CSC - I4722: HIGH PERF NETWORKS

Professor Kaliappa Ravindran

Spring 2012

Programming Assignment Report

by

Jayati Das

Date: May 20, 2012

Purpose:

The purpose of this programming assignment is to learn various traffic shaping techniques, specially the leaky bucket algorithm, and understand and compare the results of these shaping methods applied on a continuous data feed.

Specification:

The goal of this project is to simulate a system that receives various sized video data packet at a continuous frame rate. Then shape this received data using following algorithms.

1. Leaky Bucket algorithm
2. (r,T) Smooth algorithm
3. Token Bucket algorithm.

And compare the results of these shaping algorithm by plotting various parameter such as Delay vs. Leak Rate, Brustyness Vs. Leak rate etc. The shaping should such a way that following goal is accomplished.

1. The shaping scheme should work with wide range of traffic behavior
2. Shaping rule should make the traffic descriptions more concise, accurate and predictable.
3. Shaping should be such that the traffic is easy to be policed. “Outlier” should be easy to detect.

Leaky Bucket algorithm:

The Leaky Bucket Algorithm is based on an analogy of a bucket (figure 1) that has a hole in the bottom through which any water it contains will leak away at a constant rate, until or unless it is empty. Water can be added intermittently, i.e. in bursts, but if too much is added at once, or it is added at too high an average rate, the water will exceed the capacity of the bucket, which will overflow.[1]

For leaky bucket experiment I took first 10,000 frame of the video file. In every 33 milliseconds a new frame is added to the bucket. I run the simulation with various leak rates and observe the. We as average input rate is 5,295, we start with the leak rate of 6,000 then increase the leak rate by 100 and run the simulation up to 9000.

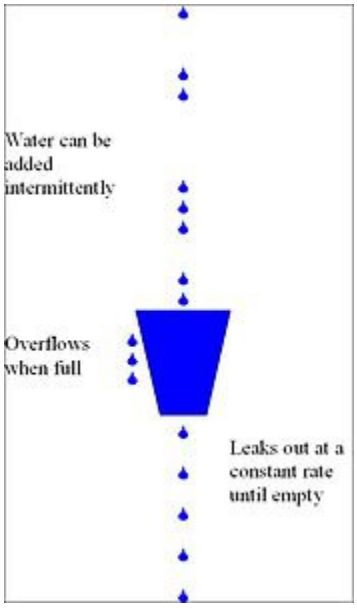


Figure 1

[1] http://en.wikipedia.org/wiki/Leaky_bucket

Input data has following specification. For better view I only graph first 1000 frames.

--- Input ---
Input Average Packet Size: 5295.77
Input Peak Packet Size: 41667
Input Brust Rate: 0.87

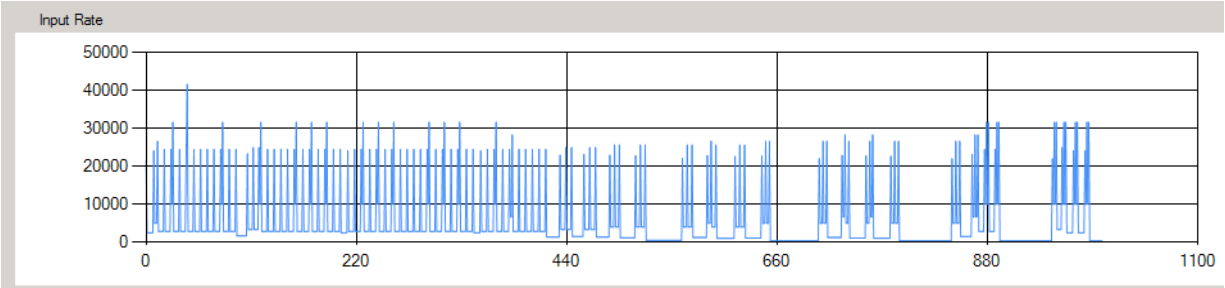


Figure 2: Input data

At leak rate 6000 the output data has following specification Leak rate: 6000

--- Output ---
Output Average Packet Size: 5290.04761904762
Output Peak Packet Size: 6000
Output Brust Rate: 0.118325396825397
Average Delay: 805.629173989455

