Lab
**2b**

# Traffic Shaping (Part 2)

## Purpose of this lab:

This lab uses the token bucket implementation (from Lab 2a) for experiments with traffic shaping. The traffic for testing the token bucket will be the Poisson, VBR video, and LAN traffic traces from Lab 1.

## Software Tools:

- This lab will use the software code from Lab 2a.

## What to turn in:

- Turn in a report with your answers to the questions in this lab, including the plots, your Java code, and the anonymous feedback form.

Version 2 (February 12, 2011)
Version 3 (January 25, 2013)

## Table of Content



## Comments

- **Part 4 is optional for an extra credit of (max.) 20%.**

## Part 1. A Reference Traffic Generator

In this part of the lab you create a simple traffic generator that can be used for testing and tuning your token bucket implementation. This reference traffic generator creates a predictable periodic traffic pattern of UDP datagrams. The traffic generator takes as input parameters three values *T, N, L,* with the interpretation:

"Every *T* msec transmit a group of *N* packets back-to-back, each with a size of *L* bytes"

If the group of N packets are sent quickly, then the traffic generator sends traffic according to an arrival function A with:

$$\begin{aligned} A(t) &= \lceil t/T \rceil \cdot N \cdot L \ \text{Bytes} \\ &= \lceil t/T \rceil \cdot N \cdot L \cdot 8 \ \text{Bits} \end{aligned}$$

The long term average traffic rate of the traffic generator is given by $Rate_{source} = NL8/T$ bps (bits per second).

For each transmitted packet, write a line to an output file that records the size of the packet and the time since the transmission of the previous packet (For the first packet, the time is zero), e.g., the file has the format:

| SeqNo | Time since last arrival (in μsec) | Size (in Bytes) |
|---|---|---|
| 1 | 0 | 320 |
| 2 | 934 | 99 |
| 3 | 2293 | 27 |
| … | | |

### Exercise 1.1 Create and Test Reference Traffic Generator

Build the traffic generator by modifying the implementation of the traffic generator from Lab 2a (Part 2).

Run experiments for specified values of T, N, and L, where you transmit traffic from the traffic generator to the traffic sink (without a token bucket). 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.

## Part 2. Token Bucket Implementation

You are asked to implement the missing features from the token bucket implementation provided to you at:

`http://www.comm.utoronto.ca/~jorg/teaching/ece466/labs/lab2/TokenBucket`

You only need to modify the implementation of the TokenBucket class. The provided implementation does not correctly update the value of TB and does not correctly compute the time for waking up the Bucket Sender. An instance of the TokenBucket is created by calling the constructor of this class with two parameters *size* and *rate*, where

- `size_TB` is the maximum content of the token bucket ($TB_{max}$),
- `rate_TB` is the token generation rate in tokens per second (r).

Other classes can query the TokenBucket about the content of the token bucket, and can request to remove tokens from the bucket. The following requests are available:

- `getNoTokens`: Returns number of tokens in bucket.
- `removeTokens(X)`: Request to remove X tokens from the bucket. The method returns false if there are less than X tokens in the bucket.
- `getWaitingTime(X)`: Returns the waiting time (in nanoseconds) until bucket has X tokens. Note that there is no guarantee that there will be enough tokens at the returned time (someone else may have removed tokens).

### Exercise 2.1 Complete the implementation of the Token Bucket

Complete the implementation of the TokenBucket class by providing the algorithms for updating the contents of the token bucket, and executing the requests `getNoTokens, removeTokens, getWaitingTime.`

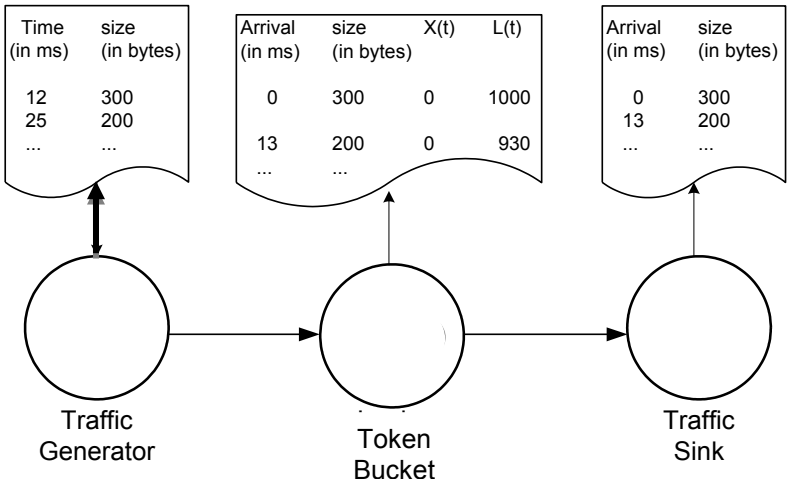
Hints:

- The contents of the token bucket does not need to be kept updated at all times. It is sufficient to update the token bucket only when there is a request made to TokenBucket.
- Be consistent about whether a single token corresponds to a Bit or a Byte.
- Refer to the documentation of the classes available at:

`http://www.comm.utoronto.ca/~jorg/teaching/ece466/labs/lab2/doc/index.html`

### Exercise 2.2 Validate your implementation

Design and conduct a series of measurement experiments that show that your implementation of the token bucket is correct. The setup of the experiments is as seen in Lab 2a:




Use the reference traffic generator from Part 1 to create a deterministic traffic stream with known properties.

When you conduct a measurement experiment, record relevant data (The code for this should already be available):

- **At the traffic generator:** For each transmitted packet, record the size of the packet and the time since the previous packet transmission (For the first packet, the time is zero).

- **At the Token Bucket:** Upon each packet arrival, write a line to an output file that records the size of the packet and the time since the arrival of the last packet (For the first packet, the time is zero). Also recorded are the number of tokens (TB(t)) in the token bucket and the backlog in the buffer (B(t)) after the arrival of the packet.

- **At the traffic sink:** For each incoming packet, write a line to an output file that records the size of the packet and the time since the arrival of the previous packet (For the first packet, the time is zero).

Use a packet size L=100 Bytes for all experiments. For the rate parameter at the traffic generator ($Rate_{source}$ < rate_TB) and at the Token Bucket ($Rate_{source}$ < rate_TB), select rates in the range 0.5 – 5 Mbps.

Conduct the following measurement experiments:

1. **$Rate_{source}$ < rate_TB, N =1, size_TB = 100 Bytes:**
   Here, the long term rate of the source is smaller than the rate allowed by the token bucket. With N=1, the traffic source generates single packets that are equally spaced.

   Expected observations: The Token Bucket has always sufficient tokens. There is never a backlog in the Buffer.

2. **$Rate_{source}$ < rate_TB, N =10, size_TB = 500 Bytes:**

Here, the long term rate of the source is smaller than the rate allowed by the token bucket. The maximum burst size at the source is 10 packets, however, the token bucket allows bursts to consist of at most five packets.

Expected observations: Whenever a burst (group) of 10 packets arrives, the Token Bucket releases 5 packets quickly, and then spaces the remaining packets out as determined by the rate parameter. The backlog in the Buffer depends on the rate at which the burst of packets arrives, and the rate at which the Token Bucket can transmit a packet. The maximum backlog should be somewhere between 5-9 packets.

3. $Rate_{source} \approx$ **rate_TB, N =1, size_TB = 100 Bytes:**
Here, the arrival rate of traffic is approximately equal to the long term rate of the Token Bucket. (To prevent persistent overflows, it is recommended that the arrival rate is a little smaller than the rate of the Token Bucket).

Expected observations: This setting can be used to test if your Token Bucket implementation can support the data rates of the source. If the Token Bucket can keep up with the arrival rate, then there should never be a backlog in the Buffer. On the other hand, if the Buffer builds up and eventually overflows, then the Token Bucket implementation is too slow.

## Exercise 2.3 Generate plots for the experiments

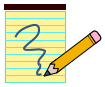
Use the data saved in the measurement experiments to generate a series of plots. For each experiment above, prepare a single plot that shows the cumulative arrival function as a function of time of:

- The transmissions of the reference traffic generator;
- The arrivals at the token bucket;
- The arrivals at the traffic sink.

For each experiment, provide a second plot that shows the content of the token bucket and the backlog in the Buffer as a function of time.

## Exercise 2.4 Maximum rate of Token Bucket

Determine the maximum arrival rate of traffic that can be supported by your Token Bucket. Support your findings with an experiment and (at least) one data plot.



