ETSF10 Part 2 Lect 2

Performance, UDP, TCP, Congestion Control (TCP), QoS

Jens A Andersson

Electrical and Information Technology

Bandwitdh-Delay Product

- How much data fills the link
- Important for example in congestion avoidance
  - Delay = Round Trip Time (RTT)
  - Go-Back-N can send more bits per RTT

Figure 23.6 *Multiplexing and demultiplexing*

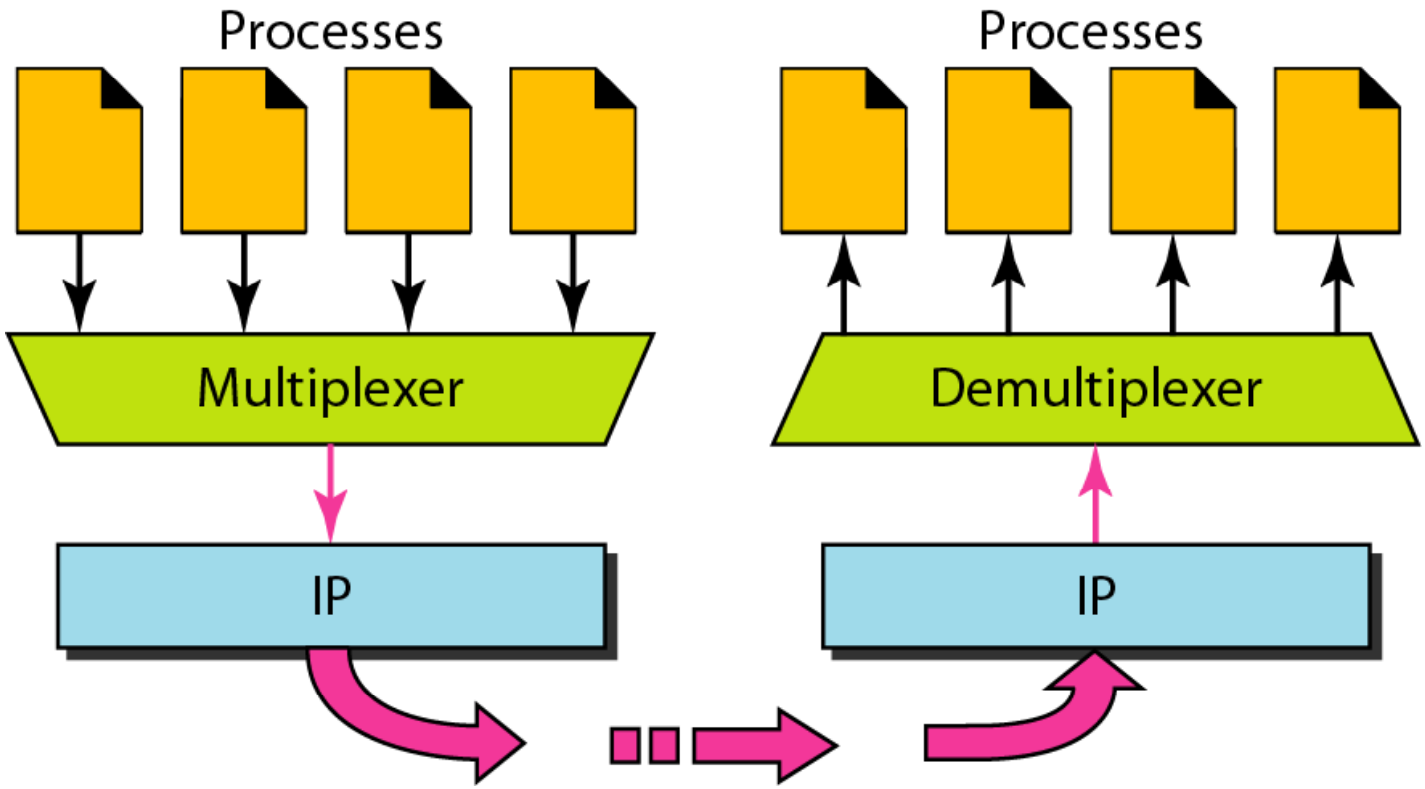
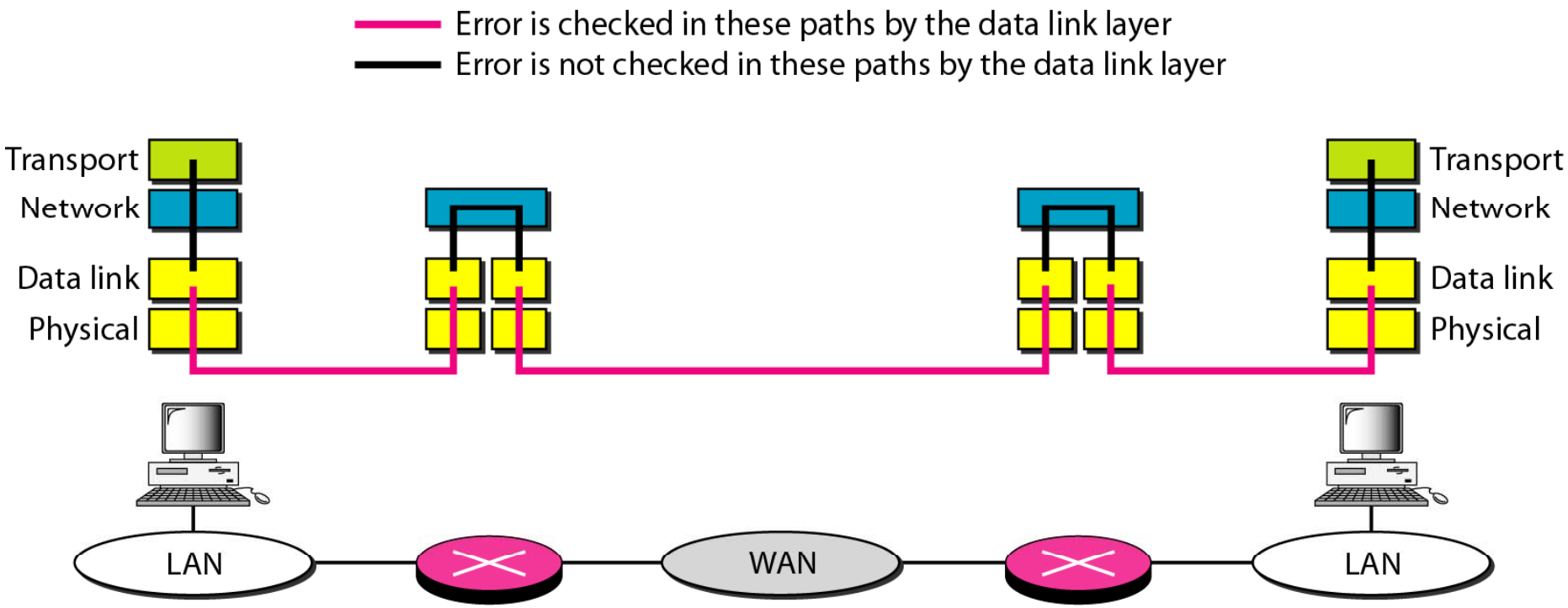


Figure 23.7 *Error control*



Today

- TCP
- Congestion Control in TCP
- Quality of Service (QoS)

