BMS INSTITUTE OF TECHNOLOGY AND MANAGEMENT
YELAHANKA, BENGALURU - 560064

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Report on Leaky Bucket Algorithm

Name	Raghavendra K M
USN	1BY18IS093
Semester/Section	5B
Course Code	18CSL57
Course Name	Computer Network Laboratory
Faculty	Prof. Gireesh babu C N
Title	Leaky Bucket Algorithm
Date	26-11-2020

Signature of a Student **Signature of a Faculty**

Experiment No. 11

Write a program for simple Leaky Bucket Algorithm to encrypt and decrypt the data.

Leaky Bucket Algorithm:

In the network layer, before the network can make Quality of service guarantees, it must know what traffic is being guaranteed. One of the main causes of congestion is that traffic is often bursty.

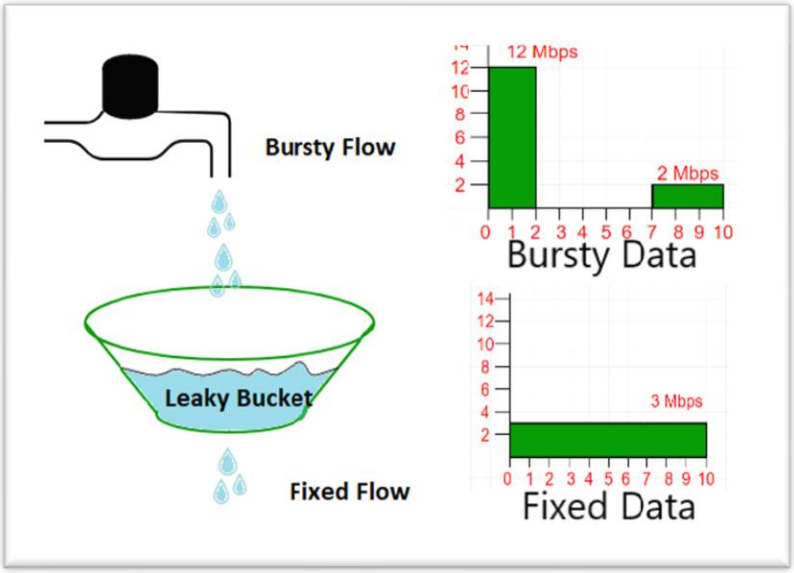
To understand this concept first we have to know little about traffic shaping. Traffic Shaping is a mechanism to control the amount and the rate of the traffic sent to the network. Approach of congestion management is called Traffic shaping. Traffic shaping helps to regulate rate of data transmission and reduces congestion.

There are 2 types of traffic shaping algorithms:

i. Leaky Bucket
ii. Token Bucket

Suppose we have a bucket in which we are pouring water in a random order but we have to get water in a fixed rate, for this we will make a hole at the bottom of the bucket. It will ensure that water coming out is in a some fixed rate, and also if bucket will full we will stop pouring in it.

The input rate can vary, but the output rate remains constant. Similarly, in networking, a technique called leaky bucket can smooth out bursty traffic. Bursty chunks are stored in the bucket and sent out at an average rate.



In the figure, we assume that the network has committed a bandwidth of 3 Mbps for a host. The use of the leaky bucket shapes the input traffic to make it conform to this commitment. In Figure the host sends a burst of data at a rate of 12 Mbps for 2 s, for a total of 24 Mbits of data. The host is silent for 5 s and then sends

data at a rate of 2 Mbps for 3 s, for a total of 6 Mbits of data. In all, the host has sent 30 Mbits of data in 10 s. The leaky bucket smooths the traffic by sending out data at a rate of 3 Mbps during the same 10 s.

Without the leaky bucket, the beginning burst may have hurt the network by consuming more bandwidth than is set aside for this host. We can also see that the leaky bucket may prevent congestion.

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:

Step 1: Initialize a counter to n at the tick of the clock.

Step 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.

Step 3: Reset the counter and go to step 1.