**Lab Report:**

- Describe the design of your implementation for the Token Buffer. Include your source code in the lab report. (Only include the source code of methods that you have modified).
- For Exercises 2.2 and 2.3, provide the set of plots and include a description of the plots. Describe your observations in each of the experiments.
- For Exercise 2.4 describe how you determined the maximum supported arrival rate, and discuss any graph that you generated to support your findings.

## Part 3. Engineering Token Bucket Parameters

The task in this part of the lab is to determine token bucket parameters for the three traffic sources used in Labs 1 and 2a, consisting of Poisson traffic, compressed video, and the Bellcore Ethernet traffic trace.

The tracefiles are available from the usual places:

- **Poisson traffic:**
  http://www.comm.utoronto.ca/~jorg/teaching/ece466/labs/lab1/poisson3.data
- **Video trace file:**
  http://www.comm.utoronto.ca/~jorg/teaching/ece466/labs/lab1/movietrace.data
- **Ethernet traffic:**
  http://www.comm.utoronto.ca/~jorg/teaching/ece466/labs/BC-pAug89-small.TL

The Token Bucket parameters must be selected subject to (all) of the following constraints:

- The objective is to smooth the traffic as much as possible, by spacing out packets, but without delaying traffic by more than 100ms (milliseconds).
- The required size of the buffer in the Token Bucket should be as small as possible, but the buffer should be large enough so that no overflows occur.
- The rate of the token bucket should be at least equal to the average rate of the traffic source, and less than the peak rate of the traffic source (In Lab 1, average and peak rate where calculated for each of the traffic sources).

There is generally a whole family of solutions for size_TB and rate_TB that can satisfy the above constraints. Your choice of parameters must trade-off the desire to smooth the traffic rate (small value for rate_TB) or to reduce the burst size of the output of the token bucket (small value for size_TB)

Note: Using the topics covered in the lecture, you can compute the required parameters without resorting to trial and error. To do this, you need to compute the empirical envelope of each traffic source, and use the service curve S(t) = size_TB + rate_TB * t.

## Exercise 3.1 (Long-term) Bandwidth is cheap

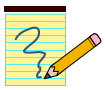
For each of the three traffic sources (Poisson, Video, Ethernet), determine the token bucket parameters that satisfy the above constraints. When you have to choose between increasing rate_TB or size_TB, your preference should be to increase rate_TB.

- For the parameters that you select, generate a plot that shows the cumulative arrival function as a function of time for
    - arrivals at the token bucket;
    - arrivals at the traffic sink.

  Also plot the content of the token bucket and the backlog in the Buffer as a function of time.

## Exercise 3.2 (Long-term) Bandwidth is expensive

Repeat the steps of Exercise 3.2, but with a different assumption. Now, when you have to choose between increasing rate_TB or size_TB, your preference should be to increase size_TB.



**Lab Report:**

- Discuss your method for selecting the token bucket parameters.
- For Exercises 3.1 and 3.2, provide the set of plots and include a description of the plots. Describe your observations in each of the experiments. Compare the results for the cheap bandwidth and expensive bandwidth scenarios.
- 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? Support your findings with data (plots).

## Part 4. (Optional) Shaping VBR video with Multiple Token Buckets

The regulation of VBR video traffic with a token bucket has to address the following two problems:

- Since the size of a frame can be much larger than the maximum packet size (in the video trace file, the maximum frame size exceeds 600 KB), yet the time lag between frames is relatively long, one would like to select token bucket parameters that stretch the transmission of a frame over the entire 33 ms.

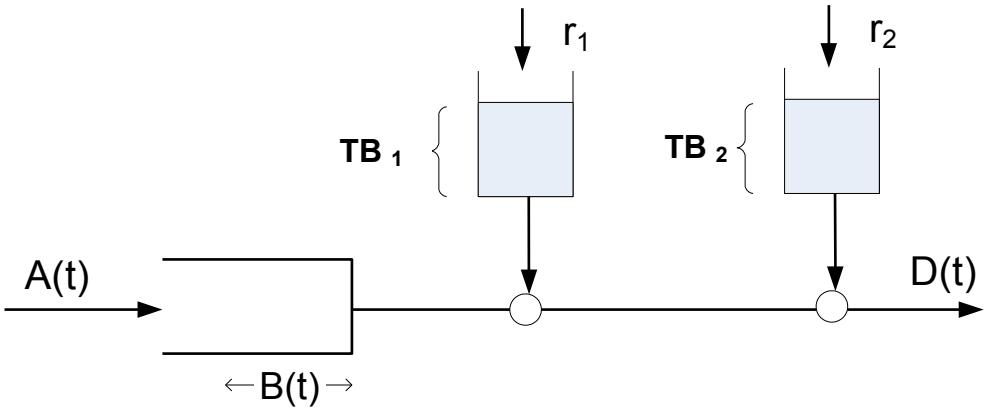
- In order to reduce the long-term resource consumption of the VBR video source, one would like to set the token bucket rate parameter to a value that is close to the average rate of the VBR source.

