Leaky Bucket

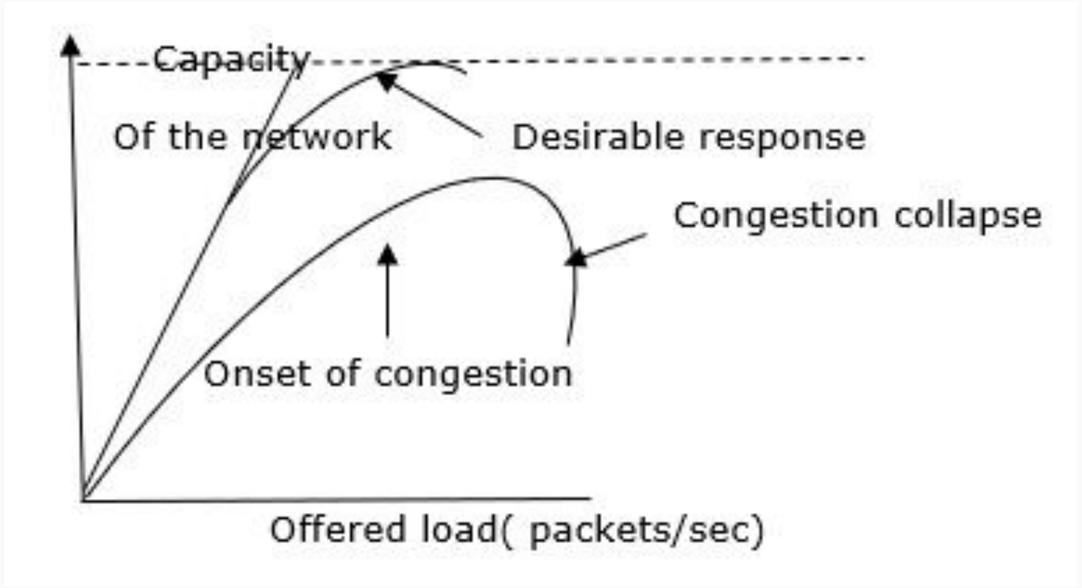
Kanishkvardhan A N:2SD19IS023
Karthik Kavathekar :2SD19IS024

What is Congestion?

When too many packets are present in the network it causes packet delay and loss of packet which degrades the performance of the system. This situation is called congestion.

How to control Congestion?

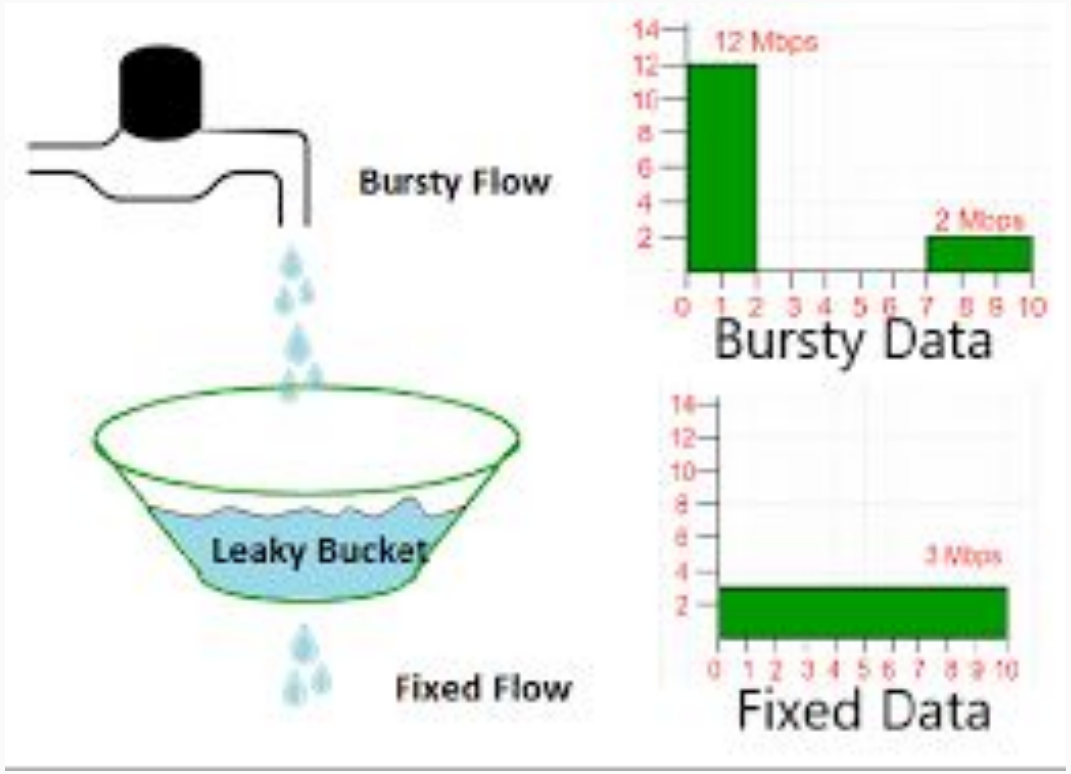
The network layer and transport layer share the responsibility for handling congestions. One of the most effective ways to control congestion is trying to reduce the load that transport layer is placing on the network. To maintain this, the network and transport layers have to work together.



