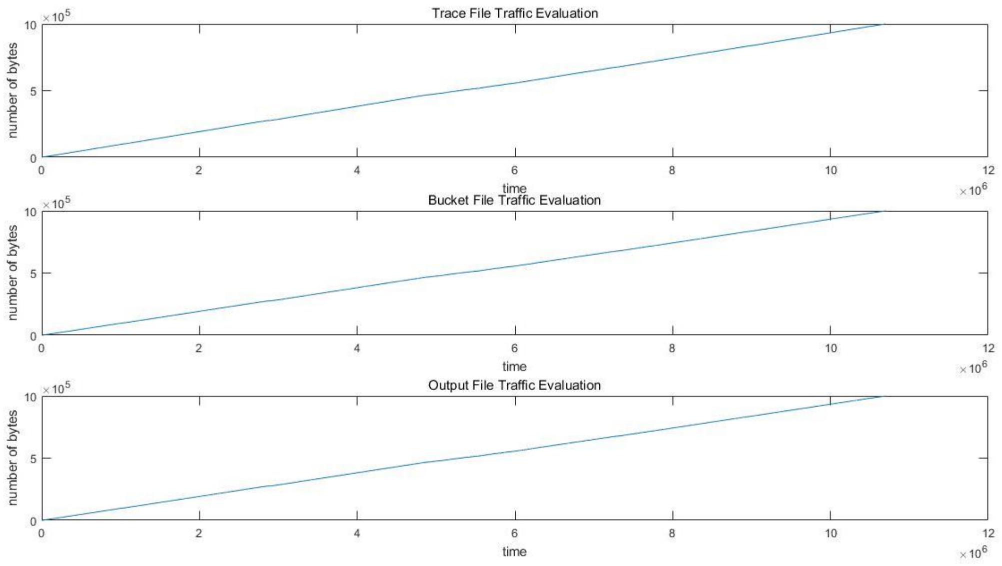
# ECE 466 Lab 2b

Xin Chen
1004391865

## Part1. A Reference Traffic Generator

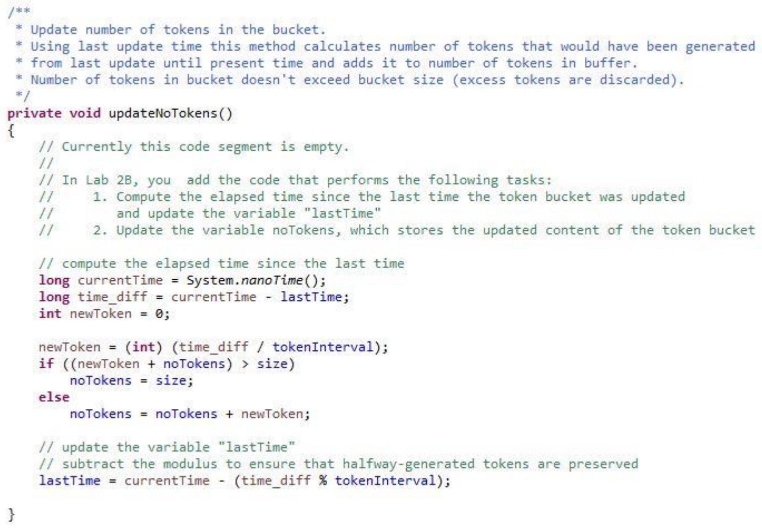
### Exercise 1.1 Create and Test Reference Traffic Generator



**Plot1. The above graph shows the cumulative arrivals of the trace file and the bottom graph show the cumulative arrivals of the output file with respect to time**