Source code:

LeakyBucket.java

```
import java.util.Scanner;

public class LeakyBucket {
    int I, B, N;
    int[] dt;
    int t, p, bs;
    boolean inCtrl = true;
    boolean outCtrl = true;

    void read() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the value of I (size of packet in bytes)");
        I = scanner.nextInt();
        System.out.println("Enter the Bucket Size");
        B = scanner.nextInt();
        System.out.println("Enter number of packets of size I to transfer");
        N = scanner.nextInt();
        dt = new int[N];
        System.out.println
("Enter the arrival time (in seconds) for all to packets in ascending order");
        for (int i = 0; i < N; i++)
            dt[i] = scanner.nextInt();
        p = bs = 0;
        t = 0;
        scanner.close();
    }
```

```
    void insert() {
        if (p < N) {
            if (dt[p] == t) {
                if (bs + I <= B) {
                    bs += I;
                    System.out.println
("Packet received at t = " + t + ". " + bs + " Bytes still left in Bucket");
                } else
                    System.out.println
                         ("Bucket overflow. Packet Lost at t = " + t);
                p++;
            }
            t++;
        } else
            inCtrl = false;
    }

    void delete() {
        if (bs > 0) {
            bs--;
            System.out.println
                 ("1 Byte Leaked.\t\t " + bs + " Bytes still left in Bucket");
        } else {
            System.out.println("Bucket is Empty. Waiting for Incoming Packets");
        }
        if (p == N && bs == 0)
            outCtrl = false;
    }

    public static void main(String[] args) {
        LeakyBucket leakyBucket = new LeakyBucket();
        leakyBucket.read();
        InsertThread insThread = new InsertThread(leakyBucket);
                                 //Call insert thread
        DeleteThread delThread = new DeleteThread(leakyBucket);
                                 //Call delete thread
    }
}

class InsertThread implements Runnable {
    LeakyBucket bkt1;

    public InsertThread(LeakyBucket bkt) {
        bkt1 = bkt;
        Thread insertThread = new Thread(this);
        insertThread.start();
    }

    public void run() {
        try {
            while (bkt1.inCtrl) {
                bkt1.insert();
                Thread.sleep(1000);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```java
class DeleteThread implements Runnable {
    LeakyBucket bkt1; //bkt1 is an instance of Bucket class

    public DeleteThread(LeakyBucket bkt) {
        this.bkt1 = bkt; //create a reference
        Thread deleteThread = new Thread(this);
        deleteThread.start();
    }

    public void run() {
        try {
            while (bkt1.outCtrl) {
                bkt1.delete();
                Thread.sleep(1000);
            }
            System.out.println("\nAll Packets Have Been Sent");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Outputs:

```
C:\Windows\System32\cmd.exe
D:\1by18is093\ISE V SEM\Subjects\18CSL57 - Computer Network Laboratory\Leaky Bucket algorithm>javac LeakyBucket.java

D:\1by18is093\ISE V SEM\Subjects\18CSL57 - Computer Network Laboratory\Leaky Bucket algorithm>java LeakyBucket
Enter the value of I (size of packet in bytes)
4
Enter the Bucket Size
10
Enter number of packets of size I to transfer
6
Enter the arrival time (in seconds) for all to packets in ascending order
1
2
6
8
9
10
Bucket is Empty. Waiting for Incoming Packets
Packet received at t = 1. 4 Bytes still left in Bucket
1 Byte Leaked.          3 Bytes still left in Bucket
Packet received at t = 2. 6 Bytes still left in Bucket
1 Byte Leaked.          6 Bytes still left in Bucket
1 Byte Leaked.          5 Bytes still left in Bucket
1 Byte Leaked.          4 Bytes still left in Bucket
1 Byte Leaked.          3 Bytes still left in Bucket
1 Byte Leaked.          6 Bytes still left in Bucket
Packet received at t = 6. 6 Bytes still left in Bucket
1 Byte Leaked.          5 Bytes still left in Bucket
Packet received at t = 8. 9 Bytes still left in Bucket
1 Byte Leaked.          8 Bytes still left in Bucket
1 Byte Leaked.          7 Bytes still left in Bucket
Bucket overflow. Packet Lost at t = 9
1 Byte Leaked.          6 Bytes still left in Bucket
Packet received at t = 10. 10 Bytes still left in Bucket
1 Byte Leaked.          9 Bytes still left in Bucket
1 Byte Leaked.          8 Bytes still left in Bucket
1 Byte Leaked.          7 Bytes still left in Bucket
1 Byte Leaked.          6 Bytes still left in Bucket
1 Byte Leaked.          5 Bytes still left in Bucket
1 Byte Leaked.          4 Bytes still left in Bucket
1 Byte Leaked.          3 Bytes still left in Bucket
1 Byte Leaked.          2 Bytes still left in Bucket
1 Byte Leaked.          1 Bytes still left in Bucket
1 Byte Leaked.          0 Bytes still left in Bucket

All Packets Have Been Sent

D:\1by18is093\ISE V SEM\Subjects\18CSL57 - Computer Network Laboratory\Leaky Bucket algorithm>
```