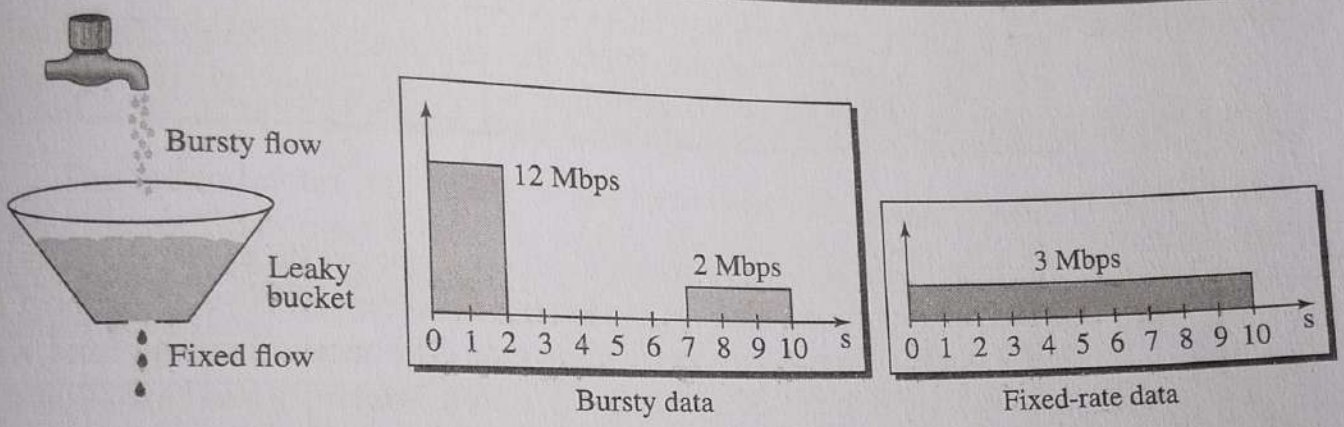
30.2.2 Traffic Shaping or Policing

To control the amount and the rate of traffic is called *traffic shaping* or *traffic policing*. The first term is used when the traffic leaves a network; the second term is used when the data enters the network. Two techniques can shape or police the traffic: leaky bucket and token bucket.

Leaky Bucket

If a bucket has a small hole at the bottom, the water leaks from the bucket at a constant rate as long as there is water in the bucket. The rate at which the water leaks does not depend on the rate at which the water is input unless the bucket is empty. If the bucket is full, the water overflows. 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. Figure 30.4 shows a leaky bucket and its effects.

Figure 30.4 *Leaky bucket*

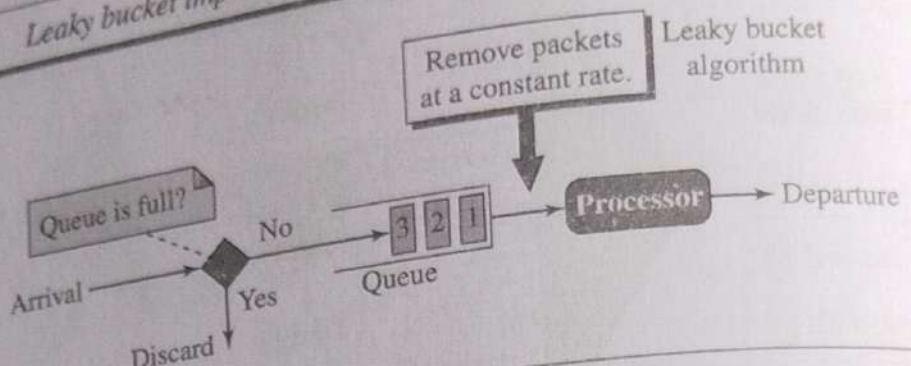


In the figure, we assume that the network has committed a bandwidth of 3 Mbps for a host. The use of the leaky bucket shapes the input traffic to make it conform to this commitment. In Figure 30.4 the host sends a burst of data at a rate of 12 Mbps for 2 seconds, for a total of 24 Mb of data. The host is silent for 5 seconds and then sends data at a rate of 2 Mbps for 3 seconds, for a total of 6 Mb of data. In all, the host has sent 30 Mb of data in 10 seconds. The leaky bucket smooths the traffic by sending out data at a rate of 3 Mbps during the same 10 seconds. 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 implementation is shown in Figure 30.5. 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.