## Part2. Token Bucket Implementation

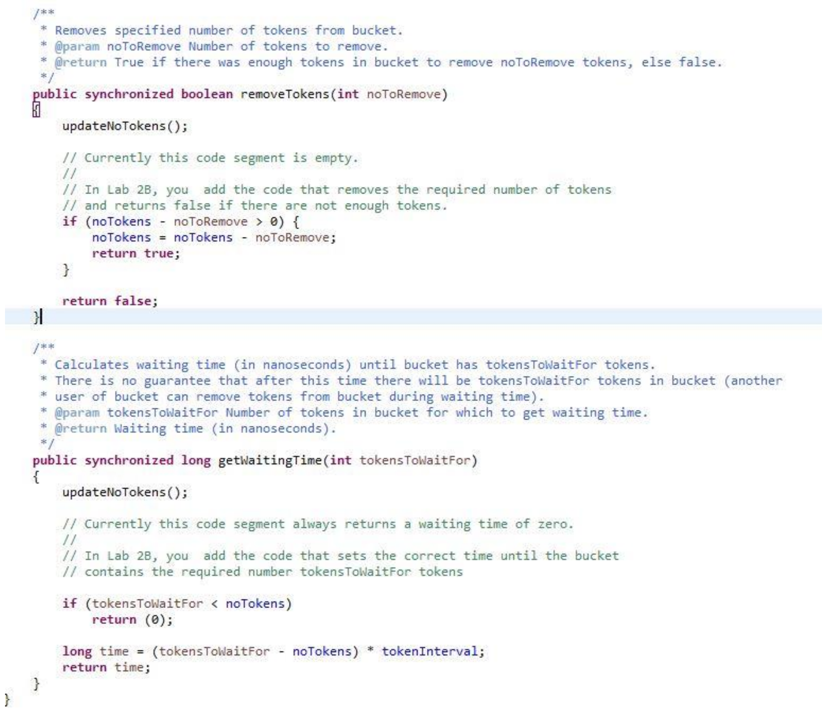
Exercise 2.1 Complete the implementation of the Token Bucket



```
/**
 * Update number of tokens in the bucket.
 * Using last update time this method calculates number of tokens that would have been generated
 * from last update until present time and adds it to number of tokens in buffer.
 * Number of tokens in bucket doesn't exceed bucket size (excess tokens are discarded).
 */
private void updateNoTokens()
{
    // Currently this code segment is empty.
    //
    // In Lab 2B, you  add the code that performs the following tasks:
    //     1. Compute the elapsed time since the last time the token bucket was updated
    //        and update the variable "lastTime"
    //     2. Update the variable noTokens, which stores the updated content of the token bucket

    // compute the elapsed time since the last time
    long currentTime = System.nanoTime();
    long time_diff = currentTime - lastTime;
    int newToken = 0;

    newToken = (int) (time_diff / tokenInterval);
    if ((newToken + noTokens) > size)
        noTokens = size;
    else
        noTokens = noTokens + newToken;

    // update the variable "lastTime"
    // subtract the modulus to ensure that halfway-generated tokens are preserved
    lastTime = currentTime - (time_diff % tokenInterval);

}
```

**Plot2. Implementation of the function "updateNoTokens()" in Token Bucket class**

The function "updateNoTokens()" calculates the time difference between last update time of the token bucket and the current time. According to the token adding rates to update the token size before deciding on the incoming packets. Also need to consider about the tokens do not exceed bucket size.



```
/**
 * Removes specified number of tokens from bucket.
 * @param noToRemove Number of tokens to remove.
 * @return True if there was enough tokens in bucket to remove noToRemove tokens, else false.
 */
public synchronized boolean removeTokens(int noToRemove)
{
    updateNoTokens();

    // Currently this code segment is empty.
    //
    // In Lab 2B, you  add the code that removes the required number of tokens
    // and returns false if there are not enough tokens.
    if (noTokens - noToRemove > 0) {
        noTokens = noTokens - noToRemove;
        return true;
    }

    return false;
}

/**
 * Calculates waiting time (in nanoseconds) until bucket has tokensToWaitFor tokens.
 * There is no guarantee that after this time there will be tokensToWaitFor tokens in bucket (another
 * user of bucket can remove tokens from bucket during waiting time).
 * @param tokensToWaitFor Number of tokens in bucket for which to get waiting time.
 * @return Waiting time (in nanoseconds).
 */
public synchronized long getWaitingTime(int tokensToWaitFor)
{
    updateNoTokens();

    // Currently this code segment always returns a waiting time of zero.
    //
    // In Lab 2B, you  add the code that sets the correct time until the bucket
    // contains the required number tokensToWaitFor tokens

    if (tokensToWaitFor < noTokens)
        return (0);

    long time = (tokensToWaitFor - noTokens) * tokenInterval;
    return time;
}
}
```

**Plot3. Implementation of the function "removeTokens()" and "getWaitingTime()" in Token Bucket class**

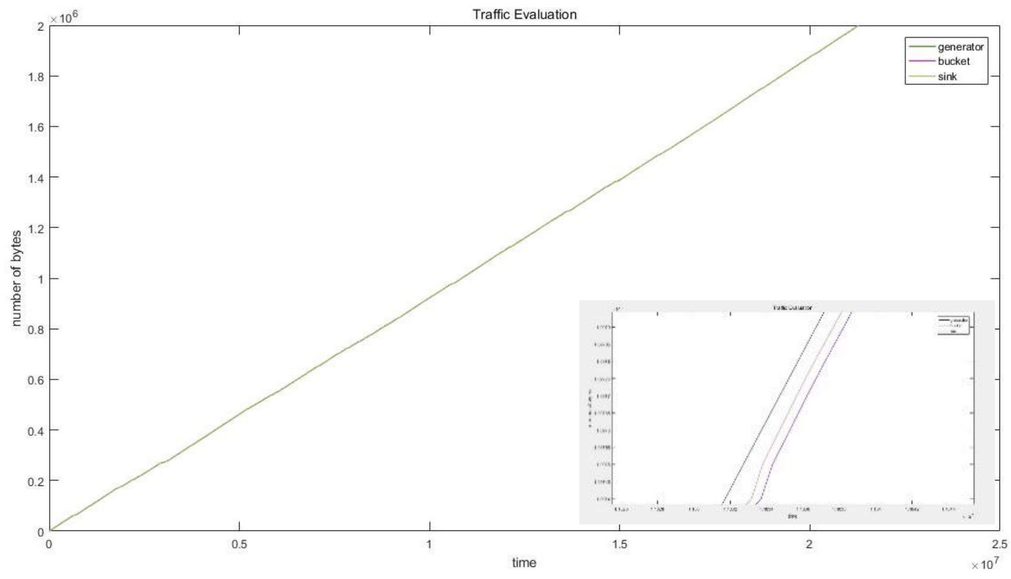
The function "removeTokens()" first update the token bucket. Then checks if the token in the token bucket has enough tokens to send out the incoming packets. Otherwise will return false and buffer will store the packets.

