
# Chapter 23

# Process-to-Process Delivery: UDP, TCP, and SCTP

# 23-1 PROCESS-TO-PROCESS DELIVERY

***The transport layer is responsible for process-to-process delivery—the delivery of a packet, part of a message, from one process to another. Two processes communicate in a client/server relationship, as we will see later.***

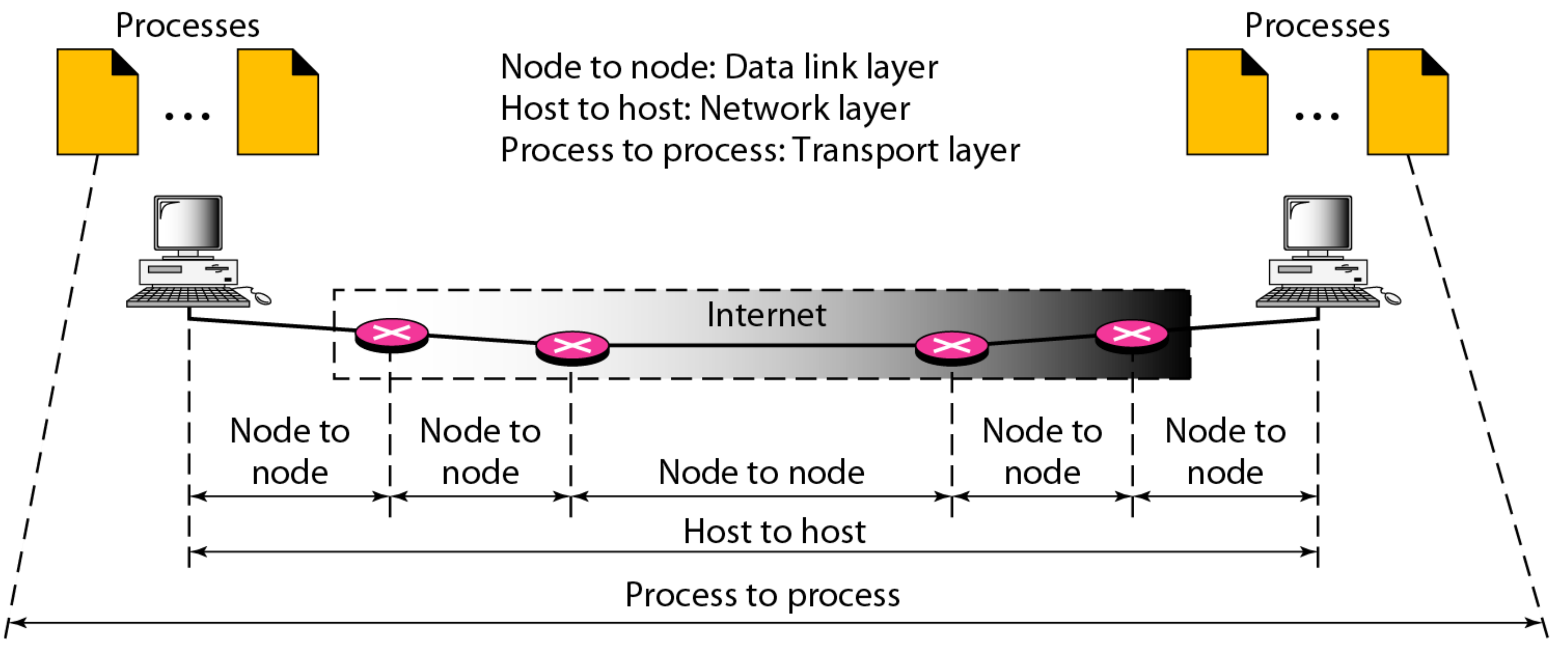
***Topics discussed in this section:***

**Client/Server Paradigm**
**Multiplexing and Demultiplexing**
**Connectionless Versus Connection-Oriented Service**
**Reliable Versus Unreliable**
**Three Protocols**