Figure 23.16 *TCP segment format*

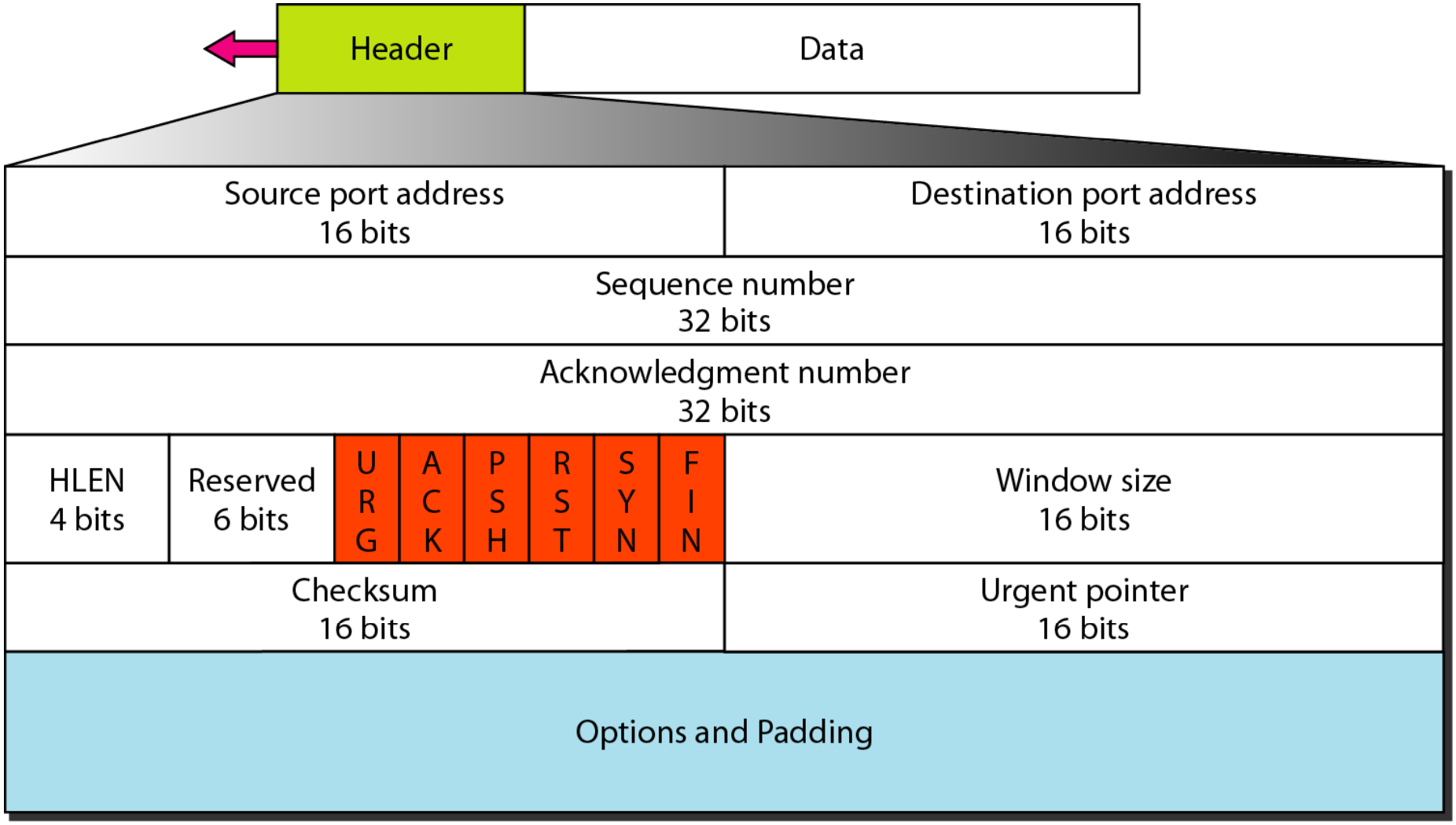


Figure 23.13 *Stream delivery*

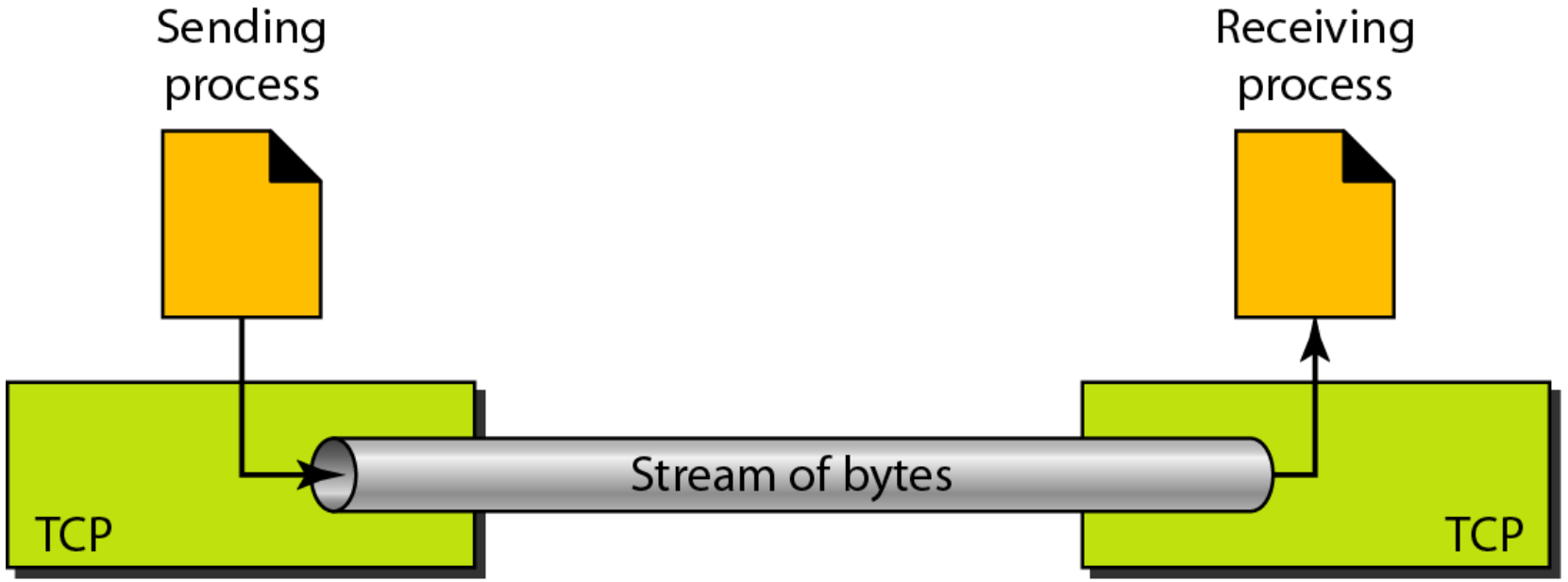


Figure 23.14 *Sending and receiving buffers*

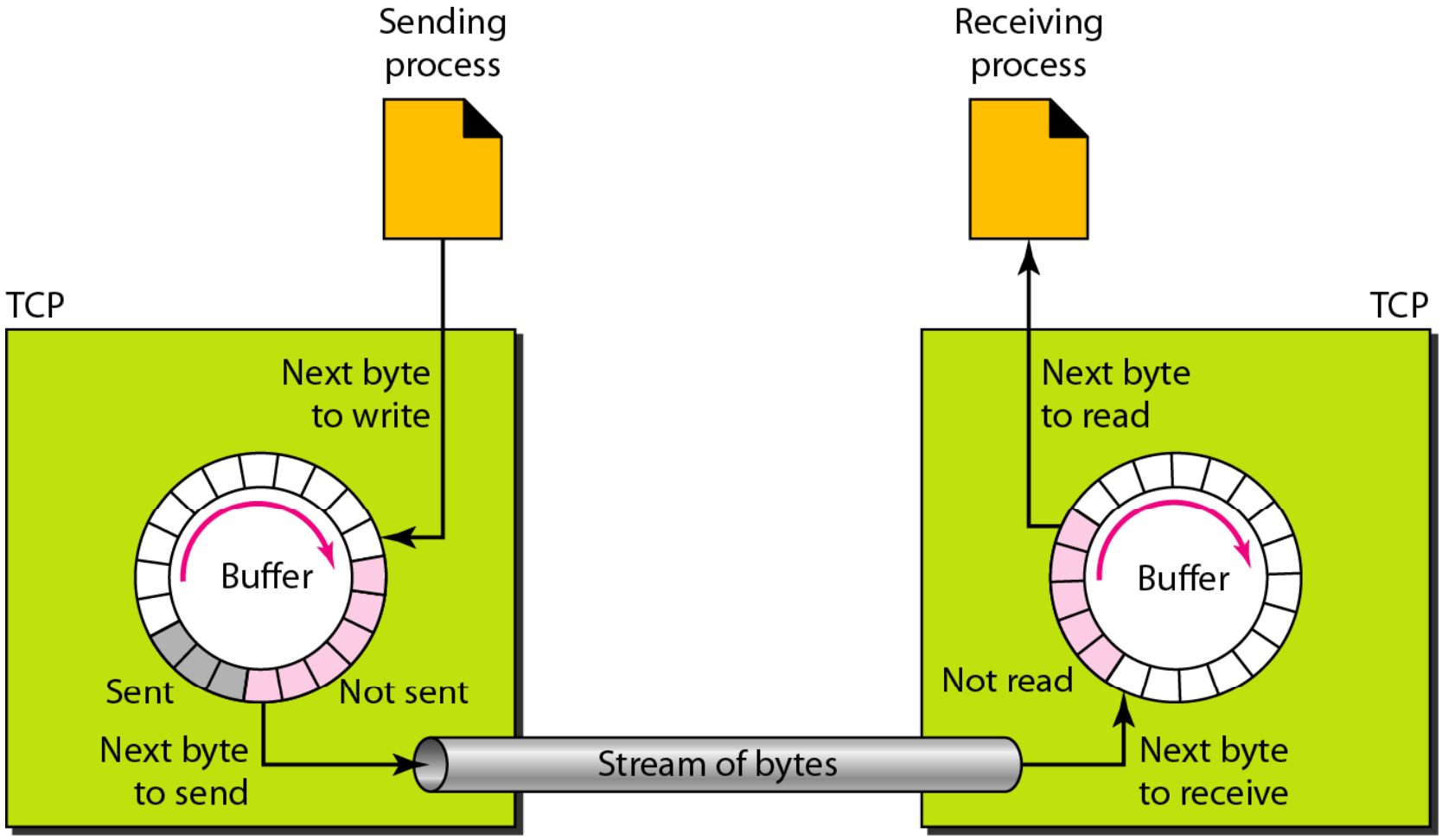
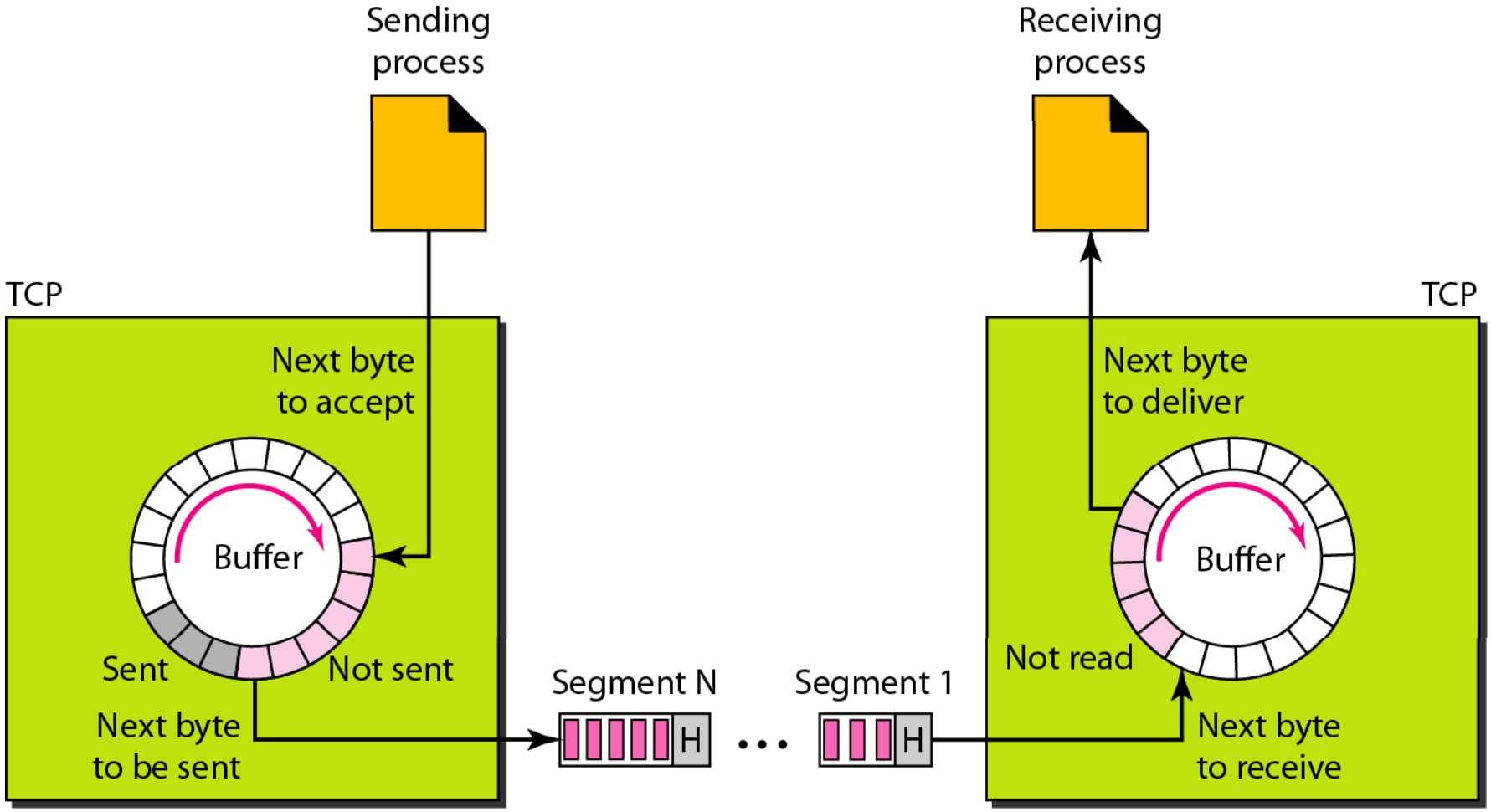


Figure 23.15 *TCP segments*



Example 23.3

The following shows the sequence number for each segment:

Segment 1	→	**Sequence Number: 10,001 (range: 10,001 to 11,000)**
Segment 2	→	**Sequence Number: 11,001 (range: 11,001 to 12,000)**
Segment 3	→	**Sequence Number: 12,001 (range: 12,001 to 13,000)**
Segment 4	→	**Sequence Number: 13,001 (range: 13,001 to 14,000)**
Segment 5	→	**Sequence Number: 14,001 (range: 14,001 to 15,000)**

