Quality of Service

Quality of Service

- Application requirements
- Traffic shaping
- Packet scheduling
- Admission control
- Integrated services
- Differentiated services

Application Requirements (1)

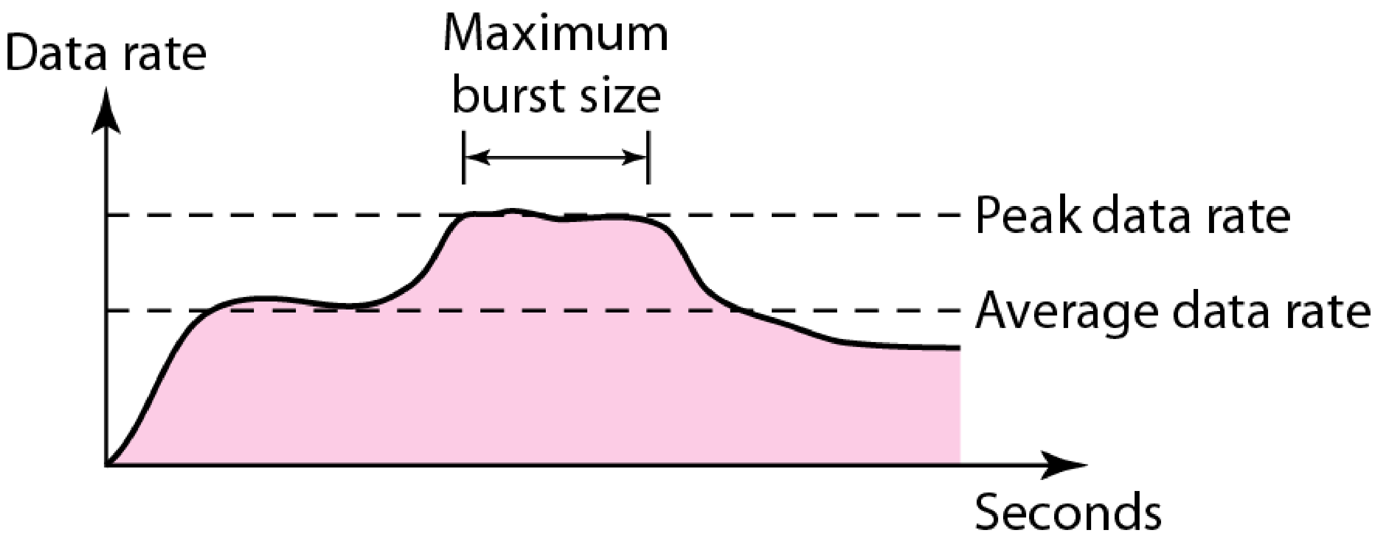
Application	Bandwidth	Delay	Jitter	Loss
Email	Low	Low	Low	Medium
File sharing	High	Low	Low	Medium
Web access	Medium	Medium	Low	Medium
Remote login	Low	Medium	Medium	Medium
Audio on demand	Low	Low	High	Low
Video on demand	High	Low	High	Low
Telephony	Low	High	High	Low
Videoconferencing	High	High	High	Low

How stringent the quality-of-service requirements are.