The function "getWaitingTime()" first update the token bucket. Then compute the time that needs to wait to have enough tokens to send out the packet.

Exercise 2.2 Validate Your Implementation

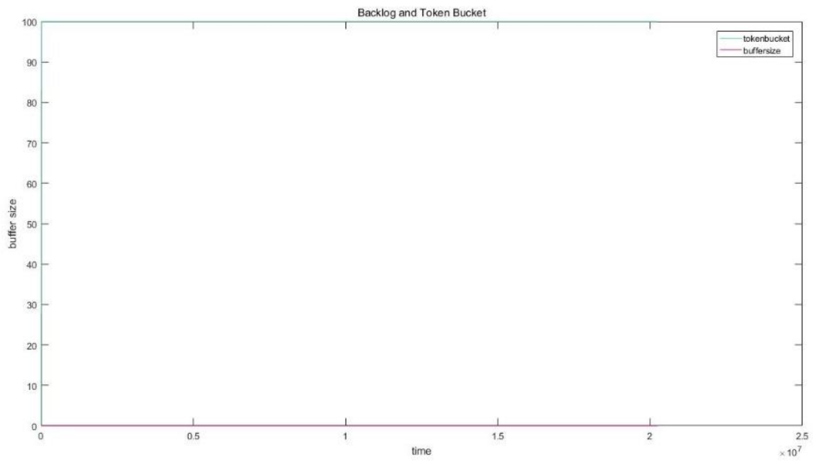
Exercise 2.3 Generate Plots for the Experiments

1. $\text{Rate}_{\text{source}}$ < rate TB, N = 1, size_TB = 100 Bytes



**Plot4. The above graph shows the cumulative arrival function as a function of time of the data of the traffic generator, the arrivals at the token bucket and the arrivals at the traffic sink respectively**

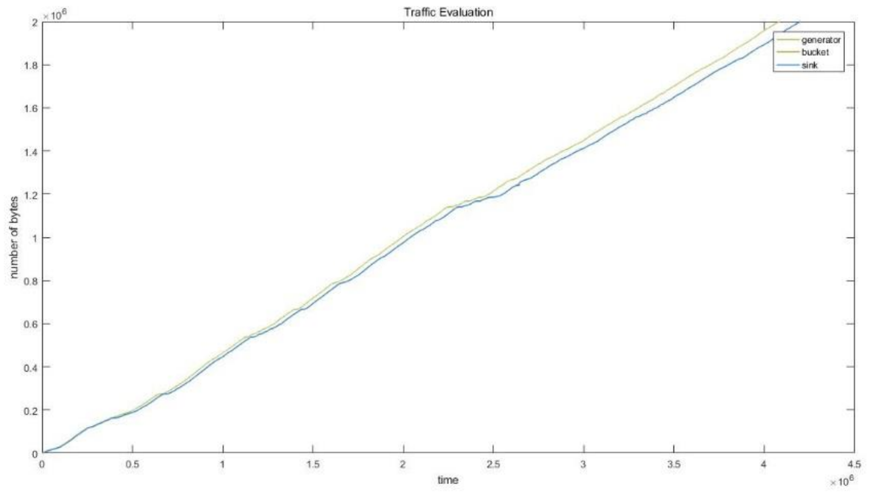
The cumulative number of bytes in the trace file and arrivals at token bucket and traffic sink has a similar trend. There might be some tiny difference between the generator traffic and token bucket and sink traffic as the time delay need to account for.



