RYERSON UNIVERSITY

Congestion Control – Open Loop

Muhammad Jaseemuddin
Dept. of Electrical & Computer Engineering
Ryerson University
Toronto, Canada

References

1. A. Leon-Garcia and I. Widjaja, Communication Networks: Fundamental Concepts and Key Architectures, McGraw Hill, 2000.
2. A. Tenenbaum, Computer Networks, Prentice Hall.

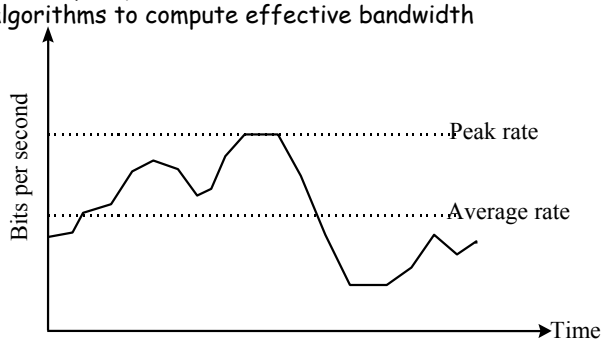
Open-Loop Congestion Control

- It prevents congestion from happening
- It entails explicit resource allocation along the path
- It is more versatile in providing different service models
  - Quality of Service
- It involves:
  - Signaling to specify Resource Requirement
  - Connection Admission Control (CAC)
  - Policing
  - Traffic Shaping

Admission Control

- Flow is an abstraction of connection in IP network
- At connection setup time (or before launching flow)
  - application expresses to the network:
    - flow traffic characteristics using traffic descriptors
      - e.g. peak rate, average rate, max. burst size etc
    - flow QoS requirements using QoS parameters
      - e.g. max. bandwidth, max. delay, delay variance, loss probability etc
  - network applies Connection Admission Control (CAC) to either accept or reject the connection setup request
    - CAC employs algorithms to compute effective bandwidth

Effective Bandwidth:
The amount of bandwidth typically lies between the average and the peak rate, and is called the effective bandwidth



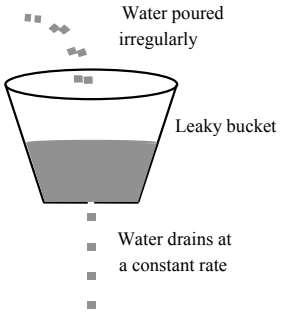
Policing

- Once a connection is accepted by CAC, the network can provide QoS as long as the traffic obeys the characteristics specified in the traffic descriptors
  - traffic descriptor and QoS parameters are contractual binding that both parties – application and network, must follow
- **Policing:** Network monitors the traffic to ascertain that it follows the characteristics specified in the traffic descriptor
  - traffic that violates the descriptor is dealt appropriately
    - discarded
    - tagged as low priority: at congestion point more vulnerable to drop
    - shaped to conform to the descriptor
  - metering
    - measure the real time traffic characteristics
  - marking
    - weigh against the traffic descriptor and tag or discard

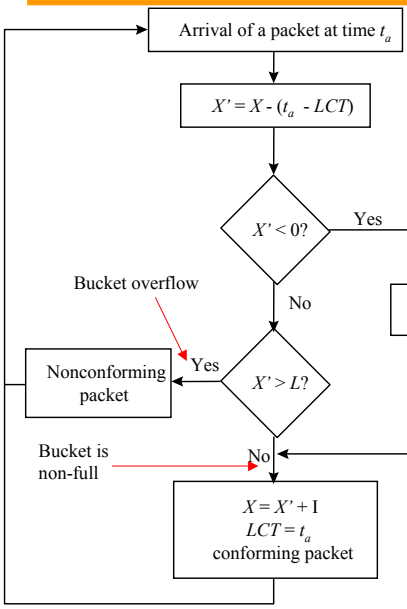
Leaky Bucket Policer

Leaky Bucket Policer

A bucket with a hole through which bucket content leaks at certain fixed rate ρ. The bucket is filled with the content at varying rate. The bucket depth indicates the allowable variation in the arrival rate. The bucket overflow indicates arrival rate exceeds allowable variation, that is the traffic violates its descriptor. Simile is leaky bucket of water. No matter how fast water can be poured in, it seeps at constant rate ρ. When bucket overflows water is disposed.