Figure 30.5 *Leaky bucket implementation*



The following is an algorithm for variable-length packets:

1. Initialize a counter to n at the tick of the clock.
2. If n is greater than the size of the packet, send the packet and decrement the counter by the packet size. Repeat this step until the counter value is smaller than the packet size.
3. Reset the counter to n and go to step 1.

A leaky bucket algorithm shapes bursty traffic into fixed-rate traffic by averaging the data rate. It may drop the packets if the bucket is full.

Token Bucket

The leaky bucket is very restrictive. It does not credit an idle host. For example, if a host is not sending for a while, its bucket becomes empty. Now if the host has bursty data, the leaky bucket allows only an average rate. The time when the host was idle is not taken into account. On the other hand, the **token bucket** algorithm allows idle hosts to accumulate credit for the future in the form of tokens.

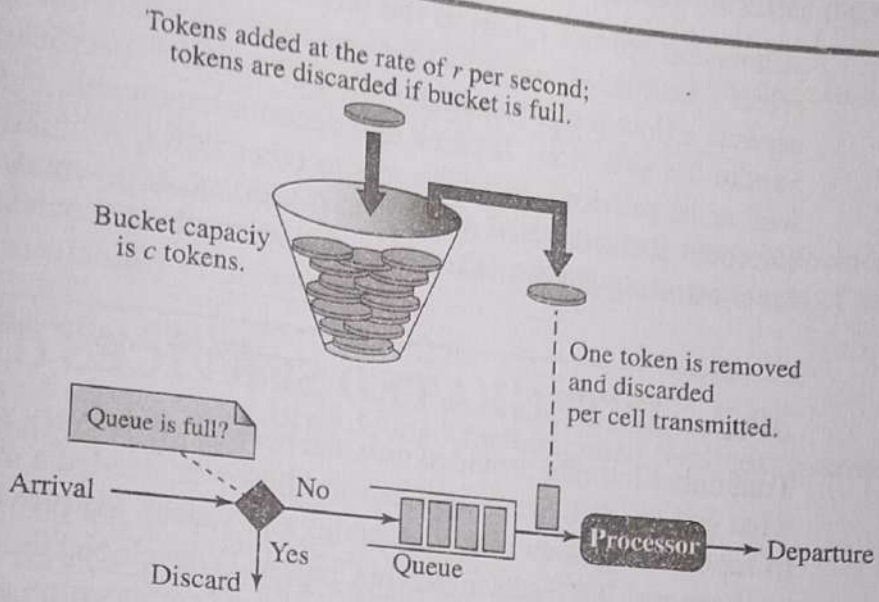
Assume the capacity of the bucket is c tokens and tokens enter the bucket at the rate of r tokens per second. The system removes one token for every cell of data sent. The maximum number of cells that can enter the network during any time interval of length t is shown below.

$$\textbf{Maximum number of packets} = r \times t + c$$

The maximum average rate for the token bucket is shown below.

$$\textbf{Maximum average rate} = (r \times t + c)/t \textbf{ packets per second}$$

This means that the token bucket limits the average packet rate to the network. Figure 30.6 shows the idea.



Example 30.2

Let's assume that the bucket capacity is 10,000 tokens and tokens are added at the rate of 1000 tokens per second. If the system is idle for 10 seconds (or more), the bucket collects 10,000 tokens and becomes full. Any additional tokens will be discarded. The maximum average rate is shown below.

$$\text{Maximum average rate} = (1000t + 10{,}000)/t$$

The token bucket can easily be implemented with a counter. The counter is initialized to zero. Each time a token is added, the counter is incremented by 1. Each time a unit of data is sent, the counter is decremented by 1. When the counter is zero, the host cannot send data.

The token bucket allows bursty traffic at a regulated maximum rate.

Combining Token Bucket and Leaky Bucket

The two techniques can be combined to credit an idle host and at the same time regulate the traffic. The leaky bucket is applied after the token bucket; the rate of the leaky bucket needs to be higher than the rate of tokens dropped in the bucket.