**Plot5. The content of the token bucket and the backlog in the Buffer as a function of time**

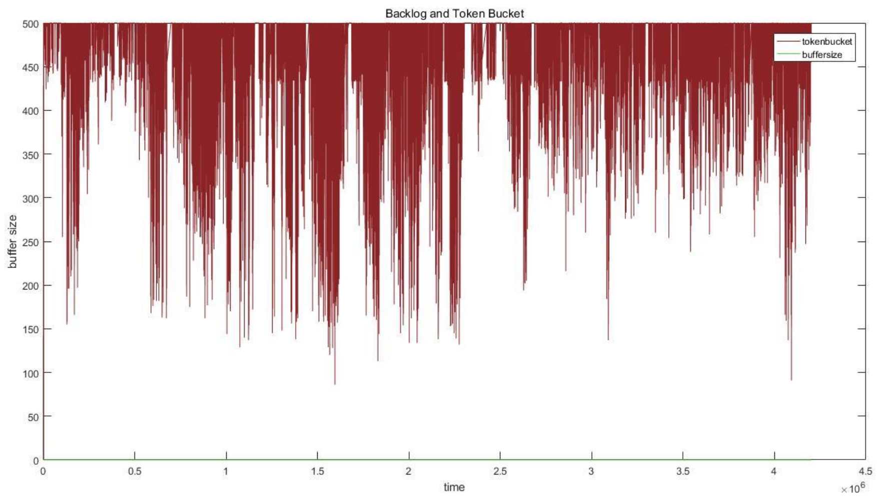
The content of the token bucket always has the value of 100 and the backlog in the buffer is 0. Even though the rate TB is 1Mbps, which means 1us will only add 1 token in the token bucket, but the transmission of file will almost cause 100us, which has enough time to fill up the token bucket and the backlog buffer will be zero.

2. $\text{Rate}_{\text{source}}$ < rate TB, N = 10, size_TB = 500 Bytes



**Plot6. The above graph shows the cumulative arrival function as a function of time of the data of the traffic generator, the arrivals at the token bucket and the arrivals at the traffic sink respectively**

The cumulative number of bytes in the trace file and arrivals at token bucket and traffic sink has a similar trend. There might be some tiny difference between the generator traffic and token bucket and sink traffic as the time delay need to account for.

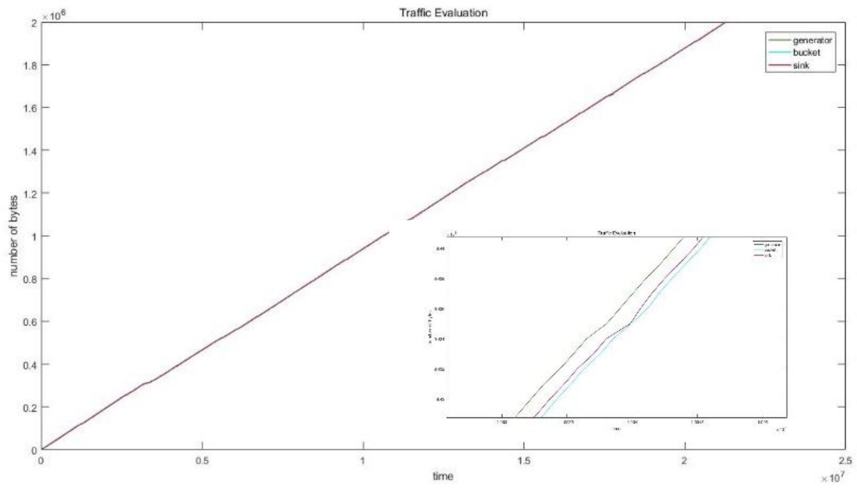


**Plot7. The content of the token bucket and the backlog in the Buffer as a function of time**

The content of the token bucket always has the value of 500. As the rate TB is large enough, which means there is will enough tokens added into the token bucket and the backlog buffer will be zero. While in the latter data(not shown in the graph) should the buffer increases while there is still enough tokens in the token bucket, this is because the socket buffer is full and the

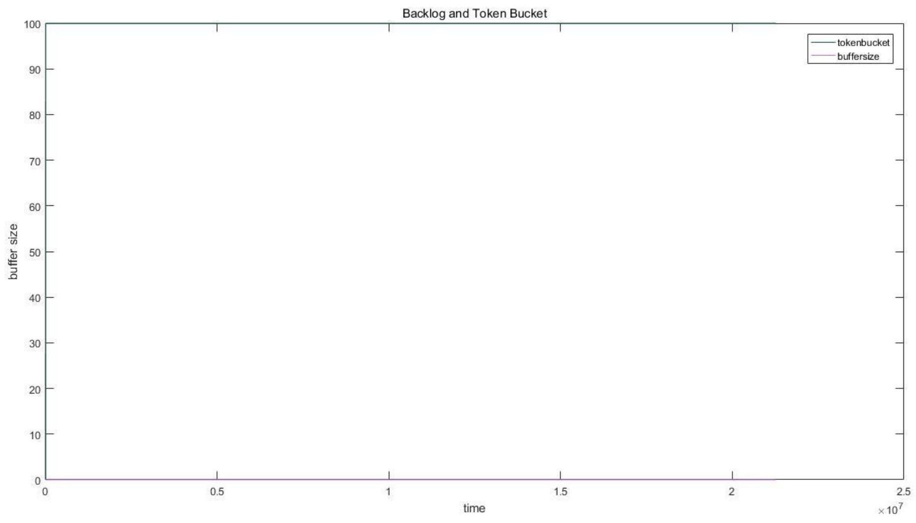
“IsInProgress” flag will always be true as the token bucket has enough token to transmit data.

