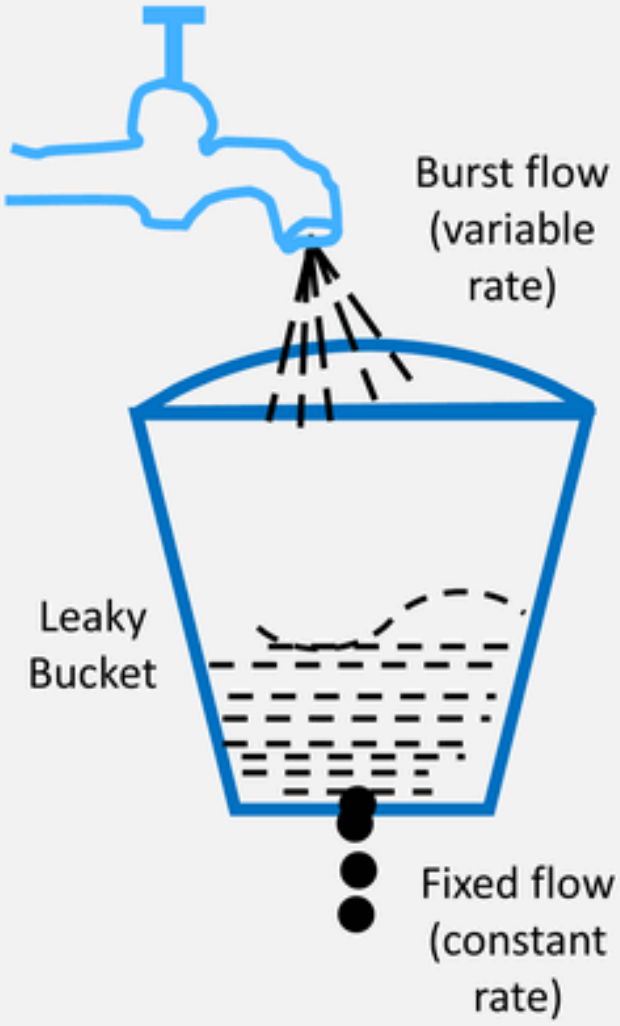
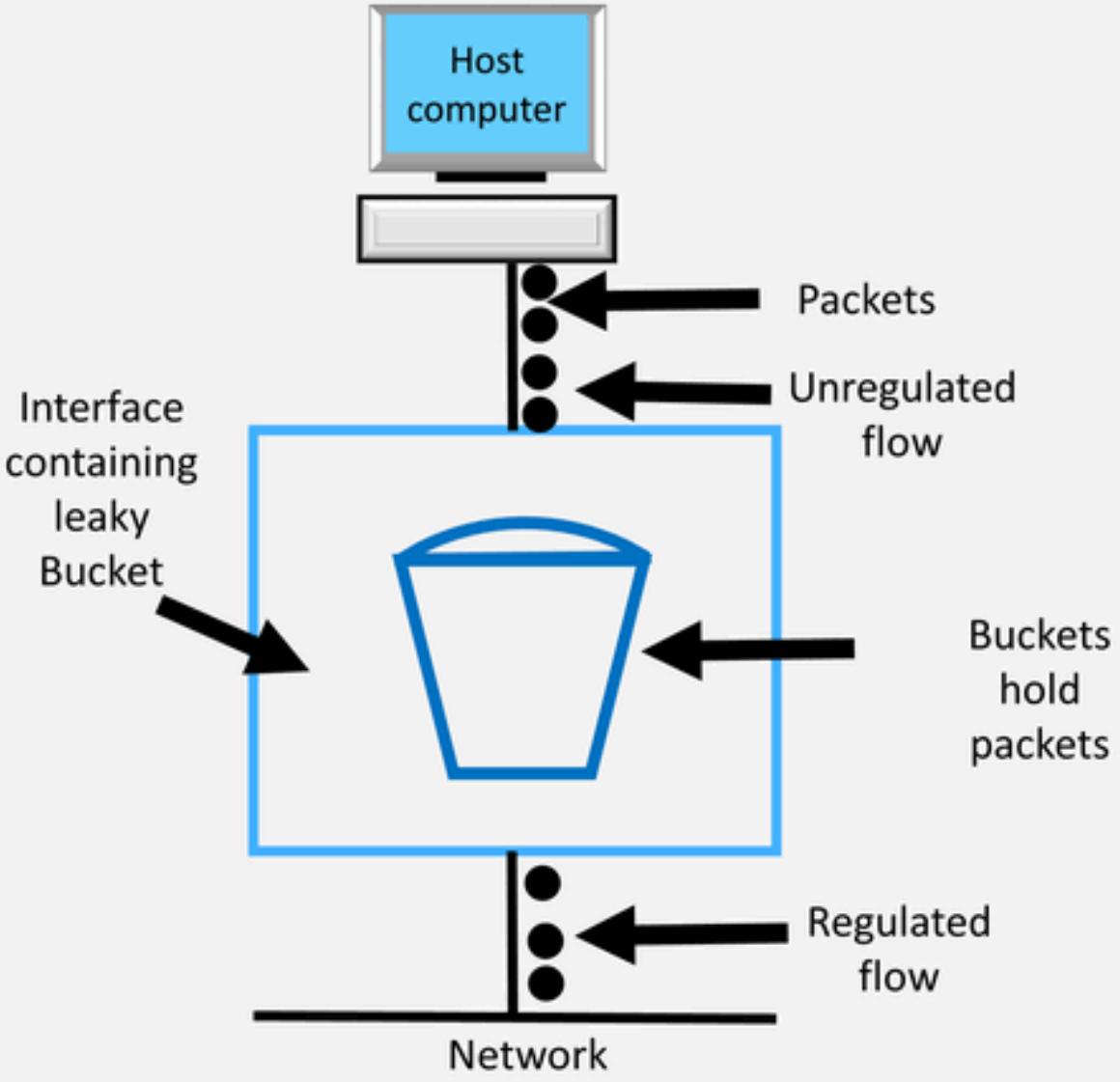
Policing

