# Lab No. 5: Application Development using Socket Programming

## Objectives

* To apply the socket programming concepts in developing the real-world applications
* communication using Python)

In the network layer, before the network can make Quality of service guarantees, it must know what traffic is being guaranteed. One of the main causes of congestion is that traffic is often bursty. The congesting control algorithms are basically divided into two groups: open loop and closed loop. Open loop solutions attempt to solve the problem by good design, in essence, to make sure it does not occur in the first place. Once the system is up and running, midcourse corrections are not made. Open loop algorithms are further divided into ones that act at source versus ones that act at the destination.

In contrast, closed loop solutions are based on the concept of a feedback loop if there is any congestion. Closed loop algorithms are also divided into two subcategories: explicit feedback and implicit feedback. In explicit feedback algorithms, packets are sent back from the point of congestion to warn the source. In implicit algorithm, the source deduces the existence of congestion by making local observation, such as the time needed for acknowledgment to come back.

The presence of congestion means that the load is (temporarily) greater than the resources (in part of the system) can handle. For subnets that use virtual circuits internally, these methods can be used at the network layer. Another open loop method to help manage congestion is forcing the packet to be transmitted at a more predictable rate. This approach to congestion management is widely used in ATM networks and is called traffic shaping.

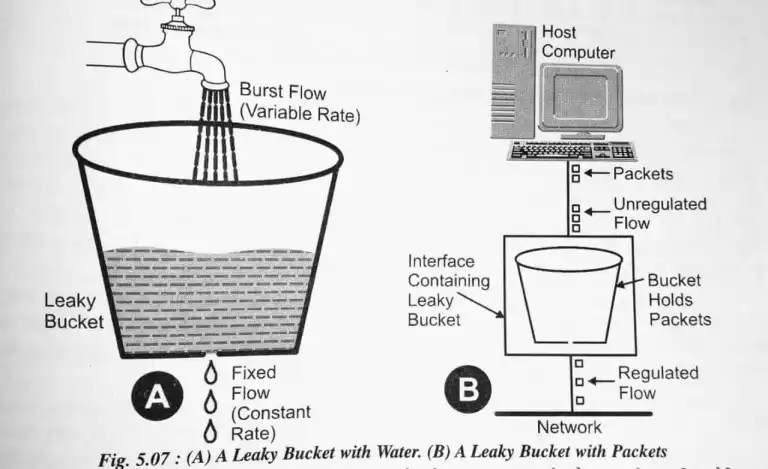
To understand this concept first we have to know little about traffic shaping. Traffic Shaping is a mechanism to control the amount and the rate of traffic sent to the network. Approach of congestion management is called Traffic shaping. Traffic shaping helps to regulate the rate of data transmission and reduces congestion.

There are 2 types of traffic shaping algorithms:

* Leaky Bucket
* Token Bucket

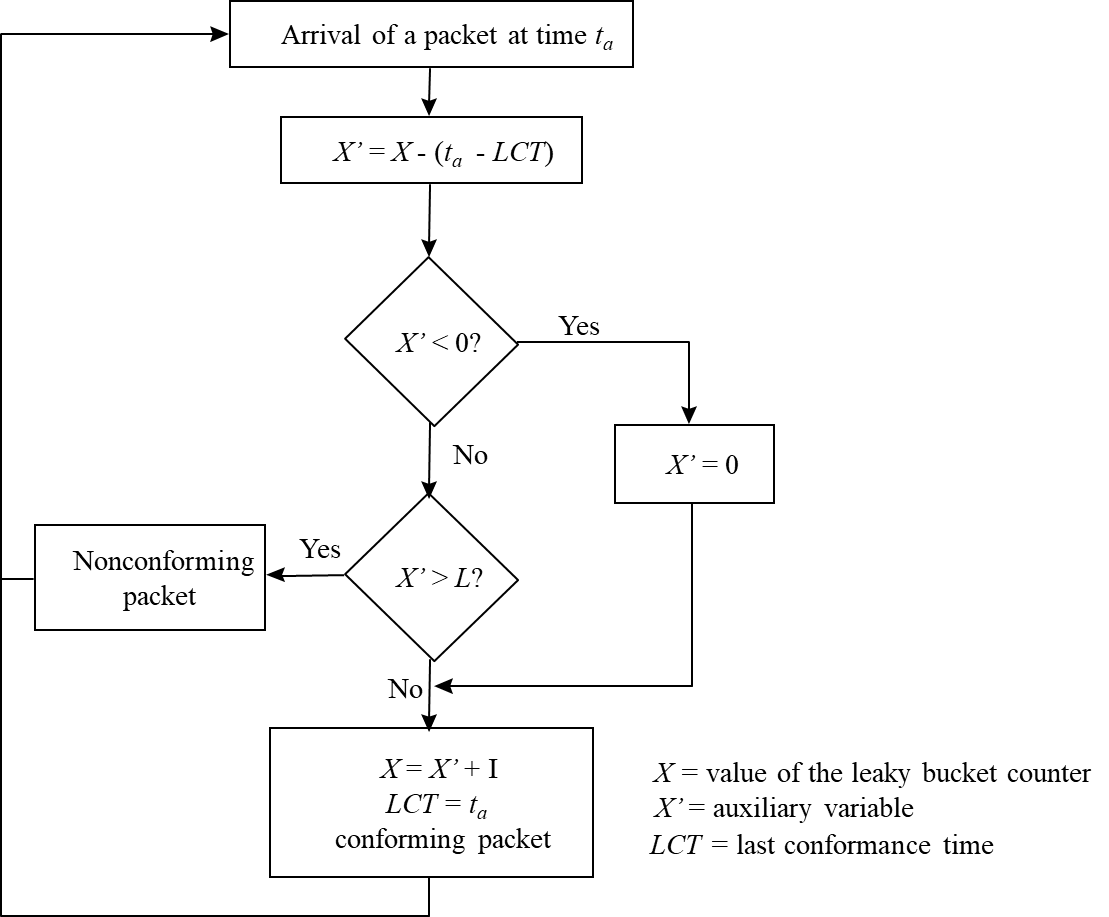
**Leaky Bucket**

Suppose we have a bucket in which we are pouring water, at random points in time, but we have to get water at a fixed rate, to achieve this we will make a hole at the bottom of the bucket. This will ensure that the water coming out is at some fixed rate, and also if the bucket gets full, then we will stop pouring water into it. The input rate can vary, but the output rate remains constant. Similarly, in networking, a technique called leaky bucket can smooth out bursty traffic. Bursty chunks are stored in the bucket and sent out at an average rate.



Each host is connected to the network by an interface containing a leaky bucket, that is, a finite internal queue. If a packet arrives at the queue when it is full, the packet is discarded. In other words, if one or more process are already queued, the new packet is unceremoniously discarded. This arrangement can be built into the hardware interface or simulate d by the host operating system. In fact it is nothing other than a single server queuing system with constant service time.

The host is allowed to put one packet per clock tick onto the network. This mechanism turns an uneven flow of packet from the user process inside the host into an even flow of packet onto the network, smoothing out bursts and greatly reducing the chances of congestion.



A graph of a graph and a chart of a graph

Description automatically generated with medium confidence

**Algorithm:**

1. Start the program
2. Set the bucket size or the buffer size.
3. Set the output rate.
4. Transmit the packets such that there is no overflow.
5. Repeat the process of transmission until all packets are transmitted. (Reject packets where its size is greater than the bucket size)
6. Stop

**Token Bucket**

The leaky bucket algorithm enforces output patterns at the average rate, no matter how busy the traffic is. So, to deal with more traffic, we need a flexible algorithm so that the data is not lost. One such approach is the token bucket algorithm. When compared to the Leaky bucket the token bucket algorithm is less restrictive which means it allows more traffic. The limit of busyness is restricted by the number of tokens available in the bucket at a particular instant of time.

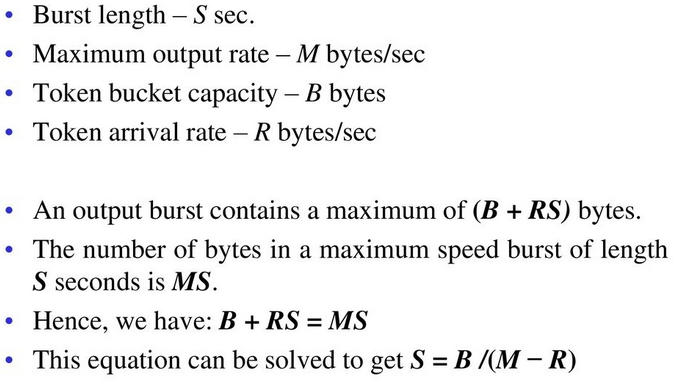
In some applications, when large bursts arrive, the output is allowed to speed up. This calls for a more flexible algorithm, preferably one that never loses information. Therefore, a token bucket algorithm finds its uses in network traffic shaping or rate-limiting. It is a control algorithm that indicates when traffic should be sent. This order comes based on the display of tokens in the bucket. The bucket contains tokens. Each of the tokens defines a packet of predetermined size. Tokens in the bucket are deleted for the ability to share a packet. When tokens are shown, a flow to transmit traffic appears in the display of tokens. No token means no flow sends its packets. Hence, a flow transfers traffic up to its peak burst rate in good tokens in the bucket.

The implementation of the token bucket algorithm is easy − a variable is used to count the tokens. For every t seconds the counter is incremented and then it is decremented whenever a packet is sent. When the counter reaches zero, no further packet is sent out.

A diagram of a machine

Description automatically generatedA diagram of a block diagram

Description automatically generated



**Algorithm:**

1. Start the program
2. In regular intervals, tokens are thrown into the bucket f.
3. The bucket has a maximum capacity f.
4. If the packet is ready, then a token is removed from the bucket, and the packet is sent.
5. Suppose, if there is no token in the bucket, the packet cannot be sent.
6. Stop

## Lab Exercises

1. Write a C program to avoid the congestion in network by using the leaky bucket method. Suppose the network packets arrive at a leaky bucket policer at time interval t = 1, 2, 3, 5, 6, 8, 11, 12, 15, 16, and 19 seconds with a packet size of 4 bytes each. Assume that the leaky bucket size X= 10 (queue size) bytes and the outgoing rate is one byte for every second. Identify the conforming and nonconforming packets in the queue and display the process on the screen.
2. Write a C program to avoid the congestion in network by using the Token bucket method. Assume that we have a token bucket shaper that has a replenishment rate r = 10 KBps, an infinite maximum rate R, a bucket size b = 50 Kbytes and that the bucket starts off full. Also assume that a sender emits 15 Kbytes packet size every 0.5 seconds in a periodic manner, starting at t = 0.5 seconds. For this question, you can assume that if sufficient tokens are available, packets pass through the token bucket instantaneously, otherwise they are queued until there are.
3. How many tokens are left in the bucket after 1.5 seconds?
4. How long will it take until packets start to be queued or dropped?
5. Now, presume the sender can send as much as they want, whenever they want. If the token bucket is changed to enforce a maximum rate R of 20 KBps, what would the maximum possible burst size be?
6. (1e), Packt Publishing 2015