To satisfy both constraints, one token bucket is clearly not sufficient. A possible (and realized) solution to this problem is to regulate VBR video traffic by two token buckets put in series:

- The first token bucket controls the peak rate, by spacing out the transmission of a frame over a longer duration;

- The second token bucket controls the average rate of the VBR source.

An illustration of a dual token bucket is shown in the figure below. An arriving packet must withdraw tokens from both token buckets. If one of the two buckets does not have enough tokens, the packet must wait until sufficient tokens are available in both buckets.




### Exercise 4.1 Dual Token Bucket Traffic Shaper

Revise the Token Bucket implementation to build a traffic shaper that realizes a dual token bucket. The token bucket has four parameters ($TB_1$, $r_1$, $TB_2$, $r_2$), for the size and rate of the token buckets. As before, the Token Bucket receives data from a traffic generator and transmits the output to a traffic sink.

- Build a traffic generator (of your choice) that can be used to demonstrate the correctness and accuracy of your implementation.

- Provide plots that illustrate how you tested the implementation.

## Exercise 4.2 Selecting dual token bucket parameters for a VBR video source

Your task is to control the video trace with your dual-token bucket implementation. Transmitted packets are not permitted to exceed 1480 Bytes. So, some frames must be divided up into multiple packets.

Your task is to make a `good' selection for the parameters of the dual leaky. The selection of parameters requires you to tradeoff multiple considerations:

1. The maximum buffer size (and the maximum waiting time) should not be too large;

2. The average rate allocation should not be much larger than the average rate of the VBR source (to avoid over-allocation of bandwidth);

3. The number of packets that can be transmitted at once (this is determined by $\min(TB_1, TB_2)$), should be small.

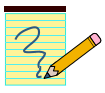
- Transmit the video source from the traffic generator using the data in file: http://www.comm.utoronto.ca/~jorg/teaching/ece466/labs/lab1/movietrace.data.

- Prepare a plot that shows the differences between the trace file and the output file, as a function of time. Prepare plots that depict the size of the token bucket TB(t) and the backlog in the buffer B(t) as a function of time.

    - What percentage of time is there a backlog at the token buckets?

    - What is the longest backlog at the token buckets?

- Prepare a plot that shows the differences between the trace file and the output file, as a function of time. Also provide a plot that shows the backlog in the buffer B(t) as a function of time. Mark the longest backlog in the buffer and the maximum waiting time.



**Lab Report:**

- Describe the design of your implementation for the Dual Token Bucket. Include your source code in the lab report. Include and discuss how you have tested the implementation.

- Discuss your method for selecting the token bucket parameters. Use plots to justify your choices.

- Provide the plots and the answers performed in Exercise 4.2.

## Feedback Form for Lab 2b

- Complete this feedback form at the completion of the lab exercises and submit the form when submitting your lab report.
- The feedback is anonymous. **Do not put your name on this form** and keep it separate from your lab report.
- For each exercise, please record the following:

| | **Difficulty (-2,-1,0,1,2)**<br>-2 = too easy<br>0 = just fine<br>2 = too hard | **Interest Level (-2,-1,0,1,2)**<br>-2 = low interest<br>0 = just fine<br>2 = high interest | **Time to complete (minutes)** |
|---|---|---|---|
| Part 1. A Reference Traffic Generator | | | |
| Part 2. Token Bucket Implementation | | | |
| Part 3. Engineering Token Bucket Parameters | | | |
| Part 3. Shaping VBR video with Multiple Token Buckets | | | |

Please answer the following questions:

- What did you like about this lab?
- What did you dislike about this lab?
- Make a suggestion to improve the lab.