Figure 23.17 *Control field*

URG: Urgent pointer is valid
ACK: Acknowledgment is valid
PSH: Request for push

RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection

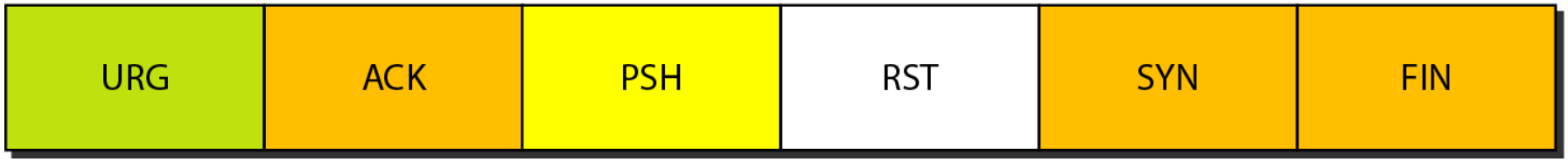


Figure 23.18 *Connection establishment using three-way handshaking*

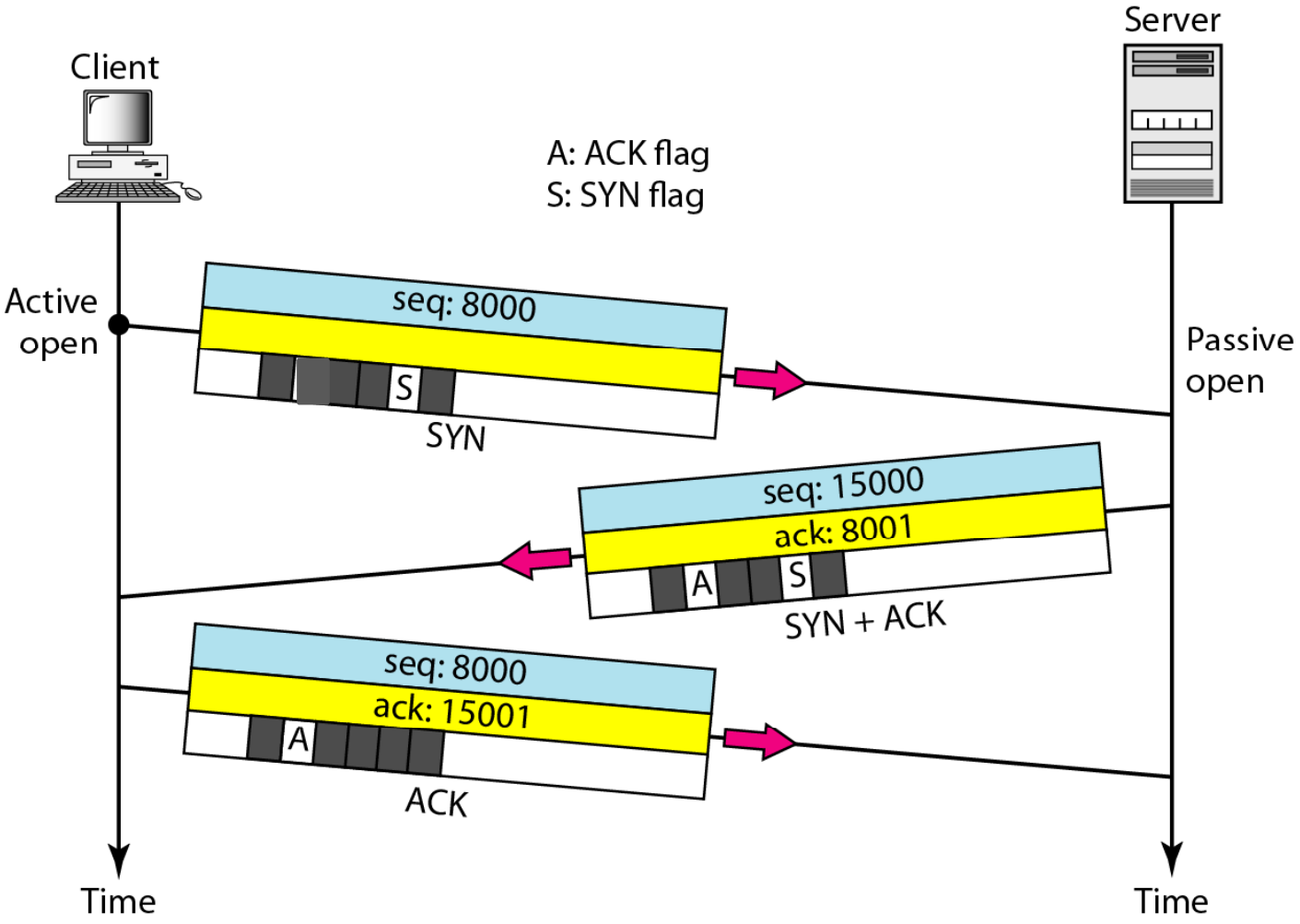


Figure 23.19 *Data transfer*

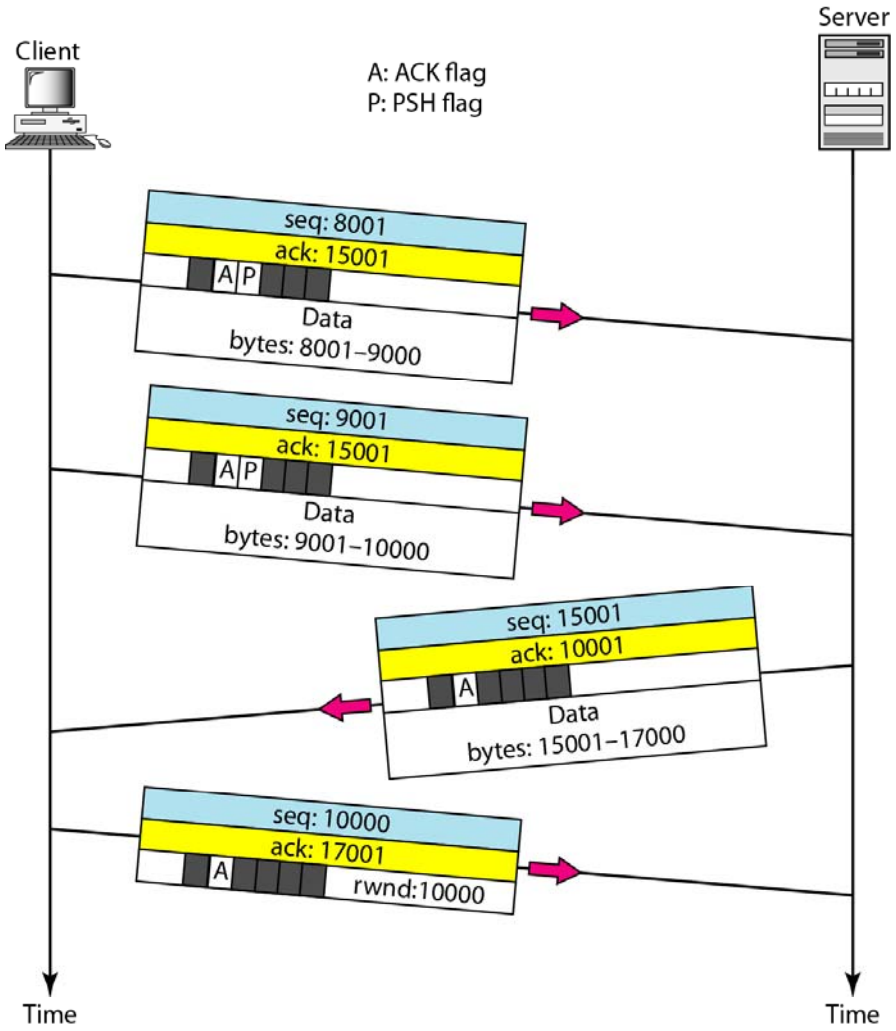


Figure 23.20 *Connection termination using three-way handshaking*

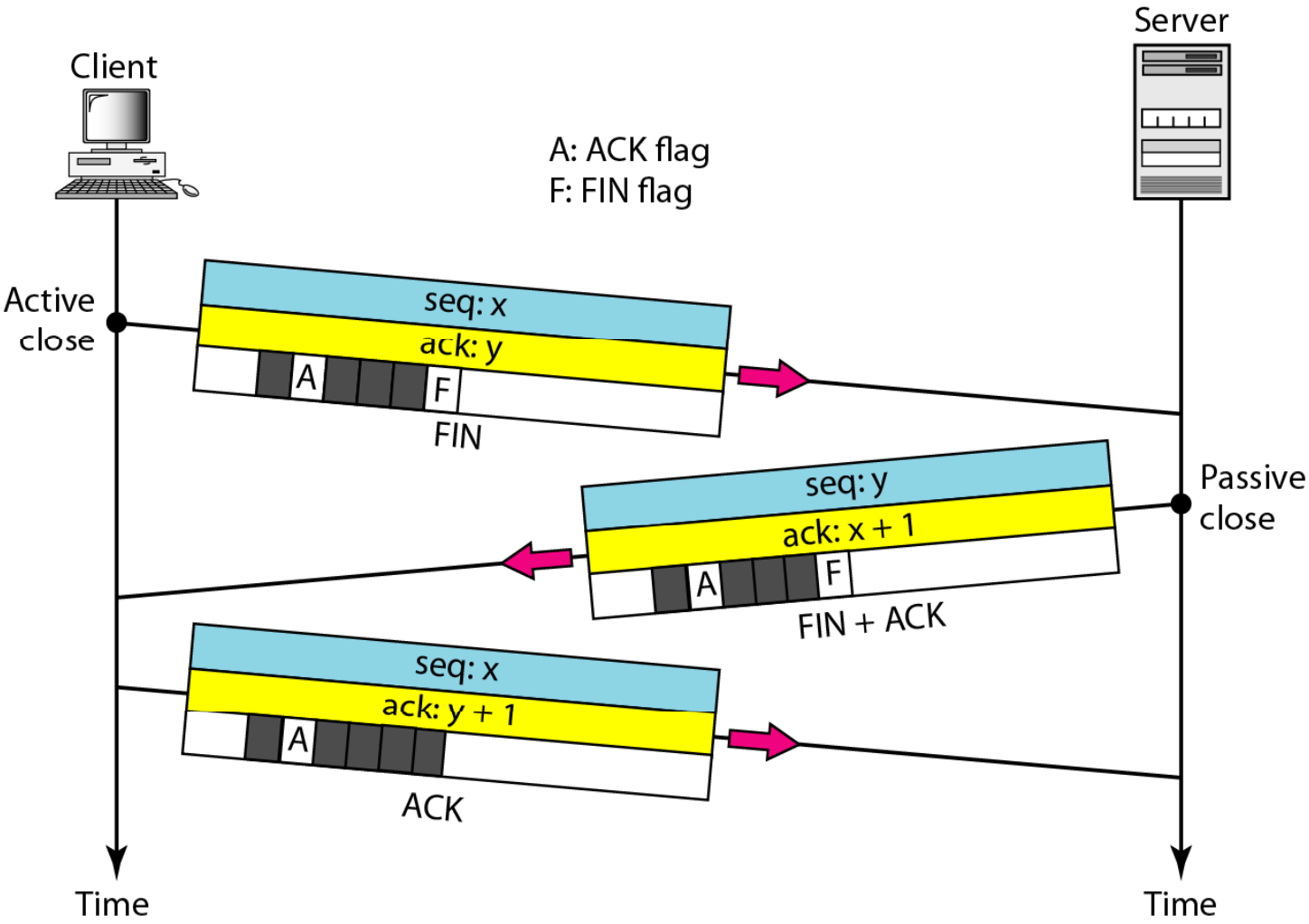
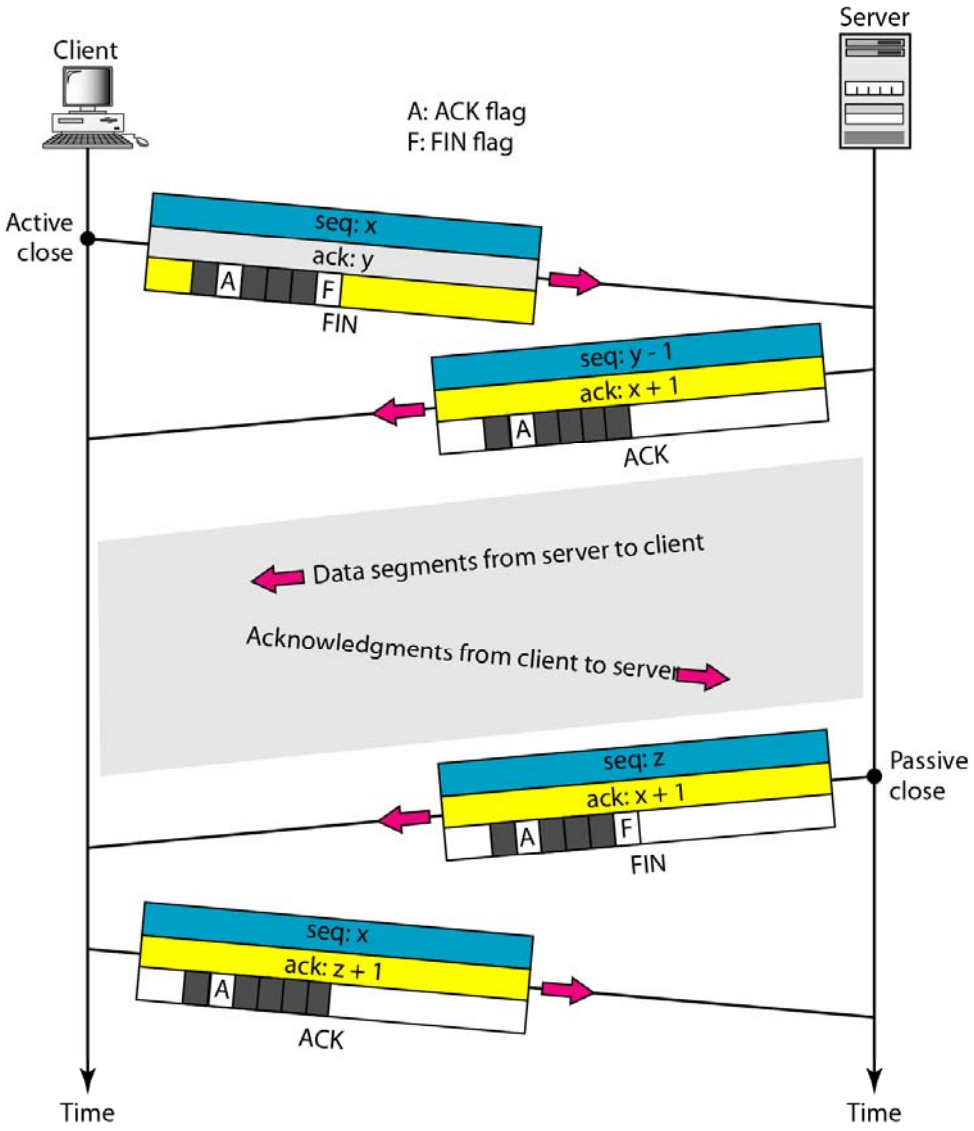


Figure 23.21 *Half-close*



Flow control

- Sliding window
- Byte oriented
  - Send chunks of bytes in segment
- Implemented as pointers into a memory buffer

Figure 23.22 *Sliding window*

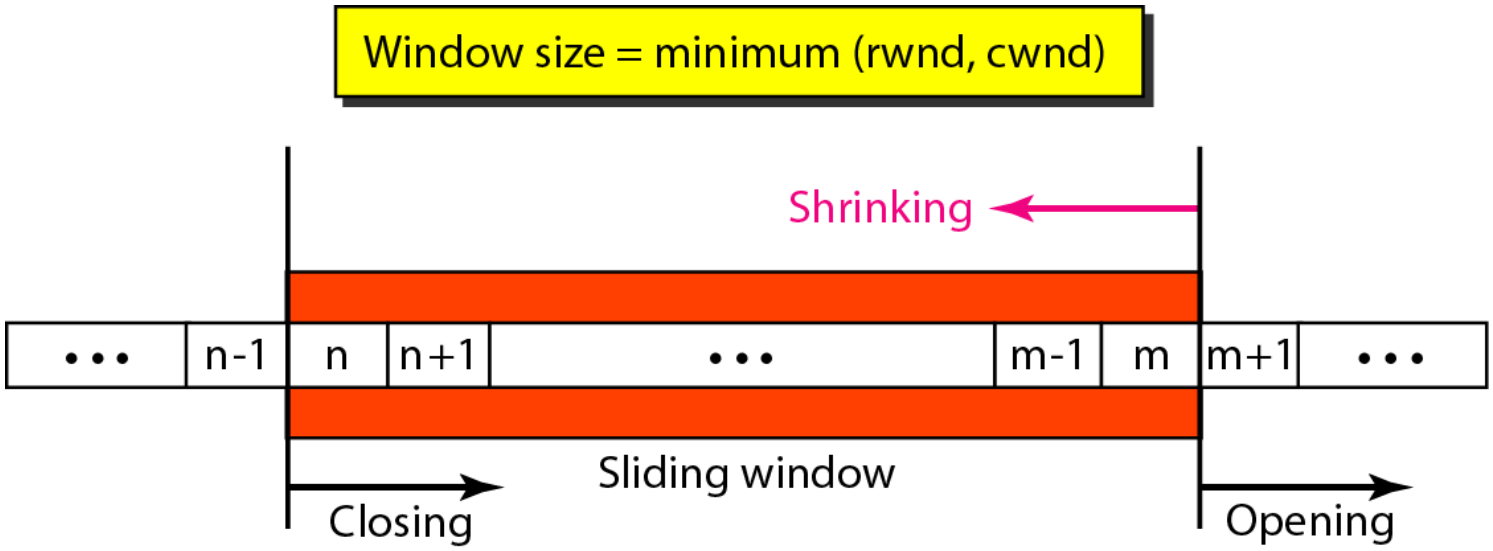
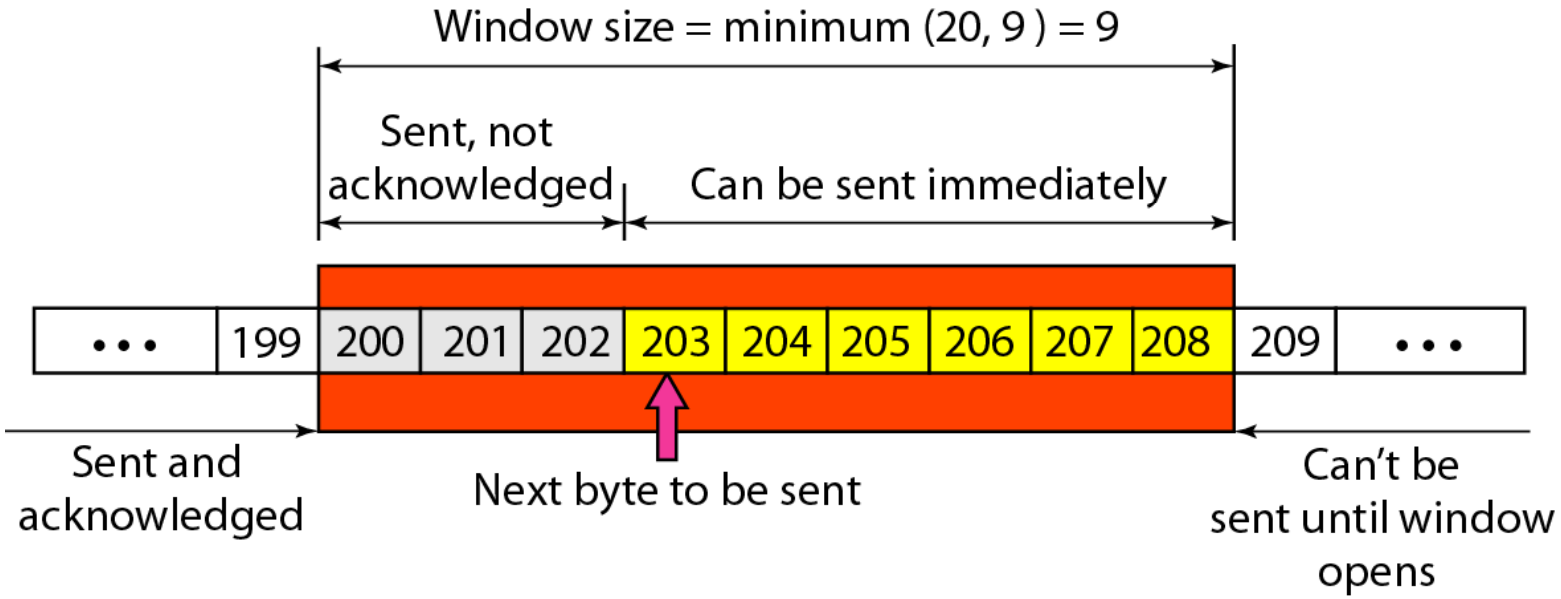


