo.java

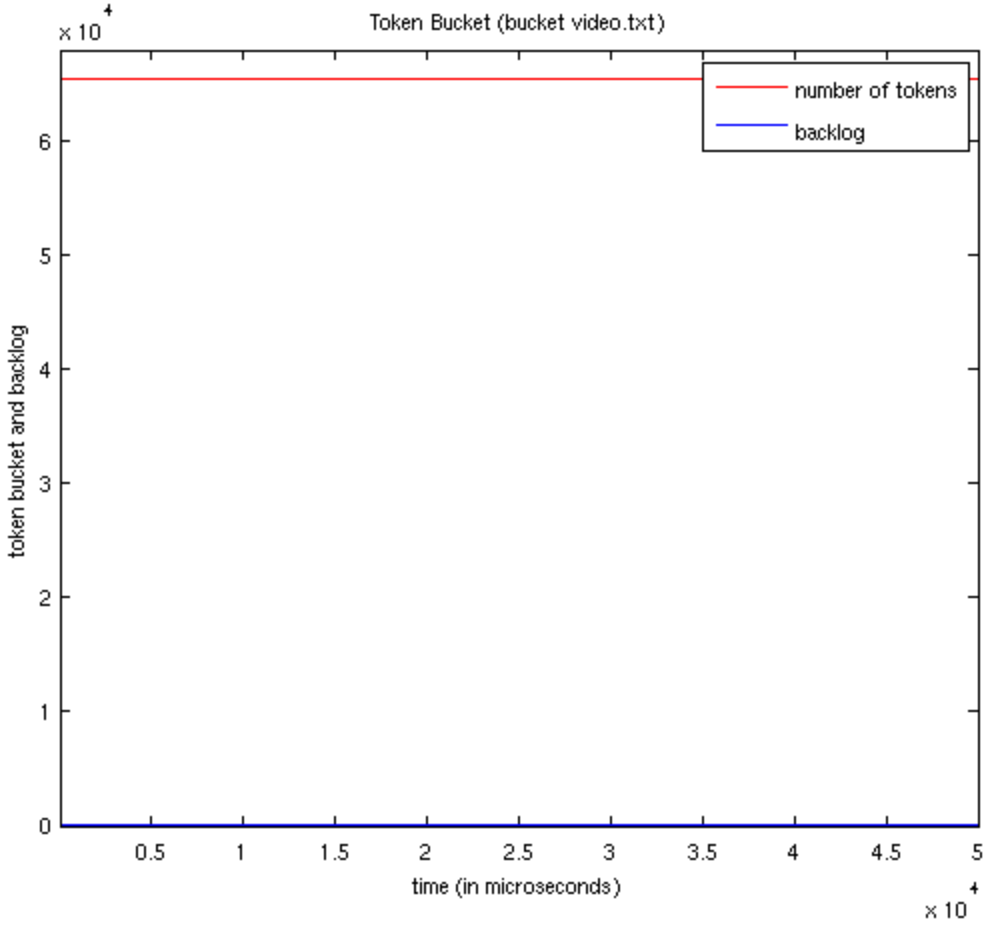
Prepare a single plot that shows the cumulative arrival function as a function of time of:

- The data of the trace file (as read by the traffic generator)
- The arrivals at the token bucket
- The arrivals at the traffic sink



As we can see, with the reference implementation, we have successfully transported our packets from the traffic generator over to the traffic shaper and over to the traffic sink. Since the UDP datagrams are limited to 65507 bytes as a safety precaution to ensure a consistent rate of successful packet transfers, we have introduced an 30-60 nanosecond delay to send back-to-back packets for one large frame. This is not a significant delay since it is a factor of a 1000 times smaller compared to the 33000 nanoseconds elapsed time between two frames.

Provide a second plot that shows the content of the token bucket and the backlog in the Buffer as a function of time



