Part II: Policing

Dr. Yeali S. Sun
National Taiwan University

Outline

- GCRA
- Leaky bucket
- Token bucket

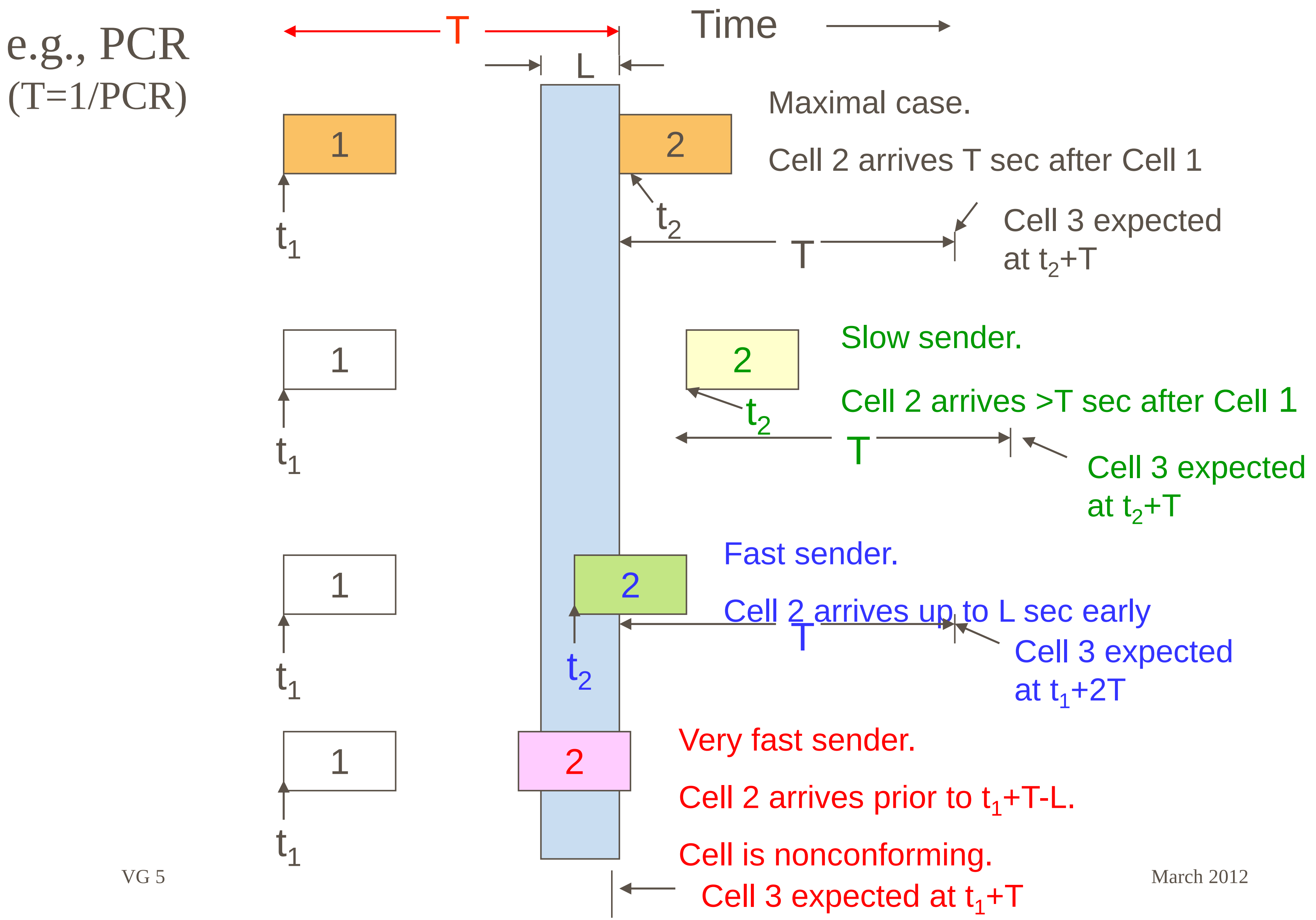
Policing

- The <u>*goal*</u> is to hold sources that are actually used to *committed* resources.
- **Mechanisms** are used to <u>*constraint*</u> traffic arrivals injected into the network
- Mechanisms
  - Generic Cell Rate Algorithm
  - Leaky bucket
  - Token bucket

Generic Cell Rate Algorithm (GCRA)

