Experiment No. 11

Aim: To implement leaky bucket algorithm using programming language.

Requirement: Windows/MAC/Linux OS in P.C and JAVA/Python programming Language in OS.

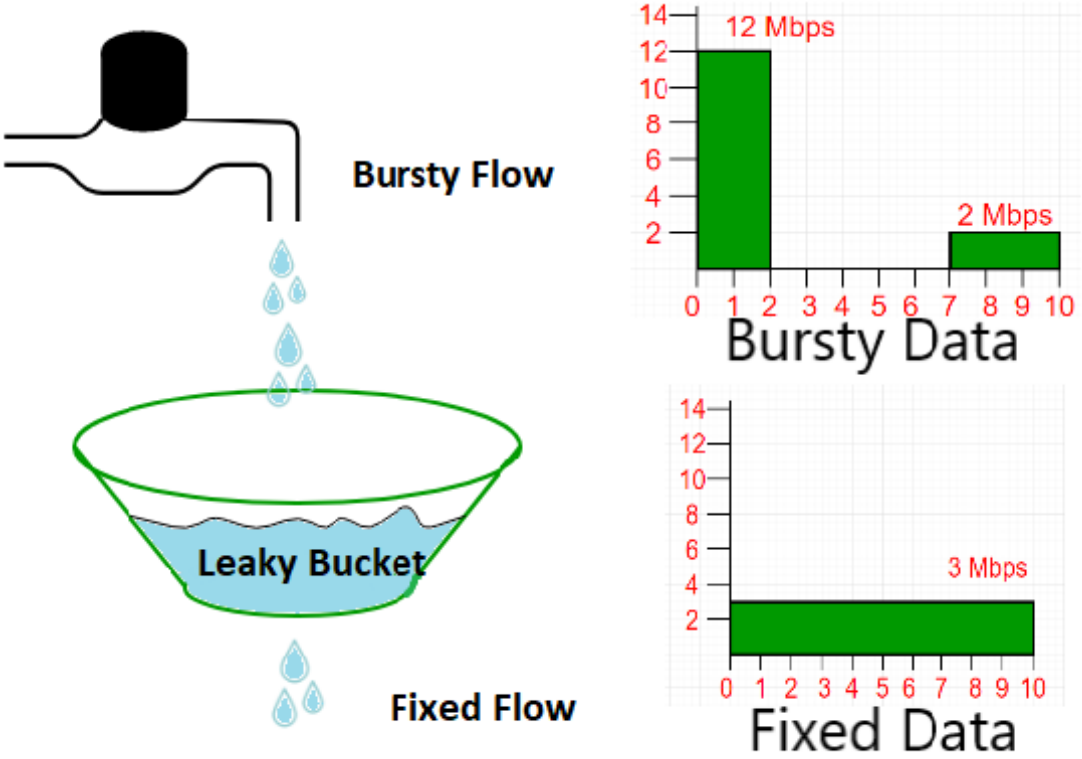
Theory:

The leaky bucket algorithm is a method of temporarily storing a variable number of requests and organizing them into a set-rate output of packets in an asynchronous transfer mode (ATM) network.

The leaky bucket is used to implement traffic policing and traffic shaping in Ethernet and cellular data networks. The algorithm can also be used to control metered-bandwidth Internet connections to prevent going over the allotted bandwidth for a month, thereby avoiding extra charges.

The algorithm works similarly to the way an actual leaky bucket holds water: The leaky bucket takes data and collects it up to a maximum capacity. Data in the bucket is only released from the bucket at a set rate and size of packet. When the bucket runs out of data, the leaking stops. If incoming data would overfill the bucket, then the packet is considered to be non-conformant and is not added to the bucket. Data is added to the bucket as space becomes available for conforming packets.

The leaky bucket algorithm can also detect both gradually increasing and dramatic memory error increases by comparing how the average and peak data rates exceed set acceptable background amounts.



A simple leaky bucket algorithm can be implemented using FIFO queue. A FIFO queue holds the packets. If the traffic consists of fixed-size packets (e.g., cells in ATM networks), the process removes a fixed number of packets from the queue at each tick of the clock. If the traffic consists of variable-length packets, the fixed output rate must be based on the number of bytes or bits.

The following is an algorithm for variable-length packets:

1. Initialize a counter to n at the tick of the clock.
2. If n is greater than the size of the packet, send the packet and decrement the counter by the packet size. Repeat this step until n is smaller than the packet size.
3. Reset the counter and go to step 1.

Code:

```
import java.io.*;
import java.util.*;
class LB{
   public static void main (String[] args) {
      int no_of_queries,storage,output_pkt_size;
      int input_pkt_size,bucket_size,size_left;
      storage=0;
      no_of_queries=4;
      bucket_size=10;
      input_pkt_size=4;
      output_pkt_size=1;
      for(int i=0;i<no_of_queries;i++)
      {
         size_left=bucket_size-storage;
         if(input_pkt_size<=(size_left))
         {
            storage+=input_pkt_size;
            System.out.println("Buffer size= "+storage+
               " out of bucket size= "+bucket_size);
         }
         else
         {
```

```
            System.out.println("Packet loss = " +(input_pkt_size-(size_left)));
            storage=bucket_size;
            System.out.println("Buffer size= "+storage+" out of bucket size= "+bucket_size);
        }
        storage-=output_pkt_size;
    }
  }
}
```

Output:

```
C:\Users\adnan\OneDrive\Desktop\College\sem5\CN\Practical 11>java LB
Buffer size= 4 out of bucket size= 10
Buffer size= 7 out of bucket size= 10
Buffer size= 10 out of bucket size= 10
Packet loss = 3
Buffer size= 10 out of bucket size= 10
```

Conclusion: We have understand the concept of Leaky Bucket algorithm and successfully implemented it in Java Programming Language.