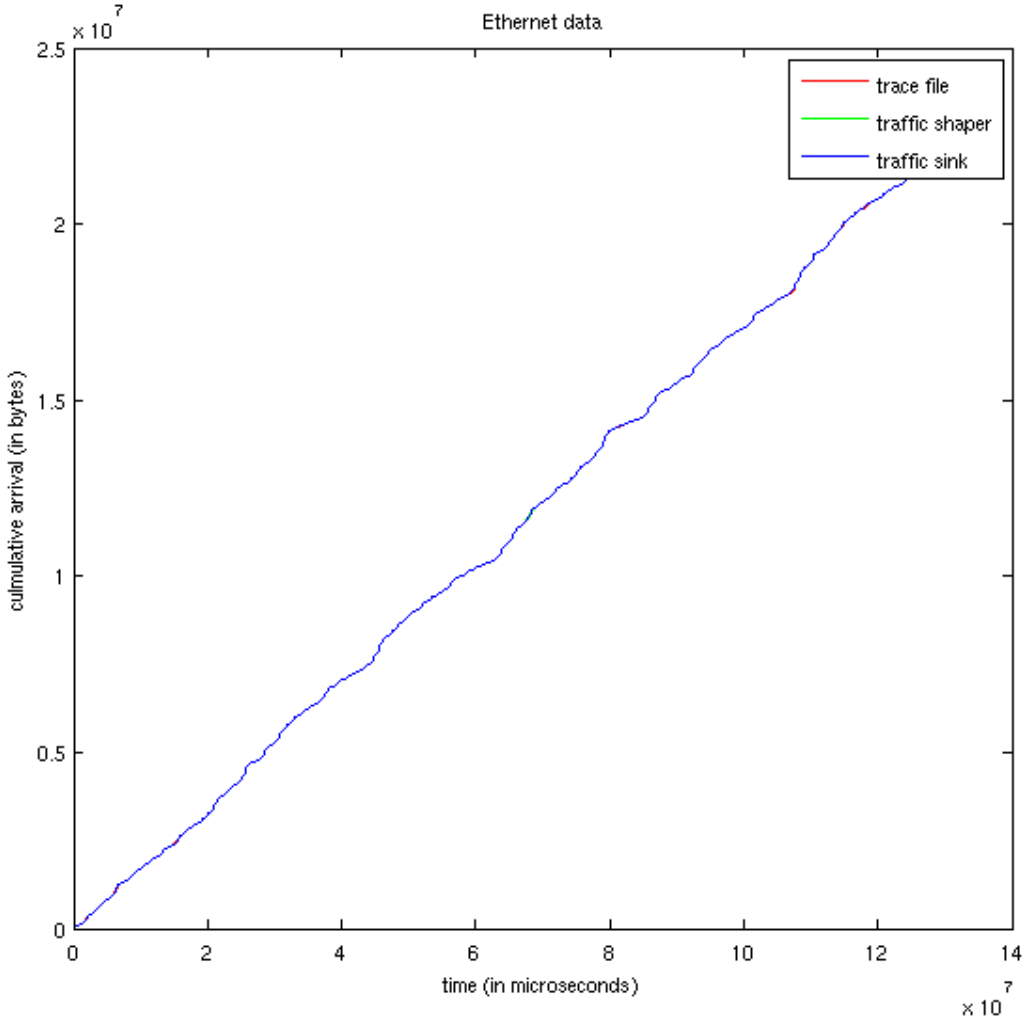
Since we have increased the maximum packet size to the limit which is imposed by the underlying IPv4 protocol of 65,507 bytes (65,535 - 8 byte UDP header - 20 byte IP header). We also had to increase the number of tokens in the token bucket to be greater or equal to the maximum size of a packet. Again, since we are using the reference implementation, the tokens are not used and there is no backlog.

Ethernet Traffic (BC-pAug89-small.TL)

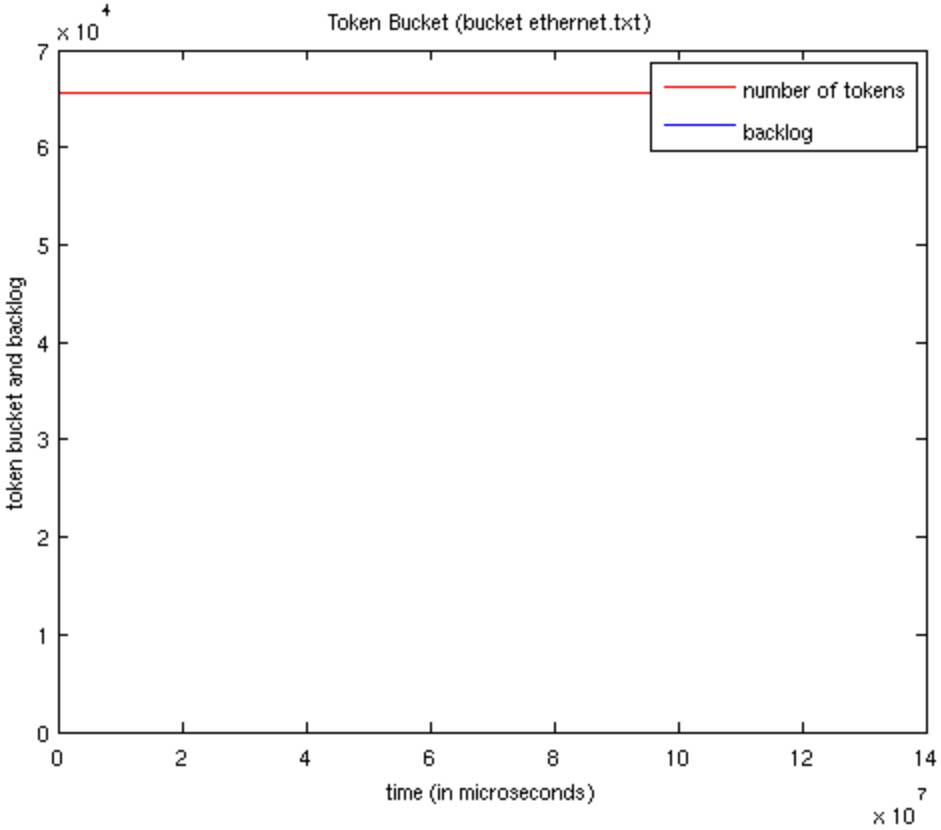
The traffic generator can be found in part3/TrafficGeneratorEthernet.java

Prepare a single plot that shows the cumulative arrival function as a function of time of:

- The data of the trace file (as read by the traffic generator)
- The arrivals at the token bucket
- The arrivals at the traffic sink


Ethernet data
x 10^7
2.5
2
1.5
1
0.5
0
0
2
4
6
8
10
12
14
x 10^7
time (in microseconds)
cumulative arrival (in bytes)
trace file
traffic shaper
traffic sink

Provide a second plot that shows the content of the token bucket and the backlog in the Buffer as a function of time



This is the same case as before with the video trace file. We have left the max packet size to 65507 bytes to represent the UDP packet size limit.