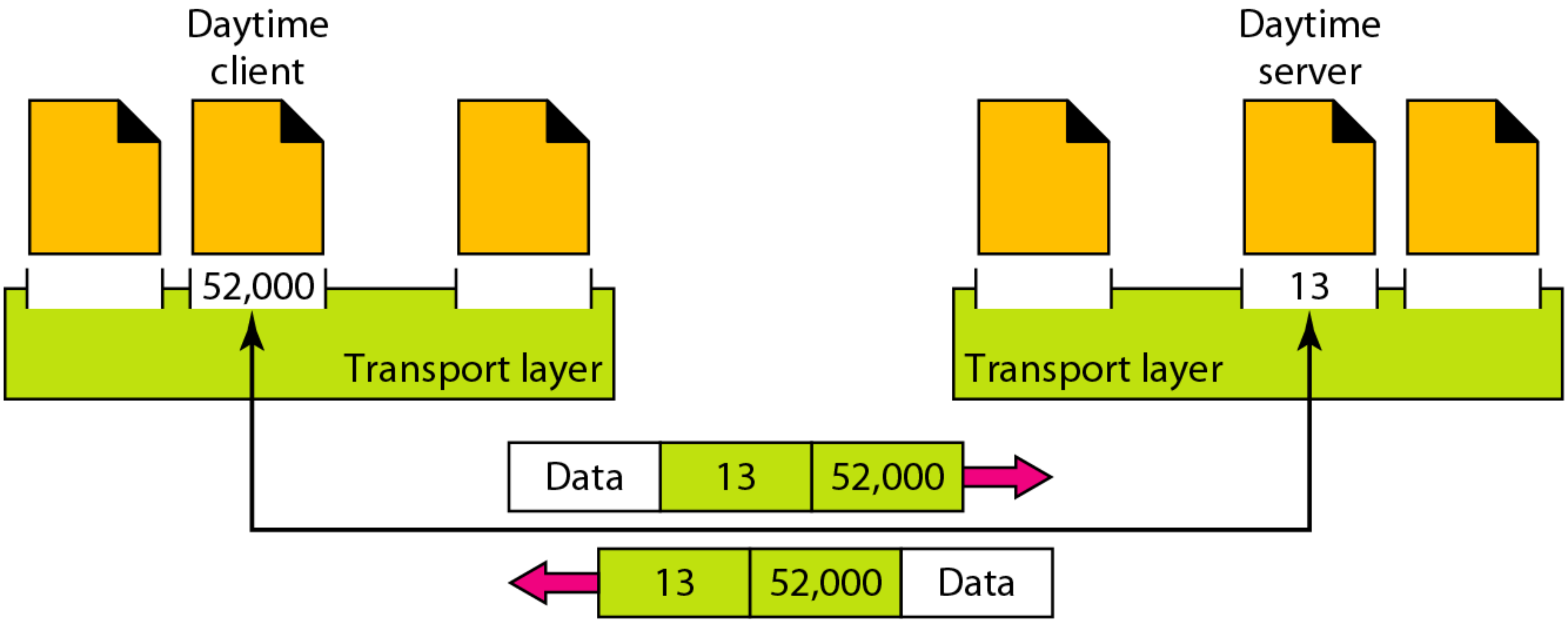
***Note***

**The transport layer is responsible for process-to-process delivery.**

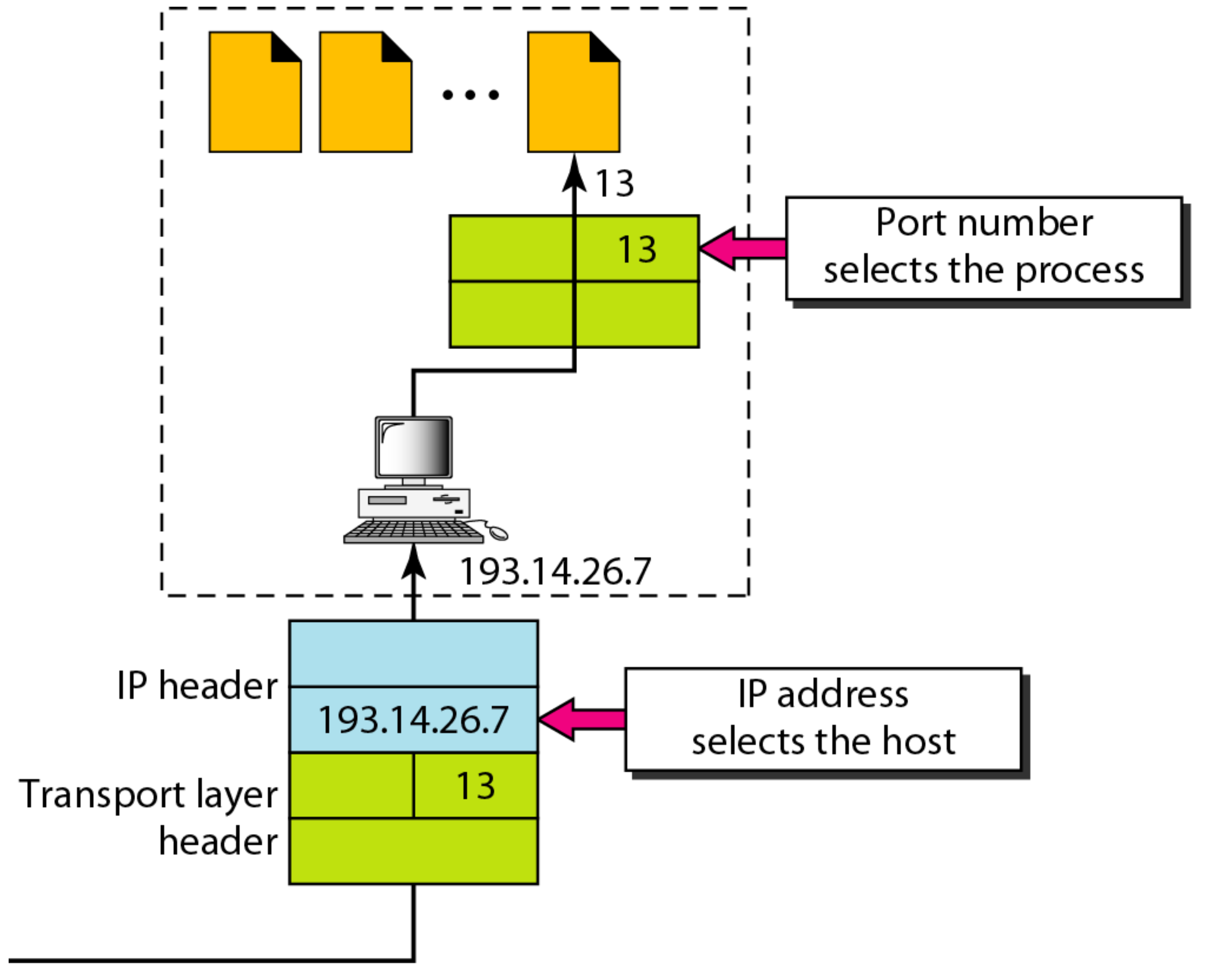
## Figure 23.1 *Types of data deliveries*



