Modeling Traffic and Access Control Protocols

Access Control

- Computer traffic is seldom uniform and is characterized by periods of burstiness. Traffic bursts tax the network resources such as switch buffers and lead to network congestion and data loss. Because it is impossible for the network to accept only uniform traffic, mechanisms have been proposed to regulate or smooth out these bursts.

Access Control

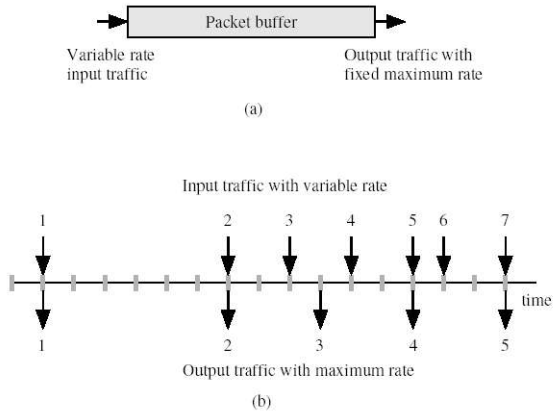
- Assume a connection has been established, it is necessary to monitor and control the traffic actually generated by the connection to ensure it conforms to the traffic descriptors originally specified.
- The procedure of controlling the traffic during the period of a connection is variously referred to as access control, credit management, or traffic “policying”.

The Leaky Bucket Algorithm

The Leaky Bucket Algorithm

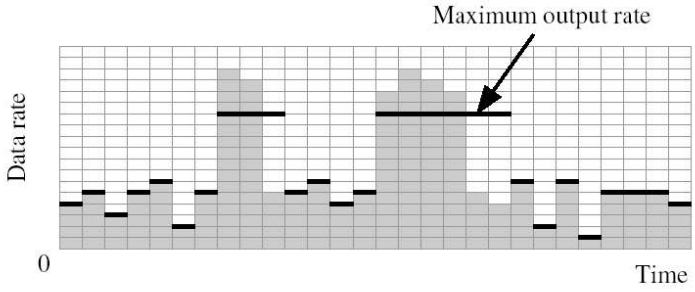
- Leaky bucket is a **rate-based** algorithm for controlling the **maximum rate** of traffic arriving from a source. If the input data rate is less than the maximum rate specified by the algorithm, leaky bucket accepts the data.
- If the input data rate exceeds the maximum rate, leaky bucket passes the data at the maximum rate and excess data is buffered.
- If the buffer is full then excess data is discarded.

The Leaky Bucket Algorithm



Leaky bucket: (a) Packet buffer. (b) Packet arrival and departure.

The Leaky Bucket Algorithm



Control of data rate by the leaky bucket algorithm. Data rate at the input is indicated by the grey areas; and data rate at the output is indicated by the black lines.

The Leaky Bucket Algorithm

To model the source burstiness in a simple manner, we assume the data source has the following parameters:

- λ_a Average data rate of source.
- σ Source burst rate when it is nonconforming.

The Leaky Bucket Algorithm

On the other hand, packets leave the buffer with an output rate λ_{out} given by

$$\lambda_{out} = \begin{cases} \min(\lambda_{in},\ \lambda_b) & \text{buffer is empty} \\ \lambda_b & \text{buffer is not empty} \end{cases}$$

Notice that the output data rate is governed by the state of the data buffer and not by the input data rate.

M/M/1/B Model

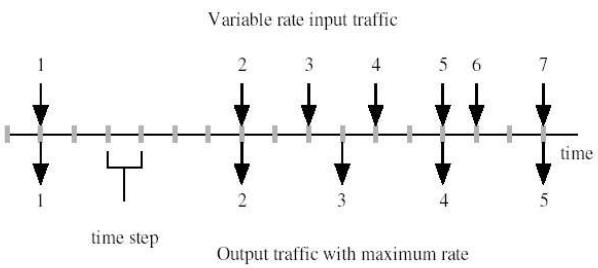
The time step equal to the inverse of the maximum data rate on the line.

$$T = \frac{1}{\lambda_l}$$

T is measured in units of seconds and λ_l is the *maximum* input line rate (in units of packets/s) such that

$$\lambda_b < \lambda_l$$

M/M/1/B Model



The leaky bucket algorithm where the time step is chosen equal to the inverse of the maximum line rate.

M/M/1/B Model Arrival Probability

Number of packets arriving in time frame t is

$$N(in) = \lambda_{in}\, t$$

But during this time period, we have N time steps with $N = t/T$.