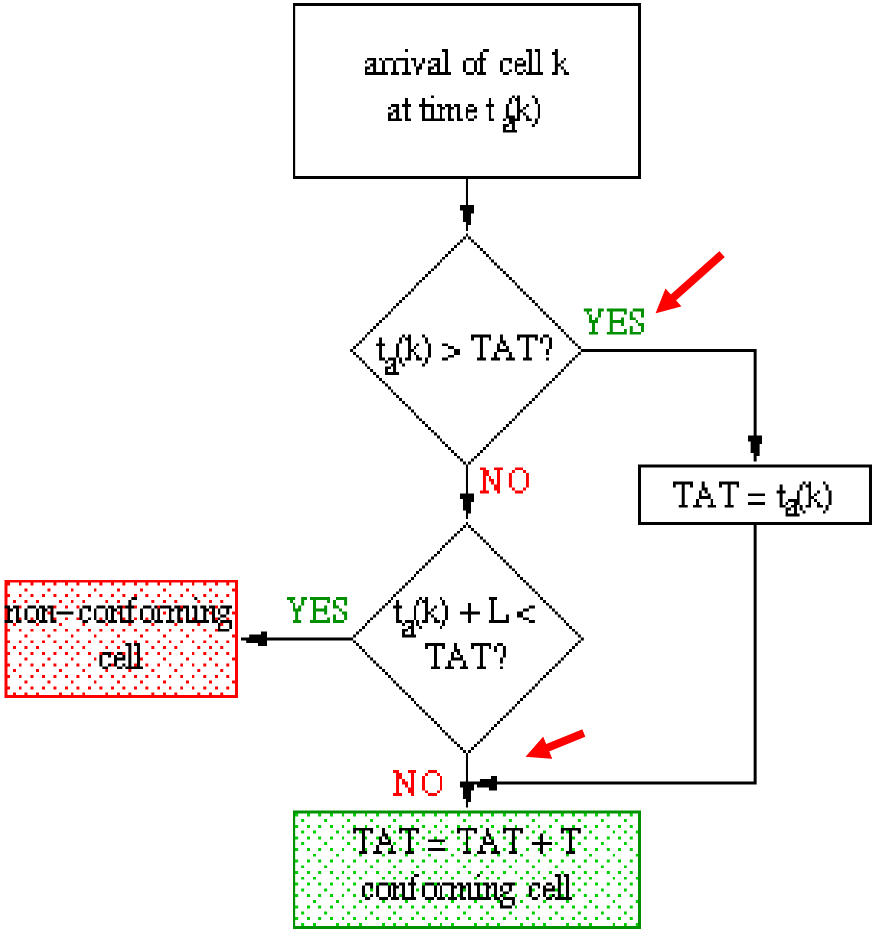
- Aka. Virtual Scheduling
- Used for conformance check
- Two parameters
  - *T* - Increment
  - *L* - Limit
- Regulated arrivals (T) with occasional *earlier* arrivals but not earlier than the limit (L)

Conformance Check: cell arrival variation control

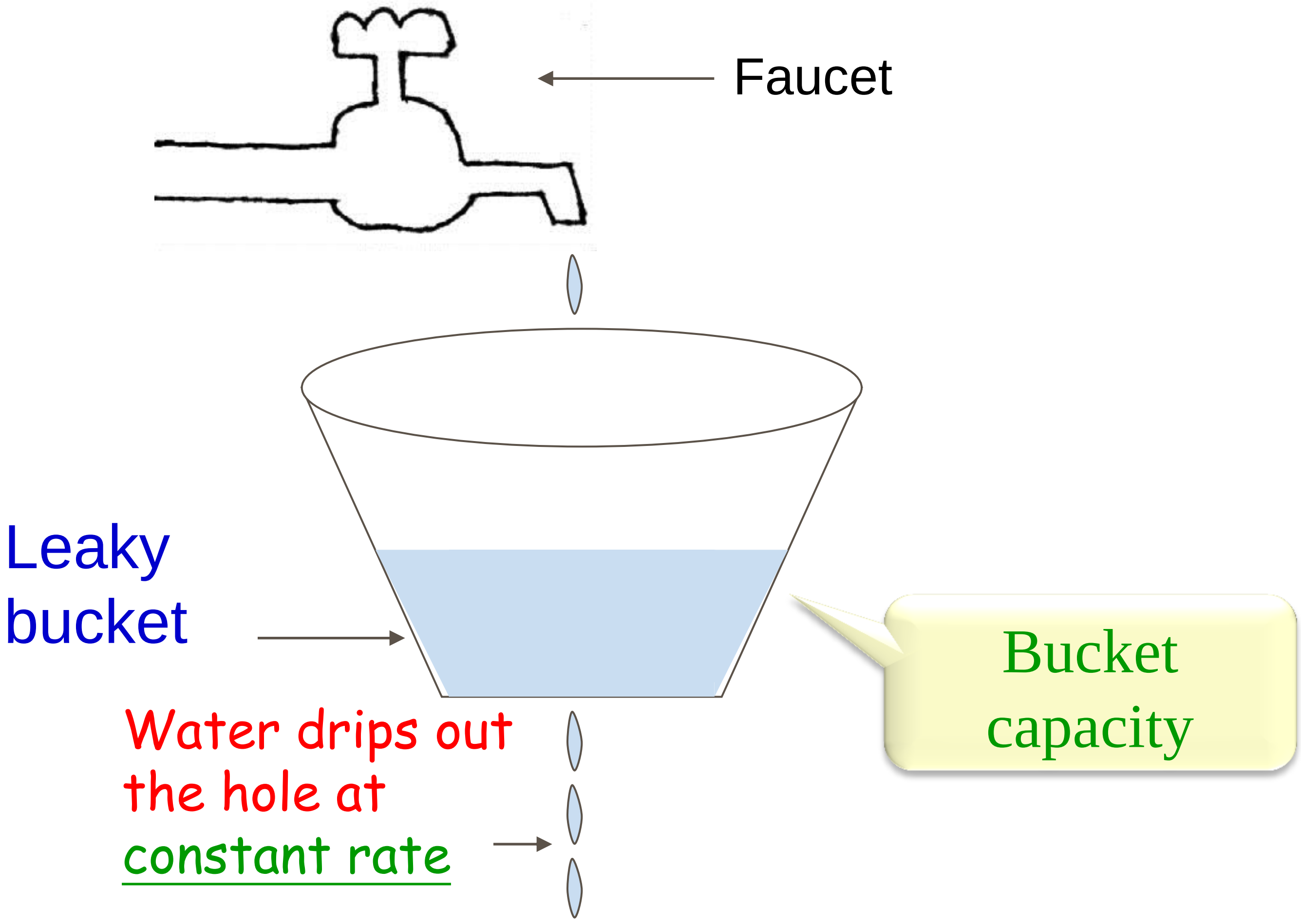


GCRA Algorithm

- Cell arrival time vs. TAT (Theoretical Arrival Time)
  - (1) If cell arrival time > curnt_TAT,
    cell is *conforming* and
    next_TAT = cell arrival time + T // slow arrival
  - (2) If cell arrival time < curnt_TAT but *not* earlier than limit L,
    cell is *conforming*;
    next_TAT = curnt_TAT + T // early arrival but ok
  - (3) Otherwise, cell arrives too earlier than the limit L and *non-conforming*