Figure 23.23 *Example 23.6*



Error control

- ACK recieved data
- Go-Back-N ARQ
  - Resend from lost segment or error
  - Even segments after error/lost that are correct
- Retransmission timer!
  - Adjusted according to RTT
  - Or three duplicate ACKs (fast retransmission)

Figure 23.24 *Normal operation*

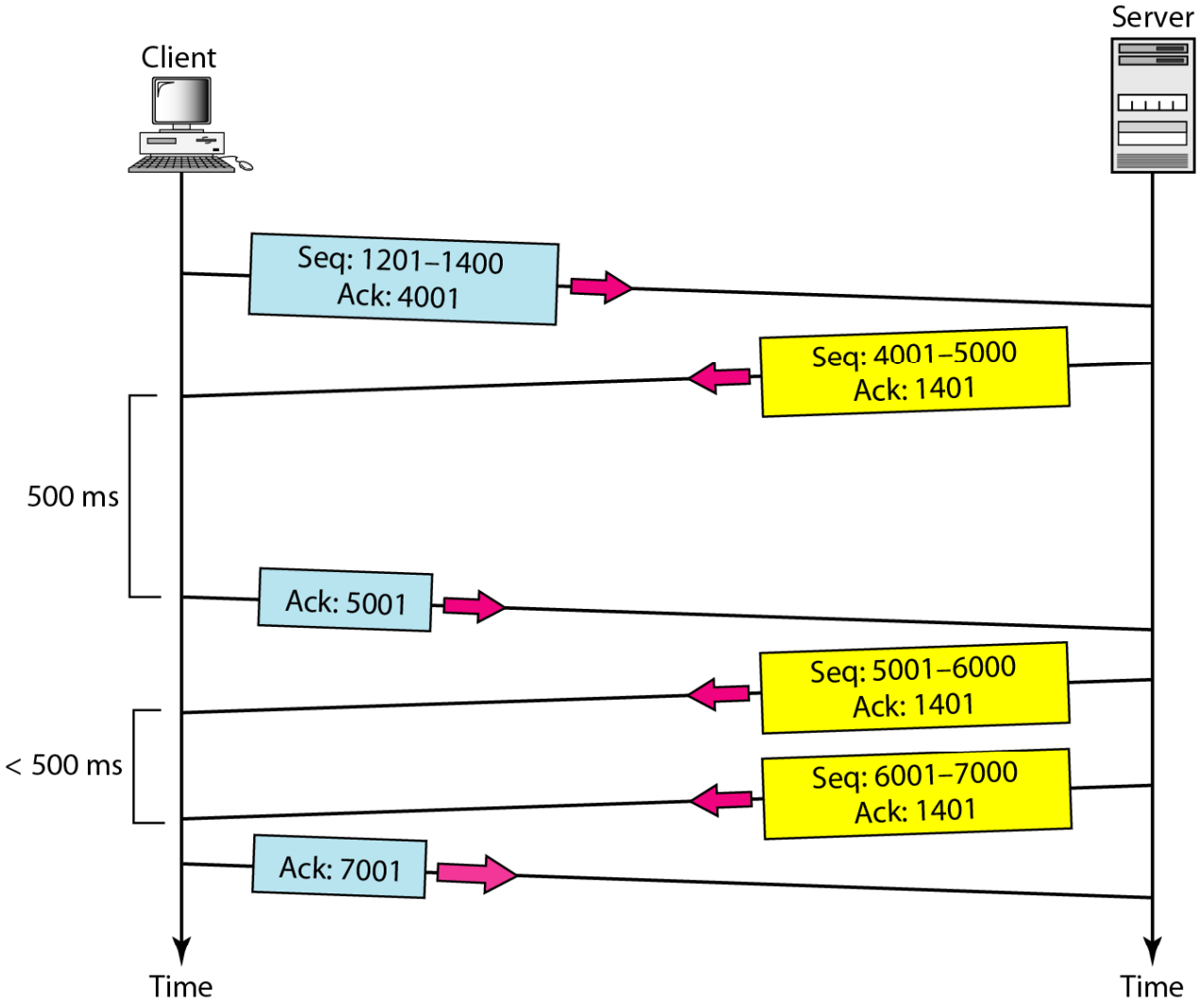


Figure 23.25 *Lost segment*

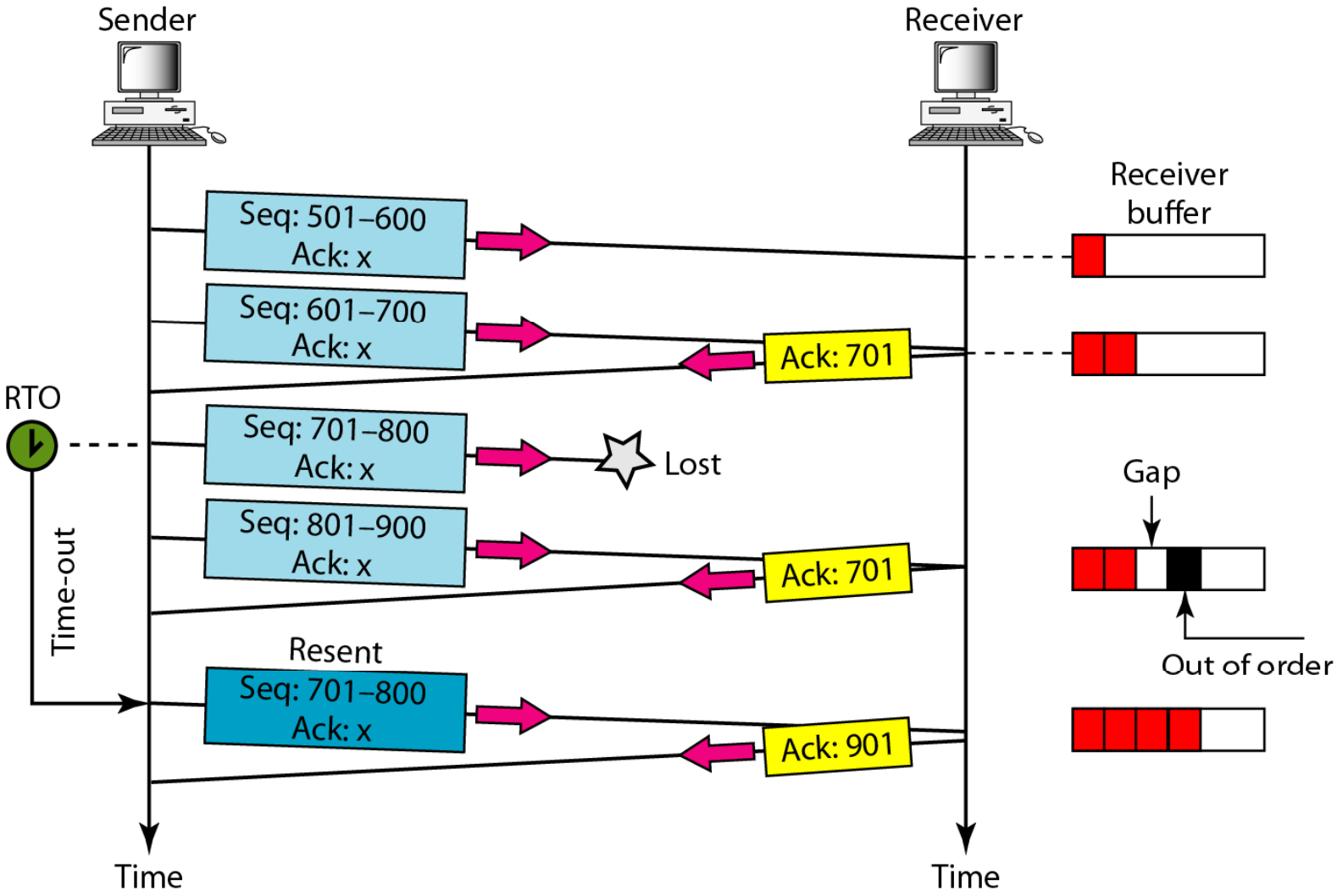
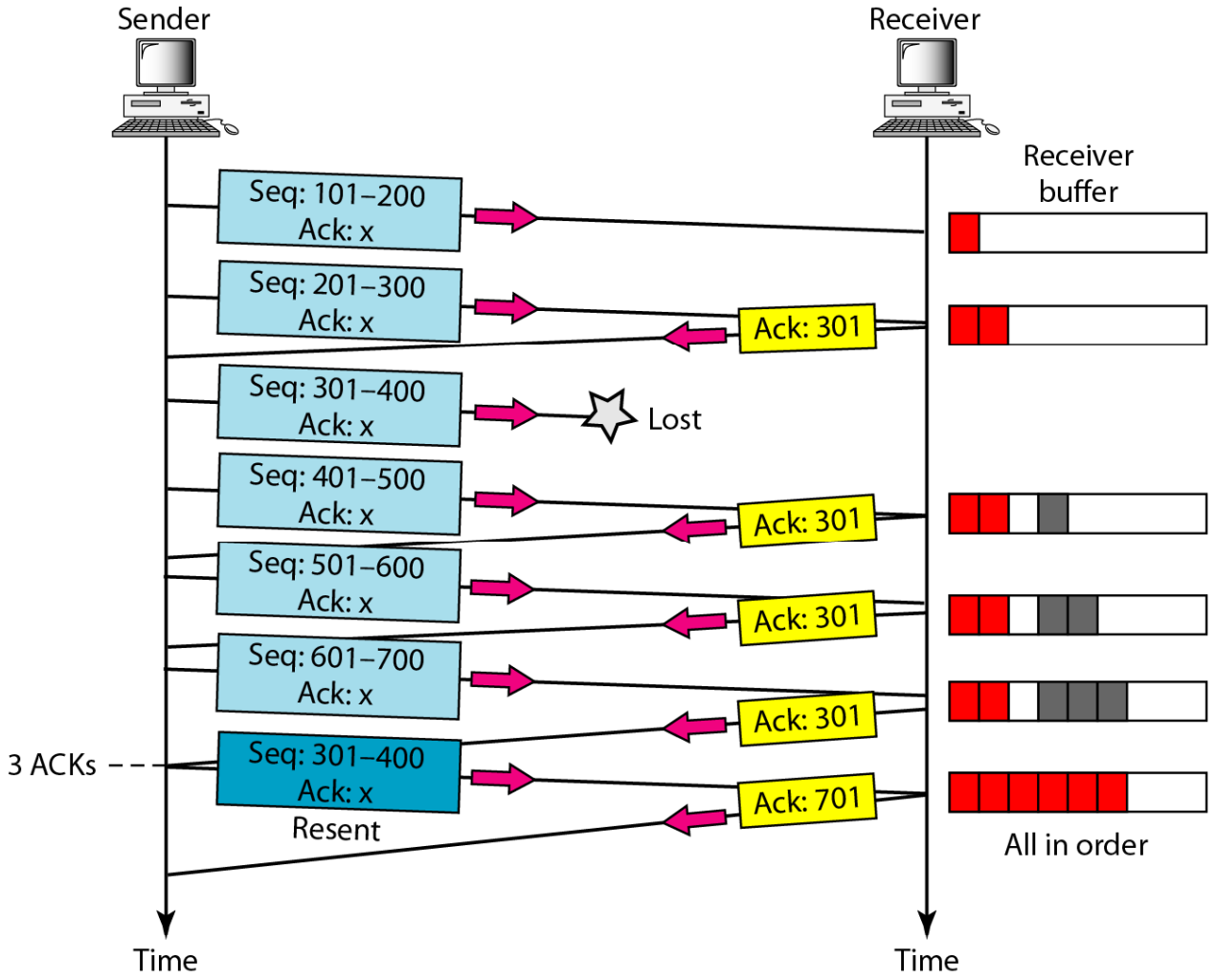


Figure 23.26 *Fast retransmission*



Congestion Control and QoS

- Congestion Control
  - Try to avoid congestion
- Quality of Service
  - Create an appropriate environment for the traffic
  - Not the same as Service Quality(?)