M/M/1/B Model Arrival Probability

The number of arriving packets is estimated also using the binomial distribution as

$$N(in) = a\,N = a\,\frac{t}{T}$$

From the above two equations we get

$$a = \lambda_{in}\,T = \frac{\lambda_{in}}{\lambda_l}$$

M/M/1/B Model Departure Probability

Using a similar argument, the packet departure probability (c) is give by

$$c = \frac{\lambda_{out}}{\lambda_l}$$

M/M/1/B Transition Matrix

Transition matrix is $(B+1) \times (B+1)$ and is given by

$$\mathbf{P} = \begin{bmatrix} f_0 & bc & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ ad & f & bc & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & ad & f & bc & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & ad & f & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & f & bc & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & ad & f & bc & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & ad & f & bc \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & ad & 1-bc \end{bmatrix}$$

where $b = 1 - a$, $d = 1 - c$, $f_0 = 1 - ad$, and $f = 1 - ad - bc$.

Leaky Bucket Performance (*M/M/1/B* Model)

The throughput of the leaky bucket algorithm when the $M/M/1/B$ model is given by

$$Th = c\ (1 - b\,s_0)$$

The throughput is measured in units of packets/time step.

Leaky Bucket Performance (*M/M/1/B* Model)

The average number of lost or tagged packets per time step is obtained using the results of the $M/M/1/B$ queue

$$N_a(lost) = s_B\ a\ d$$

The lost traffic is measured in units of packets/time step.

The packet loss probability is given by

$$L = \frac{N_a(lost)}{N_a(in)} = \frac{s_B\ a\ d}{\text{a}}$$

Leaky Bucket Performance (*M/M/1/B* Model)

Using Little's result, the average wait time in the buffer is

$$W = \frac{Q_a}{Th}$$

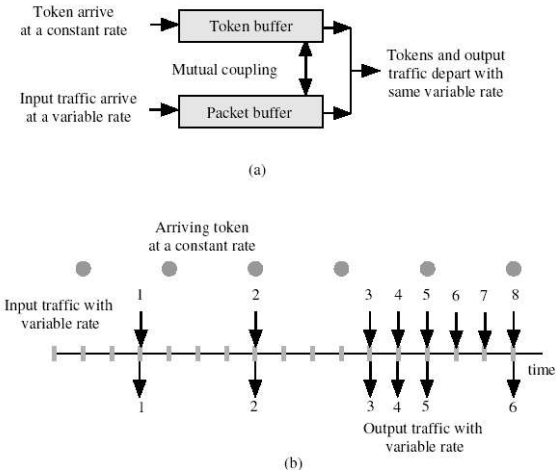
The wait time is measured in units of time steps.

The Token Bucket Algorithm

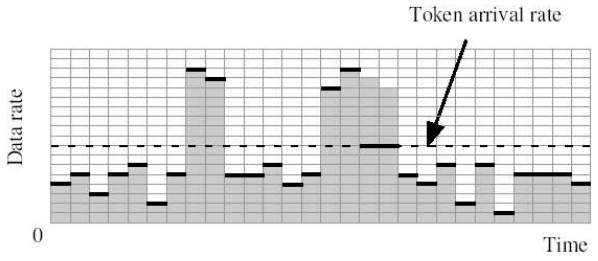
The Token Bucket Algorithm

- Token bucket is a **credit-based** algorithm for controlling the **volume** of traffic arriving from a source.
- Tokens are issued at a constant rate and arriving packets can leave the system only if there are tokens available in the token buffer.

The Token Bucket Algorithm



The Token Bucket Algorithm



Control of data rate by the token bucket algorithm. Data rate at the input is indicated by the grey areas, data rate at the output is indicated by the black lines and the dashed line is the token issue rate.

Modeling the Token Bucket Algorithm

- We perform Markov chain analysis of the token bucket algorithm.
- The states of the Markov chain represent the number of tokens stored in the token buffer or bucket; and the number of packets stored in the packet buffer.
- We cannot model the token buffer separately from the packet buffer since the departure from one buffer depends on the state of occupancy of the other buffer. In a sense we have two mutually coupled queues

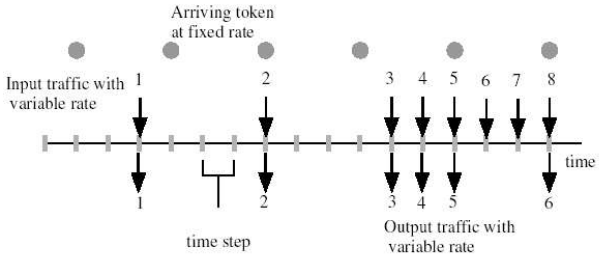
Single Arrival/Departures Model

In this approach to modeling the token bucket algorithm we take the time step equal to the inverse of the maximum data rate on the line

$$T = \frac{1}{\lambda_l}$$

where T is measured in units of seconds and λ_l is the *maximum* input line rate in units of packets/s.

Modeling the Token Bucket Algorithm



The token bucket algorithm where the time step is chosen equal to the inverse of the maximum line rate.

Modeling the Token Bucket Algorithm

To model the source burstiness in a simple manner, we assume the data source has the following parameters:

- λ_a Source average data rate.
- σ Source burst rate.

Modeling the Token Bucket Algorithm

λ_a and σ typically satisfy the relations

$$\begin{aligned} \lambda_a &< \lambda_t \\ \sigma &> \lambda_t \end{aligned}$$

where λ_t is the token arrival rate.