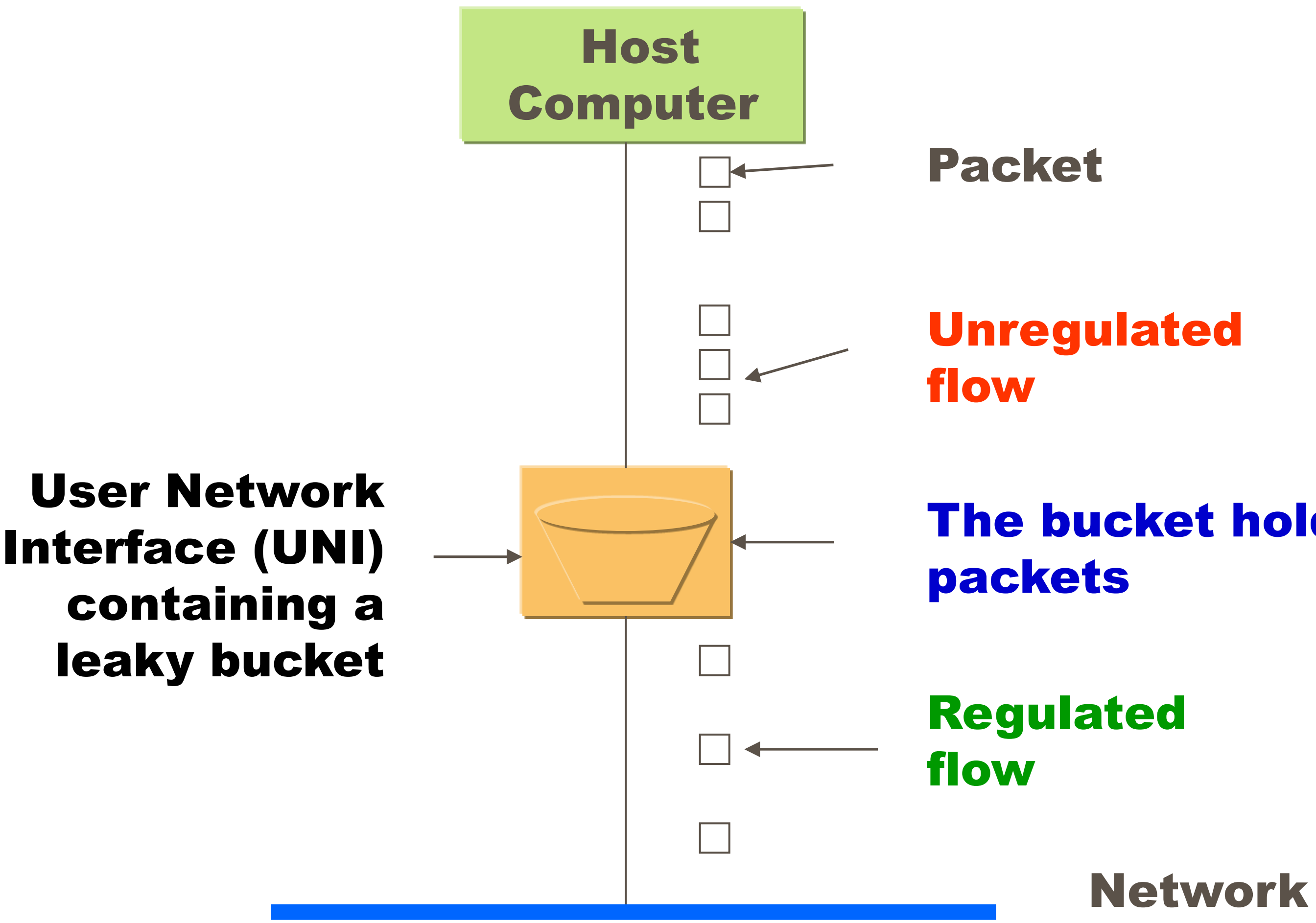
A "Leaky Bucket" with Water

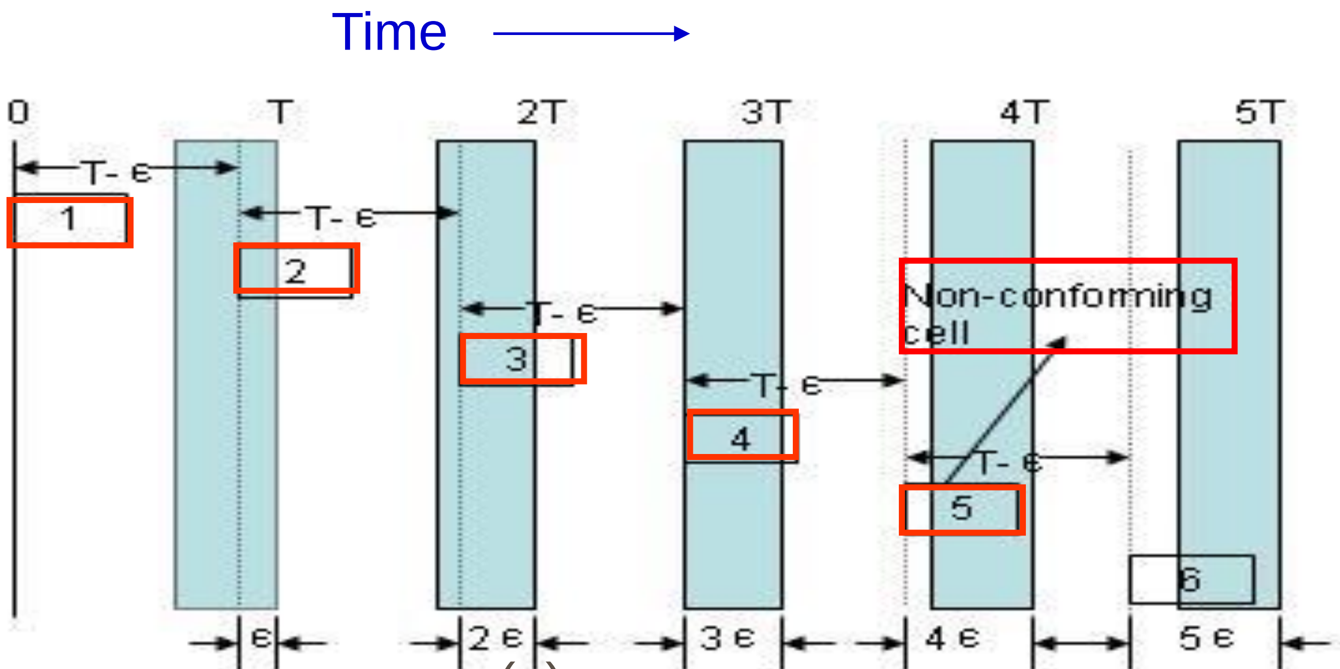


“Continuous-State” Leaky Bucket

- Draining rate - drain-out at a continuous rate of one unit of content per time unit
- Content increased by the increment T for each conforming cell (i.e. work load brought by a cell)
- Follow GCRA
- Bucket capacity is T+L

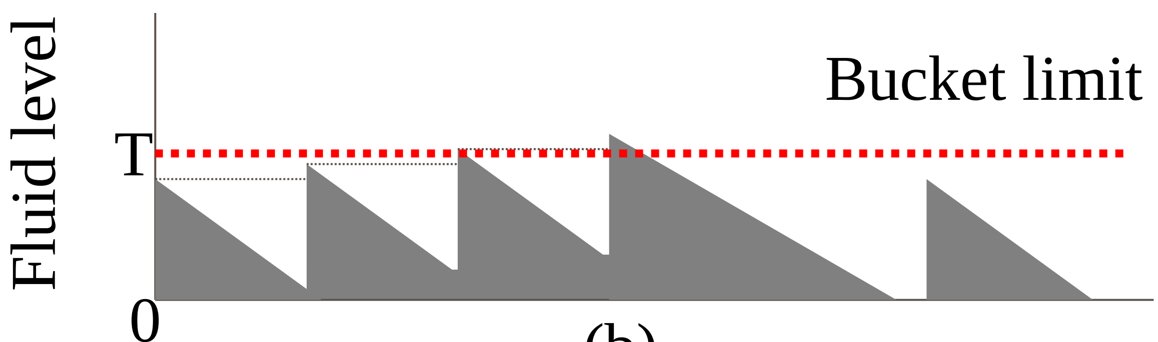
A Leaky Bucket with "Packet"





(a)

(a) Packet arrivals - a sender trying to cheat



(b)

(b) Viewed in terms of leaky bucket

Maximum Burst Tolerance (MBS)

- The maximum number of cells that can be transmitted back-to-back given the peak rate (PCR) (bits/sec) constraint
  - L: allowable amount of early arrivals (bucket capacity = T+L)
  - δ: cell transmission time (sec/cell)
  - T: 1/PCR (sec/cell)
- Assume drainage does not start until the first cell has been entirely transmitted.
- Used in traffic conformance check (i.e., cell variation delay tolerance (CVDT))

- N: the number of conforming cells arrived back-to-back at the peak cell rate.
- Total_in – Total_out = left_over_in_bucket