- Network monitors traffic flows continuously to ensure they meet their traffic contract
- When a packet violates the contract, network can discard or tag the packet giving it lower priority
- If congestion occurs, tagged packets are discarded first
- *Leaky Bucket Algorithm* is the most commonly used policing mechanism
  - Bucket has specified leak rate for average contracted rate
  - Bucket has specified depth to accommodate variations in arrival rate
  - Arriving packet is *conforming* if it does not result in overflow

LEAKY BUCKET

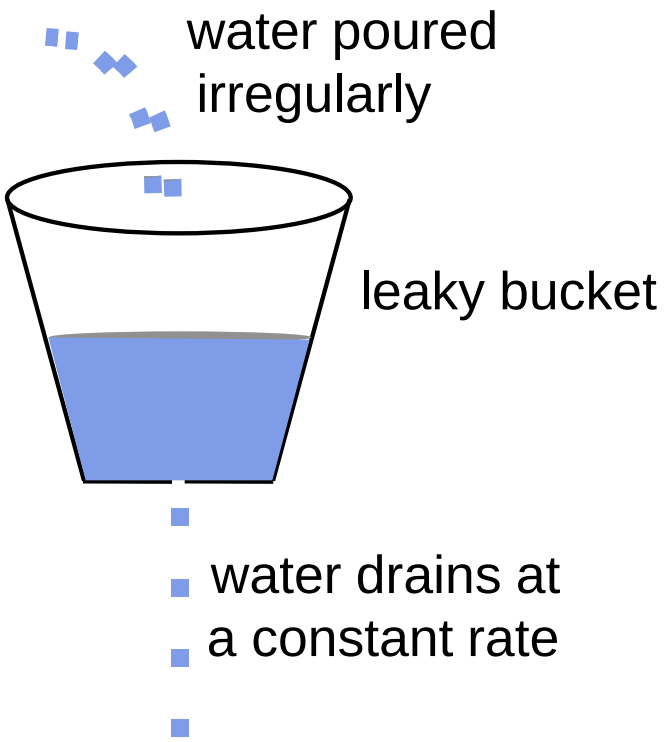


(a) A leaky Bucket with water



(b) A leaky Bucket with packets

Leaky Bucket algorithm can be used to police arrival rate of a packet stream



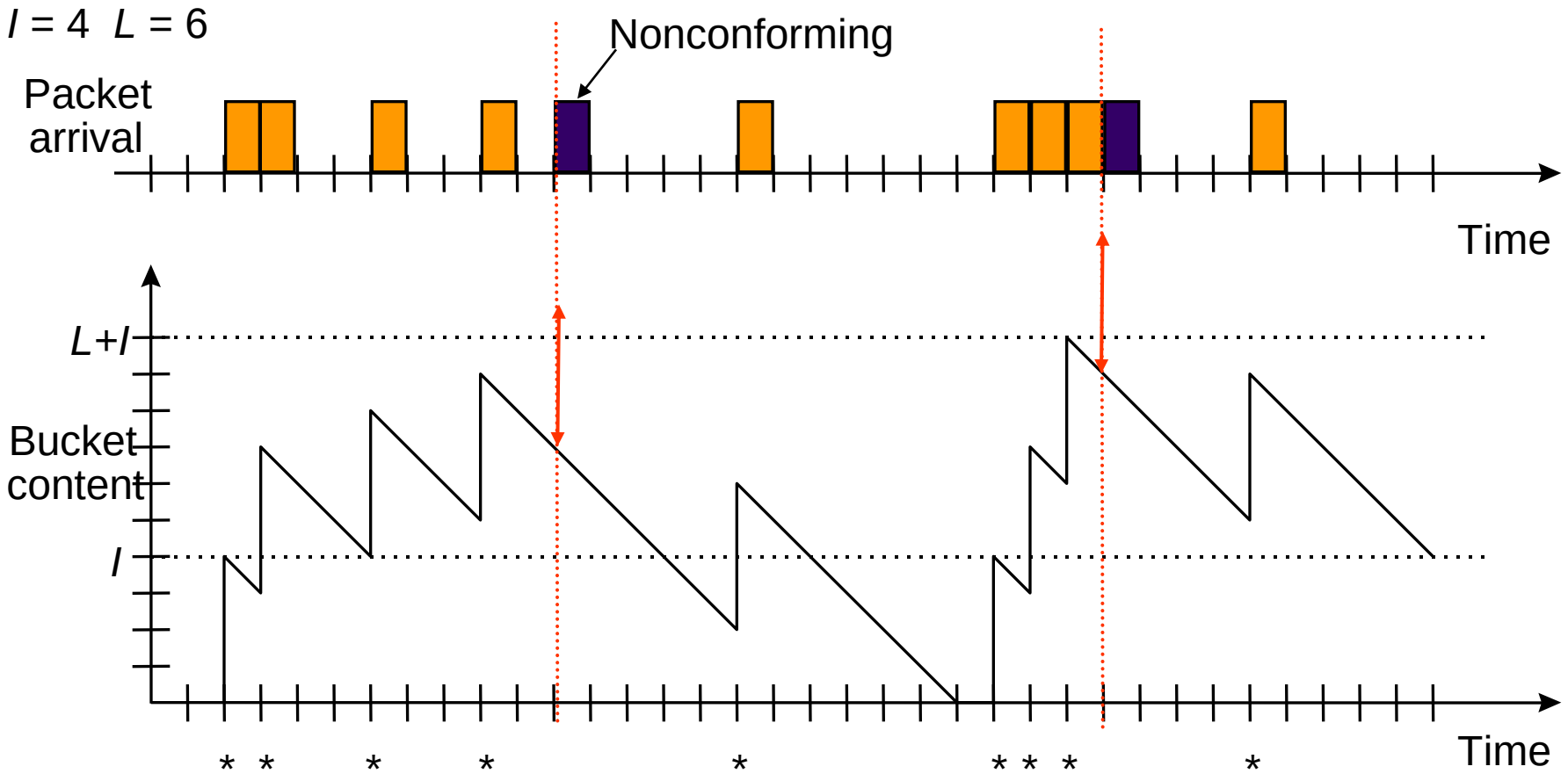
Leak rate corresponds to long-term rate

Bucket depth corresponds to maximum allowable burst arrival

1 packet per unit time
Assume constant-length packet as in ATM

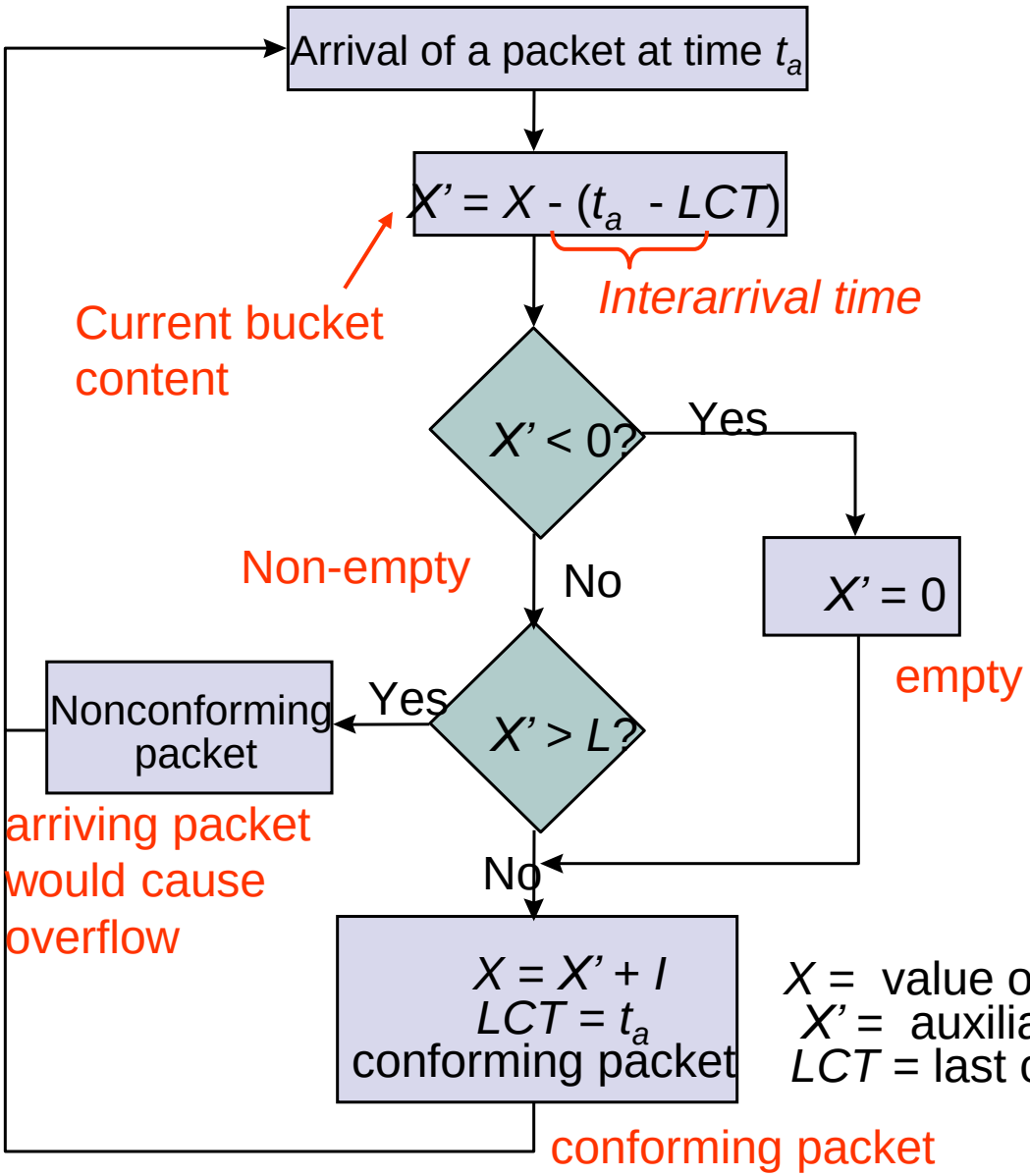
Let X = bucket content at last conforming packet arrival
Let t_a – last conforming packet arrival time = depletion in bucket

Leaky Bucket Example



