

Name : Ansari M.Saeem M.Saleem

Uid : 2019430001

Subject : NAD

Expt no : 2

Aim : Write a program to archive Traffic management at Flow level by implementing Closed Loop Control technique. (Leaky Bucket Algorithm)

Date Of Performance : 21/01/2020

Aim:

Write a program to archive Traffic management at Flow level by implementing Closed Loop Control technique. (Leaky Bucket Algorithm).

Objectives:

- To ensure network transmission meet specific/required traffic condition.
- To detect and remove congestion on the network while transmission of packets.
- To maintain a steady traffic by continuously transmitting packets and to obtain a constant output rate.

Theory:

Congestion control refers to the mechanisms and techniques used to control congestion and keep the traffic below the capacity of the network. The congestion control techniques can be broadly classified two broad categories:

- Open loop : Protocols to prevent or avoid congestion, ensuring that the system (or network under consideration) never enters a Congested State.
- Close loop : Protocols that allow system to enter congested state, detect it, and remove it.

Consider a Bucket with a small hole at the bottom, whatever may be the rate of water pouring into the bucket, the rate at which water comes out from that small hole is constant. Once the bucket is full, any additional water entering it spills over the sides and is lost (i.e. it doesn't appear in the output stream through the hole underneath). The same idea of leaky bucket can be applied to packets. Conceptually each network interface contains a leaky bucket, and the following steps are performed:

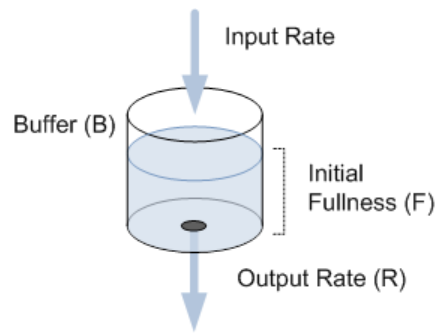
- When the host has to send a packet, the packet is thrown into the bucket.
- The bucket leaks at a constant rate, meaning the network interface transmits packets at a constant rate.
- Bursty traffic is converted to a uniform traffic by the leaky bucket.
- In practice the bucket is a finite queue that outputs at a finite rate.

This arrangement can be simulated in the operating system or can be built into the hardware. Implementation of this algorithm is easy and consists of a finite queue. Whenever a packet arrives, if there is room in the queue it is queued up and if there is no room then the packet is discarded

Methodology:

To understand the leaky bucket model, consider a bucket with a small hole at the bottom. Three parameters define the bucket:

- The capacity/Buffer (B)
- The rate at which water flows out of the bucket / Output Rate(R)
- The initial fullness of the bucket (F)



If water is poured into the bucket at exactly rate R , the bucket will remain at F , because the input rate equals the output rate. If the input rate increases while R remains constant, the bucket accumulates water. If the input rate is larger than R for a sustained period, eventually the bucket overflows. However, the input rate can vary around R without overflowing the bucket, as long as the average input rate does not exceed the capacity of the bucket. The larger the capacity, the more the input rate can vary within a given window of time.