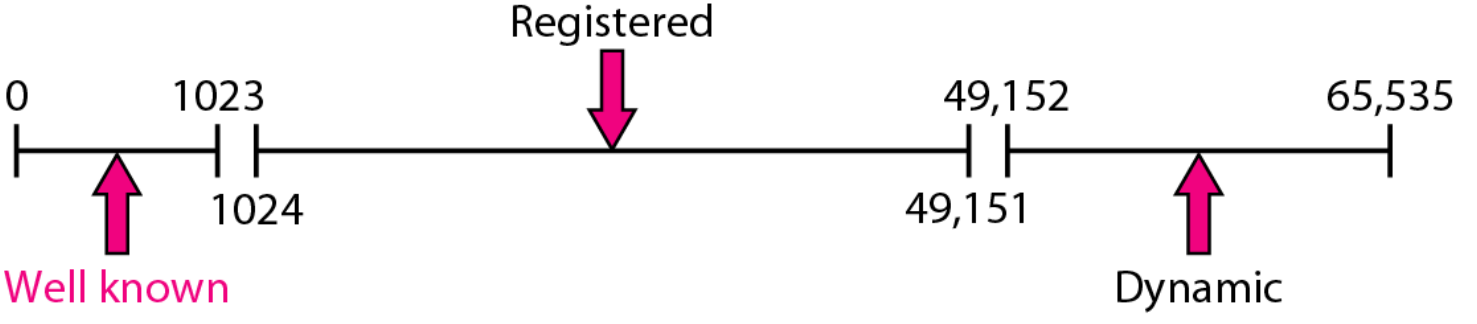
## Figure 23.2 *Port numbers*



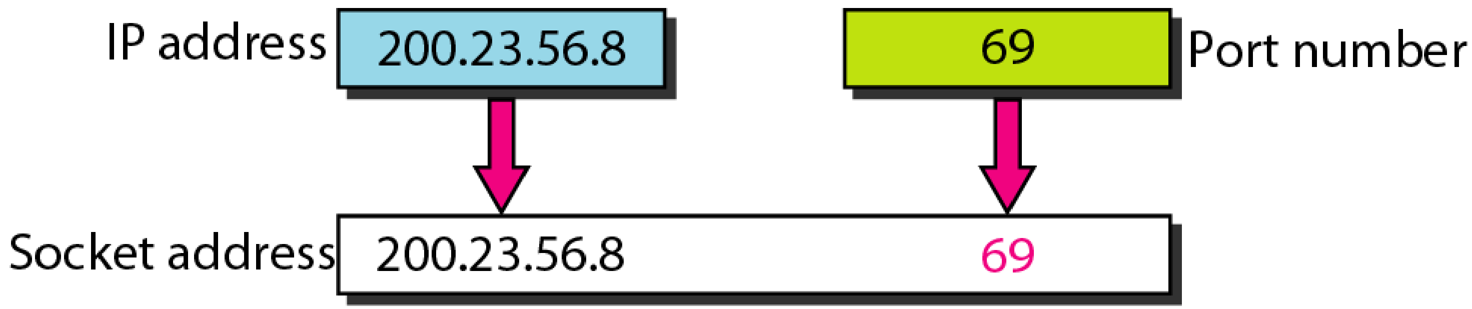
## Figure 23.3 *IP addresses versus port numbers*