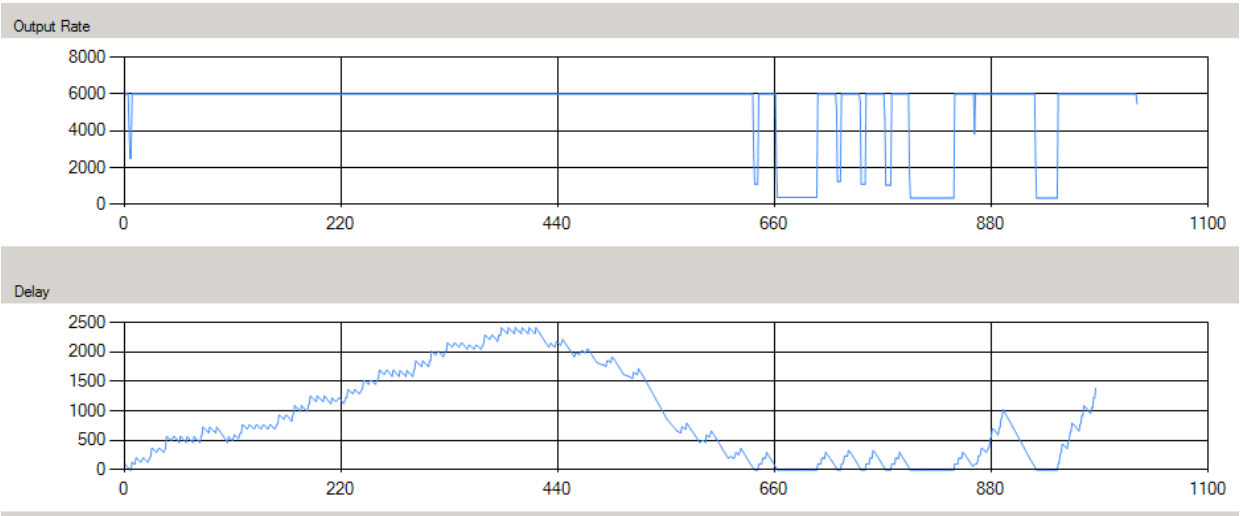


Figure 3: Leak Rate 6000

At leak rate 7000 the output data has following specification Leak rate: 6000

--- Output ---
Output Average Packet Size: 5347.20923379175
Output Peak Packet Size: 7000
Output Brust Rate: 0.236112966601179
Average Delay: 137.792270531401

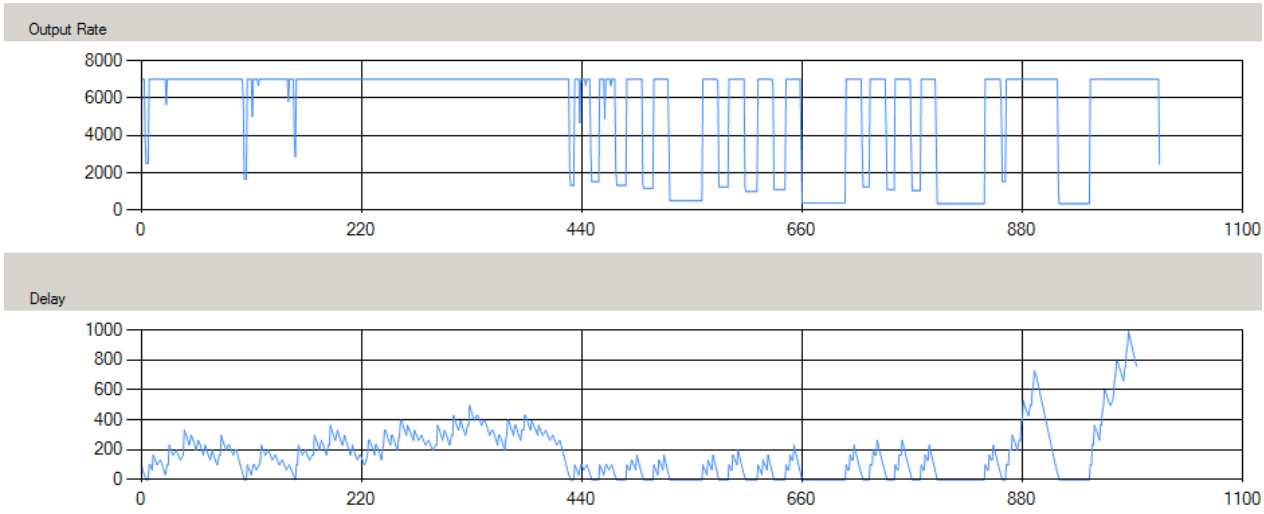


Figure 3: Leak Rate 7000

At leak rate 9000 the output data has following specification Leak rate: 6000

--- Output ---
Output Average Packet Size: 5432.59381237525
Output Peak Packet Size: 9000
Output Brust Rate: 0.396378465291639
Average Delay: 38.4549180327869

