ECE466 LAB2a

Chengchen Yang 1000285880
Yu Li 1000173554

Part1. 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.
    - Run the sender and receiver in two personal laptops

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	P Frames	B Frames
183776 bytes	111412 bytes	36093 bytes

Part2. Traffic generators

Exercise 2.1 Traffic Generator for Poisson traffic

- The program is in the folder part2/TrafficGenertor.java
- The program takes in the input file and store the packets in a list. Then, based on the time difference between packets to send the packets through the datagram.

Exercise 2.2 Build the Traffic Sink

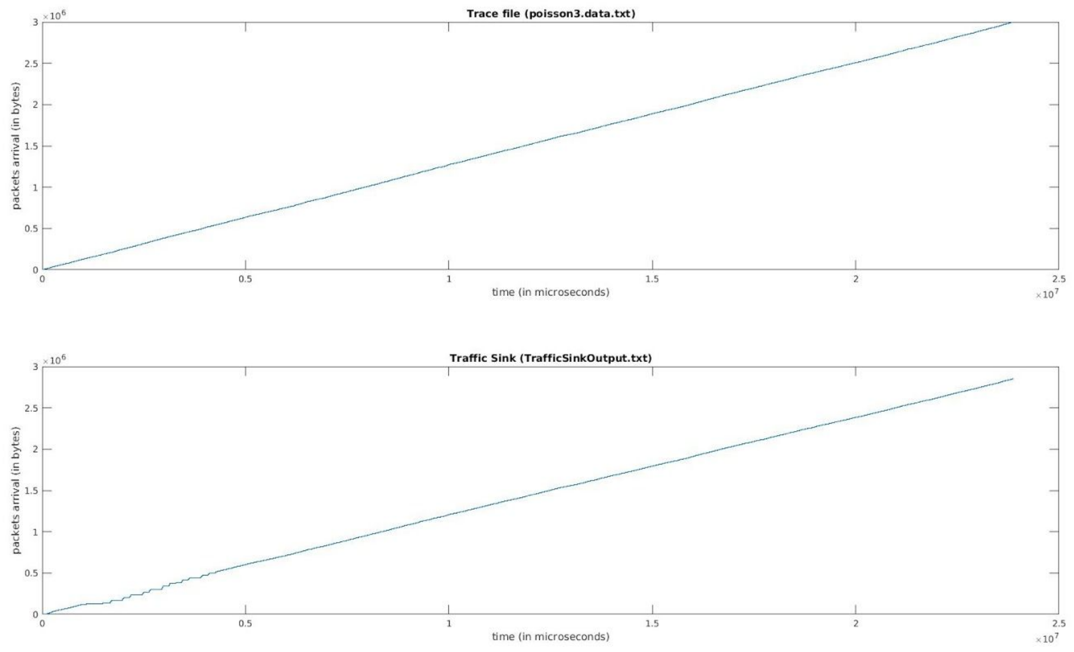
- The program is in the folder part2/TrafficSink.java
- The programs enters a while loop to wait for the data from the socket. Once the data is passed from TrafficGenerator, it reads the data information including the sequence number, frame size, and time. Write this information to the output file (TrafficSinkOutput.txt)

Exercise 2.3 Evaluation

Run experiments where you transmit traffic from the traffic generator to the traffic sink. Evaluate the accuracy of the traffic generator by comparing the entries in the trace file (at the traffic generator) to the results written to the output file (at the sink).