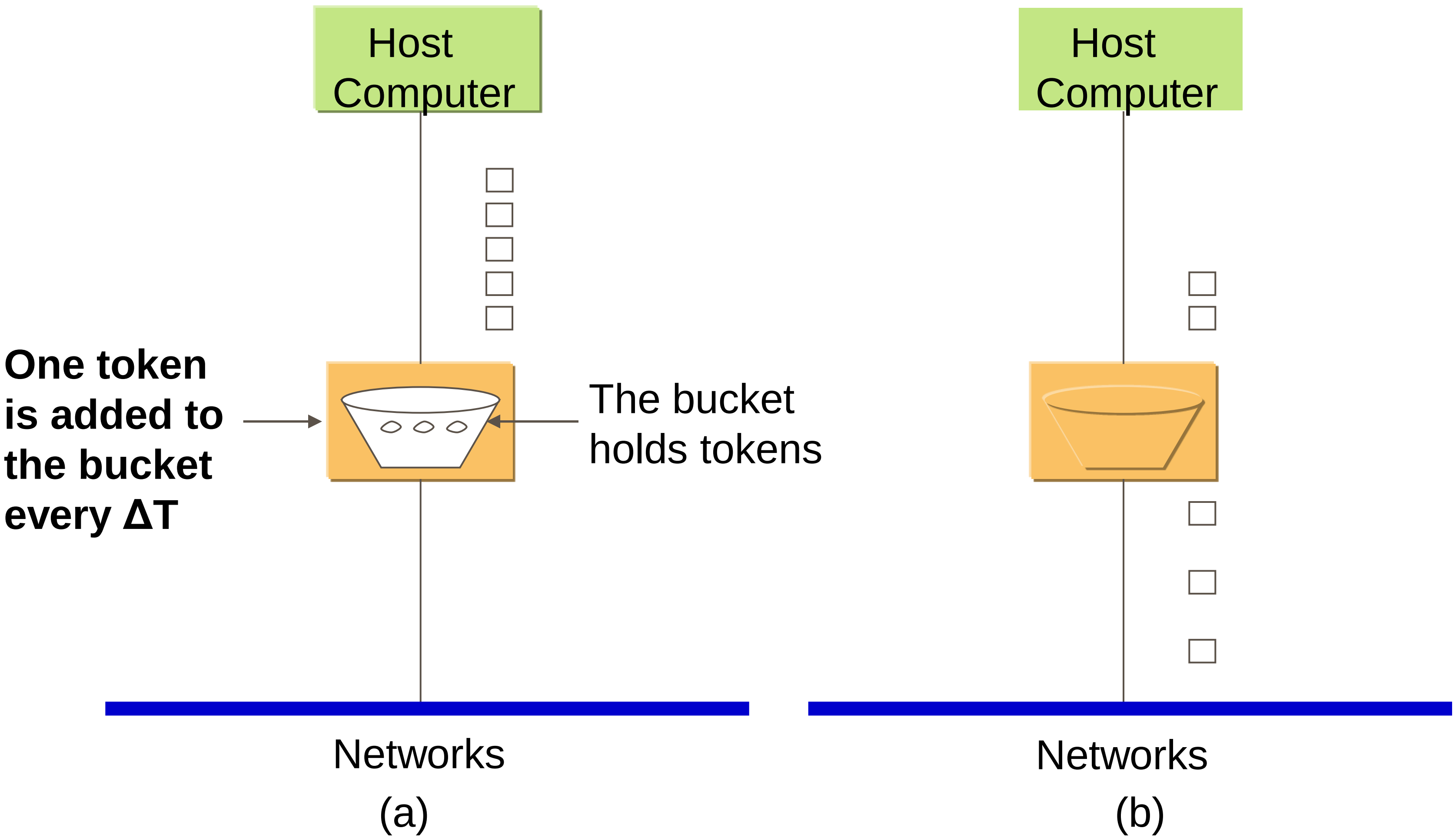
$$NT-(N-1)\delta = T+L$$

$$N = 1+\frac{L}{T-\delta}$$

SCR: sustainable cell (packet) rate

- An *upper* bound of "average rate"
- Average rate = total cells transmitted / duration of the connection

Token Bucket (1/2)



Token Bucket (2/2)

- The bucket holds tokens
- Tokens are generated by a clock at the rate of one token every ΔT sec.
- For a packet to be transmitted, it must capture and destroy a token.
- The bucket can save up tokens to allow burst arrivals and burst departure.
  - The maximum number of tokens saved up is the **bucket size.**
  - The maximum number of packets can be sent at once is the **bucket size.**
- Tokens are thrown away when the bucket fills up.
- Packets are never discarded.

Token Bucket: the length of the maximum rate burst

- Let
  - M : the maximum output rate of the token bucket
  - ρ : token generation rate
  - C: bucket size
  - S: duration to output all data
- Traffic Input = Traffic output (the balance equation)

  C (bits) + ρS (bits/sec x sec) = MS (bits/sec x sec)

$$S = C/(M - \rho)$$

- the time duration having the maximum output rate

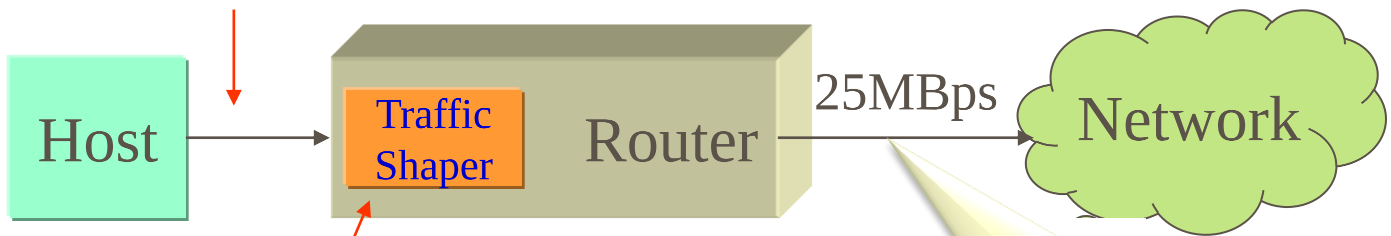
- Example: S=250KB/(25MB-2MB)