Figure 24.1 *Traffic descriptors*

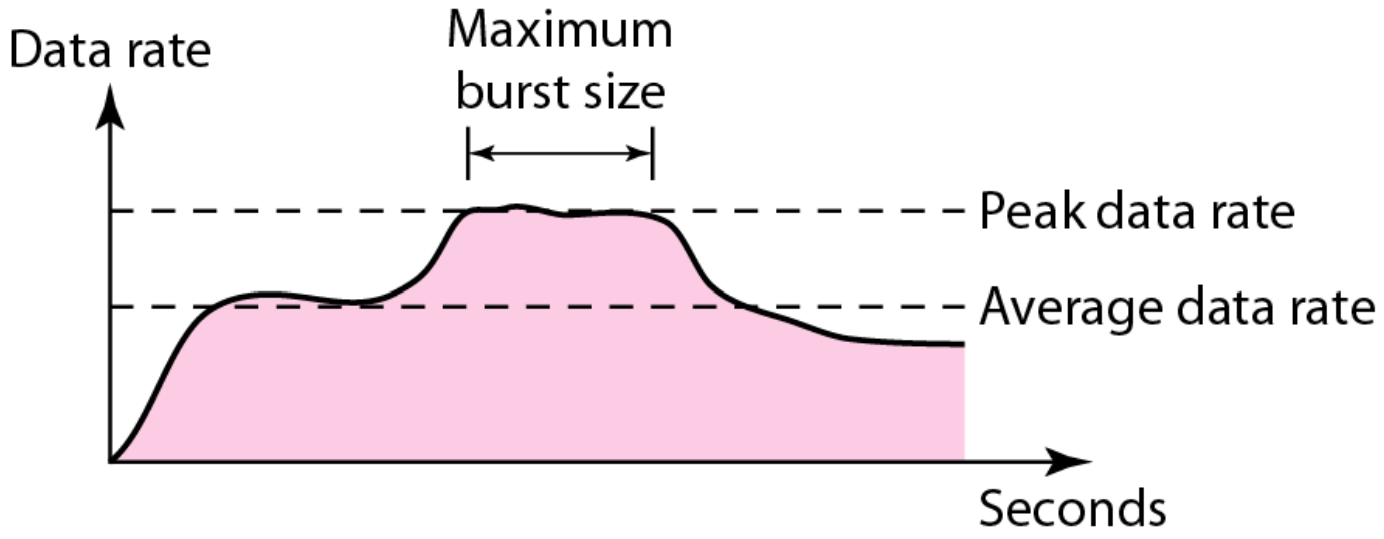
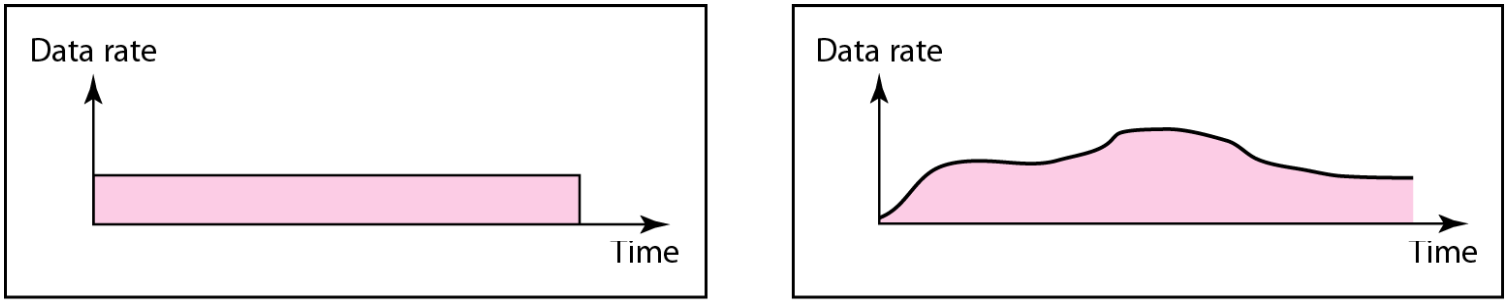


