## 12. 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 spill over. 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)).

The Leaky Bucket Algorithm

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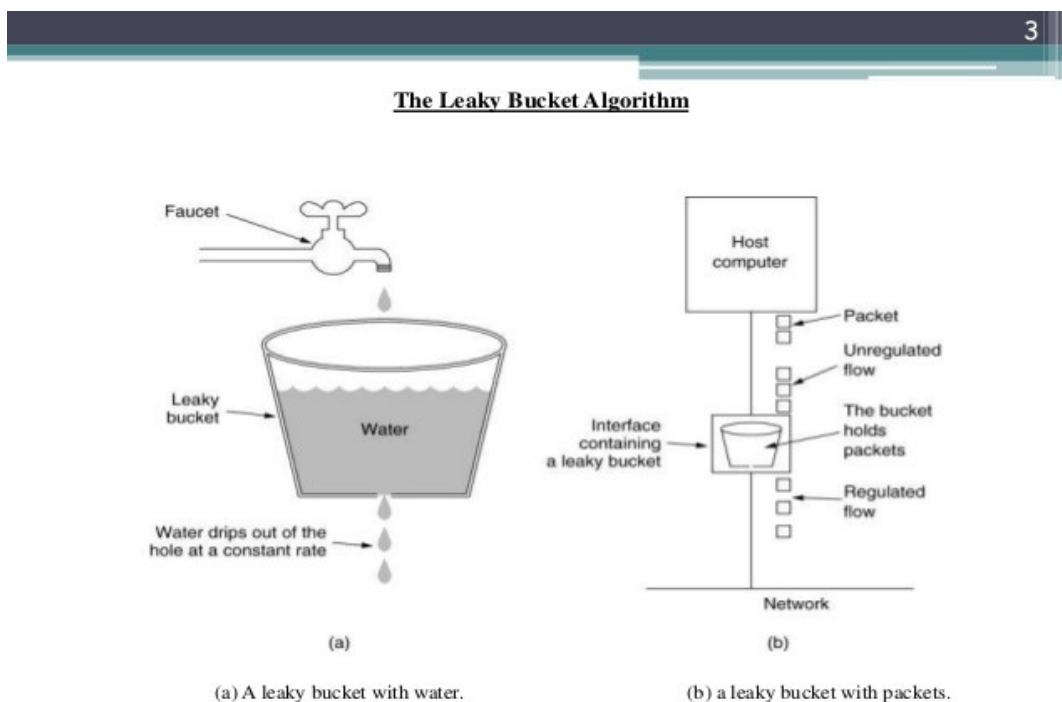
While leaky bucket eliminates completely bursty traffic by regulating the incoming data flow its main drawback is that it drops packets if the bucket is full. Also, it doesn't take into account the idle process of the sender which means that if the host doesn't transmit data for some time the bucket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

Source Code:

```java
import java.io.*;
import java.util.Scanner;
class Leaky
{
	public static int min(int x,int y)
	{
		if(x<y)
			return x;
		else
			return y;
	}
	public static void main(String args[])
	{
		Scanner sc=new Scanner(System.in);
		int drop=0,mini,n,cap,count=0,i;
		int inp[ ]= new int[25];
		int outrate;
		System.out.println("Enter The Bucket Size\n");
		cap=sc.nextInt();
		System.out.println("Enter The Output Rate\n");
		outrate=sc.nextInt();
		System.out.println("Enter the number of packets\n");
		n=sc.nextInt();
		System.out.println("Enter the size of packets to be sent:");
		for(i=0;i<n;i++)
		{
			inp[i]=sc.nextInt();
		}
		System.out.println("\nSecond|Packet Received|Packet Sent|Packet
					Left |Packet Dropped|\n");
		System.out.println("----------------------------------------------------------\n");
		for(i=0;i<n;i++)
		{
			count+=inp[i];
			if(count>cap)
			{
				drop=count-cap;
				count=cap;
			}
			System.out.print(i+1);
			System.out.print("\t" +inp[i]);
			mini=min(count,outrate);
			System.out.print("\t\t" + mini);
			count=count-mini;
			System.out.print("\t\t" +count);
			System.out.println("\t\t"+ drop);
			drop=0;
		}
		for(;count!=0;i++)
		{
			if(count>cap)
```

```java
                {
                        drop=count-cap;
                        count=cap;
                }
                System.out.print(i+1);
                System.out.print("\t0");
                mini=min(count,outrate);
                System.out.print("\t\t" +mini);
                count=count-mini;
                System.out.print("\t\t" +count);
                System.out.println("\t\t" +drop);
            }
        }
}
```

OUTPUT:

Enter The Bucket Size

5
Enter The Output Rate

2
Enter the number of packets

3
Enter the size of packets to be sent:
5
4
3

Second|Packet Recieved|Packet Sent|PacketLeft|Packet Dropped|

Second	Packet Recieved	Packet Sent	PacketLeft	Packet Dropped
1	5	2	3	0
2	4	2	3	2
3	3	2	3	1
4	0	2	1	0
5	0	1	0	0

lab3-20@lab320-Veriton-Series:~/CN$