  =10.8ms

Traffic Control Example: using Leaky Bucket

- A burst of up to 1MBytes can be handled without data loss

1MBytes per 40ms



- Host sending rate 25MBps

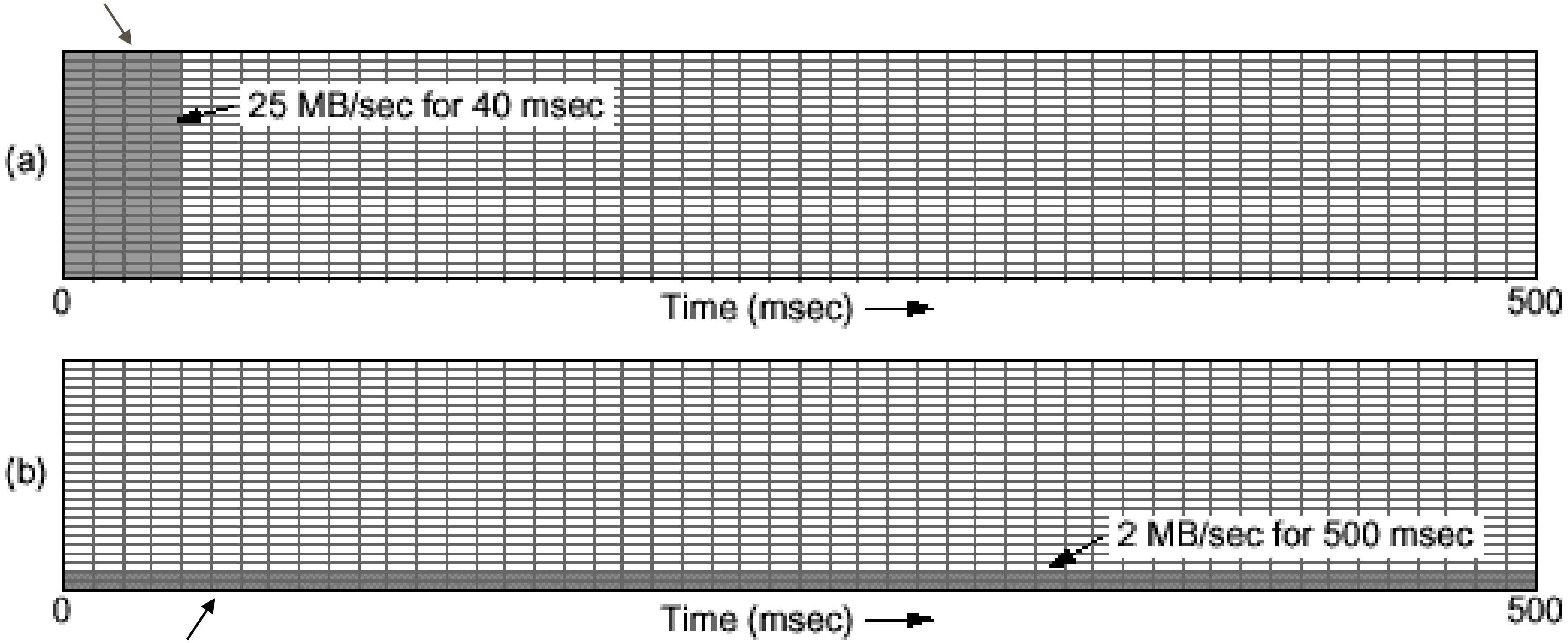
- Avg rate 2MBps
- Peak rate 25MBps

ρ = 2MBps (token generation rate)
C = 1MB (token bucket size)

- Want to enforce this sustainable rate i.e. token generation rate
- Allow burst arrivals

Input : 25MB/40ms

Leaky Bucket



Output: constant 2MBps

Legend:

- 10ms per column

ρ = 2MBps (drain-out rate)
C = 1MB (leaky bucket size)

25M x 40ms = 2M x Y
Y = 500ms

Traffic Control Example: using Token Bucket (1/3)

- Assume
  - Token bucket is full (of token) when the N=1MB burst arrives
  - Token bucket size C= 250KBytes;
  - The bucket drains at the full M=25MB/sec for 11msec
  - ρ: token arrival rate (bytes/sec) = 2MB/sec
- S: maximum allowable burst length (sec)

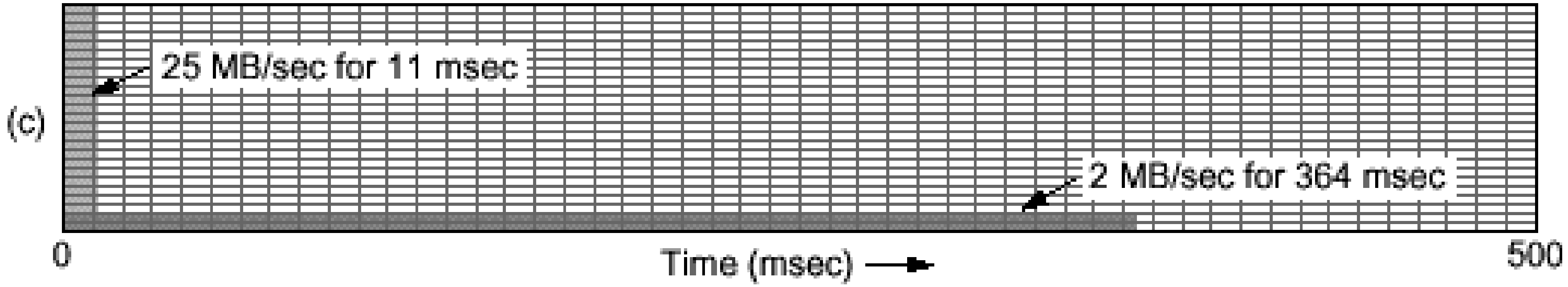
  -> 25MB x S = 250KB + 2MB x S

  -> S = 250KB/(25MB-2MB) =10.8ms
- Drain out duration Y (sec)

  -> MS + ρY = N

  -> 25MB x 10.8ms + 2MB x Y= 1MB

  -> Y = 364.5ms



Traffic Control Example: using Token Bucket (2/3)