Figure 24.2 *Three traffic profiles*

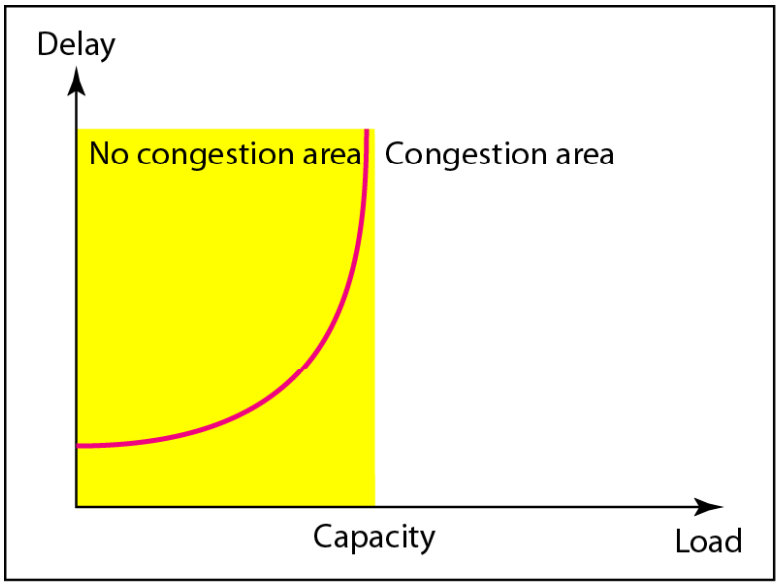


a. Constant bit rate

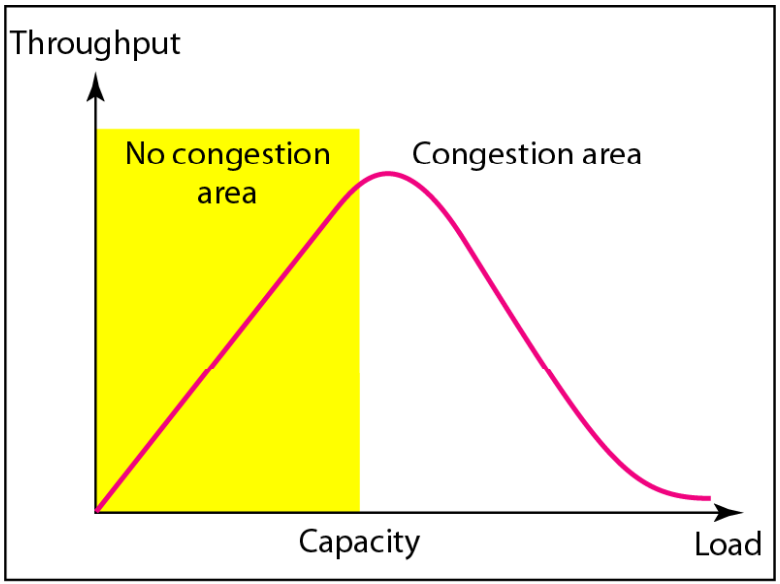
b. Variable bit rate

c. Bursty

Figure *Packet delay and throughput as functions of load*



a. Delay as a function of load



b. Throughput as a function of load

Figure 24.3 *Queues in a router*

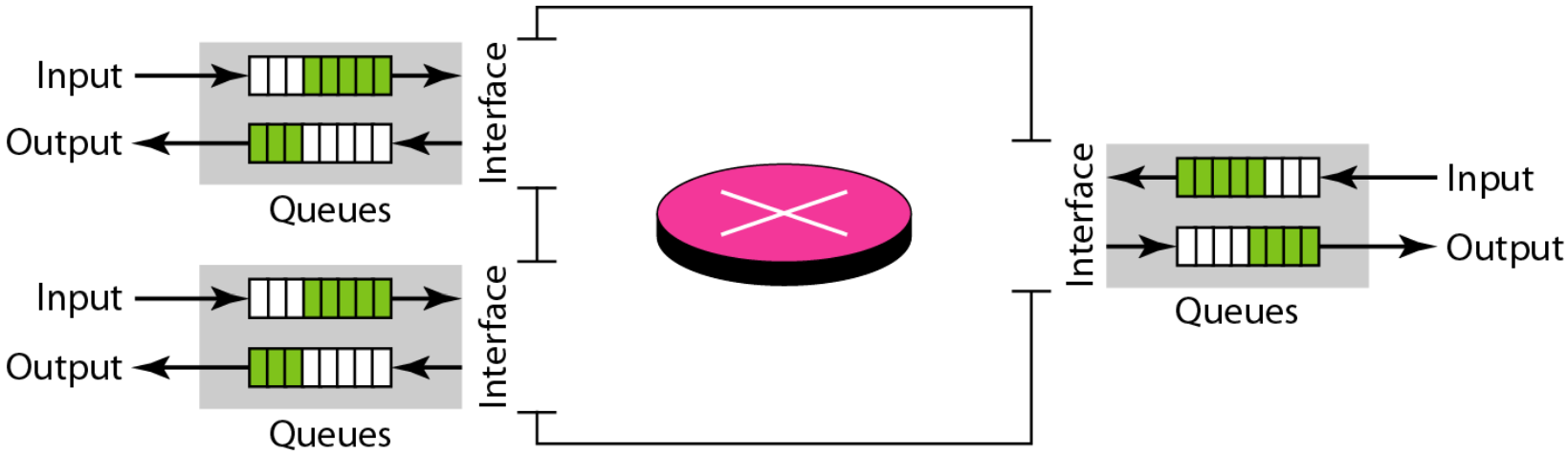


Figure 24.5 *Congestion control categories*

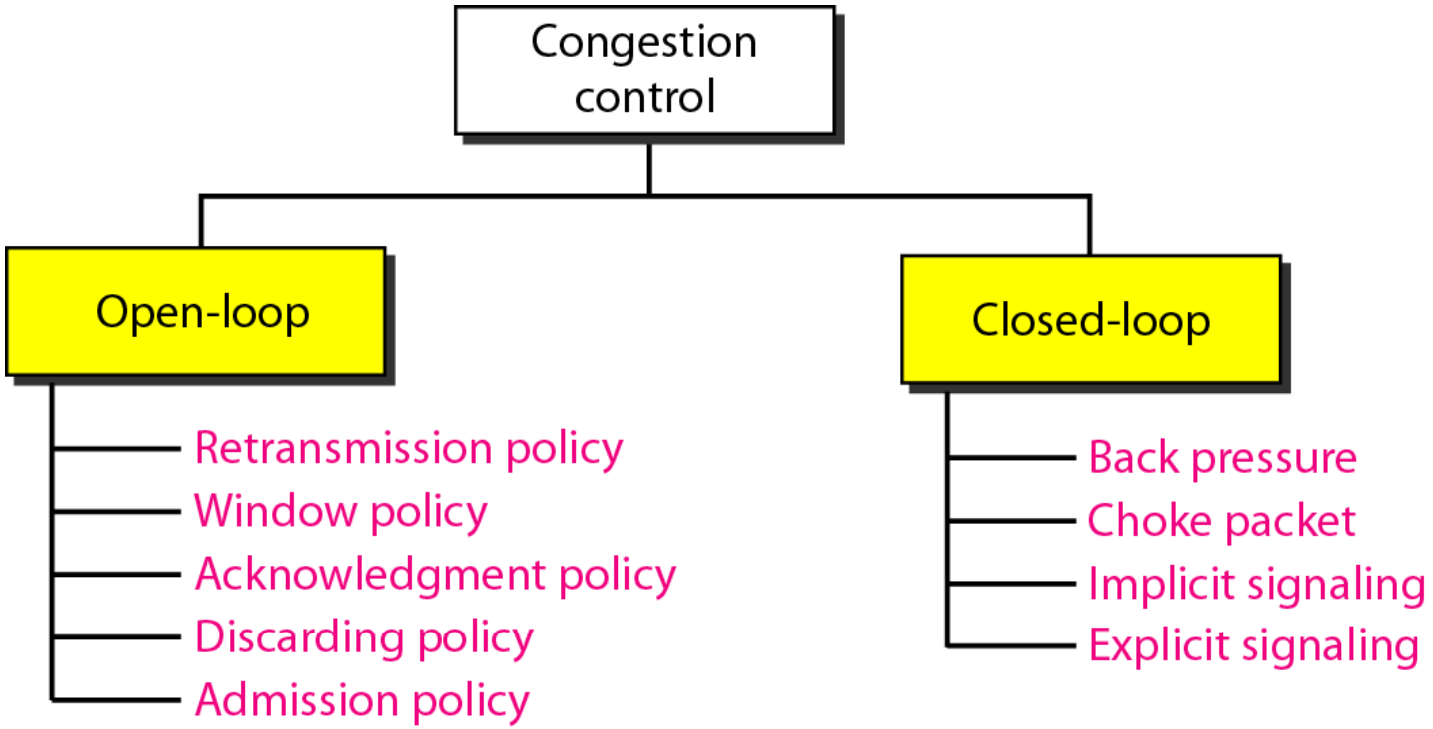


Figure 24.6 *Backpressure method for alleviating congestion*

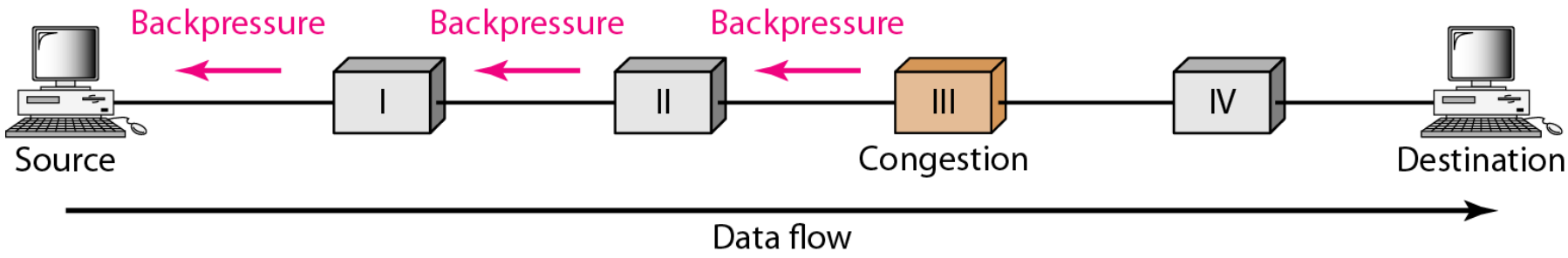
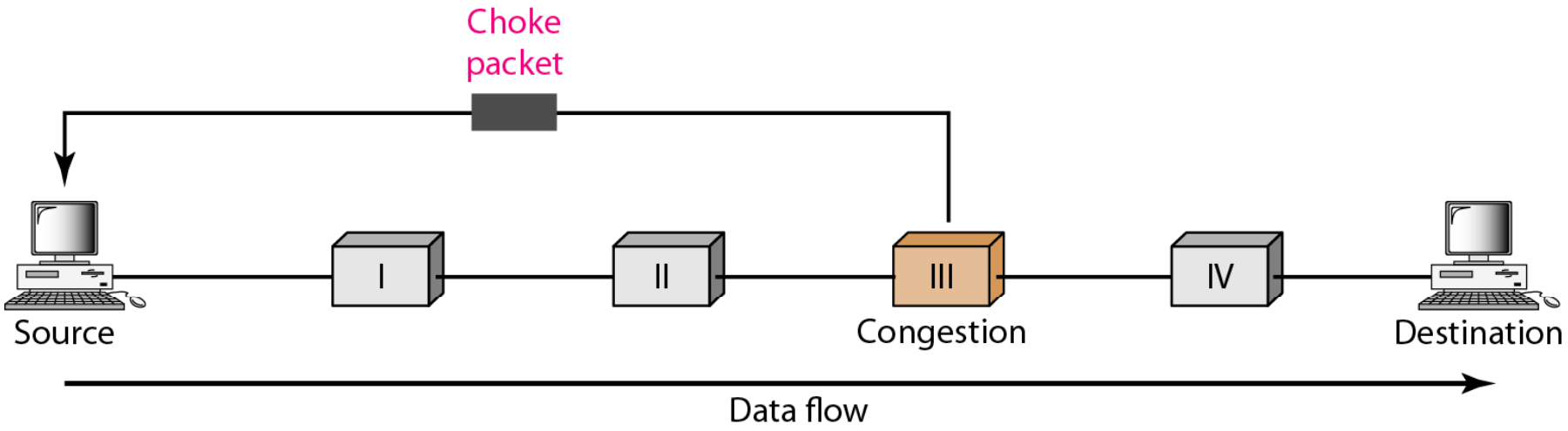


Figure 24.7 *Choke packet*



Congestion Control in TCP

- Slow start (state)
  - Exponential Increase
- Congestion avoidance (state)
  - Additive Increase
- Congestion detection (occation to act upon)
  - Multiplicative Decrease
- Congestion Windows Size (short: cwnd)
  - Remember: window size = min(receiver window, congestion window)

Figure 24.8 *Slow start, exponential increase*

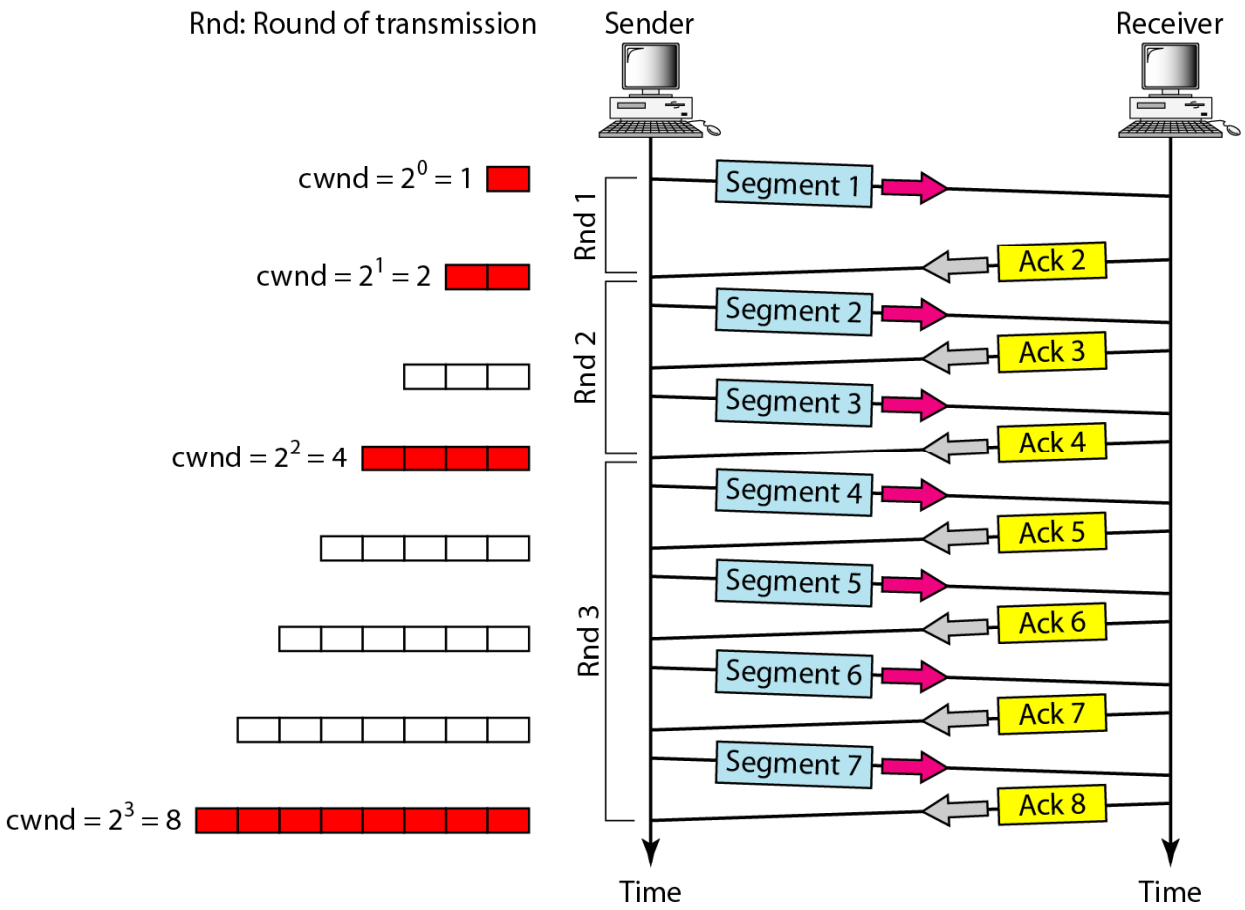
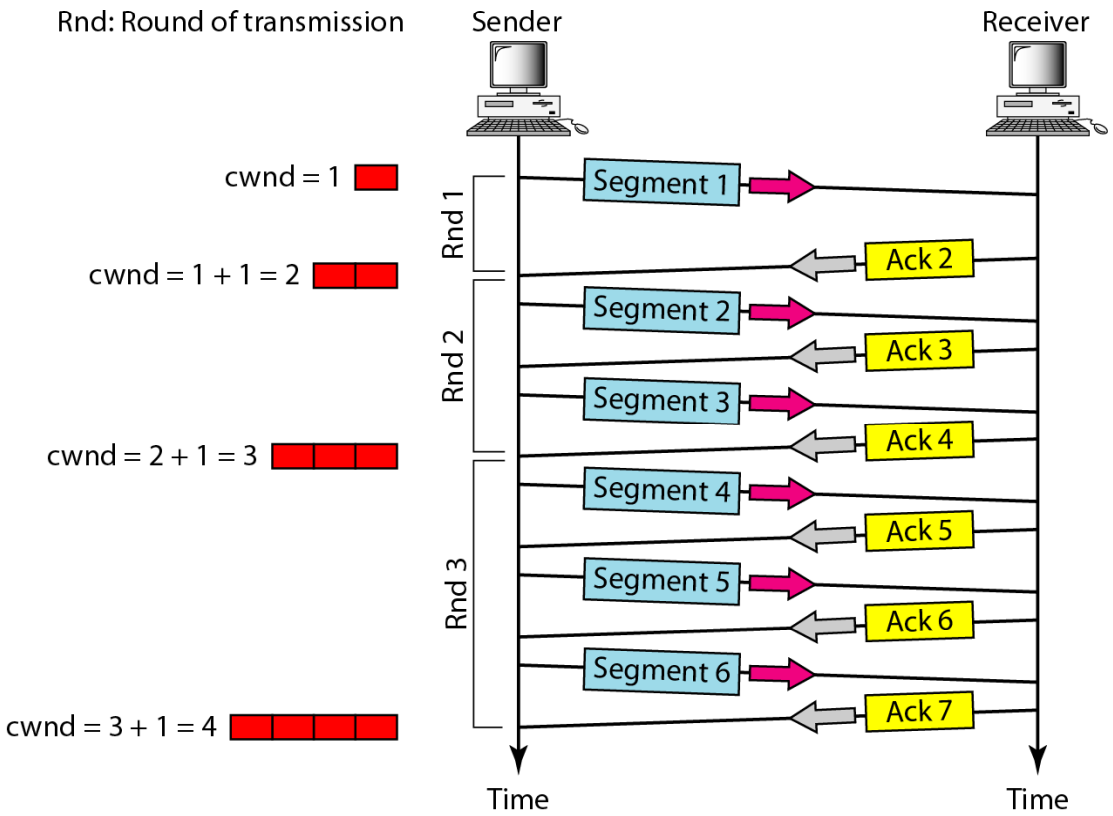


Figure 24.9 *Congestion avoidance, additive increase*



Reaction to congestion detection

- If detection is by time-out, a new slow start phase starts.
- If detection is by three ACKs, a new congestion avoidance phase starts.
- Time out = probably congestion both channels
- 3 ACKS = probably congestion only in sending channel

