**Experiment 8:**

# Aim: write a program for congestion control using leaky bucket algorithm

In computer networks, congestion occurs when data traffic exceeds the available bandwidth and leads to packet loss, delays, and reduced performance. Traffic shaping can prevent and reduce congestion in a network. It is a technique used to regulate data flow by controlling the rate at which packets are sent into the network.

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.

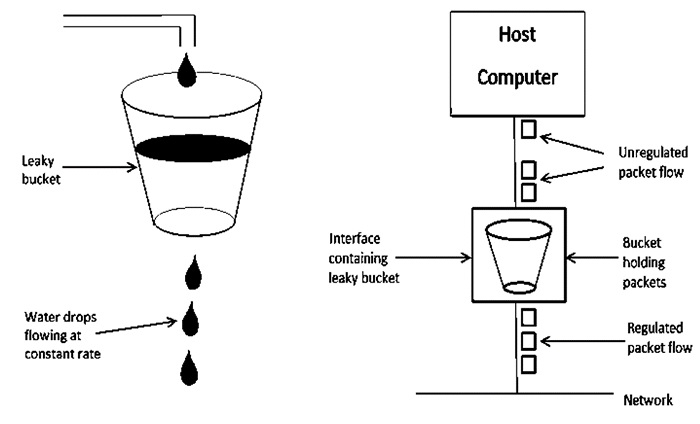
**Leaky Bucket algorithm:**

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, 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.



**ALGORITHM:**

Step 1: Start

Step 2: Set the bucket size or the buffer size

Step3: Set the output rate.

Step 4: Transmit the packets such that there is no overflow.

Step 5: Repeat the process of transmission until all packets are transmitted. (Reject packets where its size is greater than the bucket size)

Step 6: Stop

**Program:**

#include <stdio.h>

#include <stdlib.h>

int main() {

int bucket\_size, output\_rate, n, i;

int incoming, stored = 0;

printf("Enter the bucket size (in packets): ");

scanf("%d", &bucket\_size);

printf("Enter the output rate (packets per second): ");

scanf("%d", &output\_rate);

printf("Enter the number of incoming packet requests: ");

scanf("%d", &n);

for (i = 0; i < n; i++) {

printf("\nEnter the size of incoming packet %d (in packets): ", i+1);

scanf("%d", &incoming);

printf("Incoming packet size = %d\n", incoming);

// Check if packets can be stored

if (incoming + stored > bucket\_size) {

int dropped = (incoming + stored) - bucket\_size;

printf("Bucket overflow! Dropped %d packets\n", dropped);

stored = bucket\_size; // bucket is full

} else {

stored += incoming;

}

// Transmit packets at output rate

printf("Packets sent: %d\n", (stored < output\_rate) ? stored : output\_rate);

// Reduce stored packets after sending

if (stored < output\_rate)

stored = 0;

else

stored -= output\_rate;

printf("Packets remaining in bucket: %d\n", stored);

}

// Empty remaining packets in the bucket

while (stored > 0) {

printf("\nPackets sent: %d\n", (stored < output\_rate) ? stored : output\_rate);

if (stored < output\_rate)

stored = 0;

else

stored -= output\_rate;

printf("Packets remaining in bucket: %d\n", stored);

}

return 0;

}

**OUTPUT:**

Enter the bucket size (in packets): 10

Enter the output rate (packets per second): 3

Enter the number of incoming packet requests: 4

Enter the size of incoming packet 1 (in packets): 4

Incoming packet size = 4

Packets sent: 3

Packets remaining in bucket: 1

Enter the size of incoming packet 2 (in packets): 3

Incoming packet size = 3

Packets sent: 3

Packets remaining in bucket: 1

Enter the size of incoming packet 3 (in packets): 4

Incoming packet size = 4

Packets sent: 3

Packets remaining in bucket: 2

Enter the size of incoming packet 4 (in packets): 2

Incoming packet size = 2

Packets sent: 3

Packets remaining in bucket: 1

Packets sent: 1

Packets remaining in bucket: 0