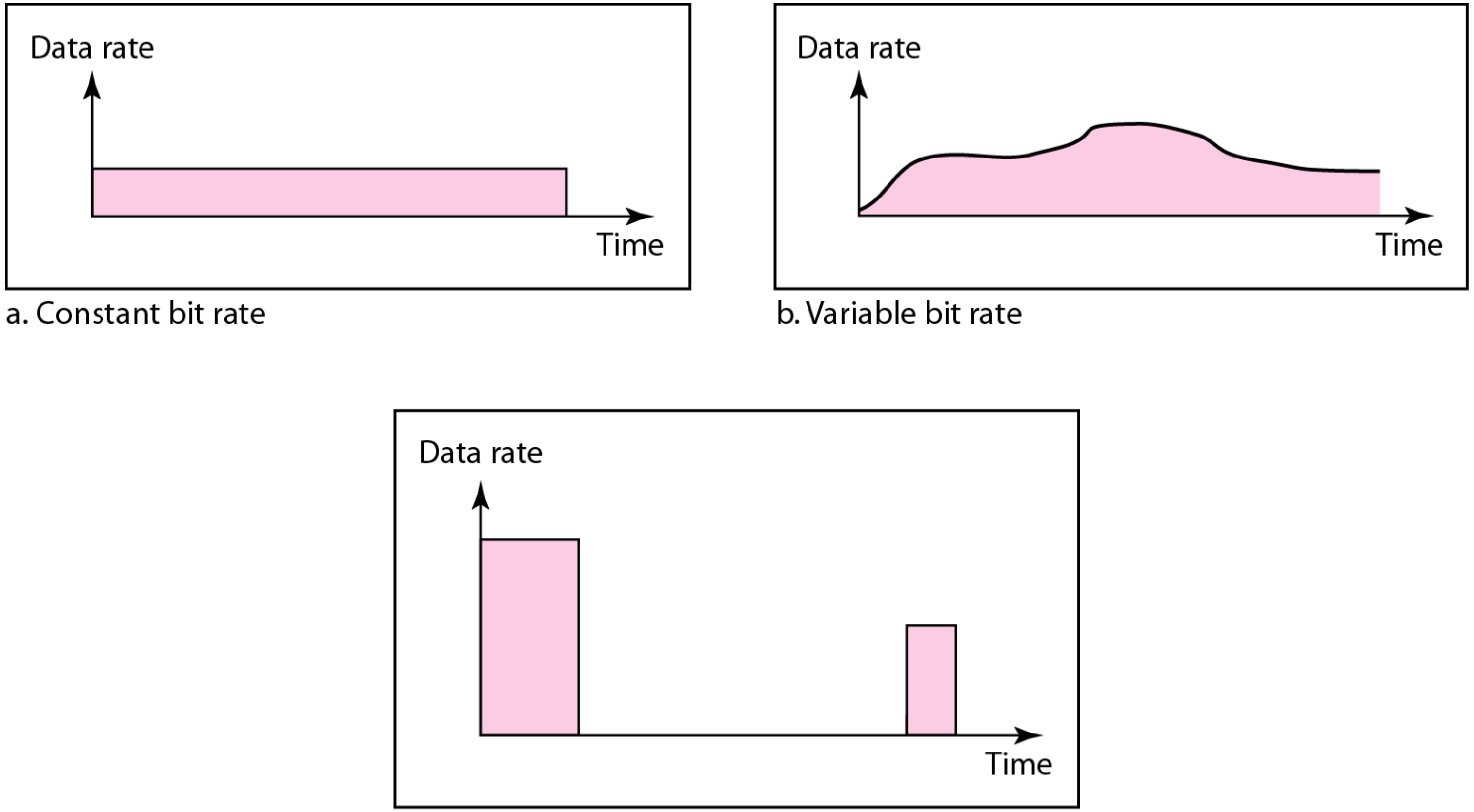
Flow Classes

1. Constant bit rate
   - Telephony
2. Real-time variable bit rate
   - Compressed videoconferencing
3. Non-real-time variable bit rate
   - Watching a movie on demand
4. Available bit rate
   - File transfer

