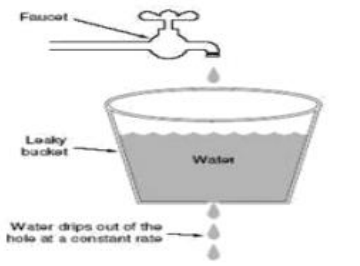
Simulation of congestion control algorithm

Aim:

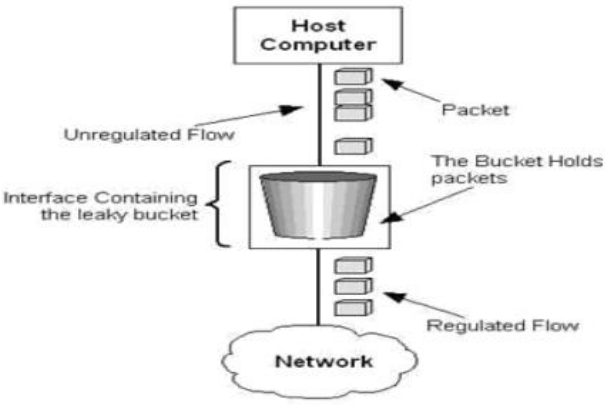
Write a program for congestion control using leaky bucket algorithm

Theory:

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 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.

Program:

```
import java.io.*;
import java.util.*;
class Queue
{
int q[],f=0,r=0,size;
void insert(int n)
{
Scanner in = new Scanner(System.in);
q=new int[10];
```

```
for(int i=0;i<n;i++)
{
System.out.print("\nEnter " + i + " element: ");
int ele=in.nextInt();
if(r+1>10)
{
System.out.println("\nQueue is full \nLost Packet: "+ele);
break;
}
else
{
r++;
q[i]=ele;
}
}
}
void delete()
{
Scanner in = new Scanner(System.in);
Thread t=new Thread();
if(r==0)
System.out.print("\nQueue empty ");
else
{
for(int i=f;i<r;i++)
{
try
{
t.sleep(1000);
}
        catch(Exception e){}
System.out.print("\nLeaked Packet: "+q[i]);
f++;
}
}
System.out.println();
}
}
class Leaky extends Thread
{
public static void main(String ar[]) throws Exception
{
Queue q=new Queue();
Scanner src=new Scanner(System.in);
System.out.println("\nEnter the packets to be sent:");
int size=src.nextInt();
```

```
q. insert(size);
q.delete();
}
}
```

Output:

```
E:\nwlab>java Leaky


Enter the packets to be sent:

12


Enter 0 element: 23


Enter 1 element: 34


Enter 2 element: 34


Enter 3 element: 334


Enter 4 element: 334


Enter 5 element: 34


Enter 6 element: 34


Enter 7 element: 34


Enter 8 element: 34


Enter 9 element: 34


Enter 10 element: 4
```

```
Queue is full
Lost Packet: 4

Leaked Packet: 23
Leaked Packet: 34
Leaked Packet: 34
Leaked Packet: 334
Leaked Packet: 334
Leaked Packet: 34
Leaked Packet: 34
Leaked Packet: 34
Leaked Packet: 34
Leaked Packet: 34

E:\nwlab>
```

Result:

Thus the program for congestion control using leaky bucket algorithm is implemented and executed successfully.