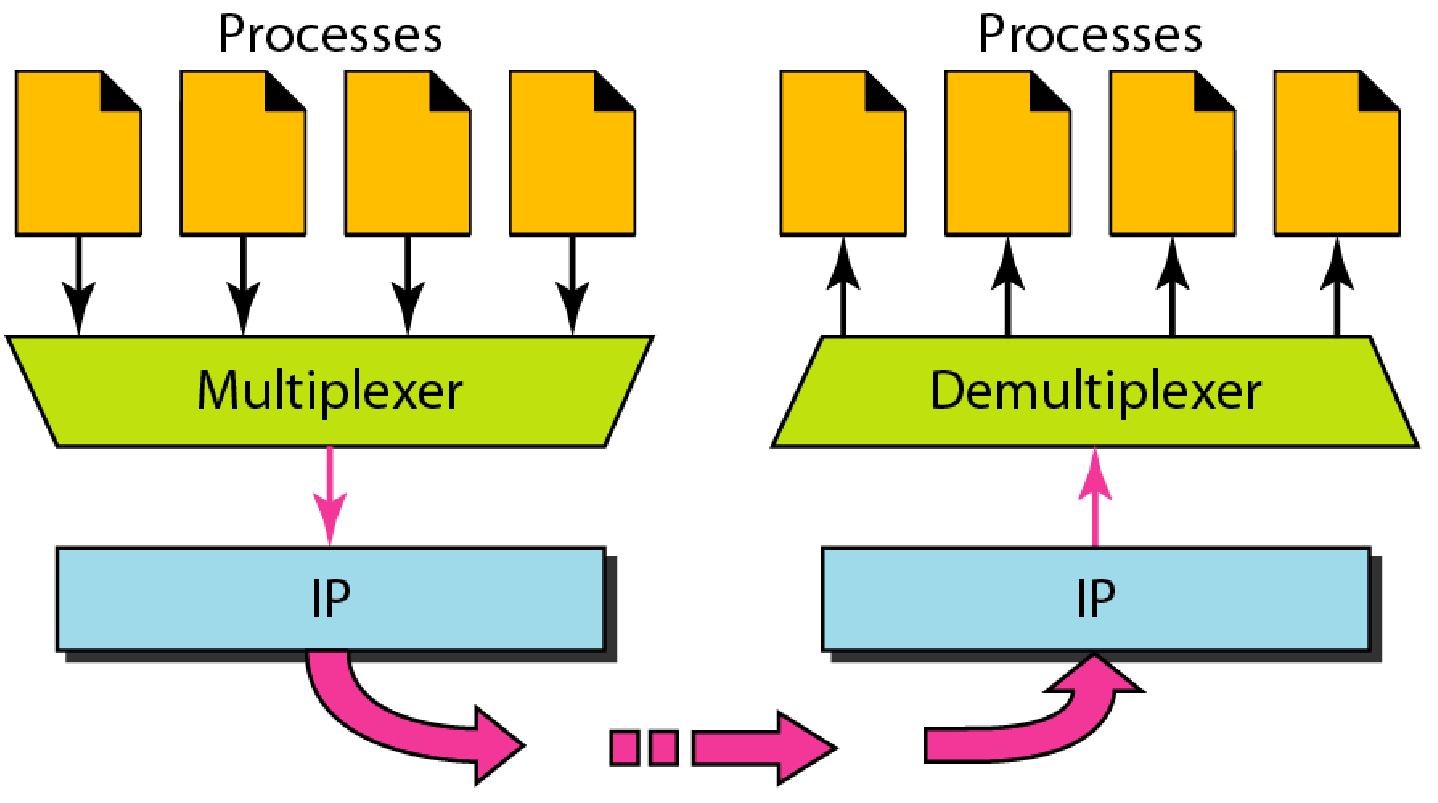
## **Figure 23.4** *IANA ranges*



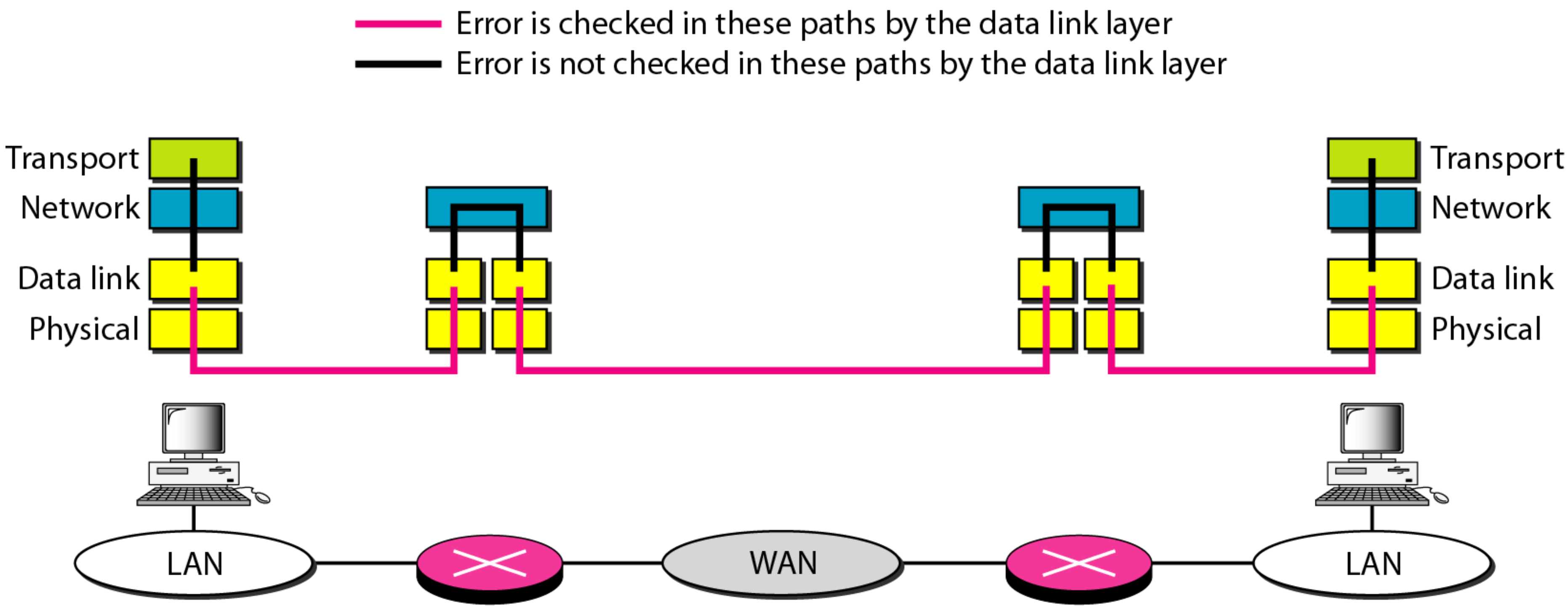