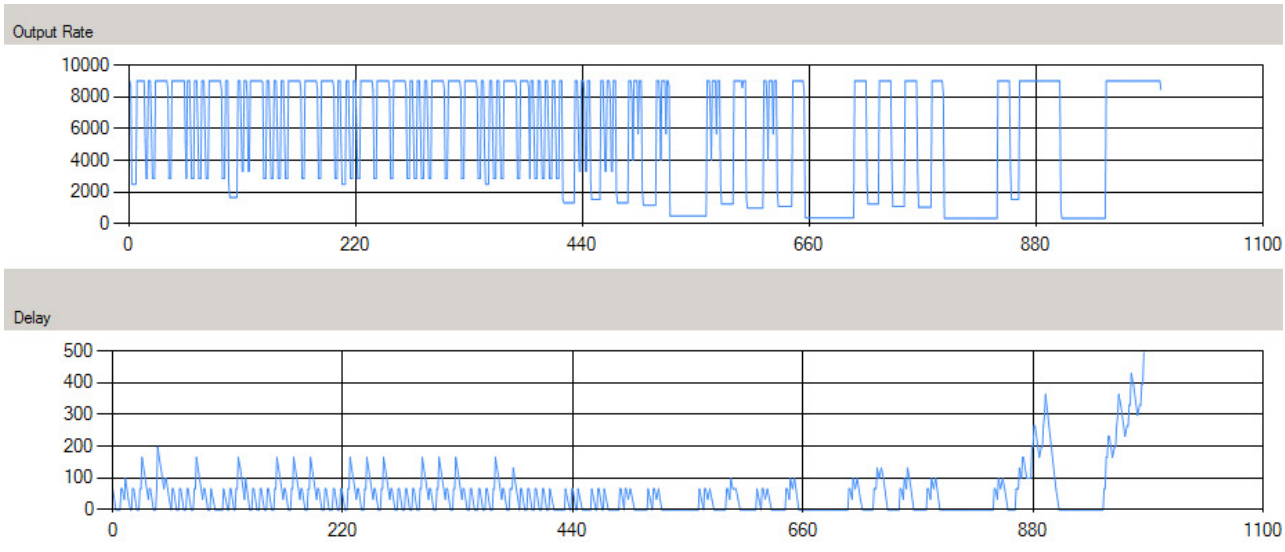
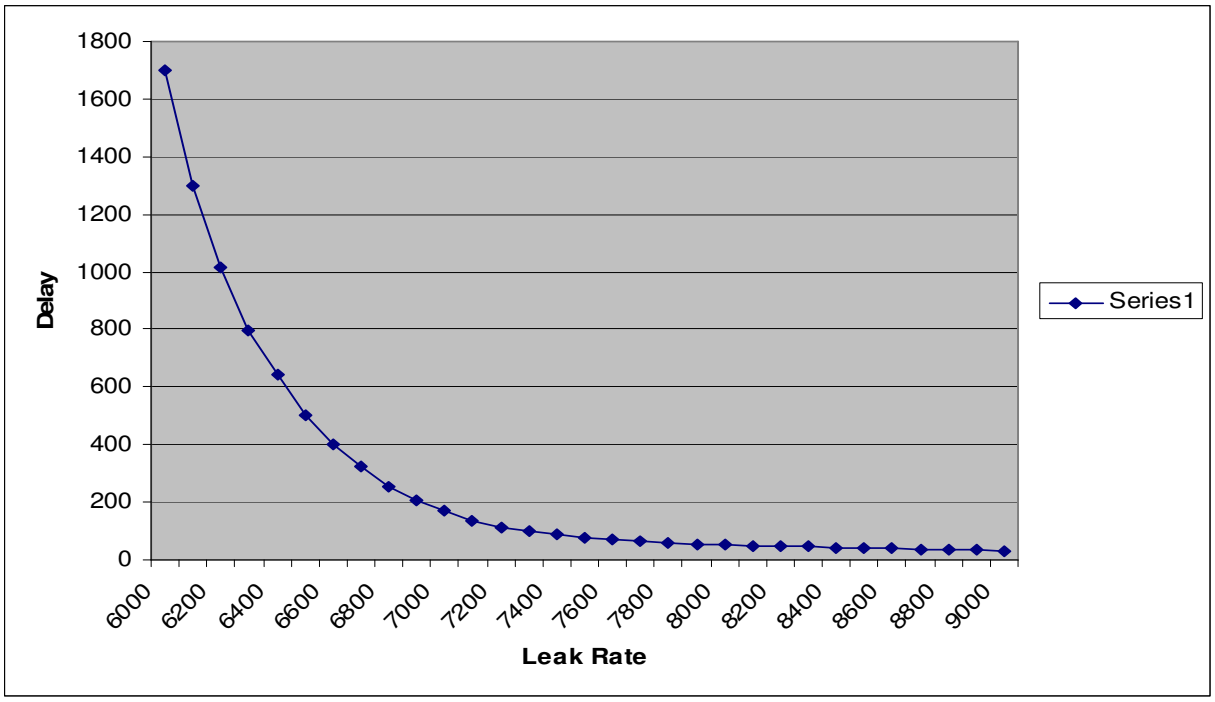