Modeling the Token Bucket Algorithm

At a given time step a maximum of one token could arrive at or leave the token buffer. Thus the token buffer is described by an $M/M/1/B$ queue. The token arrival probability (a) is given by

$$a = \frac{\lambda_t}{\lambda_l}$$

We also define $b = 1 - a$ as the probability that a token does not arrive.

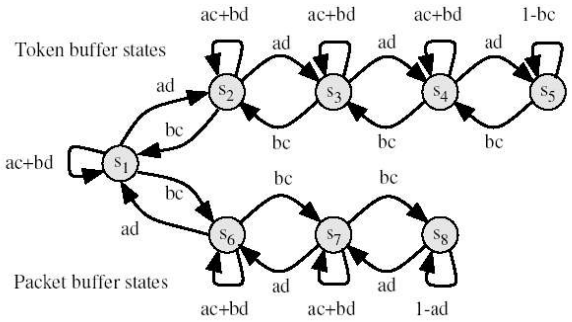
Modeling the Token Bucket Algorithm

Since a token leaves the token buffer each time a packet arrives, the token departure probability c is given by

$$c = \frac{\lambda_{in}}{\lambda_l}$$

We also define $d = 1 - c$ as the probability that a token does not leave the token buffer.

Modeling the Token Bucket Algorithm



Transition diagram for the system comprising the token and packet buffers. Token buffer size is $B_t = 4$ and packet buffer size is $B_p = 3$.

Modeling the Token Bucket Algorithm

The meaning of each state is shown below.

State	Token buffer occupancy	Packet buffer occupancy
s_1	0	0
s_2	1	0
s_3	2	0
s_4	3	0
s_5	4	0
s_6	0	1
s_7	0	2
s_8	0	3

Modeling the Token Bucket Algorithm

Transition matrix is $(B_t + B_p + 1) \times (B_t + B_p + 1)$. Here $B_t = 4$ and $B_p = 3$.

$$\mathbf{P} = \begin{bmatrix} f & bc & 0 & 0 & 0 & ad & 0 & 0 \\ ad & f & bc & 0 & 0 & 0 & 0 & 0 \\ 0 & ad & f & bc & 0 & 0 & 0 & 0 \\ 0 & 0 & ad & f & bc & 0 & 0 & 0 \\ 0 & 0 & 0 & ad & 1-bc & 0 & 0 & 0 \\ bc & 0 & 0 & 0 & 0 & f & ad & 0 \\ 0 & 0 & 0 & 0 & 0 & bc & f & ad \\ 0 & 0 & 0 & 0 & 0 & 0 & bc & 1-ad \end{bmatrix}$$

where $b = 1 - a$, $d = 1 - c$ and $f = ac + bd = 1 - ad - bc$.

Token Bucket Performance

- The throughput of the token bucket algorithm is the average number of packets per time step that are produced without being tagged or lost.
- To find the throughput we must study all the states of the combined system. It is much easier to obtain the throughput using the traffic conservation principle after we find the lost traffic.

Token Bucket Performance

Packets are lost or tagged for future discard if they arrive when the packet buffer is full and no token arrives at that time step.

The average number of lost or tagged packets per time step is

$$N_a(lost) = b\,c\;\; s_{B_t+B_p+1}$$

The lost traffic is measured in units of packets/time step.

Token Bucket Performance

The average number of packets arriving per time step is given by

$$N_a(in) = c$$

Token Bucket Performance

The packet loss probability is

$$L = \frac{N_a(lost)}{N_a(in)}$$

$$= \frac{b\,c\;\; s_{B_t+B_p+1}}{c}$$

$$= b\, s_{B_t+B_p+1}$$

Token Bucket Performance

The throughput of the token bucket algorithm is obtained using the traffic conservation principle

$$\begin{aligned} Th &= N_a(in) - N_a(lost) \\ &= c - b\,c\; s_{B_t+B_p+1} \\ &= c\left(1 - b\, s_{B_t+B_p+1}\right) \end{aligned}$$

The throughput is measured in units of packets/time step.

Token Bucket Performance

The packet acceptance probability p_a or η of the token bucket algorithm is

$$\begin{aligned} p_a &= \frac{Th}{N_a(in)} \\ &= 1 - L \\ &= 1 - b\, s_{B_t+B_p+1} \end{aligned}$$

Token Bucket Performance

The average queue size for the tokens is given by

$$Q_t = \sum_{i=1}^{B_t} i\; s_{i+1}$$

Notice the range of values of the state index in the above equation.

Token Bucket Performance

Similarly, the average queue size for the packets is given by

$$Q_p = \sum_{i=1}^{B_p} i\; s_{i+B_t+1}$$

Notice the range of values of the state index in the above equation.

Token Bucket Performance

Using Little's result, the average wait time or delay for the packets in the packet buffer is

$$W = \frac{Q_p}{Th}$$

The wait time is measured in units of time steps.