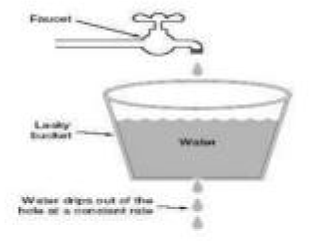
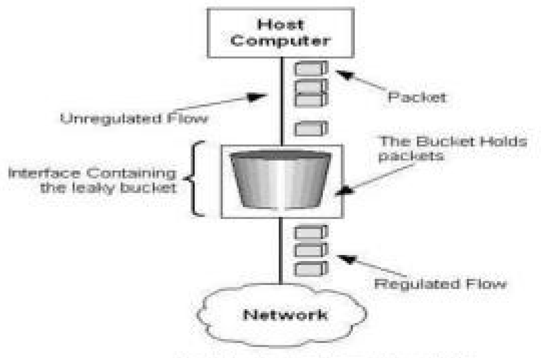
## Program- 10

Write a program for congestion control using leaky bucket algorithm.

The main concept of the leaky bucket algorithm is that the output data flow remains constant despite the variant input traffic, such as the water flow in a bucket with a small hole at the bottom. In case the bucket contains water (or packets) then the output flow follows a constant rate, while if the bucket is full any additional load will be lost because of spillover. In a similar way if the bucket is empty the output will be zero. From network perspective, leaky bucket consists of a finite queue (bucket) where all the incoming packets are stored in case there is space in the queue, otherwise the packets are discarded. In order to regulate the output flow, leaky bucket transmits one packet from the queue in a fixed time (e.g. at every clock tick). In the following figure we can notice the main rationale of leaky bucket algorithm, for both the two approaches (e.g. leaky bucket with water (a) and with packets (b)).



(a) A leaky bucket with water



(b) A leaky bucket with packets

While leaky bucket eliminates completely bursty traffic by regulating the incoming data flow its main drawback is that it drops packets if the bucket is full. Also, it doesn't take into account the idle process of the sender which means that if the host doesn't transmit data for some time the bucket becomes empty without permitting the transmission of any packet.

**Source Code**

```
import
java.util.*; public
class leaky
{
static int min(int x,int y)
{
if(x<y)
return x;
else
return y;
}
public static void main(String[] args)
{ int drop=0,mini,nsec,cap,count=0,i,process;
int inp[]=new int[25];
Scanner sc=new Scanner(System.in);
System.out.println("Enter The Bucket Size\n");
cap= sc.nextInt();
System.out.println("Enter The Operation Rate\n");
process= sc.nextInt();
System.out.println("Enter The No. Of Seconds You Want To Stimulate\n");
nsec=sc.nextInt();
for(i=0;i<nsec;i++)
{
System.out.println("Enter The Size Of The Packet Entering At 1 sec:");
inp[i] = sc.nextInt();
}
System.out.println("\nSecond | Packet Recieved | Packet Sent | Packet Left |Packet
Dropped|\n"); System.out.println("------------------------------------\n");
for(i=0;i<nsec;i++)
{
count+=inp[i];
if(count>cap)
{ drop=count-cap;
count=cap;
}
System.out.print(i+1);
System.out.print("\t\t"+inp[i])
; mini=min(count,process);
System.out.print("\t\t"+mini);
count=count-mini;
System.out.print("\t\t"+count)
;
```

```
System.out.print("\t\t"+drop);
drop=0;
System.out.println();
```

```
}
for(;count!=0;i++)
{
if(count>cap)
{
d
r
o
p
=
c
o
u
n
t
-
c
a
p
;
c
o
u
n
t
=
c
a
p
;
}
System.
out.prin
t(i+1);
System.
out.prin
t("\t\t0"
);
mini=m
in(coun
t,proces
s);
System.
out.prin
t("\t\t"+
mini);
count=c
ount-mi
ni;
System.
out.prin
t("\t\t"+
count);
System.
out.prin
t("\t\t"+
drop);
System.
out.prin
tln();
}
}
}
```

## *OUTPUT*

```
user@user-OptiPlex-3050:~
/Desktop$ javac leaky.java
user@user-OptiPlex-3050:~
/Desktop$ java leaky Enter
The Bucket Size

5
Enter The Operation Rate

2
Enter The No. Of Seconds You Want To Stimulate

3
Enter The Size Of The Packet Entering At 1 sec:
2
Enter The Size Of The Packet Entering At 1 sec:
3
Enter The Size Of The Packet Entering At 1 sec:
5
```

```
          Second | Packet Recieved | Packet Sent | Packet Left |Packet Dropped|

------------------------------------------

          1        2        2        0        0
          2        3        2        1        0
          3        5        2        3        1
          4        0        2        1        0
          5        0        1        0        0
```