Figure 3: Leak Rate 9000

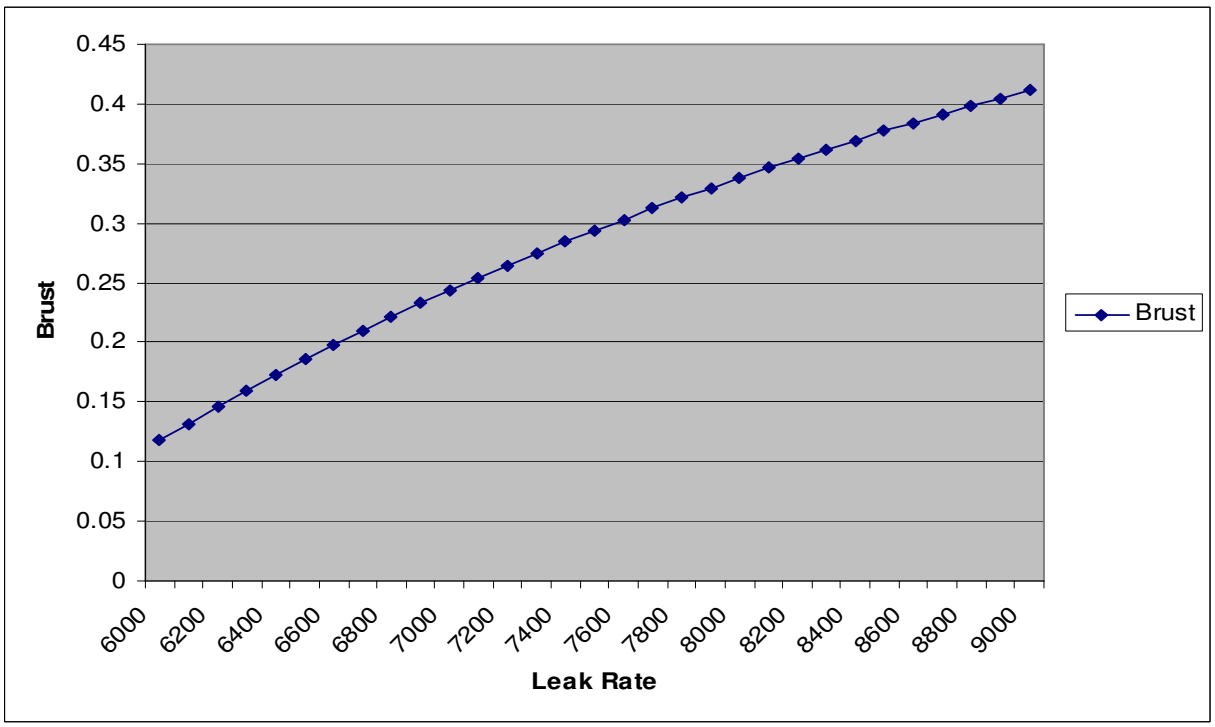
Full data table:

Leak Rate	Delay	Brust
6000	1697.96	0.11737075
6100	1300.67	0.131840082
6200	1016.1	0.145842661
6300	798.27	0.159400714
6400	640.52	0.172535078
6500	499.67	0.185265308
6600	400.64	0.197609773
6700	322.49	0.209585746
6800	255.67	0.221209485
6900	204.68	0.232496304
7000	169.82	0.243460643
7100	136.46	0.254116127
7200	113.34	0.264475625
7300	98.93	0.274551301
7400	88.08	0.284354662
7500	78.22	0.2938966
7600	71.25	0.303187434
7700	64.61	0.312236948
7800	58.77	0.321054423
7900	55.92	0.329648671
8000	51.63	0.338028063
8100	49.09	0.346200556
8200	46.25	0.35417372
8300	44.37	0.361954759
8400	42.31	0.369550536
8500	40.71	0.376967588
8600	38.7	0.384212151
8700	36.41	0.391290172
8800	34.45	0.39820733
8900	33.35	0.404969045
9000	31.19	0.4115805

Leak rate Vs Delay:



Leak Rate VS Brust



From the result we observe that as leak rate increase farther from the average input rate the delay decrease rapidly. For your given data, the decrease of delay slows down at leak rate 7000. The brustiness of the data increase almost linearly throughout the entire interval.

So in order to get better compromise between delay and brustiness the optimum leak rate will be 7000/33 = 212 byte per millisecond.