There are two types of Congestion control algorithms, which are as follows −

- Leaky Bucket Algorithm
- Token Bucket Algorithm

Leaky Bucket



Leaky Bucket Algorithm mainly controls the total amount and the rate of the traffic sent to the network.

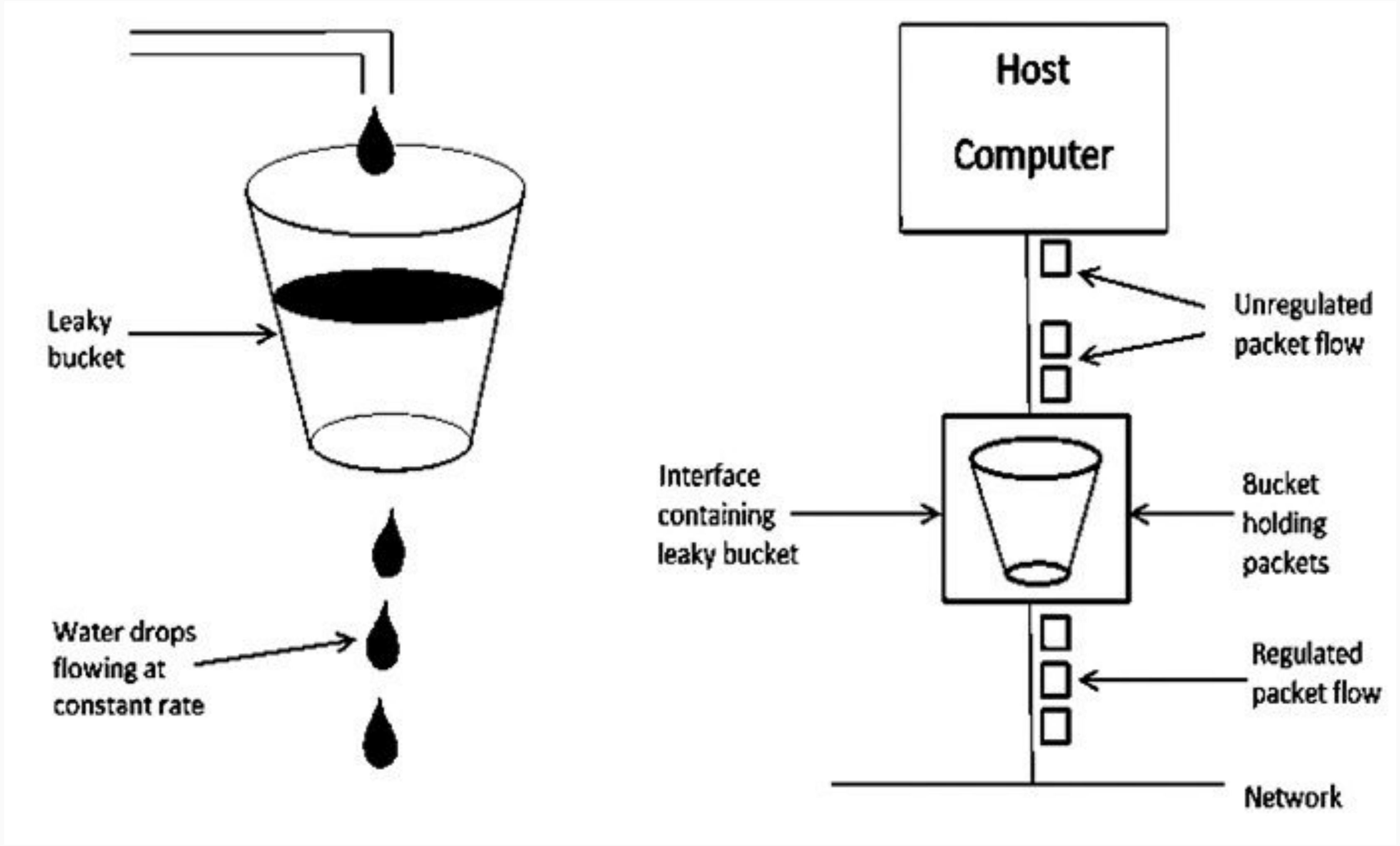
Step 1 − Let us imagine a bucket with a small hole at the bottom where the rate at which water is poured into the bucket is not constant and can vary but it leaks from the bucket at a constant rate.