## Figure 23.5 *Socket address*



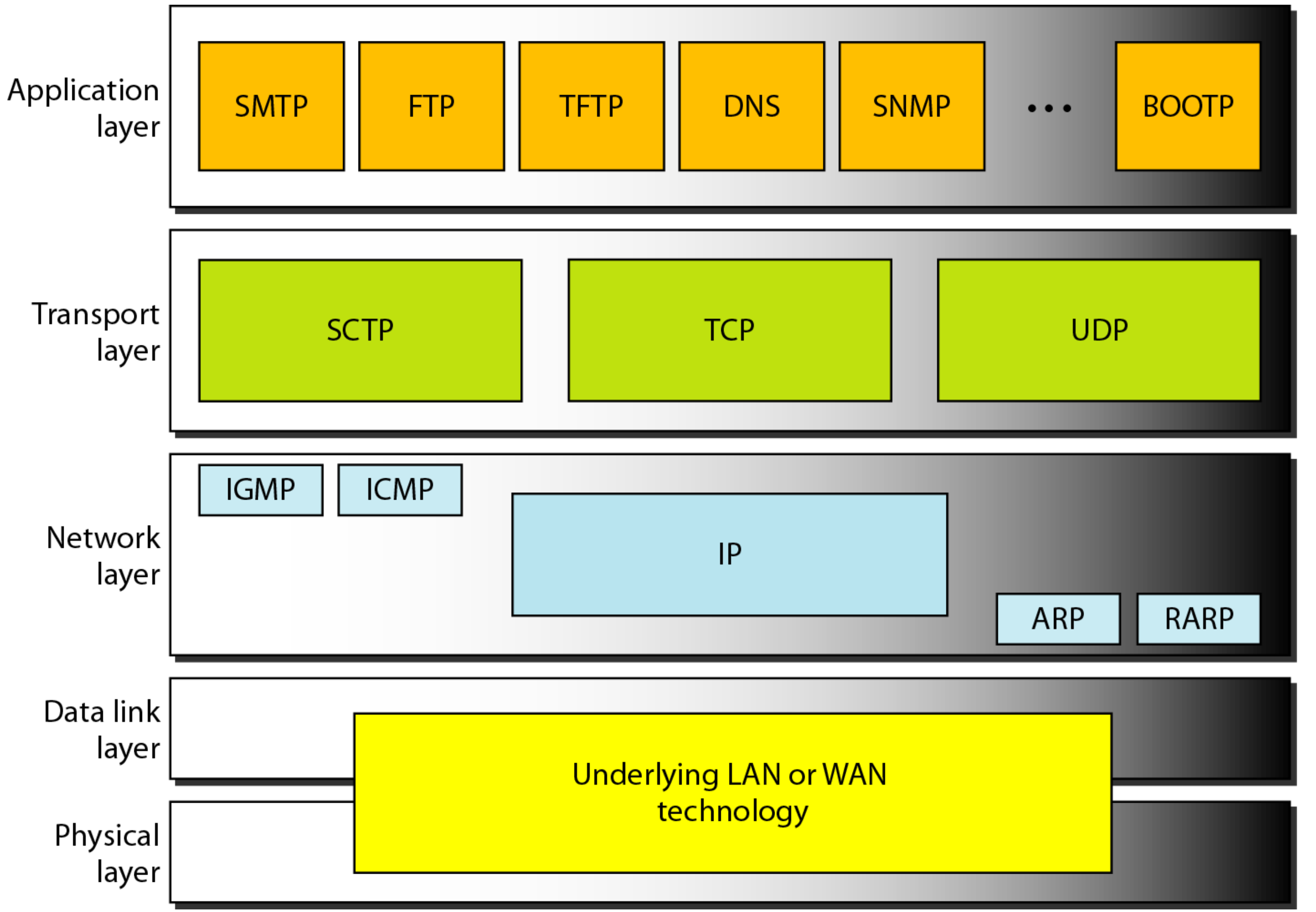
## **Figure 23.6** *Multiplexing and demultiplexing*