3. $\text{Rate}_{\text{source}}$ = rate TB, N = 10, size_TB = 100 Bytes



**Plot8. The above graph shows the cumulative arrival function as a function of time of the data of the traffic generator, the arrivals at the token bucket and the arrivals at the traffic sink respectively**

The cumulative number of bytes in the trace file and arrivals at token bucket and traffic sink has a similar trend. There might be some tiny difference between the generator traffic and token bucket and sink traffic as the time delay need to account for.

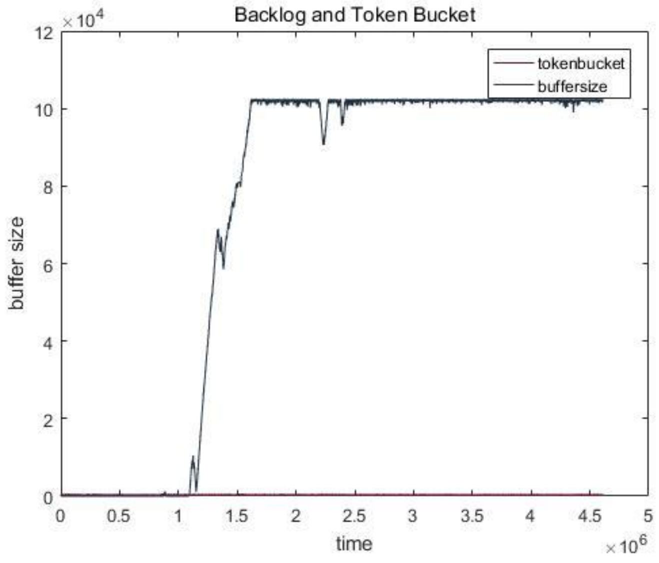


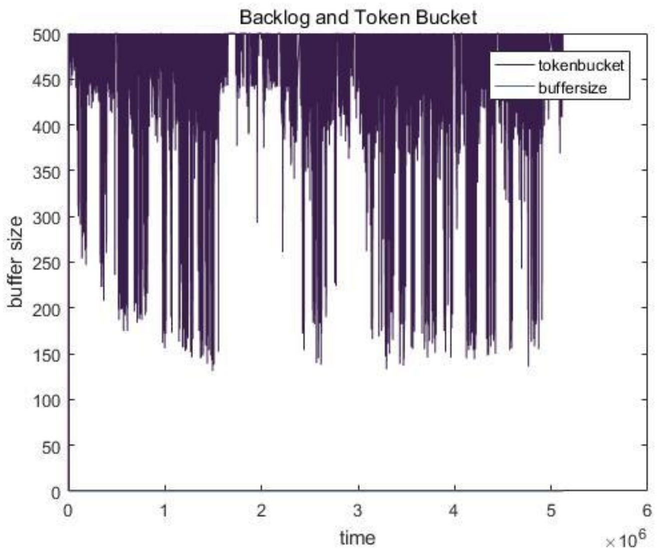
**Plot9. The content of the token bucket and the backlog in the Buffer as a function of time**

The content of the token bucket always has the value of 100 and the backlog in the buffer is 0. Even though the rate TB is 1Mbps, which means 1us will only add 1 token in the token bucket, but the transmission of file will almost cause 100us, which has enough time to fill up the token bucket and the backlog buffer will be zero.

Exercise 2.4 Maximum Rate of Token Bucket

The maximum arrival rate that can be supported by my token bucket will determine by the token bucket parameters. As the burst size is 500 bytes, and the rate of generating token is 1.5Mbps, with N = 10, L = 100bytes, T = 1us. The maximum arrival rate should less than the rate of generating token bucket, about 1Mbps, so that the backlog is not always increasing.





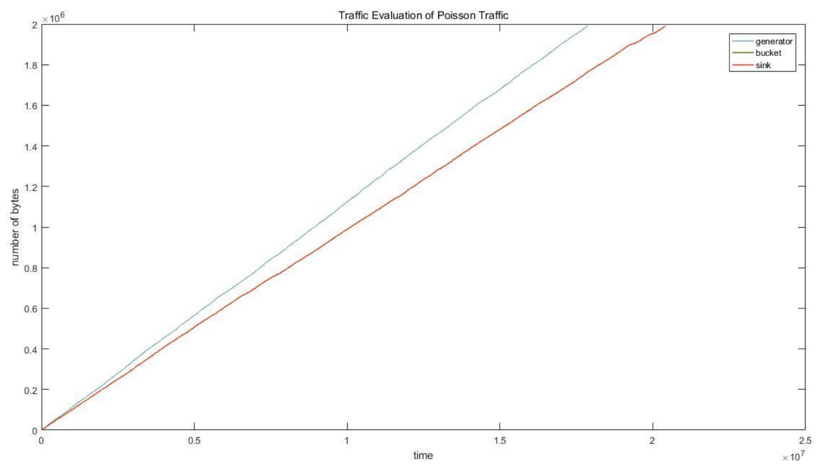
The left-hand side of the rate of generating token is about 1Mbps, which the tokens in the token bucket quickly decreases to 0 and the backlog increases to its maximum value. While the right-hand side the rate of generating token is about 1.5Mbps, which the tokens in the token bucket are always available for the transmission of data.

