Week 3

Implement the congestion control using Leaky bucket algorithm

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.

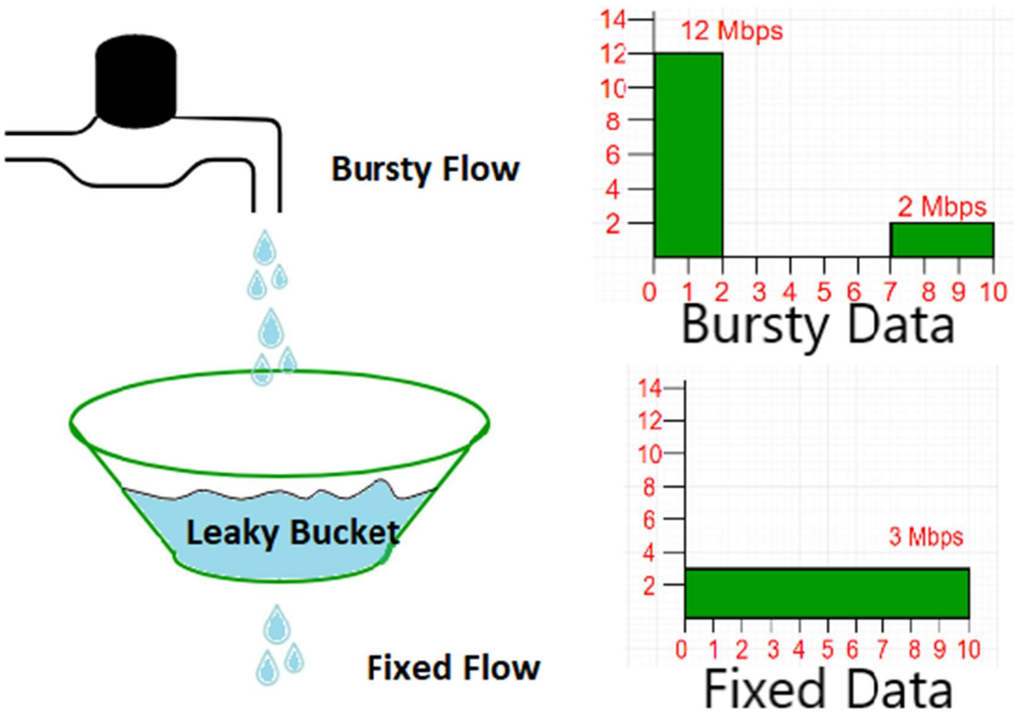
To understand this concept first we have to know little about traffic shaping. **Traffic Shaping** is a mechanism to control the amount and the rate of traffic sent to the network. Approach of congestion management is called Traffic shaping.

Traffic shaping helps to regulate the rate of data transmission and reduces congestion.
There are 2 types of traffic shaping algorithms:

1. Leaky Bucket
2. Token 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.



In the above figure, we assume that the network has committed a bandwidth of 3 Mbps for a host. The use of the leaky bucket shapes the input traffic to make it conform to this commitment. In the above figure, the host sends a burst of data at a rate of 12 Mbps for 2s, for a total of 24 Mbits of data. The host is silent for 5 s and then sends data at a rate of 2 Mbps for 3 s, for a total of 6 Mbits of data. In all, the host has sent 30 Mbits of data in 10 s. The leaky bucket smooths out the traffic by sending out data at a rate of 3 Mbps during the same 10 s.

Without the leaky bucket, the beginning burst may have hurt the network by consuming more bandwidth than is set aside for this host. We can also see that the leaky bucket may prevent congestion.

A simple leaky bucket algorithm can be implemented using FIFO queue. A FIFO queue holds the packets. If the traffic consists of fixed-size packets (e.g., cells in ATM networks), the process removes a fixed number of packets from the queue at each tick of the clock. If the traffic consists of variable-length packets, the fixed output rate must be based on the number of bytes or bits.

The following is an algorithm for variable-length packets:

1. Initialize a counter to n at the tick of the clock.
2. Repeat until n is smaller than the packet size of the packet at the head of the queue.
    1. Pop a packet out of the head of the queue, say P.
    2. Send the packet P, into the network
    3. Decrement the counter by the size of packet P.
3. Reset the counter and go to step 1.

Example: Let n=1000
Packet=

200	700	500	450	400	200

Since n > size of the packet at the head of the Queue, i.e. n > 200
Therefore, n = 1000-200 = 800
Packet size of 200 is sent into the network.

200	700	500	450	400

Now, again n > size of the packet at the head of the Queue, i.e. n > 400
Therefore, n = 800-400 = 400
Packet size of 400 is sent into the network.

200	700	500	450

Since, n < size of the packet at the head of the Queue, i.e. n < 450
Therefore, the procedure is stopped.

Initialise n = 1000 on another tick of the clock.
This procedure is repeated until all the packets are sent into the network.