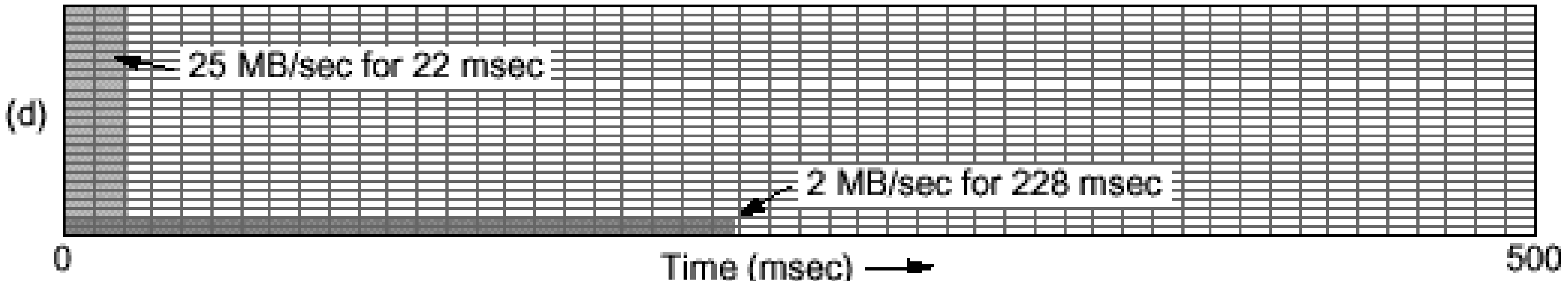


Figure 5.25 (d) - Find the duration with maximum output rate 25Mps

- Given token bucket size = 500KB
- The maximum allowable burst size (sec) (i.e., S=C/(M-ρ))
  -> S=500KB/(25MB-2MB)=21.739ms~22ms
- The duration with average output rate 2Mps
  -> 25MB x 22ms + 2MB x Y = 1MB
  -> Y=228.26ms

Traffic Control Example: using Token Bucket (3/3)

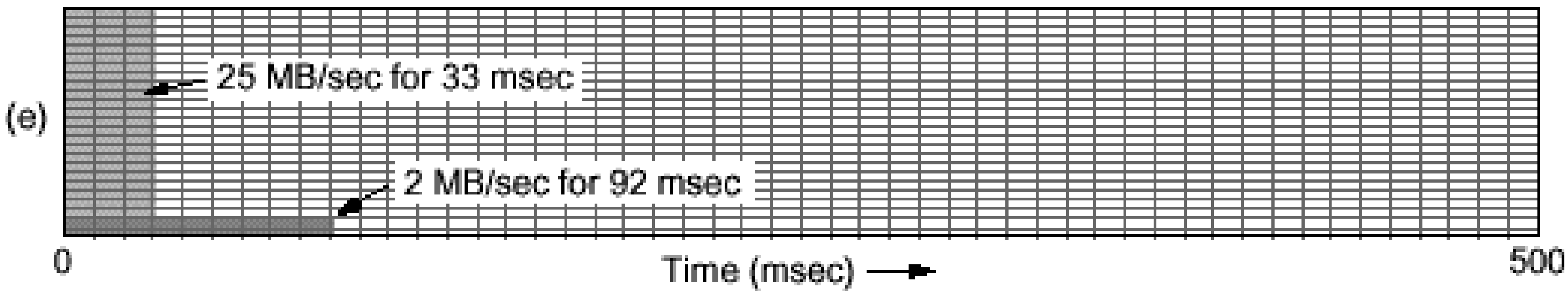


Figure 5.25 (e) - Find the duration with maximum output rate 25Mps

- Given token bucket size = 750KB
- The maximum allowable burst size (sec) (i.e., S=C/(M-ρ))
  -> S=750KB/(25MB-2MB)=32.609~33ms
- The duration with average output rate 2Mps
  -> 25MB x 33ms+2MB x Y = 1MB
  -> Y=92.391ms

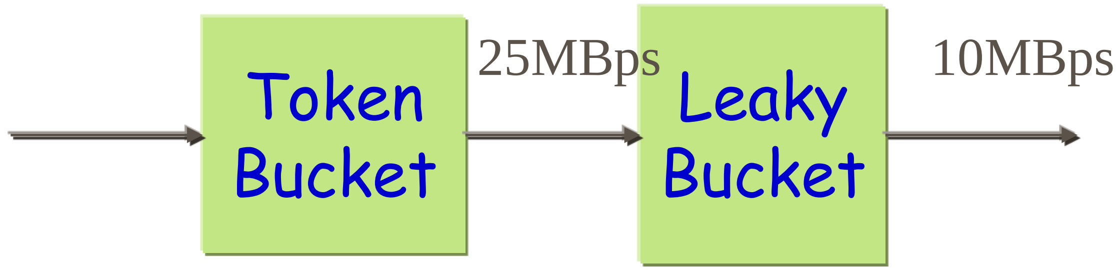
Discussion

- 250KB -> 25MB x 10.8ms, Y = 364.5ms
- 500KB -> 25MB x 22ms, Y=228.26ms
- 750KB -> 25MB x 33ms, Y=92.391ms

- Which input arrivals are more bursty to the network?
  - Allow “bursts”!