Non-conforming packets not allowed into bucket & hence not included in calculations

Leaky Bucket Algorithm



Depletion rate:
1 packet per unit time

$L+I$ = Bucket Depth

I = increment per arrival, nominal interarrival time

X = value of the leaky bucket counter
X' = auxiliary variable
LCT = last conformance time

MINGW64:/c/Users/ANANTHANUP/ARMC-CN

```
ANANTHANUP@ANANTHANUP MINGW64 ~/ARMC-CN (master)
$ javac LeakyBucket.java

ANANTHANUP@ANANTHANUP MINGW64 ~/ARMC-CN (master)
$ 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 acending order
1
4
6
8
9
10
Bucket is Empty. Waiting for Incoming Packets
Bucket is Empty. Waiting for Incoming Packets
Packet recievd at t = 1. 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.           5 Bytes still left in Bucket
Packet recievd at t = 4. 5 Bytes still left in Bucket
1 Byte Leaked.           4 Bytes still left in Bucket
Packet recievd at t = 6. 7 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.           9 Bytes still left in Bucket
Packet recievd at t = 8. 9 Bytes still left in Bucket
1 Byte Leaked.           8 Bytes still left in Bucket
Bucket overflow. Packet Lost at t = 9
1 Byte Leaked.           7 Bytes still left in Bucket
Bucket overflow. Packet Lost at t = 10
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
```

A simple example of the operation of the leaky bucket algorithm is shown in Figure 7.55. Here the value of I is four packet times, and the value of L is six packet times. The arrival of the first packet increases the bucket content by four (packet times). At the second arrival the content has decreased to three, but four more are added to the bucket resulting in a total of seven. The fifth packet is declared as nonconforming since it would increase the content to 11, which would exceed $L + I$. Packets 7, 8, 9, and 10 arrive back to back after the bucket becomes empty. Packets 7, 8, and 9 are conforming, and the last one is nonconforming. If the peak rate is one packet/packet time, then the **maximum burst size (MBS)** for this algorithm is three. Note that the algorithm does not update the content of the bucket continuously, but only at discrete points (arrival times) indicated by the asterisks. Also note that the values of I and L in general can take any real numbers.