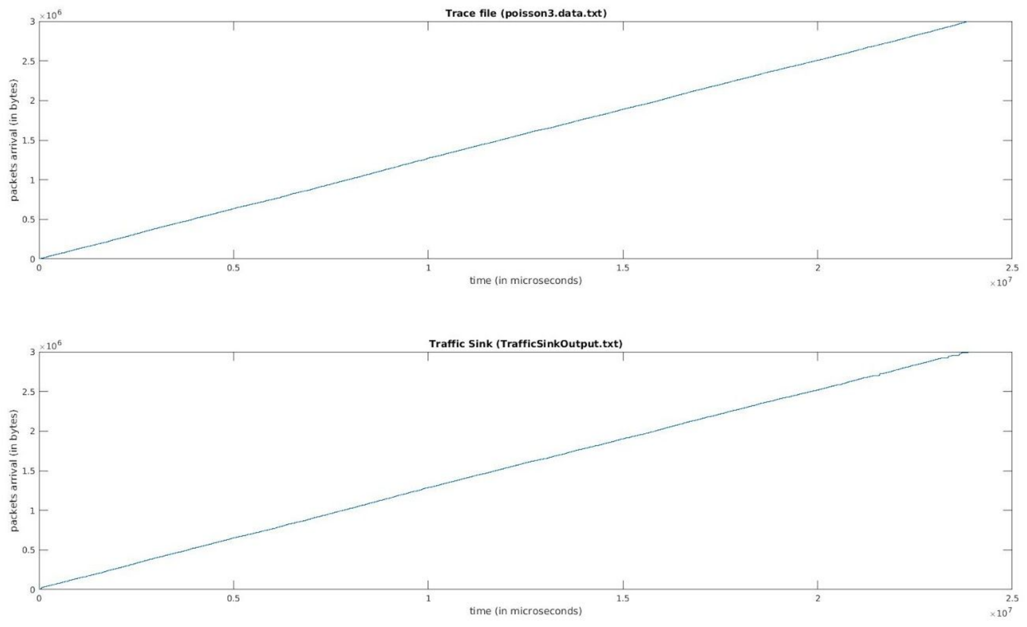
- **Use at least 10,000 data points for your evaluation.**
- **Prepare a plot that shows the difference of trace file and the output file. For example, you may create two functions that show the cumulative arrivals of the trace file and the output file, respectively, and plot them as a function of time.**

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



[The plot of the Trace file and the Traffic Sink Output]

The plot code for matlab is in the folder part2/plot.m. We use the matlab and use 30,000 data points as the input to the graphs. We can see that because of some packets loss, the output from the TrafficSink is lower than the cumulative arrival packets in the trace file. But they have the similar trend and the loss is only a small portion of the whole trace file.



[Improved plots of the trace and output]

We use the same plot code as the previous one to plot out the graphs above. We are originally use the buffer size of 512 bytes to store the packets coming from the socket. However, the maximum size of the packet size is more than this value. Therefore, we increase the size of buffer to 2048 bytes in order to cover the whole range of the packets sizes.
With the increasing of the buffer size, we can now achieve similar total arrival packets as in the trace file. We largely improve the effectiveness of sending data.

Exercise 2.4 Account for packet losses.

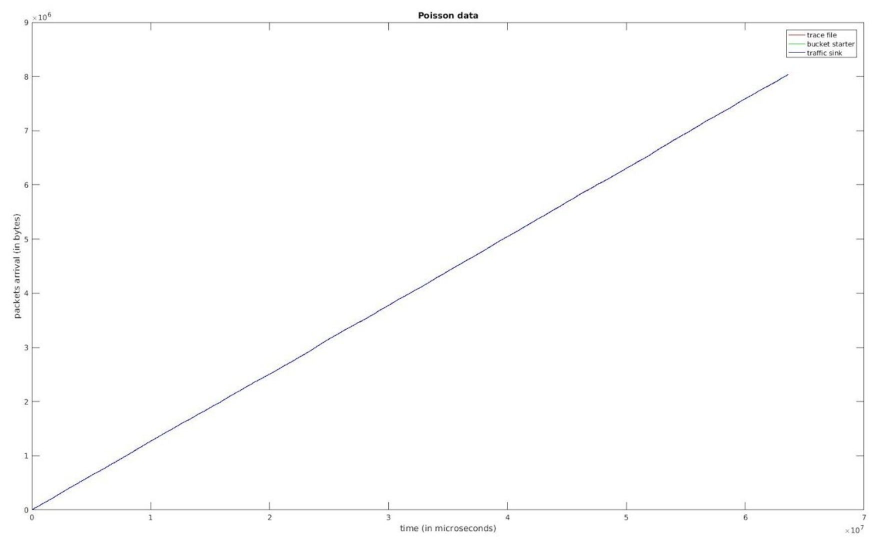
In the graphs that we show above, we can see that if the buffer size is too small, it causes some packets losses and the total size of arrival packets are smaller than the size in the trace file.When running the generator and sink in the same machine, there’s no packet loss and the curves are almost the same, the only difference is because the time scale mismatch.