Concatenate a token bucket with a leaky bucket

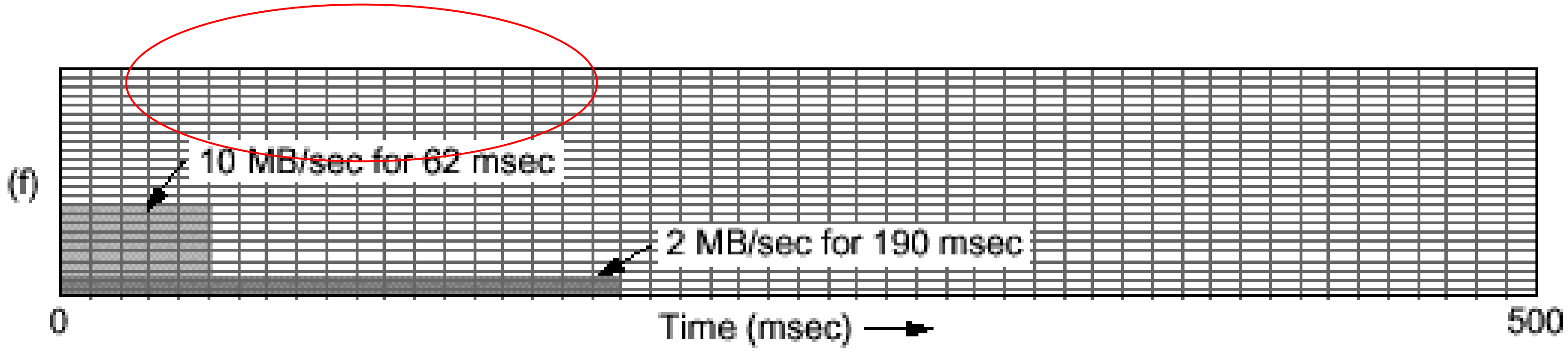
- Figure 5.25 (f)
  - To further "smooth" the traffic (less bursty output)
  - To concatenate a token bucket with a leaky bucket



- **ρ = 2MBps** (token generation rate)
- **C = 1MB** (token bucket size)

Concatenate a token bucket with a leaky bucket (cont'd)

- TokenBucket_max_output_rate x max_rate_duration + token_gen_rate x avg_rate_duration
  = LeakyBucket_max_output_rate x (max_rate_duration + avg_rate_duration)
  -> $MS + \rho * X = L * (S + X)$
- Average rate duration X
  -> $X = (M-L)*S/(L-\rho)$
  - From figure 5.25 (d), we have S=22ms
  -> X = (25MB-10MB)*22ms/(10MB-2MB)
  = 40.761ms
- The duration of maximum output rate from leaky bucket is S+X
  -> 40.71+22 = 62.71ms

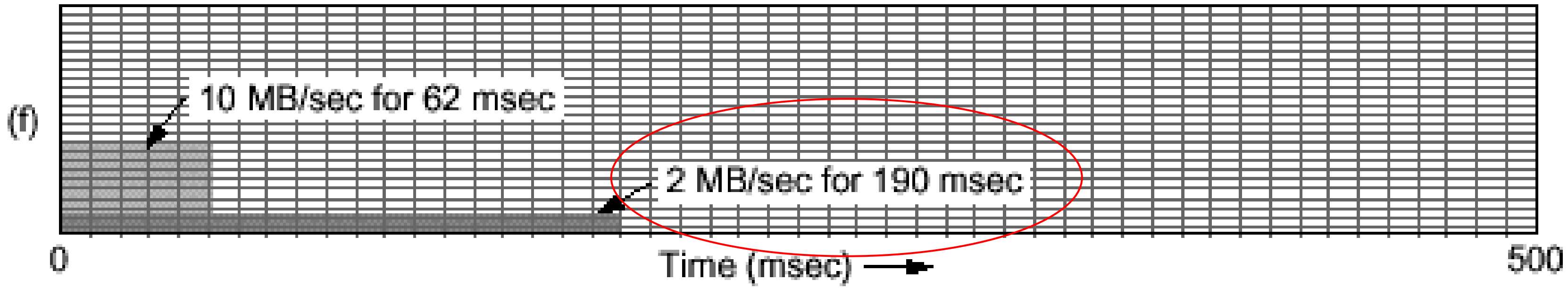


Concatenate a token bucket with a leaky bucket (cont'd)

- The duration of having average rate output after token+leaky buckets traffic shaping
  - Total_traffic_at_leaky_bucket_max_rate + total_traffic_at_avg_rate = total_traffic_amount

-> $L*(S+X)+\rho*Y=1MB$

-> Y=(1MB-10MB*62.5ms)/2MB=187.5ms



Compare Leaky bucket and Token bucket

- Leaky bucket
  - does *not* allow idle hosts to *save* up permits or tokens
  - does *not* allow *bursts of outputs*
  - enforces a rigid output pattern at the average rate no matter how bursty the traffic is.
  - Packet lost when the bucket is full
- Token bucket
  - allows *burst arrivals* and *bursty output*
  - Sustainable rate and peak rate
  - No packet loss

Implementations

- Leaky bucket algorithm
  - conformance check and traffic shaping
- Token bucket algorithm
  - transmission control
  - allows burst arrivals
- Traffic smoothing *between* routers
  - Self-similar traffic
  - Individual flows and traffic aggregate
- Traffic regulation of host traffic output

Summary

- Policing is necessary to ensure that traffic sources conform to the committed rates.
- A powerful mechanism in traffic regulation
- Leaky bucket is often used for peak rate regulation.
- Token bucket is often used to allow instantaneous bursty arrival while regulating the traffic on their average behaviors.

The end. ☺

Reference

- S. Tanenbaum, "Computer Networks," 4th edition, Prentice Hall, 2003.