## Part3. Engineering Token Bucket Parameters

<u>Exercise 3.1 (Long-term) Bandwidth is Cheap</u>

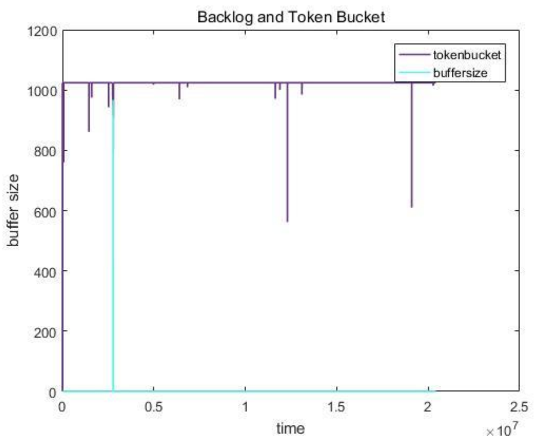
As the token bucket bandwidth is cheap, which we need to choose as small the token bucket burst size as possible. So that we used the maximum data transfer length as the token bucket size. This is because if we choose a smaller token bucket size, the data length that is larger than the burst size will never able to transmit and all the latter data will also store into the buffer and will never able to be transmitted. The rate_TB is considering the data transfer rate and the delay of the data.

1. Poisson Traffic



**Plot10. The above graph shows the cumulative arrival function as a function of time of the data of the traffic generator, the arrivals at the token bucket and the arrivals at the traffic sink respectively**

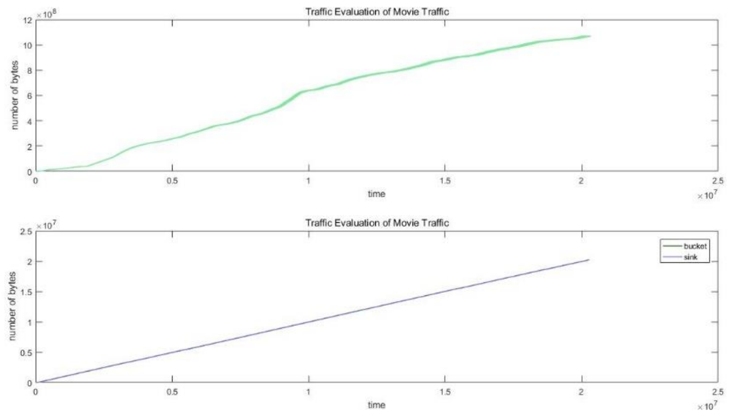
The cumulative number of bytes in the trace file and arrivals at token bucket and traffic sink has a similar trend. There might be some difference between the generator traffic and token bucket and sink traffic as the time delay in the generator traffic will have impact on the final plotting.



**Plot11. The content of the token bucket and the backlog in the Buffer as a function of time**

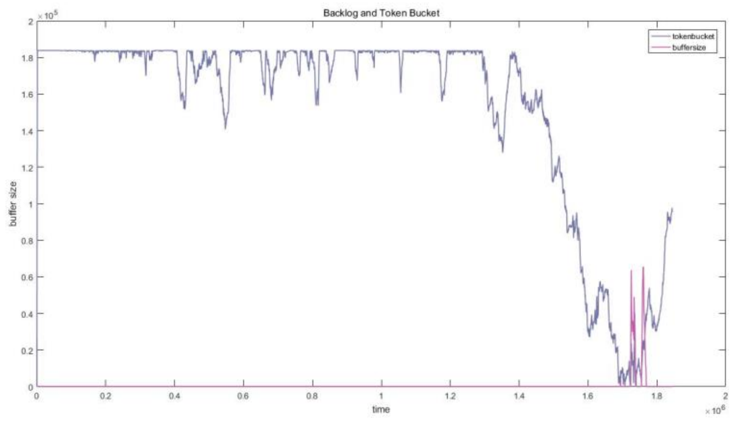
The content of the token bucket always has the value of 1024 and the backlog in the buffer is 0. There is always enough token in the token bucket and the backlog buffer will be zero. While in the graph, it shows a burst in the buffer size (1024 bytes), that might be due to the internet environment while running the program, and might have 1 or 2 packets of data not able to transmit and stored in the buffer.