Figure 24.10 *TCP congestion policy summary*

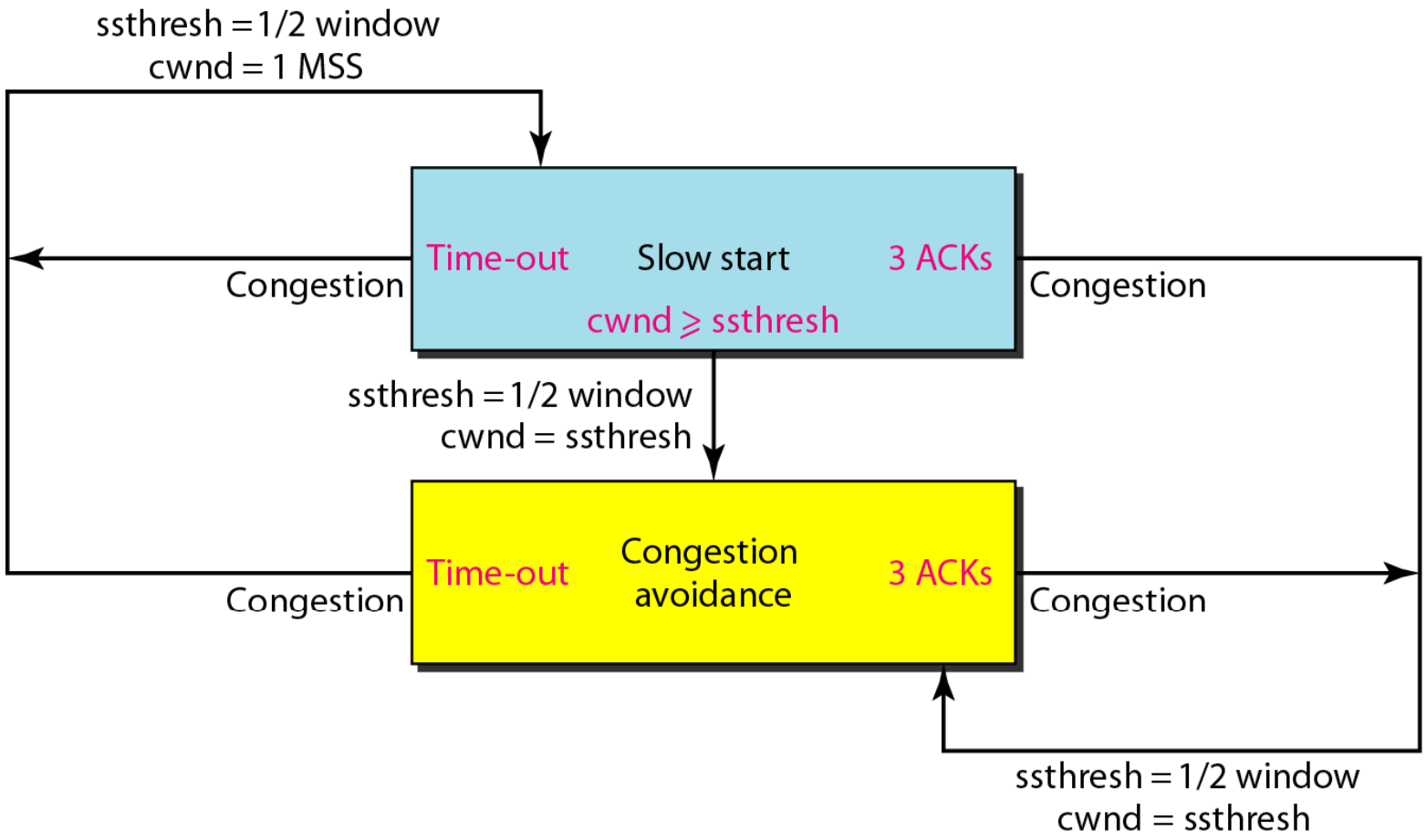
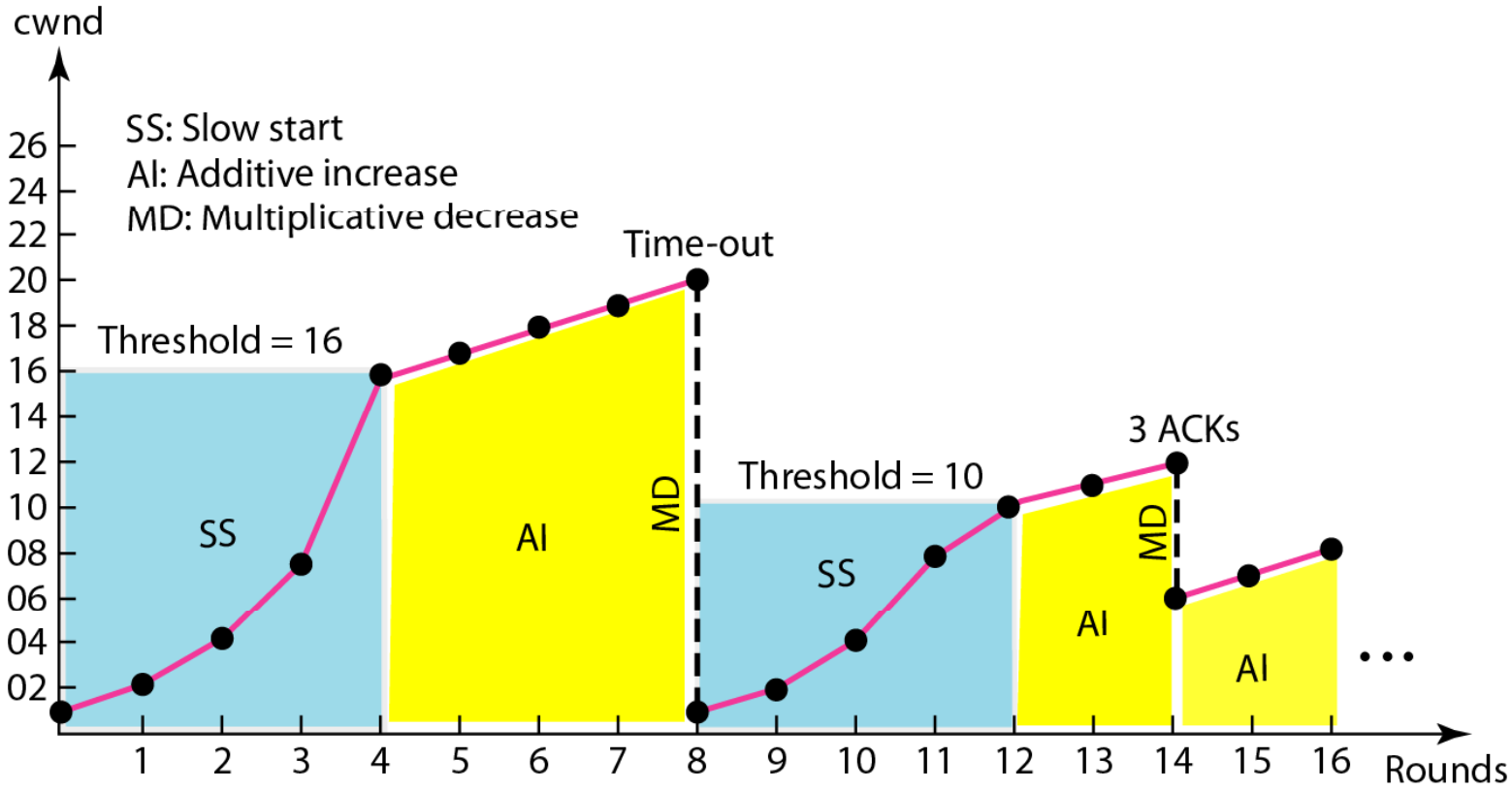