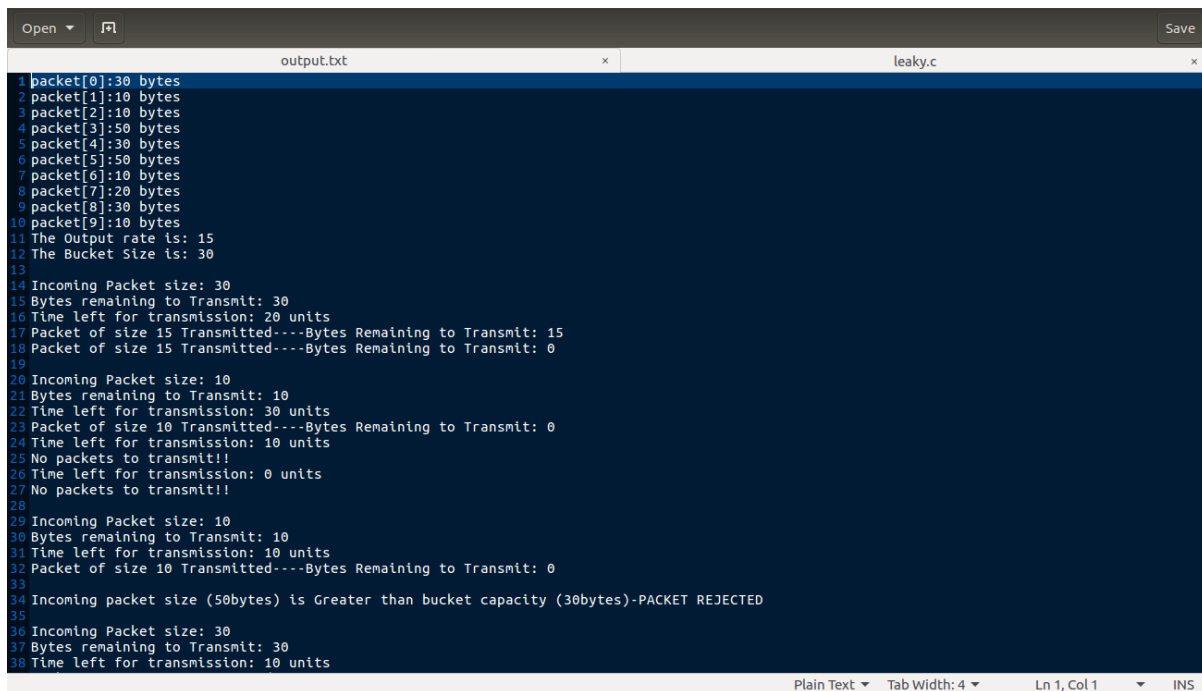
Algorithm:

1. Start
2. Set the bucket size or the buffer size.
3. Set the output rate.
4. Transmit the packets such that there is no overflow.
5. Repeat the process of transmission until all packets are transmitted. (Reject packets where its size is greater than the bucket size)
6. Stop

Results:

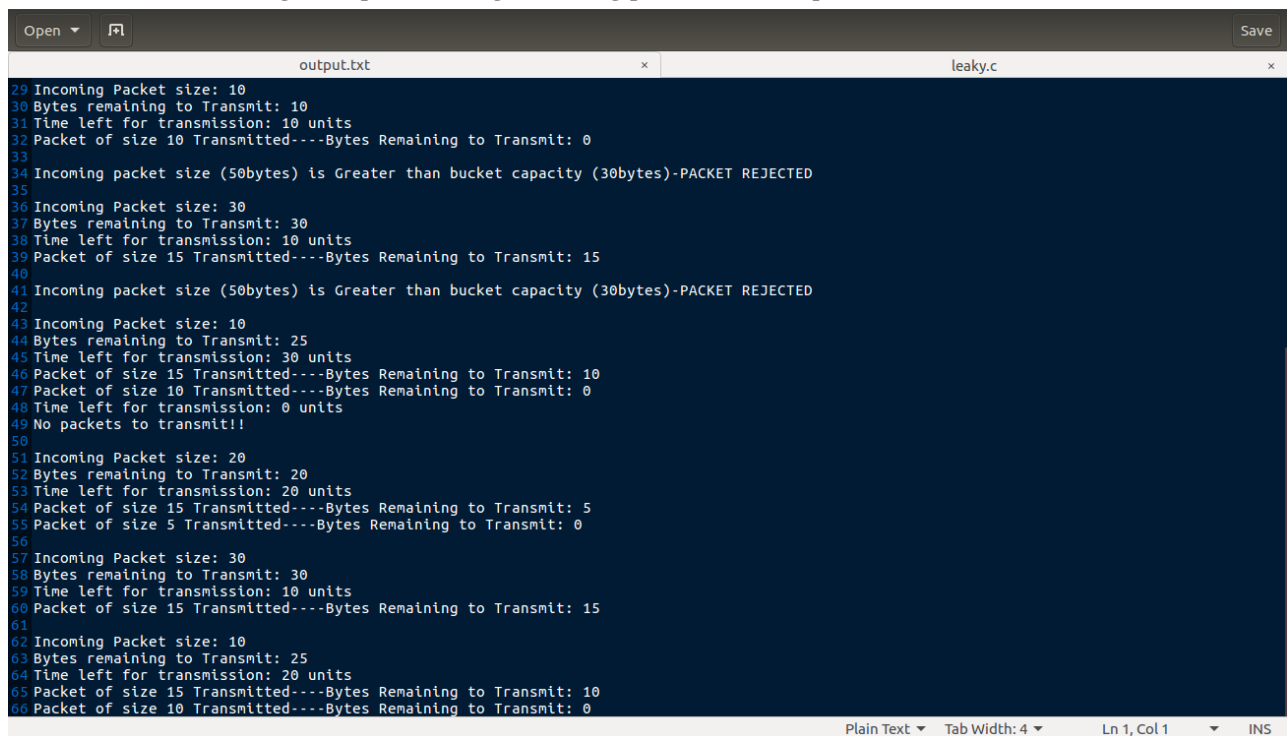
```
students@CE-Lab3-603-U22: ~/Desktop/saeem/NAD/2
students@CE-Lab3-603-U22:~/Desktop/saeem/NAD/2$ gcc leaky.c -o leaky
students@CE-Lab3-603-U22:~/Desktop/saeem/NAD/2$ ./leaky > output.txt
students@CE-Lab3-603-U22:~/Desktop/saeem/NAD/2$
```