Part 3. Token Bucket Traffic Shaper

Exercise 3.1 Running the reference implementation of the Token Bucket

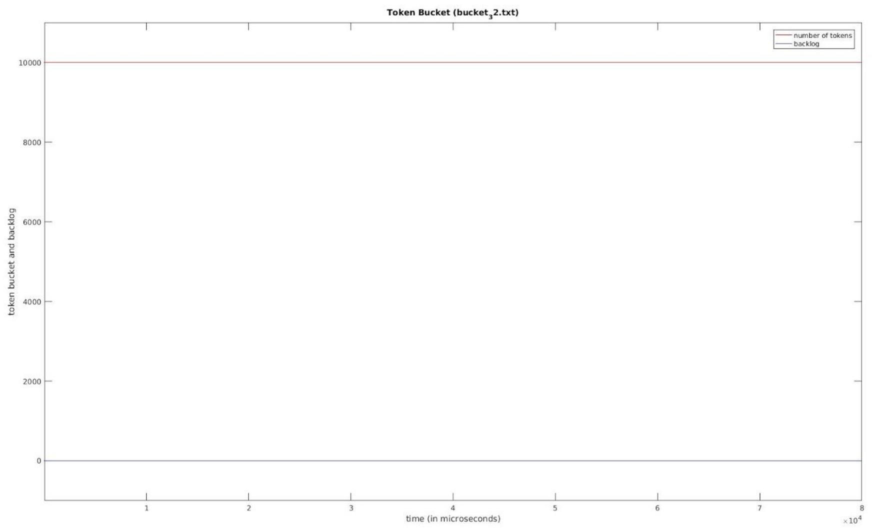
Source code can be found in part3/BucketStarter.java

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;**



Description: From the graph we can see there is almost no packet loss and very little delay, this is expected because we used two machines in the ug lab to complete this lab and network seems very stable in the lab.

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



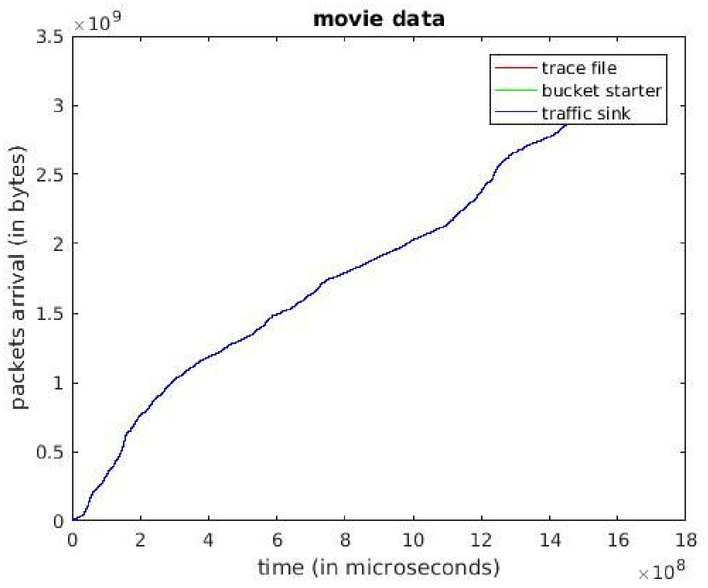
Description: Since we used provided token bucket implementation, there is no token decreasing mechanism so backlog never filled.

Exercise 3.3 Evaluate the reference implementation for the Ethernet and Video Tracefile

Video Traffic:

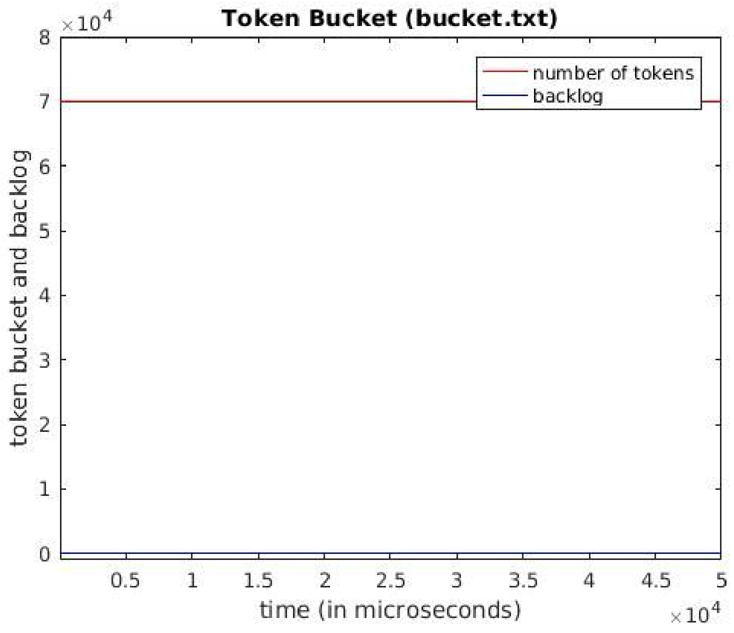
Prepare a single plot that shows the cumulative arrival function as a function of time of:

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



Description: From the graph we can see that by using the provided token bucket implementation, we have very close cumulative arrivals across traffic sink, trace file and traffic shaper. We used two machines in ug lab, and the network was quite stable therefore no packet lost. Since we use UDP to transport packets, we will have to split large packets to several smaller packets due to 65,507 bytes size limit(under ipv4). This is not a very large delay considering we have elapsed time at 33000 nanoseconds. Also we increased maximum token number to 70,000 to cover the increased maximum packet size.

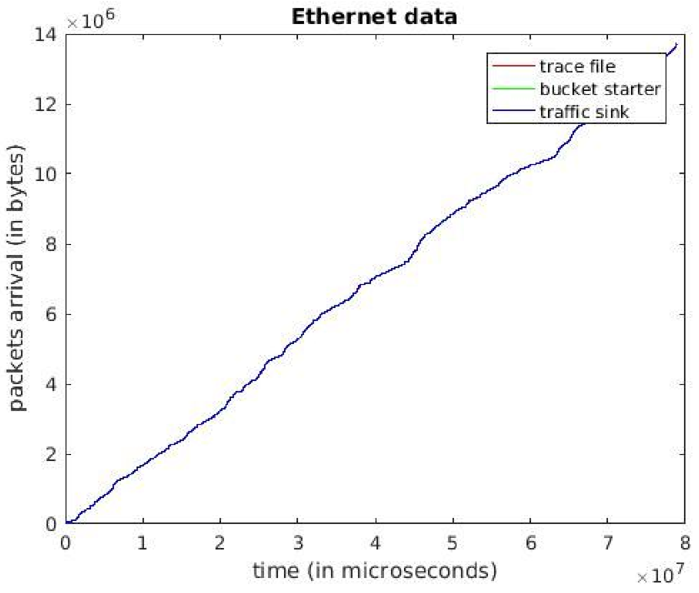
Provide a second plot that shows the content of the token bucket and the backlog in the Buffer as a function of time.



Description: Since we used provided token bucket implementation, there is no token decreasing mechanism so backlog never filled. And the number of token in the bucket is always 70,000.

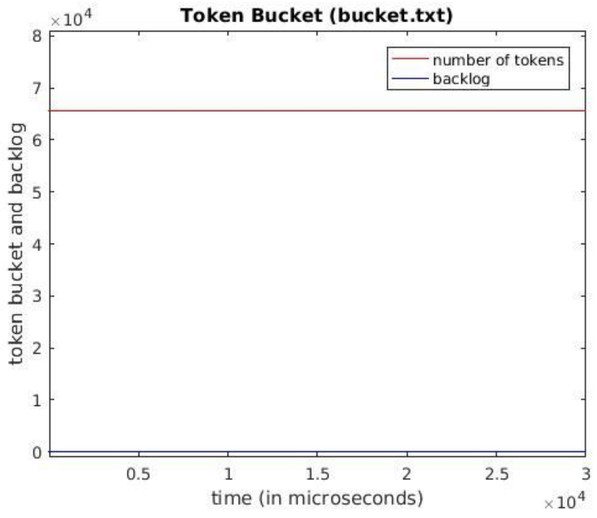
Ethernet Traffic:
Prepare a single plot that shows the cumulative arrival function as a function of time of:
The data of the trace file (as read by the traffic generator)

- **The arrivals at the token bucket;**
- **The arrivals at the traffic sink.**



Description: From the graph we can see that by using the provided token bucket implementation, we have very close cumulative arrivals across traffic sink, trace file and traffic shaper.

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



Description: Since we used provided token bucket implementation, there is no token decreasing mechanism so backlog never filled. And the number of token in the bucket is always the initial value we set for maximum datagram size.