2. Movie Trace Traffic



**Plot12. The above graph shows the cumulative arrival function as a function of time of the data of the traffic generator, the arrivals at the token bucket and the arrivals at the traffic sink respectively**

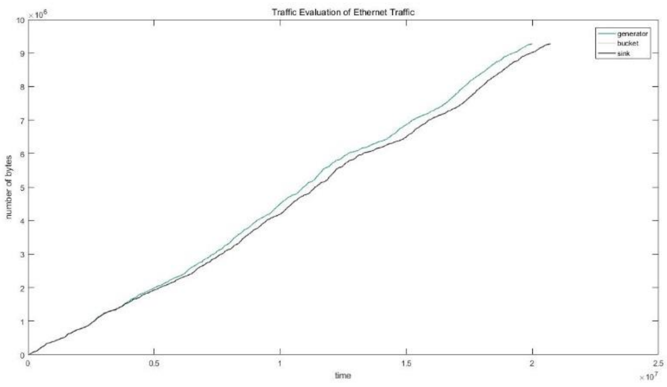
We cannot plot the traffic generator file together with the token bucket and arrivals in the same plot. As the maximum length of data is 187360 bytes, which is exceed the maximum length that can be transmitted. We slice up the data into 1024 bytes per packet. While accounting for the 20 000 data points in plotting, as the data in token bucket and traffic sink been slice up, the total time is the same, but the total data bytes will be different.



**Plot13. The content of the token bucket and the backlog in the Buffer as a function of time**

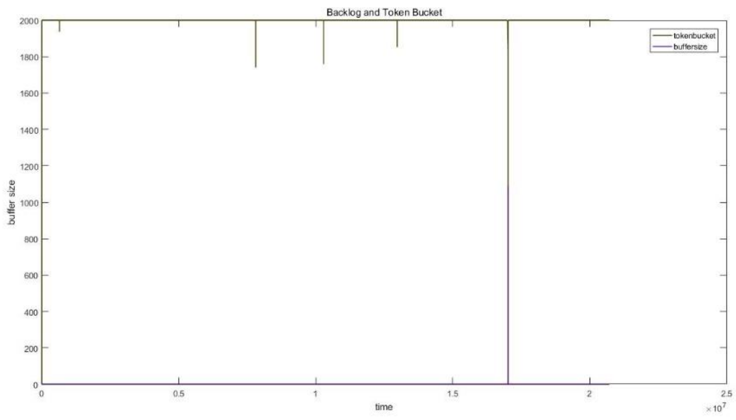
The content of the token bucket has the maximum value of 1024 and the backlog in the buffer is 0 if there is no token in token bucket.

## 3. Ethernet Traffic



**Plot14. The above graph shows the cumulative arrival function as a function of time of the data of the traffic generator, the arrivals at the token bucket and the arrivals at the traffic sink respectively**

The cumulative number of bytes in the trace file and arrivals at token bucket and traffic sink has a similar trend. There might be some difference between the generator traffic and token bucket and sink traffic as the time delay in the generator traffic will have impact on the final plotting.

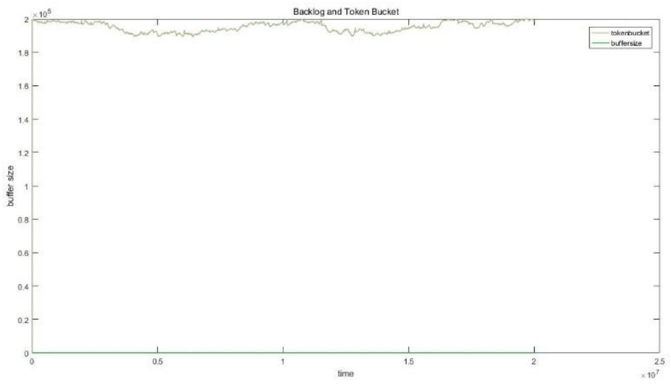


**Plot15. The content of the token bucket and the backlog in the Buffer as a function of time**

The content of the token bucket always has the value of 1024 and the backlog in the buffer is 0. There is always enough token in the token bucket and the backlog buffer will be zero. While in the graph, it shows a burst in the buffer size (1024 bytes), that might be due to the internet environment while running the program, and might have 1 or 2 packets of data not able to transmit and stored in the buffer.