fig - Command to run program and to save output in a txt file



```
1 packet[0]:30 bytes
2 packet[1]:10 bytes
3 packet[2]:10 bytes
4 packet[3]:50 bytes
5 packet[4]:30 bytes
6 packet[5]:50 bytes
7 packet[6]:10 bytes
8 packet[7]:20 bytes
9 packet[8]:30 bytes
10 packet[9]:10 bytes
11 The Output rate is: 15
12 The Bucket Size is: 30
13
14 Incoming Packet size: 30
15 Bytes remaining to Transmit: 30
16 Time left for transmission: 20 units
17 Packet of size 15 Transmitted----Bytes Remaining to Transmit: 15
18 Packet of size 15 Transmitted----Bytes Remaining to Transmit: 0
19
20 Incoming Packet size: 10
21 Bytes remaining to Transmit: 10
22 Time left for transmission: 30 units
23 Packet of size 10 Transmitted----Bytes Remaining to Transmit: 0
24 Time left for transmission: 10 units
25 No packets to transmit!!
26 Time left for transmission: 0 units
27 No packets to transmit!!
28
29 Incoming Packet size: 10
30 Bytes remaining to Transmit: 10
31 Time left for transmission: 10 units
32 Packet of size 10 Transmitted----Bytes Remaining to Transmit: 0
33
34 Incoming packet size (50bytes) is Greater than bucket capacity (30bytes)-PACKET REJECTED
35
36 Incoming Packet size: 30
37 Bytes remaining to Transmit: 30
38 Time left for transmission: 10 units
```

fig - Output showing incoming packets and output leak



```
29 Incoming Packet size: 10
30 Bytes remaining to Transmit: 10
31 Time left for transmission: 10 units
32 Packet of size 10 Transmitted----Bytes Remaining to Transmit: 0
33
34 Incoming packet size (50bytes) is Greater than bucket capacity (30bytes)-PACKET REJECTED
35
36 Incoming Packet size: 30
37 Bytes remaining to Transmit: 30
38 Time left for transmission: 10 units
39 Packet of size 15 Transmitted----Bytes Remaining to Transmit: 15
40
41 Incoming packet size (50bytes) is Greater than bucket capacity (30bytes)-PACKET REJECTED
42
43 Incoming Packet size: 10
44 Bytes remaining to Transmit: 25
45 Time left for transmission: 30 units
46 Packet of size 15 Transmitted----Bytes Remaining to Transmit: 10
47 Packet of size 10 Transmitted----Bytes Remaining to Transmit: 0
48 Time left for transmission: 0 units
49 No packets to transmit!!
50
51 Incoming Packet size: 20
52 Bytes remaining to Transmit: 20
53 Time left for transmission: 20 units
54 Packet of size 15 Transmitted----Bytes Remaining to Transmit: 5
55 Packet of size 5 Transmitted----Bytes Remaining to Transmit: 0
56
57 Incoming Packet size: 30
58 Bytes remaining to Transmit: 30
59 Time left for transmission: 10 units
60 Packet of size 15 Transmitted----Bytes Remaining to Transmit: 15
61
62 Incoming Packet size: 10
63 Bytes remaining to Transmit: 25
64 Time left for transmission: 20 units
65 Packet of size 15 Transmitted----Bytes Remaining to Transmit: 10
66 Packet of size 10 Transmitted----Bytes Remaining to Transmit: 0
```

fig - Output showing incoming packets and output leak

Conclusion:

Here we can conclude that using closed loop technique, we implemented implicit feedback algo (leaky bucket). Leaky bucket controls the bursty traffic on the network and produce steady output rate.