Figure 24.11 *Congestion example*



Improve QoS

- Scheduling
- Traffic shaping

Figure 24.15 *Flow characteristics*

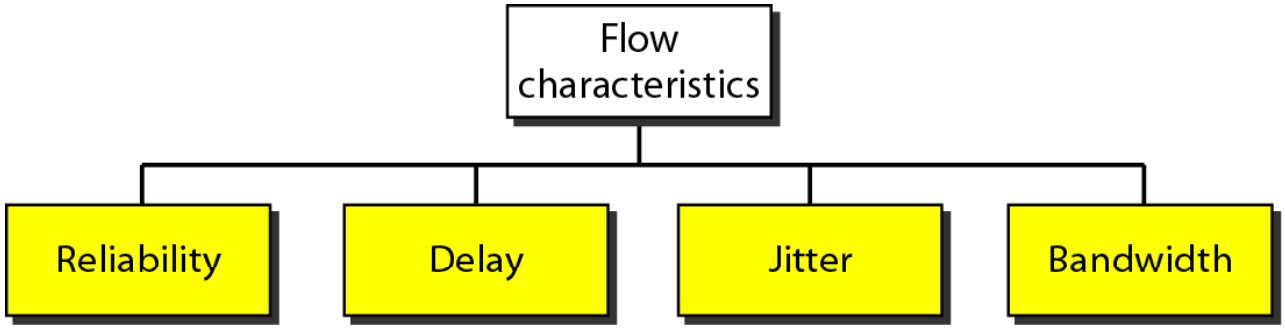


Figure 24.16 *FIFO queue*

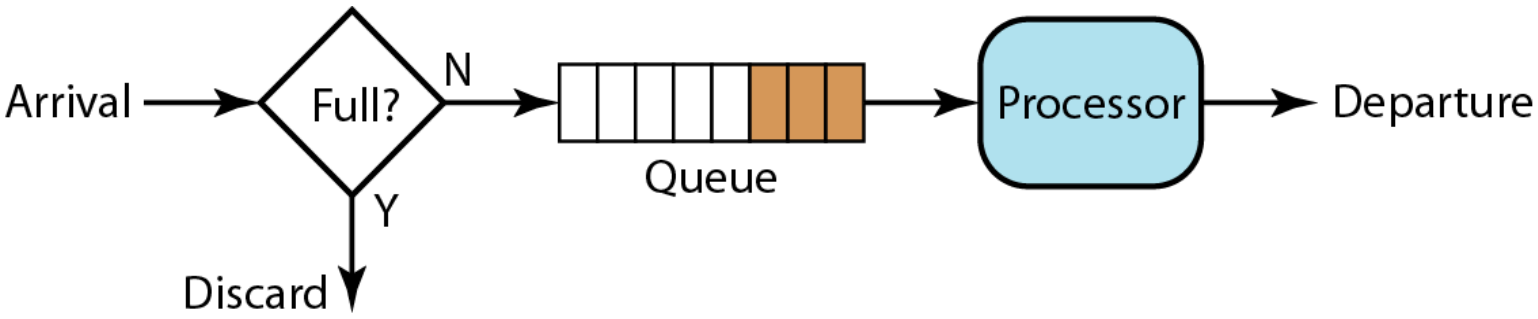


Figure 24.17 *Priority queuing*

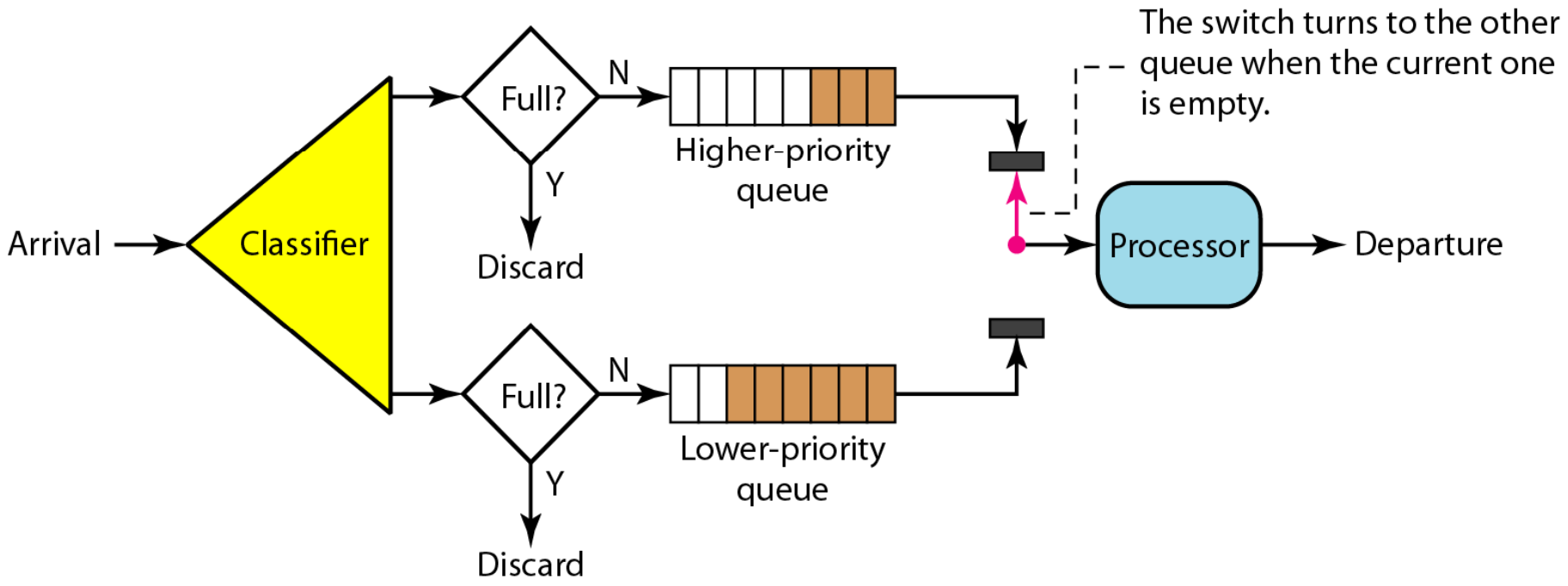


Figure 24.18 *Weighted fair queuing*

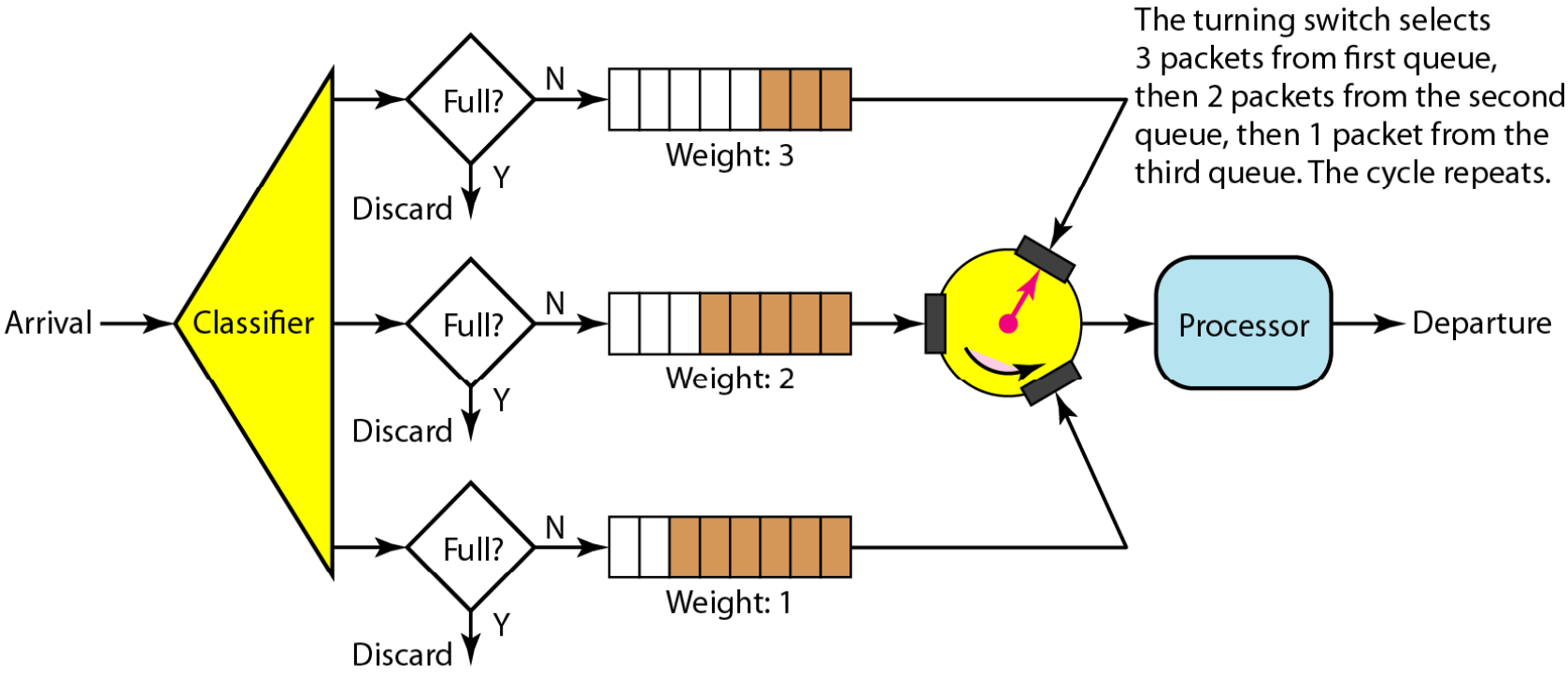


Figure 24.19 *Leaky bucket*

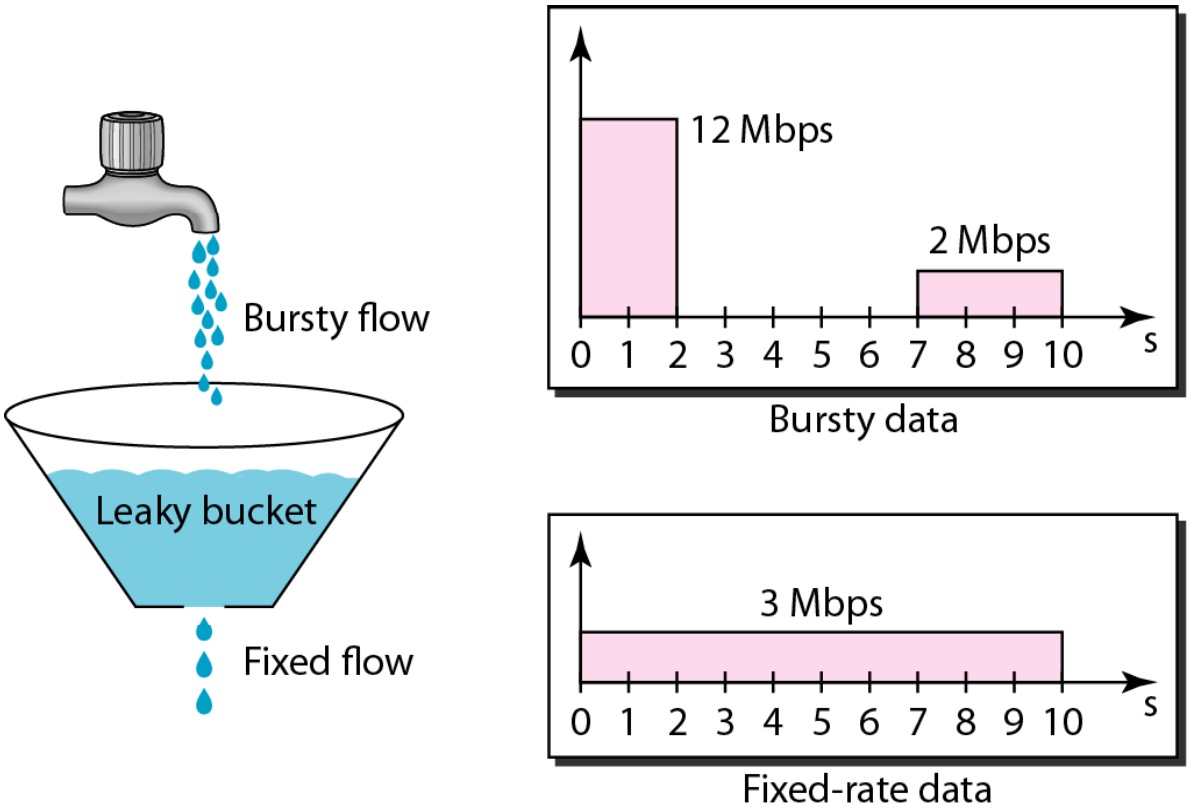


Figure 24.20 *Leaky bucket implementation*

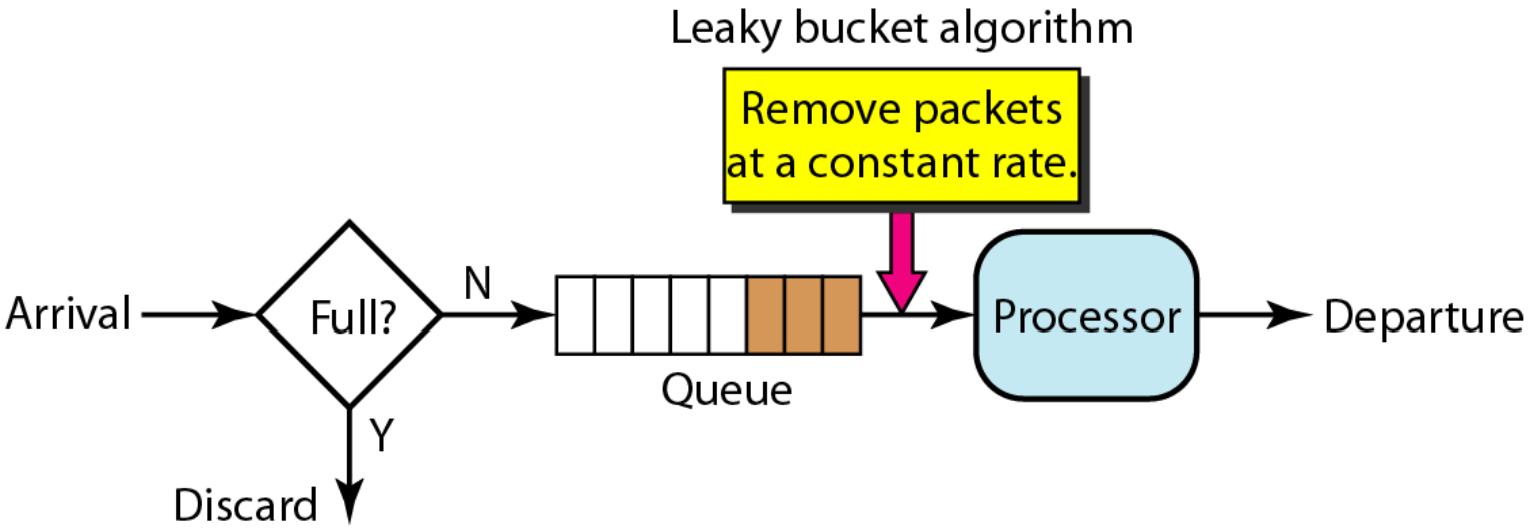
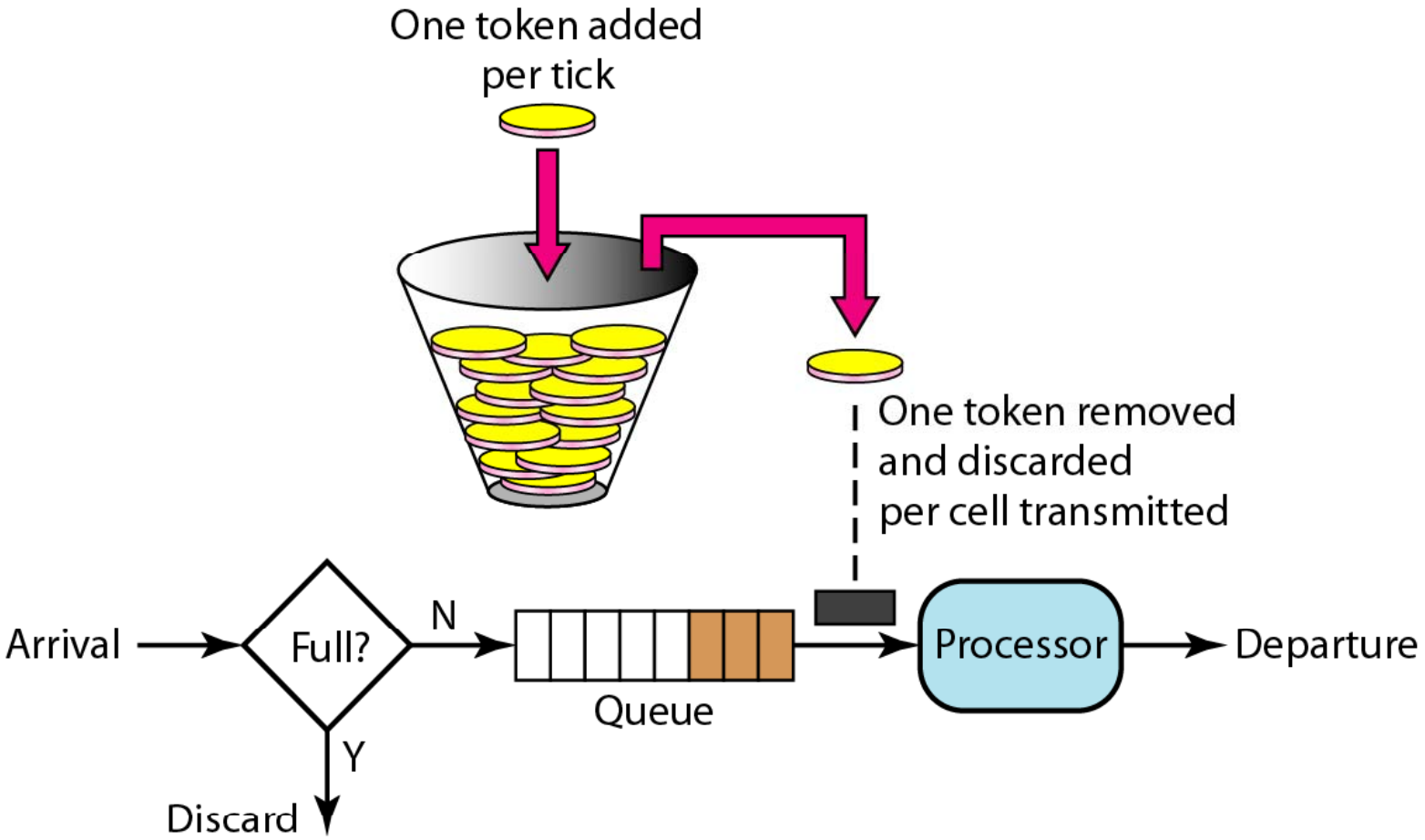
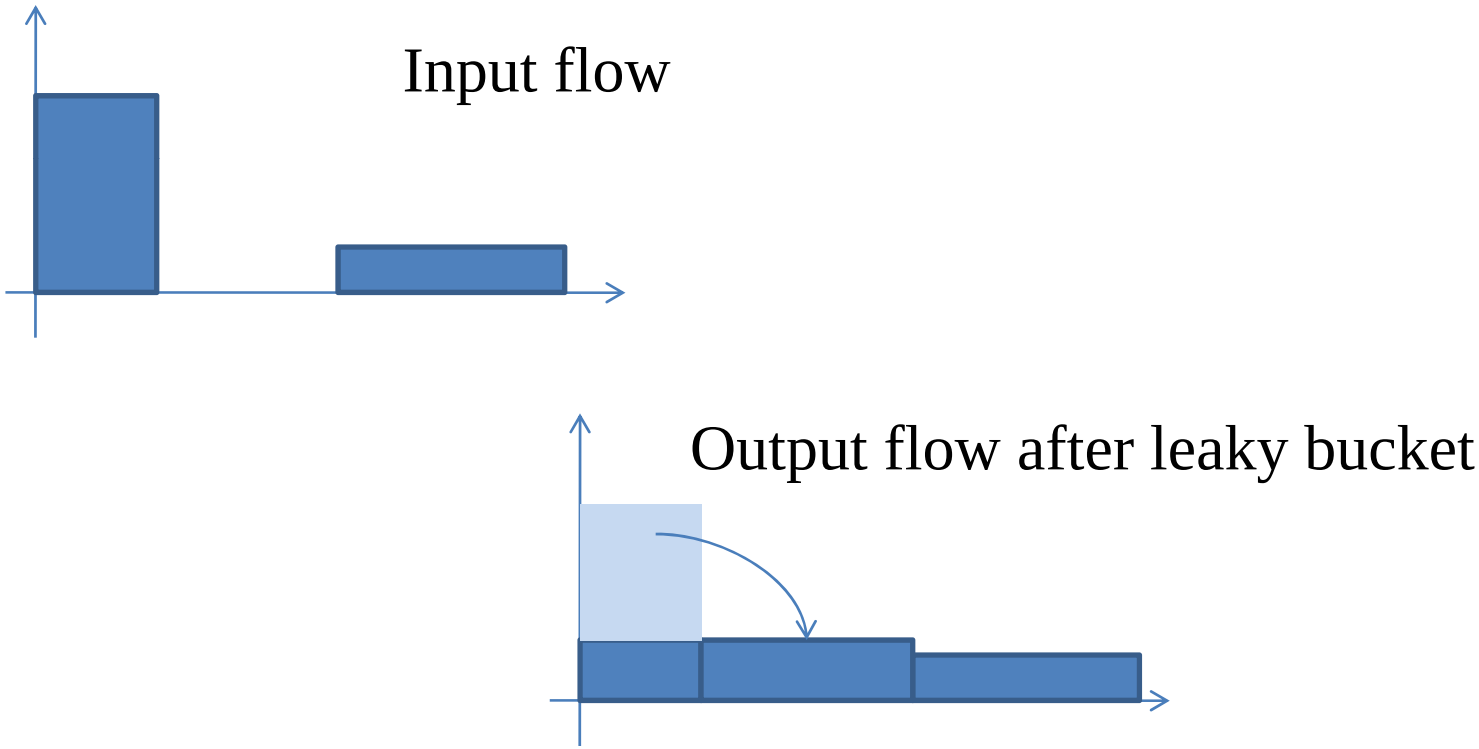


Figure 24.21 *Token bucket*



Leaky bucket



Token bucket