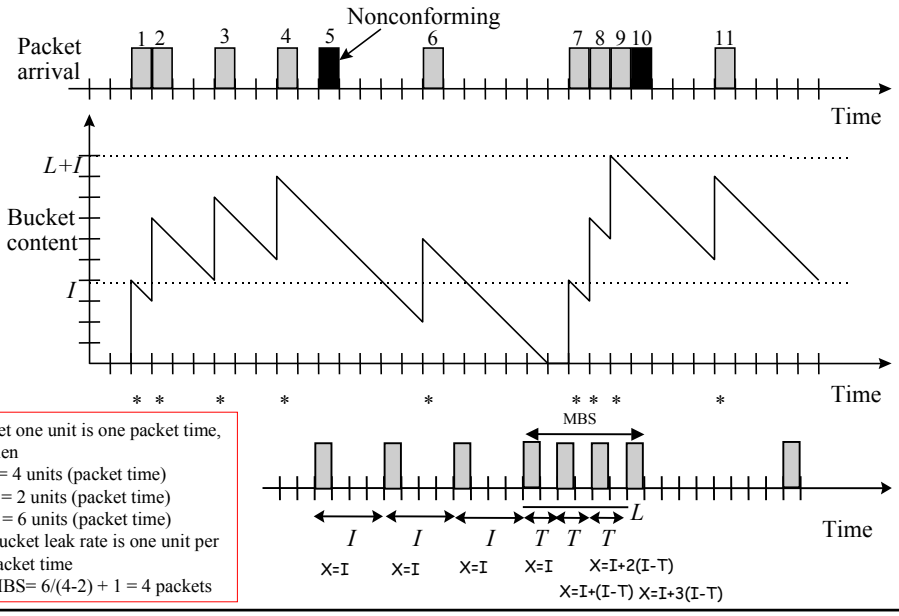
Leaky Bucket Policing Algorithm



Packet size is fixed (e.g. ATM cells)
A: Average Rate
P: Peak Rate
I : Packet inter-arrival time calculated from the sustainable average rate
T: Packet inter-arrival time calculated from the peak rate
L: Bucket depth in terms of packet time, it absorbs traffic burst
MBS: Maximum Burst Size
I = 1/A
T= 1/P
Extra tokens, L = (MBS – 1)(I – T)
MBS = 1 + [L / (I – T)]
I-T indicates how soon the packet has arrived due to peak rate as compared to if it arrives at average rate

X = value of the leaky bucket counter
X' = auxiliary variable
LCT = last conformance time
Initial Value of X = 0

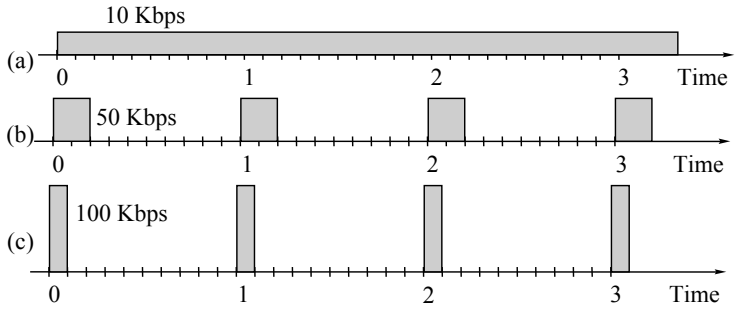
Behavior of Leaky Bucket



Let one unit is one packet time, then
I = 4 units (packet time)
P = 2 units (packet time)
L = 6 units (packet time)
Bucket leak rate is one unit per packet time
MBS= 6/(4-2) + 1 = 4 packets

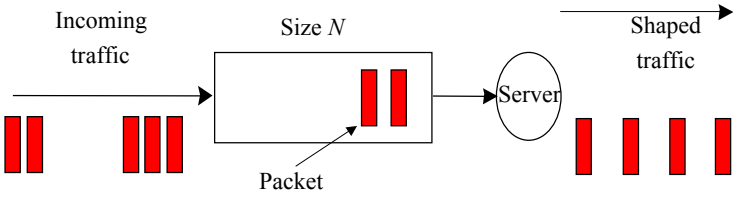
Traffic Shaping

- Traffic Shaping: The process of altering a traffic flow to another flow
- When do we need traffic shaping?
  - when a source generates traffic whose characteristic is not known and it wants to conform to the traffic policing parameters, then it employs shaping to alter the flow
  - when the network wants to regulate flow to smooth out undesirable burstiness
- How to do that?
  - Leaky bucket shaper, token bucket shaper



Leaky Bucket Traffic Shaper

- Traffic Shaper transmits the flow it receives at certain specified rate
  - if it cannot transmit all packets it has received, then it stores them in a buffer
- Leaky bucket Traffic shaper
  - stores the incoming packets in a buffer
  - empties the buffer at certain *fixed* rate
- unlike the leaky bucket policer
  - that only monitors the traffic, it regulates the traffic
  - whose bucket is a counter, its bucket is a buffer
- The buffer is used to store momentary bursts
  - its size defines the maximum burst that can be accommodated
  - packets are discarded when buffer overflows, which indicates that the incoming traffic is non-conformant