Exercise 3.2 (Long-term) Bandwidth is Expensive

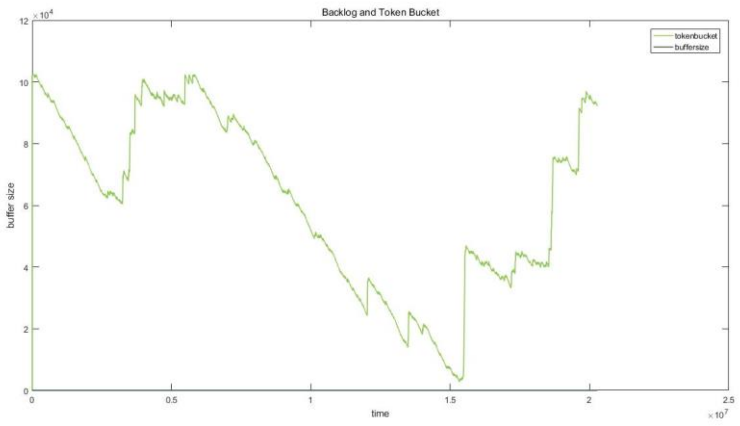
1. Poisson Traffic



**Plot16. The content of the token bucket and the backlog in the Buffer as a function of time**

We set up a large value for the burst size of token bucket and a small value 0.1 token per us to the rate_TB. Which shows there is always enough token in the token bucket to transmit data so that there will be no backlog in the buffer.

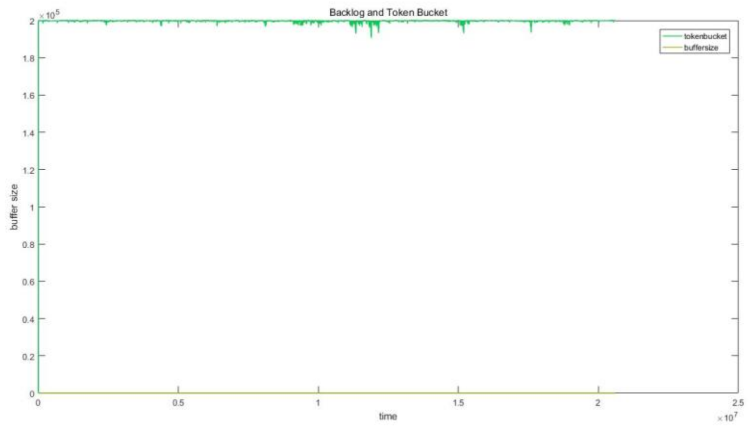
2. Movie Trace Traffic



**Plot17. The content of the token bucket and the backlog in the Buffer as a function of time**

We set up a large value for the burst size of token bucket and a small value to the rate_TB. Which shows there is always enough token in the token bucket to transmit data so that there will be no backlog in the buffer. But the token bucket decrease as the transmission rate of data is increasing. While encounter a large time gap to transmit the packet, will helps to increase the token buckets.

3. Ethernet Traffic

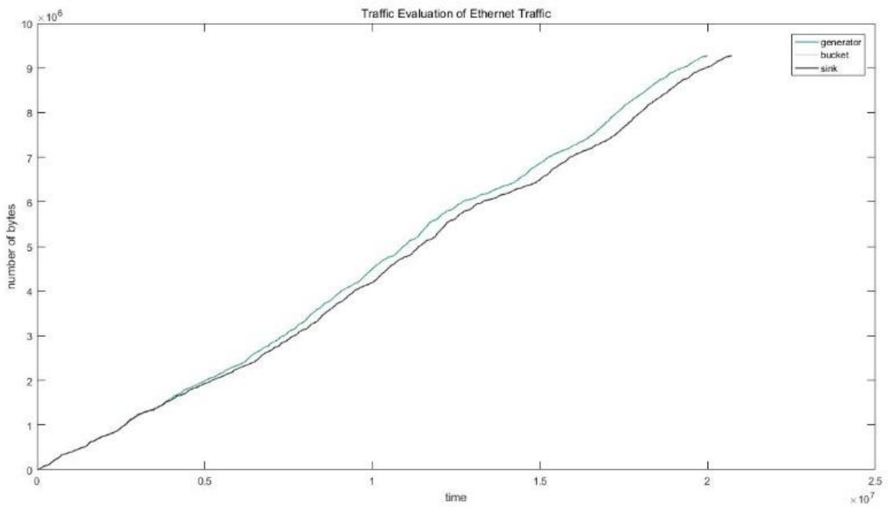


**Plot18. The content of the token bucket and the backlog in the Buffer as a function of time**

We set up a large value for the burst size of token bucket and a small value 0.1 token per us to the rate_TB. Which shows there is always enough token in the token bucket to transmit data so that there will be no backlog in the buffer.

Exercise 3.3 Analysis

1. **Compare the results for the three traffic types.**
   a) Video traffic is most demanding in terms of bandwidth and memory requirements.
   b) Ethernet traffic is the second and Poisson traffic is the least in demand of bandwidth and memory
2. **Which type of traffic benefits the most from traffic shaping?**
   The ethernet traffic benefits the most from traffic shaping


Traffic Evaluation of Ethernet Traffic
generator
bucket
sink
number of bytes
time
$\times 10^6$
$\times 10^7$