Traffic descriptors



Three traffic profiles



a. Constant bit rate

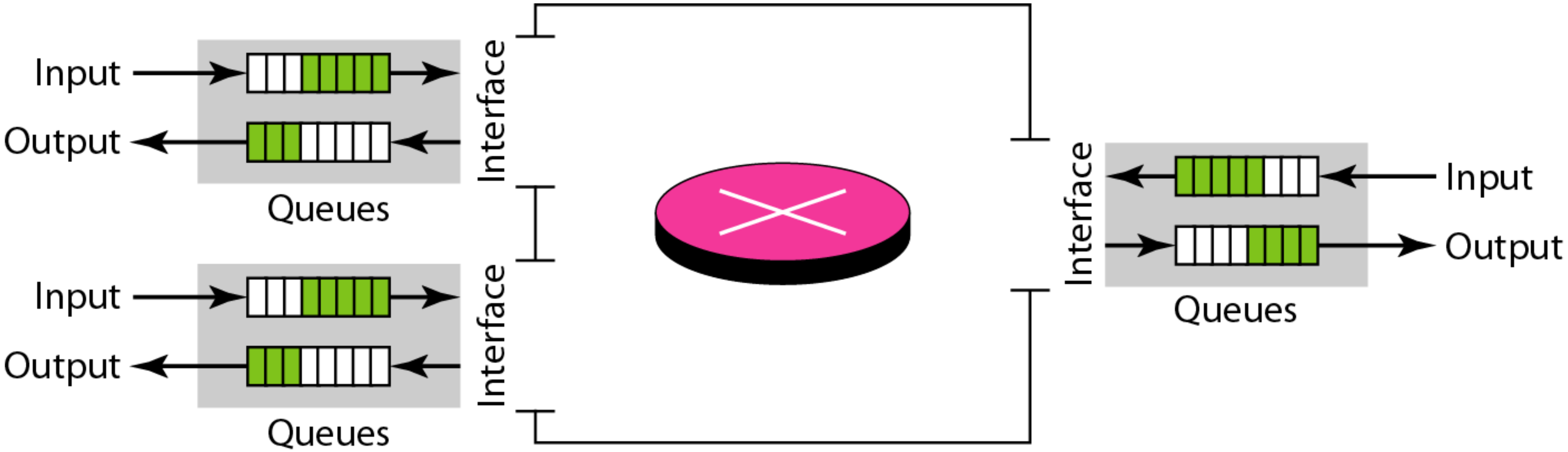
b. Variable bit rate

c. Bursty

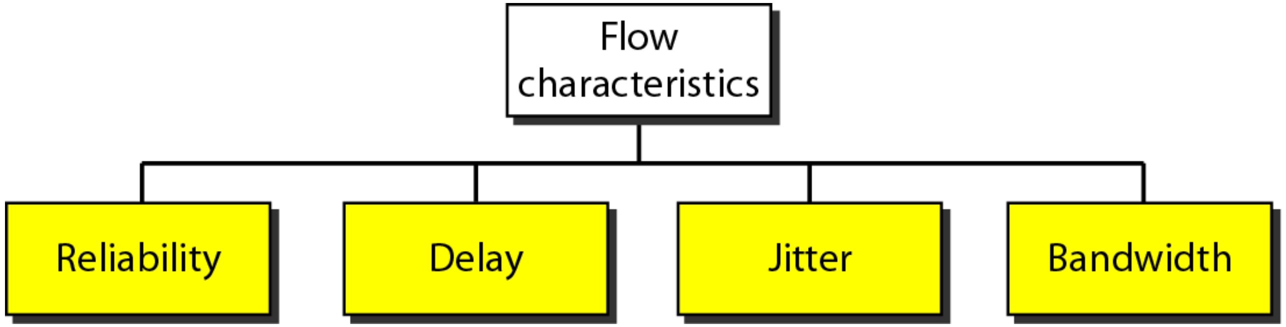
QoS Improvement

- QoS Improvement
  - Scheduling
    - FIFO
    - Priority
    - Weighted Fair Queue
  - Traffic Shaping
    - Leaky Bucket
    - Token Bucket
  - Resource Reservation
  - Admission Control

Queues in a router



Flow characteristics



Flow Characteristics

- Reliability:
  - Flow needed in order to deliver the packets safe and sound to the destination.
- Delay:
  - Should be minimum in case of telephony, audio conference, video conference, remote logging
  - Less important in case of File transfer, email, etc.
- Jitter:
  - Variation in the delay for packets belonging to the same flow.
- Bandwidth:
  - Maximum rate of data transfer across a given path

Packet Scheduling

Kinds of resources can potentially be reserved for different flows:

1. Bandwidth.
2. Buffer space.
3. CPU cycles.

Popular Algorithms:

1. FIFO
2. Priority
3. Weighted Fair / Round Robin

FIFO queue

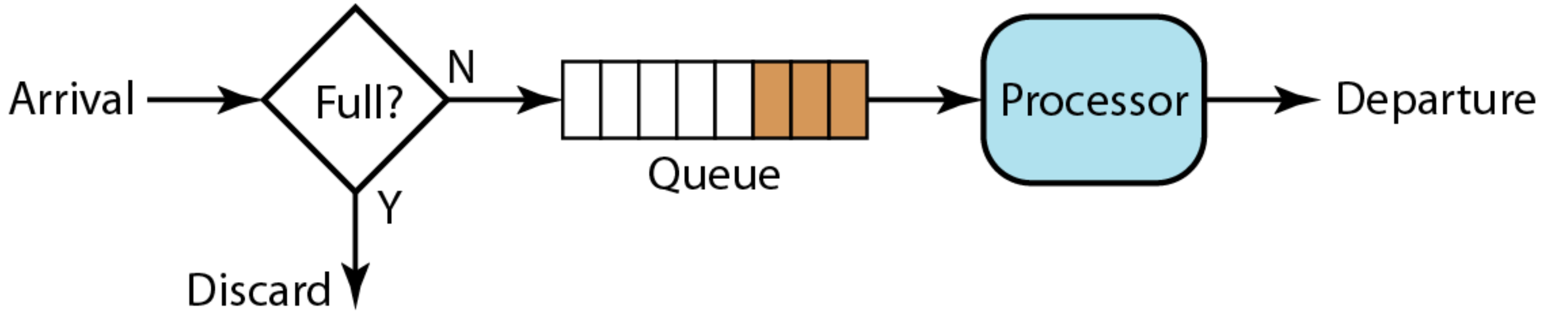
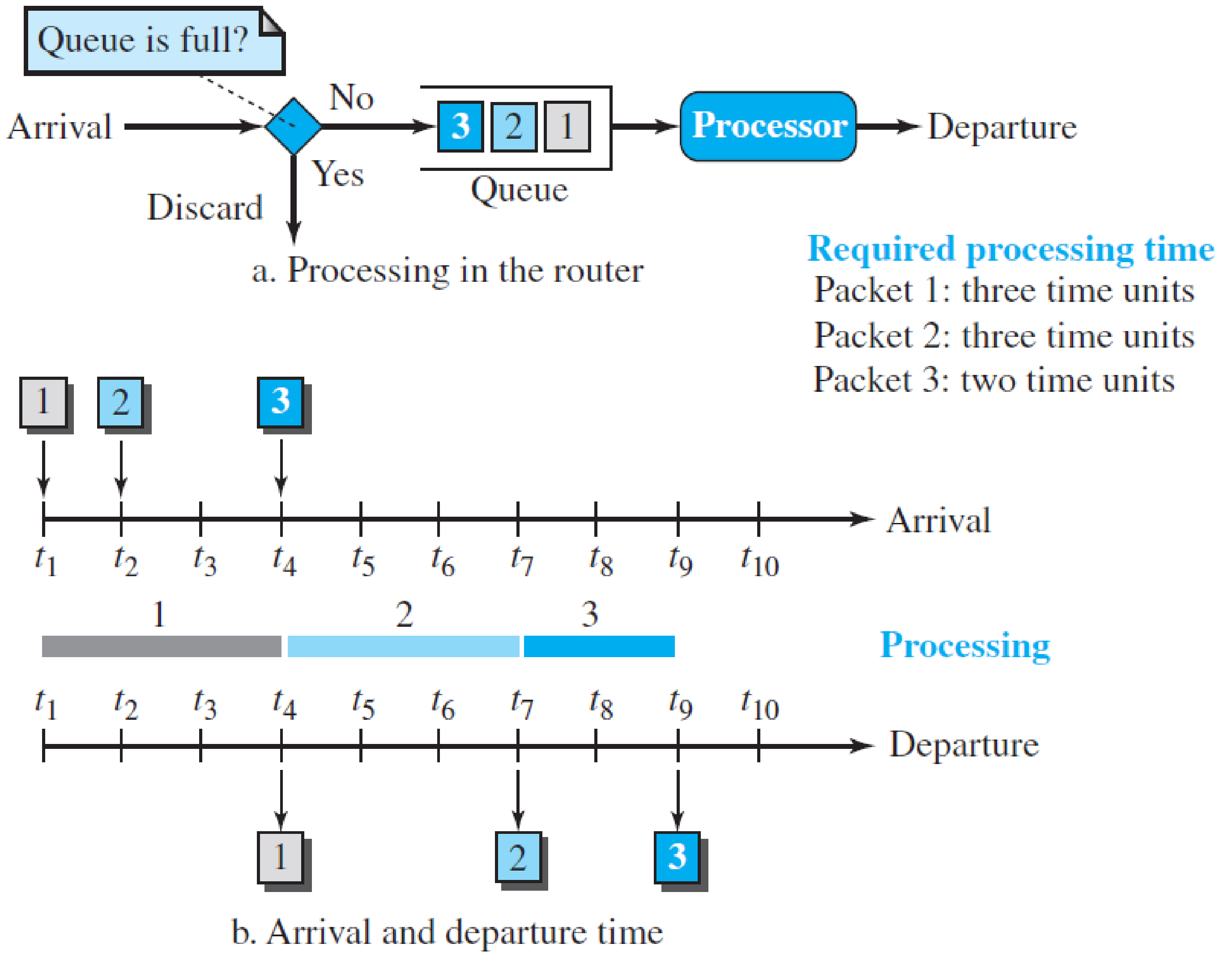