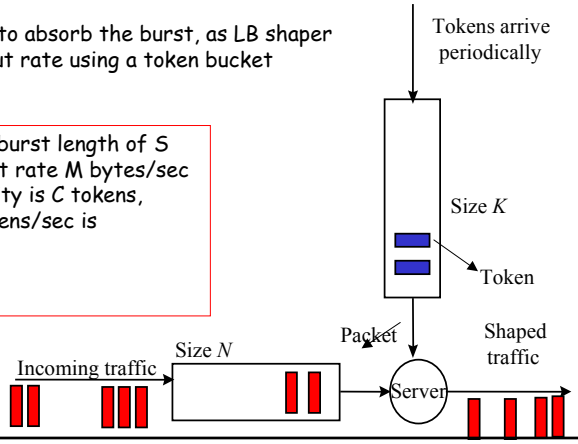
Token Bucket Traffic Shaper

- LB Traffic Shaper is restricted as its output rate is constant when the buffer is not empty
  - most applications generate variable rate traffic
  - storing packets for such traffic causes unnecessary delay
- Token Bucket Traffic Shaper is designed to allow variation in the transmission rate
  - it employs a buffer to absorb the burst, as LB shaper
  - it controls the output rate using a token bucket (counter)

The number of bytes in a burst length of S sec at the maximum output rate M bytes/sec if the token bucket capacity is C tokens, token arrival rate is R tokens/sec is

$$MS = C + RS$$

$$S = C/(M-R)$$

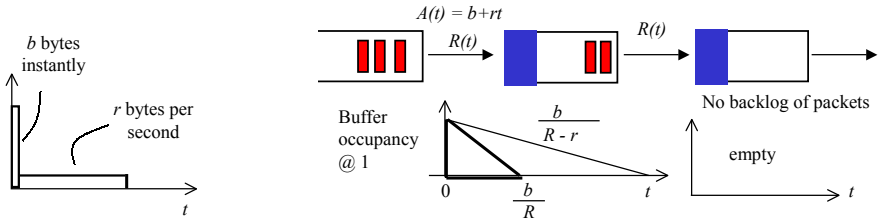


TB Algorithm

Algorithm:

- the token is generated at a constant rate and stored in the token bucket with capacity C; if the bucket is full, arriving tokens are discarded. In a packet system the token is a count of bytes, e.g. the token is generated as R tokens (bytes) in ΔT seconds.
- a packet from the buffer is transmitted only if the tokens can be drawn from the bucket, I.e. the bucket has tokens equivalent to the length of the packet, L bytes.
- if no token is available, the packet is backlogged in the buffer
- when the bucket is empty, then backlogged packets are transmitted at the rate at which token arrives in the buffer
  - it behaves like a LB shaper transmitting packets at a constant rate
- when the bucket is not empty, then the packets are drawn from the buffer as soon as they arrive
  - the traffic burstiness is preserved
  - however if burstiness continues the bucket is exhausted and the packets are backlogged
- the bucket size limits the burstiness

Delay Bound



- In time T a token bucket with rate r bytes/sec and depth b can transmit b+rT bytes
  - b bytes at time t=0 and rT upto time T
- Assume a TB shaper followed by two multiplexers in tandem each capable of transmitting R bytes/sec, where R > r
  - first multiplexer will buffer b bytes at t=0 and will subsequently receive r bytes/sec
  - the buffer occupancy will be b bytes, which will be drained at the rate R-r bytes/sec
  - a newly arrived byte experiences delay equivalent to the time it takes for the buffer occupancy to dissipate

Delay Bound Calculation

- the maximum delay experienced by the bytes following b bytes in the first multiplexer will be b/R seconds
- no queue will build up at the second multiplexer, because arrival rate = service rate
- the maximum delay through a chain of multiplexers each guaranteeing R bytes/sec bandwidth to the flow is b/R → fluid flow model
- Parekh et al comes up with delay bound in a packet network to be

$D <= b/R + (H-1)m/R + \Sigma_{j=1,H}(M/Rj)$, where m is max. packet size of the flow, M the max packet size in the network, H is the number of hops and Rj is the link bandwidth

b/R: buffer dissipation time at the first hop

(H-1)m/R: transmit time of the preceding packet of the same flow at every other hop

$\Sigma_{j=1,H}$ (M/Rj): transmit time of the preceding packet of other flow at every hop