## **Figure 23.7** *Error control*



## Figure 23.8 *Position of UDP, TCP, and SCTP in TCP/IP suite*



# 23-2 USER DATAGRAM PROTOCOL (UDP)

***The User Datagram Protocol (UDP) is called a connectionless, unreliable transport protocol. It does not add anything to the services of IP except to provide process-to-process communication instead of host-to-host communication.***

***Topics discussed in this section:***

**Well-Known Ports for UDP**
**User Datagram**
**Checksum**
**UDP Operation**
**Use of UDP**

**Table 23.1** ***Well-known ports used with UDP***

| *Port* | *Protocol* | *Description* |
|---|---|---|
| 7 | Echo | Echoes a received datagram back to the sender |
| 9 | Discard | Discards any datagram that is received |
| 11 | Users | Active users |
| 13 | Daytime | Returns the date and the time |
| 17 | Quote | Returns a quote of the day |
| 19 | Chargen | Returns a string of characters |
| 53 | Nameserver | Domain Name Service |
| 67 | BOOTPs | Server port to download bootstrap information |
| 68 | BOOTPc | Client port to download bootstrap information |
| 69 | TFTP | Trivial File Transfer Protocol |
| 111 | RPC | Remote Procedure Call |
| 123 | NTP | Network Time Protocol |
| 161 | SNMP | Simple Network Management Protocol |
| 162 | SNMP | Simple Network Management Protocol (trap) |

## *Example 23.1*

***In UNIX, the well-known ports are stored in a file called /etc/services. Each line in this file gives the name of the server and the well-known port number. We can use the***
***grep utility to extract the line corresponding to the desired application. The following shows the port for FTP. Note that FTP can use port 21 with either UDP or TCP.***

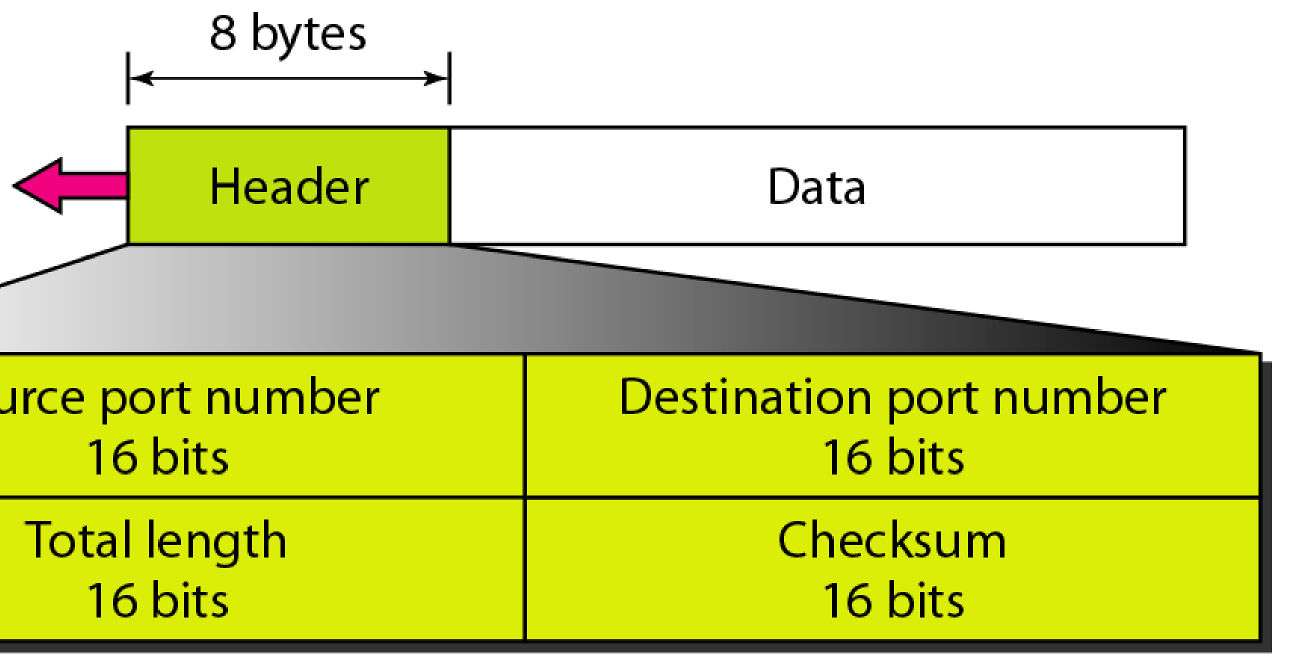
```
$ grep    ftp    /etc/services
ftp           21/tcp
ftp           21/udp
```

# Example 23.1 (continued)

***SNMP uses two port numbers (161 and 162), each for a different purpose, as we will see in Chapter 28.***

| | | |
|---|---|---|
| **$ grep** | **snmp /etc/services** | |
| snmp | 161/tcp | #Simple Net Mgmt Proto |
| snmp | 161/udp | #Simple Net Mgmt Proto |
| snmptrap | 162/udp | #Traps for SNMP |

