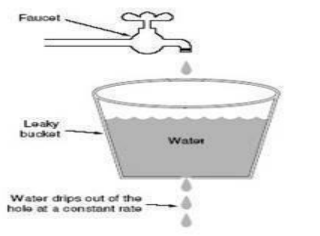
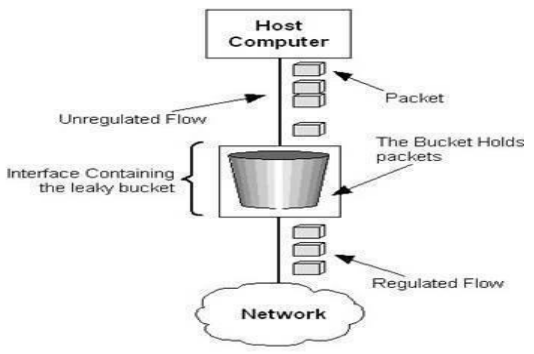
**Program 7: Write a program for congestion control using leaky bucket algorithm.**

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

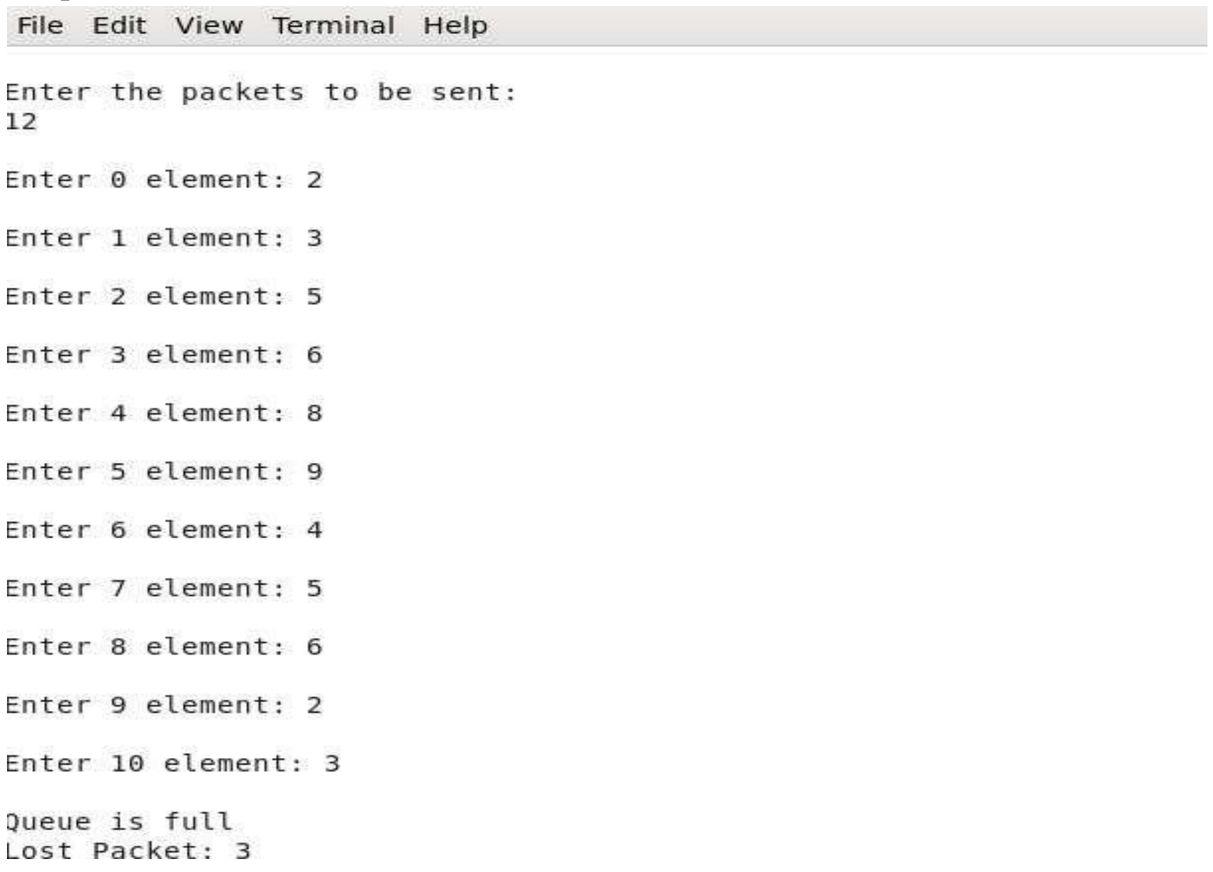


(b) A leaky bucket with packets

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.