Step 2 − So (up to water is present in the bucket), the rate at which the water leaks does not depend on the rate at which the water is input to the bucket.

Step 3 − If the bucket is full, additional water that enters into the bucket that spills over the sides and is lost.

Step 4 − Thus the same concept applied to packets in the network. Consider that data is coming from the source at variable speeds. Suppose that a source sends data at 10 Mbps for 4 seconds. Then there is no data for 3 seconds. The source again transmits data at a rate of 8 Mbps for 2 seconds. Thus, in a time span of 8 seconds, 68 Mb data has been transmitted.

That’s why if a leaky bucket algorithm is used, the data flow would be 8 Mbps for 9 seconds. Thus, the constant flow is maintained.


Leaky bucket
Water drops flowing at constant rate
Host Computer
Unregulated packet flow
Interface containing leaky bucket
8ucket holding packets
Regulated packet flow
Network

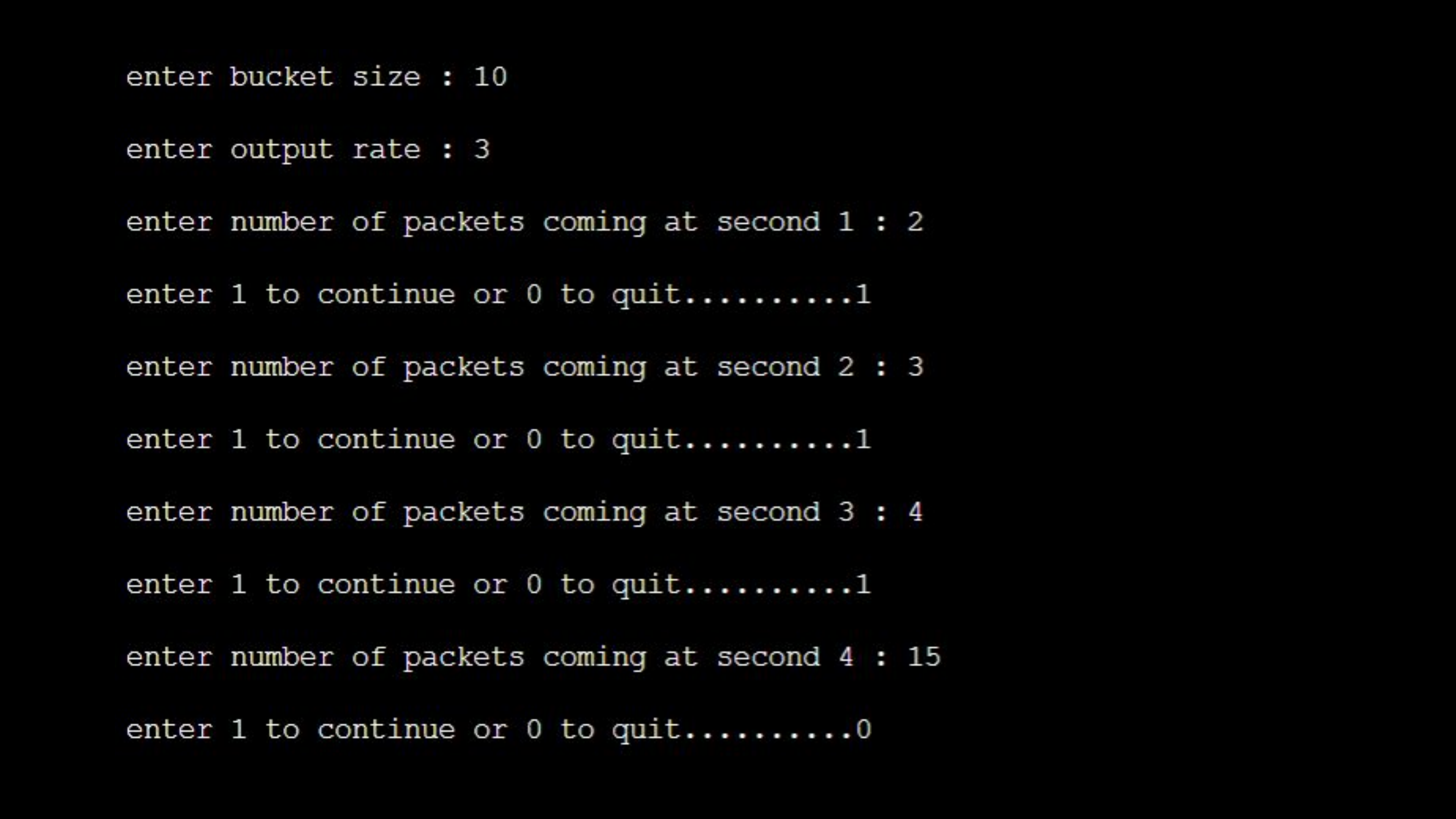
C Implementation for Leaky Bucket

```
#include<stdio.h>
#include<stdlib.h>
#define MIN(x,y) (x>y)?y:x

int main (){

int orate, drop = 0, cap, x, count = 0, inp[10] = { 0 }, i = 0, nsec, ch;

printf ("\n enter bucket size : ");
scanf ("%d", &cap);
printf ("\n enter output rate :");
scanf ("%d", &orate);
do{
    printf ("\n enter number of packets coming at second %d :", i + 1);
    scanf ("%d", &inp[i]);
    i++;
    printf ("\n enter 1 to continue or 0 to quit..........");
    scanf ("%d", &ch);
  }while (ch);
nsec = i;
printf ("\n second \t received \t sent \t dropped \t remained \n");
for (i = 0; count || i < nsec; i++){
    printf (" %d", i + 1);
    printf (" \t%d\t ", inp[i]);
    printf (" \t %d\t ", MIN ((inp[i] + count), orate));
    if ((x = inp[i] + count - orate) > 0){
        if (x > cap){
            count = cap;
            drop = x - cap;
             }
             else{
                 count = x;
                 drop = 0;
                 }
        }
    else{
        drop = 0;
        count = 0;
         }
    printf (" \t %d \t %d \n", drop, count);
}
return 0;
}
```



```
enter bucket size : 10

enter output rate : 3

enter number of packets coming at second 1 : 2

enter 1 to continue or 0 to quit..........1

enter number of packets coming at second 2 : 3

enter 1 to continue or 0 to quit..........1

enter number of packets coming at second 3 : 4

enter 1 to continue or 0 to quit..........1

enter number of packets coming at second 4 : 15

enter 1 to continue or 0 to quit..........0
```

```
second          received          sent           dropped         remained
1              2                 2              0               0
2              3                 3              0               0
3              4                 3              0               1
4              15                3              3               10
5              0                 3              0               7
6              0                 3              0               4
7              0                 3              0               1
8              0                 1              0               0
```

THANK YOU