Figure 30.1 *FIFO queue*



Priority queuing

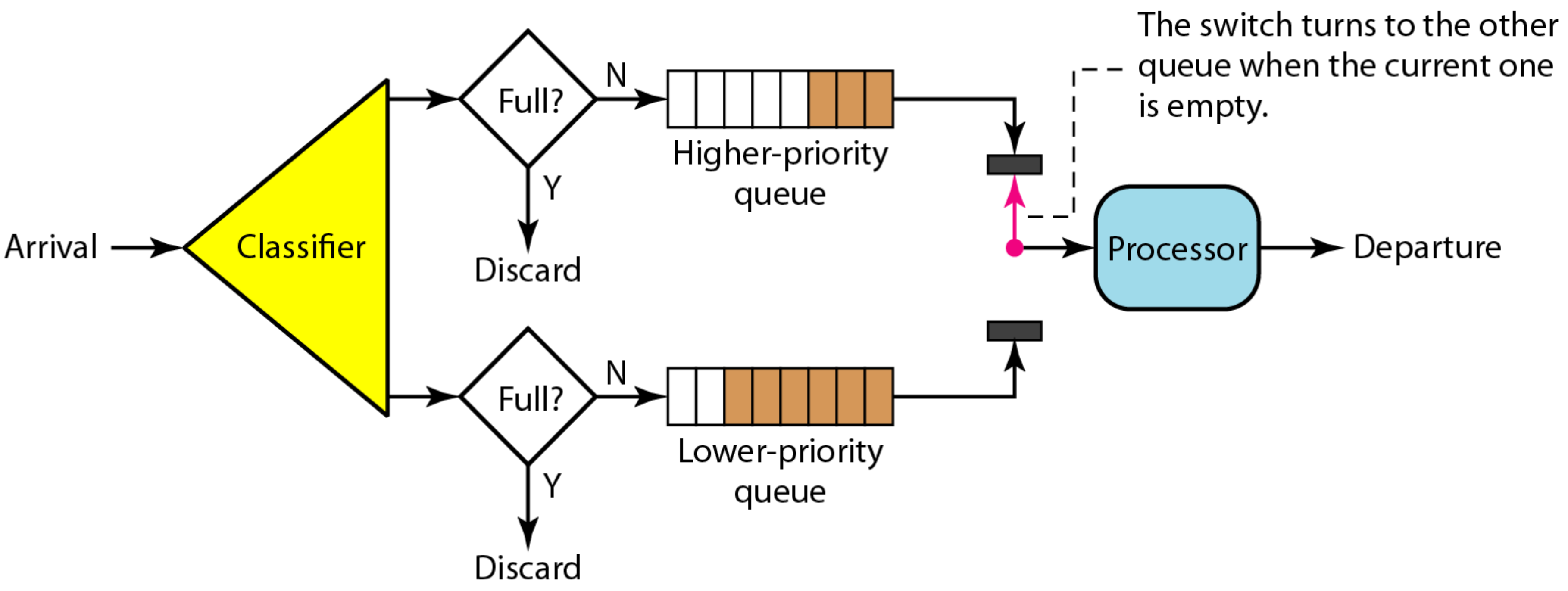
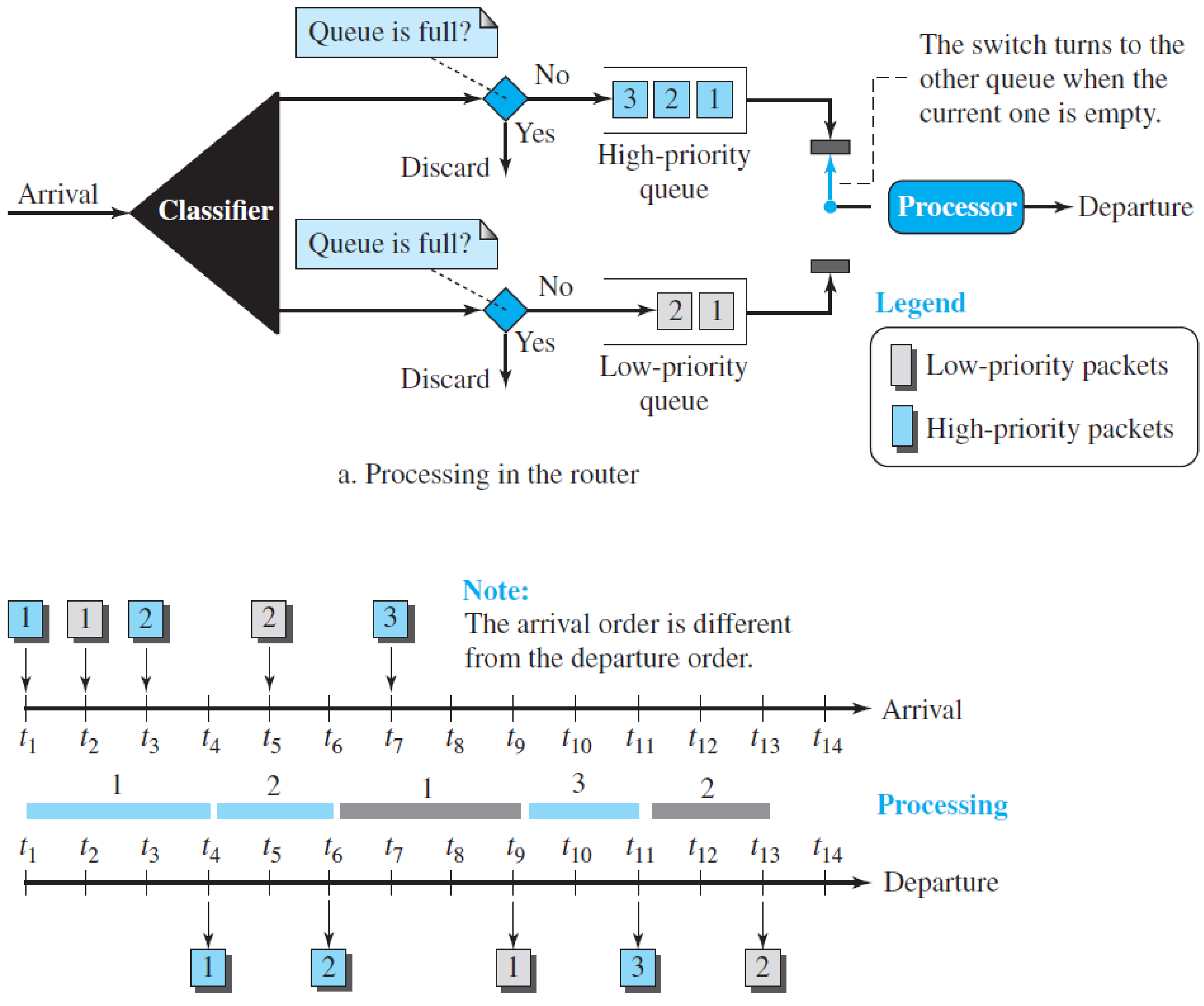


