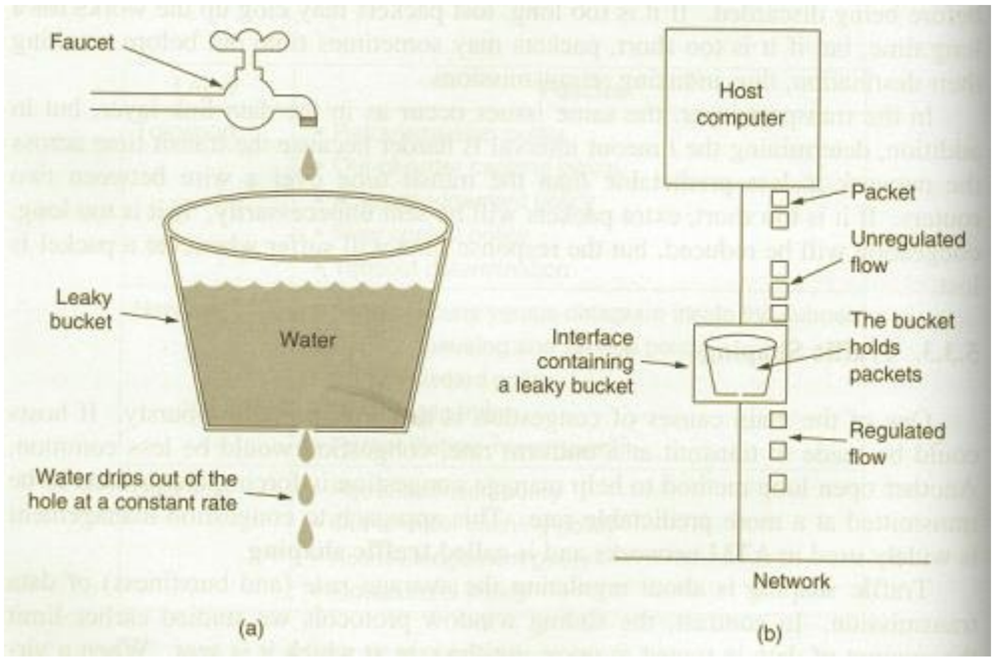
Computer Engineering 4DK4
Lab 5
Traffic Shaping Using Leaky Bucket and Token Bucket

The traffic generated by the source is often bursty, meaning that the instantaneous data arrival rate can vary over time. High burstiness is one of the main causes of network congestion. The purpose of traffic shaping is to control or reduce the burstiness of traffic from the data source so that the data flow injected into the network is less bursty and more predictable. This is typically achieved by limiting the average or maximum data rate. This lab simulates the performance of the leaky bucket and token bucket algorithms for traffic shaping.

1. Preparation

Imagine a bucket with a small hole in the bottom as shown in figure (a) below. No matter at what rate water enters the bucket, the outflow is at a constant rate when there is any water in the bucket, and zero when the bucket is empty. Also, once the bucket is full, any additional water entering it spills over the sides and is lost (does not appear in the output stream under the hole).



The same idea can be applied to packets. As shown in figure (b), the leaky bucket controller is between the host computer and the network. That is, its input to the leaky bucket controller is the host generated data traffic and the output of the leaky bucket controller is connected to the network. The leaky bucket controller can be simply considered as a single-server queueing system with a finite size queue (the "bucket" that holds packets) and a constant service rate. The service rate is the data transmission speed of the link connecting the output of the leaky bucket and the network. Upon the generation of a packet, if the queue at the leaky bucket controller is full, the packet is discarded or lost. This arrangement can be built into the hardware interface or simulated by the host operating system. It was first proposed by Turner (1986) and is called the leaky bucket algorithm.

2. Experiments

1. The original leaky bucket:

For the original leaky bucket, the host is allowed to put one packet per clock tick onto the network. That is, the clock ticks every X seconds with X the (average) packet transmission time. When a packet arrives, if there is enough room on the queue (or the "bucket"), it is appended to the queue; otherwise, it is discarded. The leaky bucket outputs the data at the rate of one packet per clock tick. That is, at every clock tick, one packet is transmitted (unless the queue is empty).

Now use discrete event simulation to simulate a leaky bucket controller that has a queue size of B packets and output data at a rate R=1000 packets per second, i.e., X =1ms, the clock ticks every 1ms, and the average the packet transmission time is 1ms. Traffic arrives to the leaky bucket input according to a Poisson process with an average rate of 100 packets per second.

a) Plot the packet loss rate (due to buffer overflow at the controller) and the mean output data rate (in number of packets) of the controller versus the queue size B (in number of packets).

b) Fix B=3 packets and vary R. Plot the packet loss rate and the mean output data rate of the controller versus R (in packets per second).

Describe your observations and explain them.

2. The bit-counting (or byte-counting) leaky bucket:
Different from the original leaky bucket that uses packets as the basic data units, the bit-counting leaky bucket uses bits as the basic units. At each clock tick, a counter is initialized to n. If the first packet on the queue has fewer bits than the current value of the counter, it is transmitted, and the counter is decremented by that number of bits. Additional packets may also be sent, as long as the counter value is high enough. When the counter drops below the length of the next packet on the queue, transmission stops until the next tick, at which time the residual bit count is overwritten and lost and the counter is initialized to n again.

Use discrete event simulation to simulate a bit-counting leaky bucket controller that has an output data rate of R=1M bps. Traffic arrives to the leaky bucket according to a Poisson process with an average rate of 100 packets per second. The packet size can take [500, 1k, 1.5k, 2k, 2.5k] bits with equal probability.

Design your own plots to study the packet loss rate and average output data rates (in bps) versus the clock ticking period, n, and R (in bps). Justify your choice of the clock ticking period. Describe your observations and explain them.

3. The token bucket algorithm:
Different from the leaky bucket, a token bucket controller uses two queues (or two buckets), one holding tokens and another holding data packets. A token bucket can hold at most $B_{\mathrm{t,max}}$ tokens, and a data bucket can hold at most $B_{\mathrm{d,max}}$ packets. Tokens are generated by a clock at the rate of one token every ΔT seconds. The generated tokens are added to the token bucket if the bucket is not full or lost otherwise. Packets generated at the source are stored at the packet bucket if the bucket is not full or lost otherwise. A packet in the data bucket can be transmitted as long as there is a token available; meanwhile, it consumes one token (the number of tokens is reduced by 1). If the token bucket is empty, no packets can be transmitted.

a) Assume all the packets are in the same size, design your own experiments to study the effect of $B_{\mathrm{t,max}}$ and $B_{\mathrm{d,max}}$ on the packet loss rate and the average output data rate.
b) Consider modifying the above token bucket algorithm so that the data transmission consumes tokens based on the number of bits transmitted not the number of packets transmitted, and then demonstrate its performance.

4. Compare the pros and cons of the above algorithms in a short paragraph (3~5 sentences).

Submission

Same as in previous labs except you should submit all your code in a zipped file.