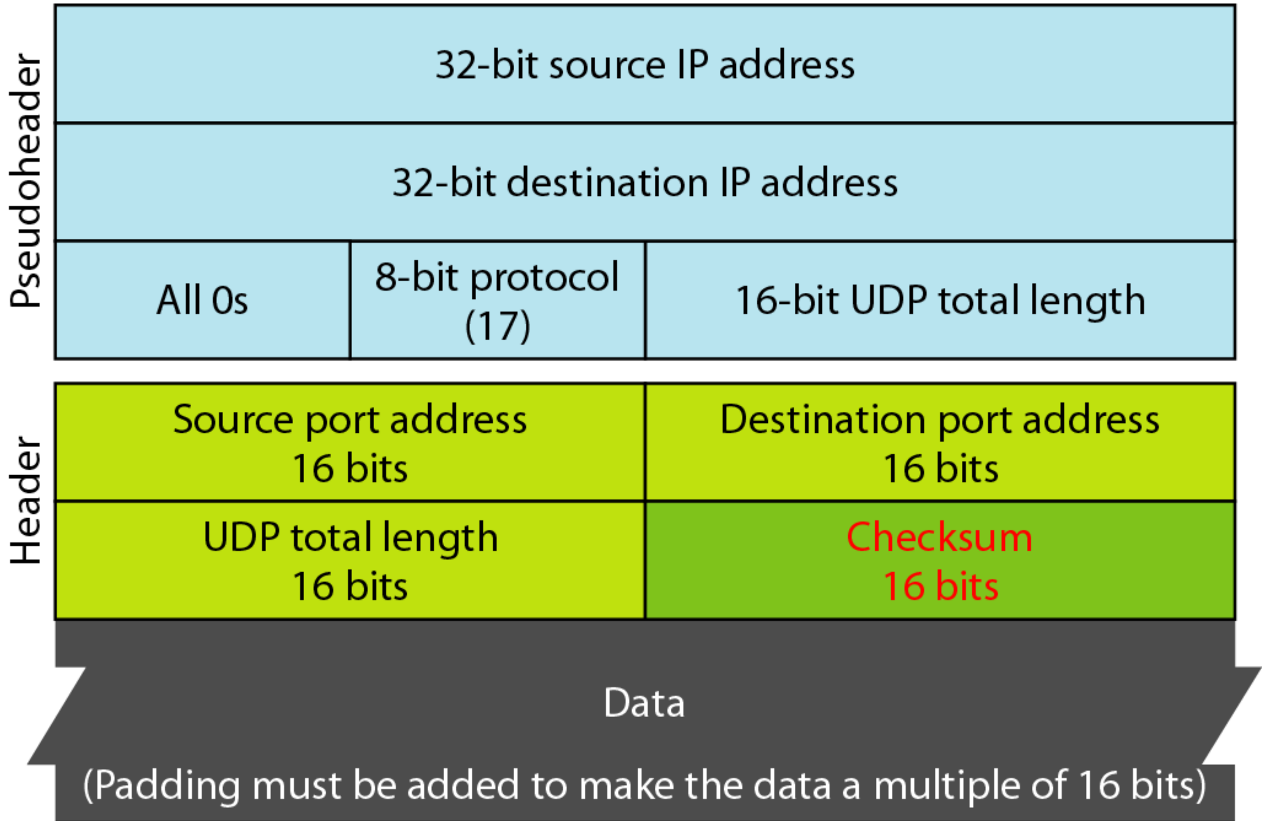
## Figure 23.9 *User datagram format*



***Note***

**UDP length**
**= IP length – IP header's length**

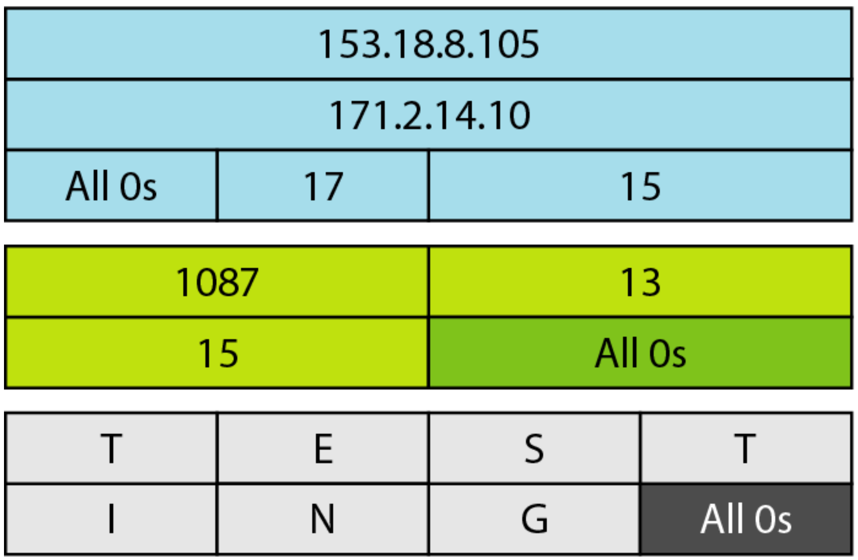
## **Figure 23.10** *Pseudoheader for checksum calculation*