Figure 30.2 *Priority queuing*



b. Arrival and departure time

Weighted fair queuing

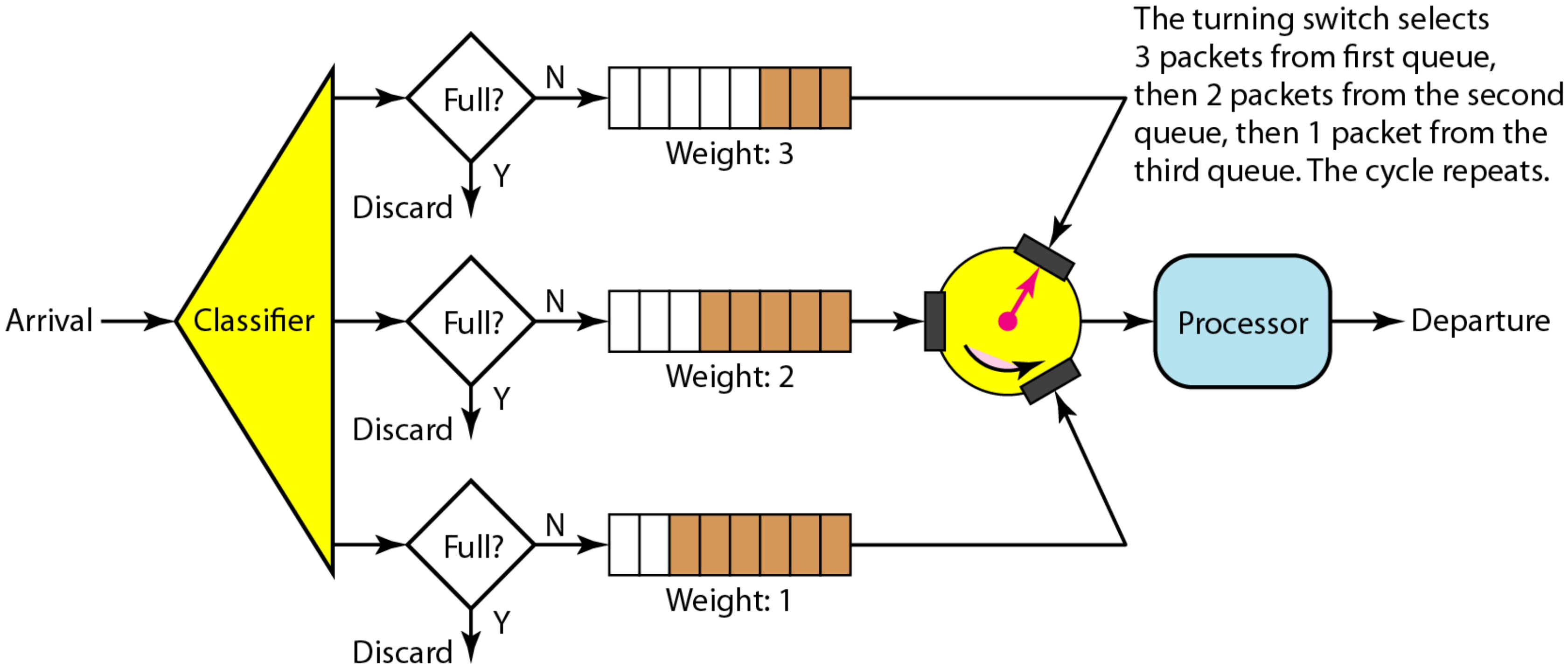


Figure 30.3 *Weighted fair queuing*

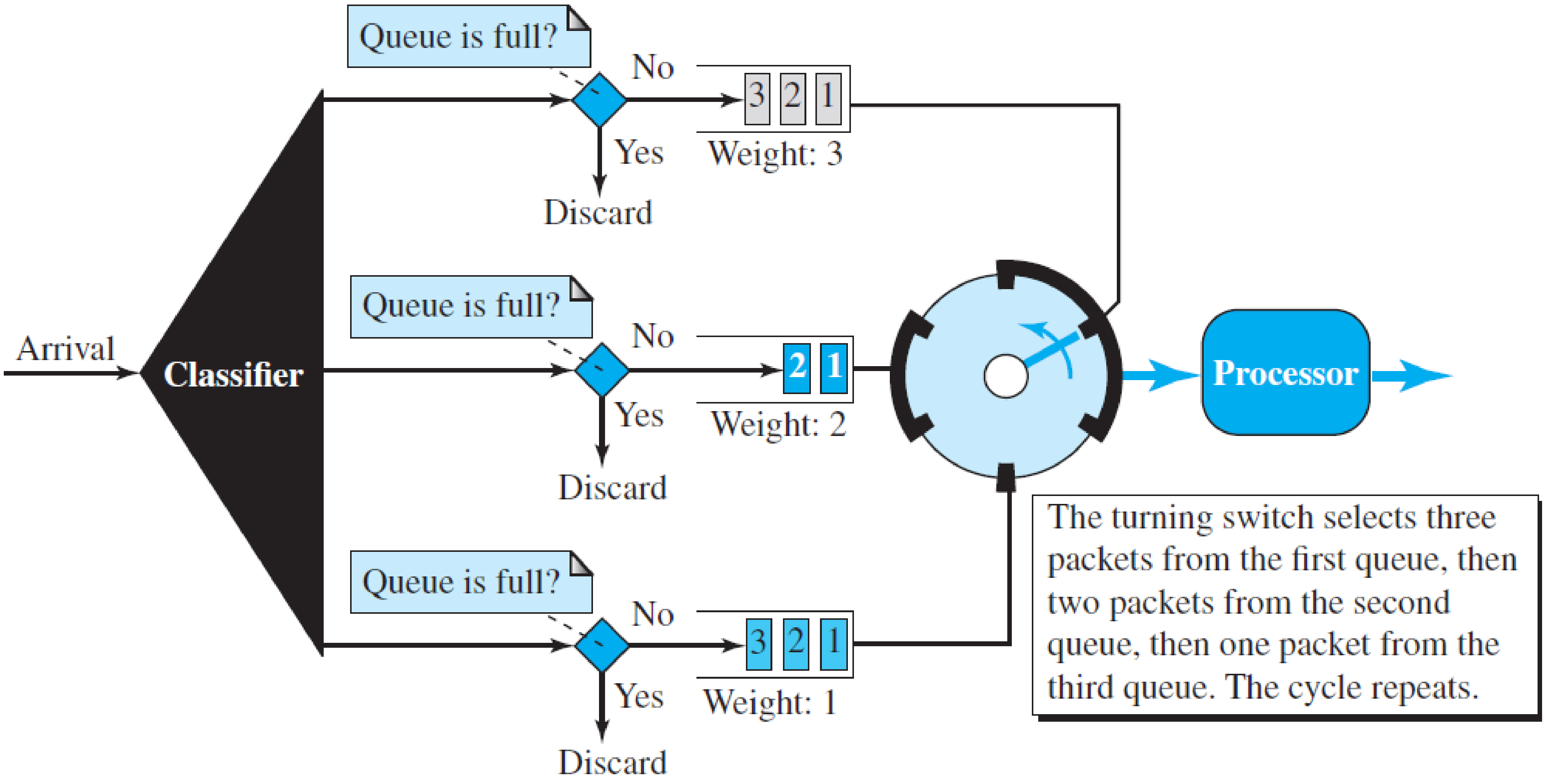
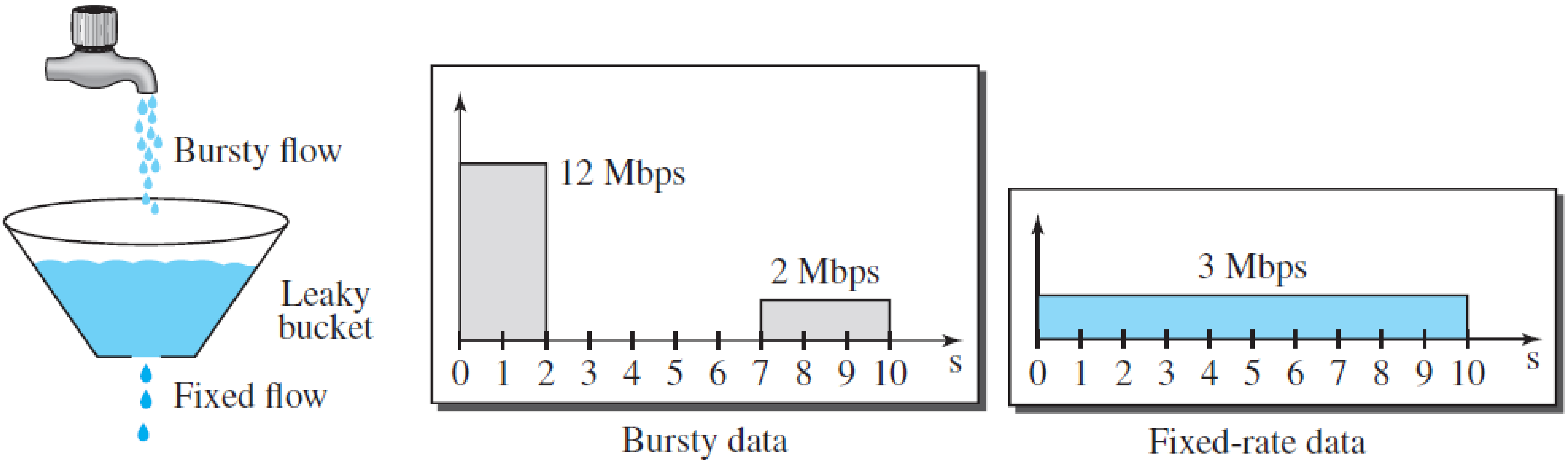


Figure 30.4 *Leaky bucket*



- The host sends a burst of data at a rate of 12 Mbps for 2 seconds, for a total of 24 Mb of data.
- The host is silent for 5 seconds and then sends data at a rate of 2 Mbps for 3 seconds, for a total of 6 Mb of data.
- In all, the host has sent 30 Mb of data in 10 seconds.
- The leaky bucket smooths the traffic by sending out data at a rate of 3 Mbps during the same 10 seconds