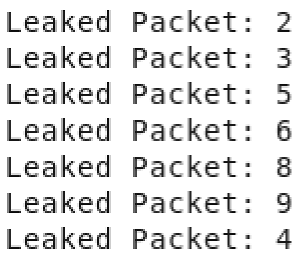
**Source Code:**
```
import java.io.*;
import java.util.*;
class Queue
{
        int q[],f=0,r=0,size;
        void insert(int n)
        {
                Scanner in = new Scanner(System.in);
                q=new int[10];
                for(int i=0;i<n;i++)
                {
                     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



File Edit View Terminal Help

```
Enter the packets to be sent:
12

Enter 0 element: 2

Enter 1 element: 3

Enter 2 element: 5

Enter 3 element: 6

Enter 4 element: 8

Enter 5 element: 9

Enter 6 element: 4

Enter 7 element: 5

Enter 8 element: 6

Enter 9 element: 2

Enter 10 element: 3

Queue is full
Lost Packet: 3
```



```
Leaked Packet: 2
Leaked Packet: 3
Leaked Packet: 5
Leaked Packet: 6
Leaked Packet: 8
Leaked Packet: 9
Leaked Packet: 4
```

Explanation:

**import java.io.*;** Imports necessary classes for input/output operations.

**import java.util.*;** Imports the Scanner class from the java.util package for user input.

**class Queue {** Defines a class named Queue to represent a simple queue.

**int q[], f = 0, r = 0, size;** Declares instance variables for the queue array (q), front (f), rear (r), and the size of the queue.

**void insert(int n) {** Defines a method insert to insert elements into the queue. It takes the number of elements to be inserted (n) as a parameter.

**Scanner in = new Scanner(System.in**); Creates a Scanner object for user input.

**q = new int[10];** Initializes the queue array with a size of 10.

**for (int i = 0; i < n; i++) {** Loop to insert n elements into the queue.

**System.out.print("\nEnter " + i + " element: ");** Prompts the user to enter the i-th element.

**int ele = in.nextInt();** Reads the user input as the element to be inserted.

**if (r + 1 > 10) {** Checks if the queue is full.

**System.out.println("\nQueue is full \nLost Packet: " + ele); break;** Prints a message indicating that the queue is full and a packet is lost. Exits the loop.

**else {** If the queue is not full:

**r++;** Increments the rear pointer.

**q[i] = ele;** Inserts the element into the queue.

**void delete() {** Defines a method delete to delete (leak) packets from the queue.

**Thread t = new Thread();** Creates a Thread object for simulating a delay.

**if (r == 0) System.out.print("\nQueue empty ");** Checks if the queue is empty.

**else {** If the queue is not empty:

**for (int i = f; i < r; i++) {** Loop to iterate through elements in the queue.

**try { t.sleep(1000); } catch (Exception e) {}** Introduces a delay of 1000 milliseconds (1 second) between printing each leaked packet.

**System.out.print("\nLeaked Packet: " + q[i]);** Prints the leaked packet.

**f++;**Increments the front pointer.

**System.out.println();** Prints a newline after leaking all packets.

**class Leaky extends Thread {:** Defines a class named Leaky that extends Thread.

**public static void main(String ar[]) throws Exception {:** The main method.

**Queue q = new Queue();:** Creates an instance of the Queue class.

**Scanner src = new Scanner(System.in);:** Creates a Scanner object for user input.

**System.out.println("\nEnter the packets to be sent:");:** Prompts the user to enter the number of packets to be sent.

**int size = src.nextInt();:** Reads the user input as the number of packets to be sent.

**q.insert(size);:** This method is responsible for inserting elements into the queue based on the specified size.

**q.delete();**
This method simulates the process of leaking packets from the queue and prints the leaked packets.