Note

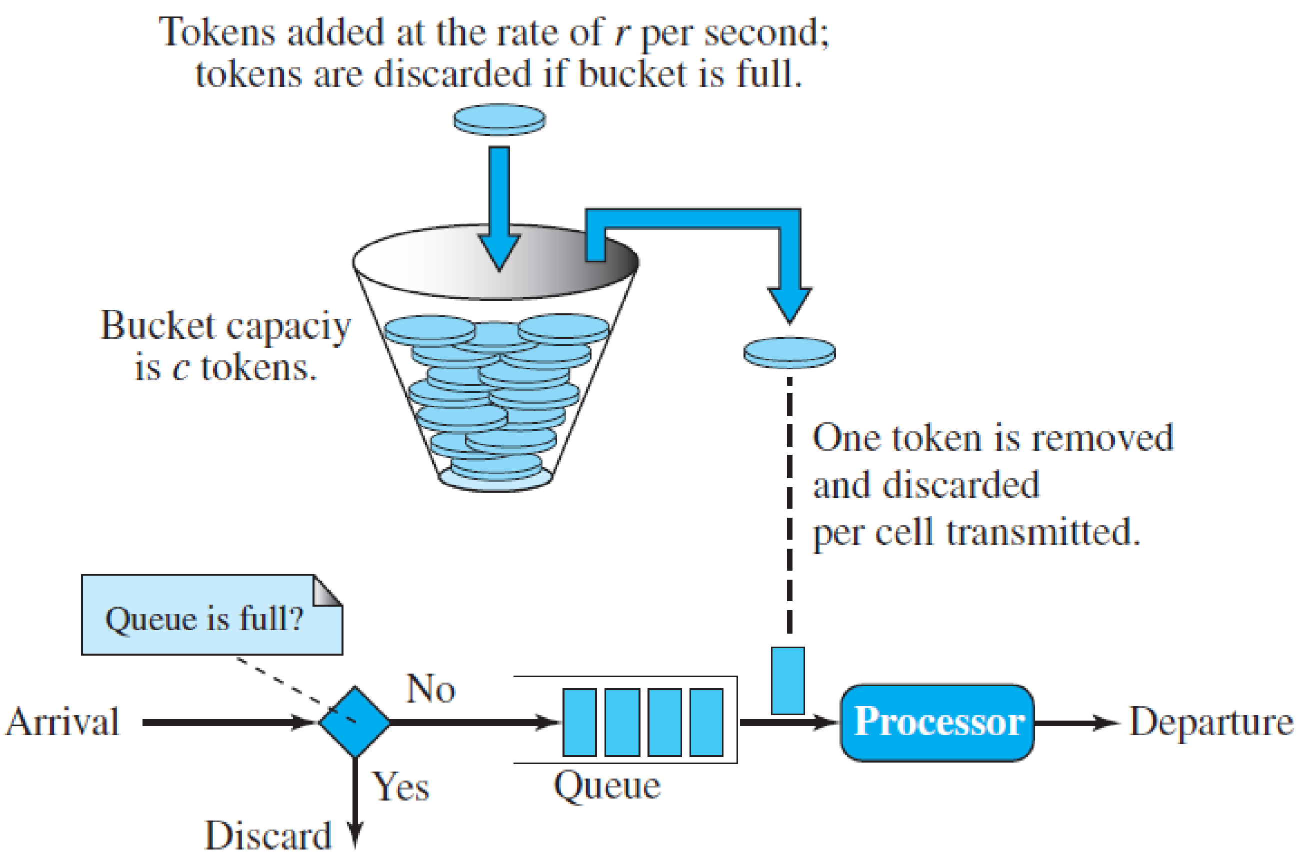
- A leaky bucket algorithm shapes bursty traffic into fixed-rate traffic by averaging the data rate. It may drop the packets if the bucket is full.

- It does not credit an idle host. For example, if a host is not sending for a while, its bucket becomes empty. Now if the host has bursty data, the leaky bucket allows only an average rate.

Note

The token bucket allows bursty traffic at a regulated maximum rate.

Figure 30.6 *Token bucket*



Assume the capacity of the bucket is c tokens and tokens enter the bucket at the rate of r tokens per second. The system removes one token for every cell of data sent.

The maximum number of cells that can enter the network during any time interval of length t is shown below.

Maximum number of packets = $r \cdot t + c$

The maximum average rate for the token bucket is shown below.

Maximum average rate = $(r \cdot t + c)/t$ packets per second

This means that the token bucket limits the average packet rate to the network.

Note: The token bucket allows bursty traffic at a regulated maximum rate.

Let's assume that the bucket capacity is 10,000 tokens and tokens are added at the rate of 1000 tokens per second. If the system is idle for 10 seconds (or more), the bucket collects 10,000 tokens and becomes full. Any additional tokens will be discarded. The maximum average rate is shown below.

$$\textbf{Maximum average rate} = (1000t + 10{,}000)/t$$

The token bucket can easily be implemented with a counter. The counter is initialized to zero. Each time a token is added, the counter is incremented by 1. Each time a unit of data is sent, the counter is decremented by 1. When the counter is zero, the host cannot send data.

Token bucket has a capacity of 1 MB and the maximum output rate is 20 MBps. Tokens arrive at a rate to sustain output at a rate of 10 MBps. The token bucket is currently full and the machine needs to send 12 megabytes of data. The minimum time required to transmit the data is __________________ seconds.

According to the token bucket algorithm, the minimum time required to send 1 MB of data or the maximum rate of data transmission is given by:

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

Where,

M = Maximum burst rate, P = Rate of arrival of a token, C = capacity of the bucket

M = 20 MB P = 10 MB C = 1 MB

$$S = 1 / (20 - 10) = 0.1 \text{ sec}$$

Since, the bucket is initially full, it already has 1 MB to transmit so it will be transmitted instantly. So, we are left with only (12 – 1), i.e., 11 MB of data to be transmitted.

Time required to send the 11 MB will be 11 * 0.1 = 1.1 sec

A computer on a 10Mbps network is regulated by a token bucket. The token bucket is filled at a rate of 2Mbps. It is initially filled to capacity with 16Megabits. What is the maximum duration for which the computer can transmit at the full 10Mbps?

- New tokens are added at the rate of p bytes/sec which is 2Mbps
- Capacity of the token bucket (C) = 16 Mbits
- Maximum possible transmission rate (M) = 10Mbps
- So the maximum burst time = C/(M-p) = 16/(10-2) = 2 seconds