# *Example 23.2*

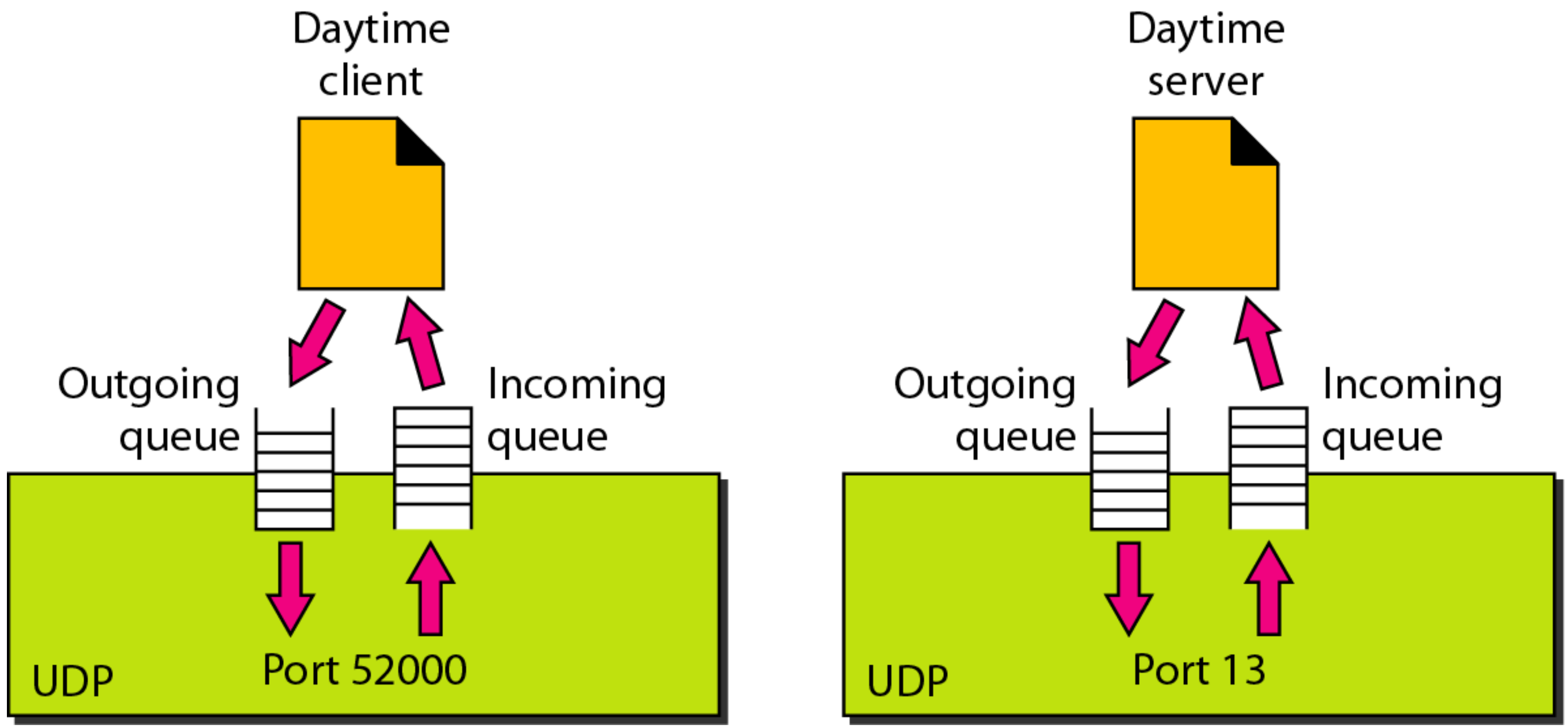
***Figure 23.11 shows the checksum calculation for a very small user datagram with only 7 bytes of data. Because the number of bytes of data is odd, padding is added for checksum calculation. The pseudoheader as well as the padding will be dropped when the user datagram is delivered to IP.***

## **Figure 23.11** *Checksum calculation of a simple UDP user datagram*




| | |
|---|---|
| 10011001 00010010 | → 153.18 |
| 00001000 01101001 | → 8.105 |
| 10101011 00000010 | → 171.2 |
| 00001110 00001010 | → 14.10 |
| 00000000 00010001 | → 0 and 17 |
| 00000000 00001111 | → 15 |
| 00000100 00111111 | → 1087 |
| 00000000 00001101 | → 13 |
| 00000000 00001111 | → 15 |
| 00000000 00000000 | → 0 (checksum) |
| 01010100 01000101 | → T and E |
| 01010011 01010100 | → S and T |
| 01001001 01001110 | → I and N |
| 01000111 00000000 | → G and 0 (padding) |
| 10010110 11101011 | → Sum |
| 01101001 00010100 | → Checksum |

## **Figure 23.12** *Queues in UDP*



# 23-3 TCP

***TCP is a connection-oriented protocol; it creates a virtual connection between two TCPs to send data. In addition, TCP uses flow and error control mechanisms at the transport level.***

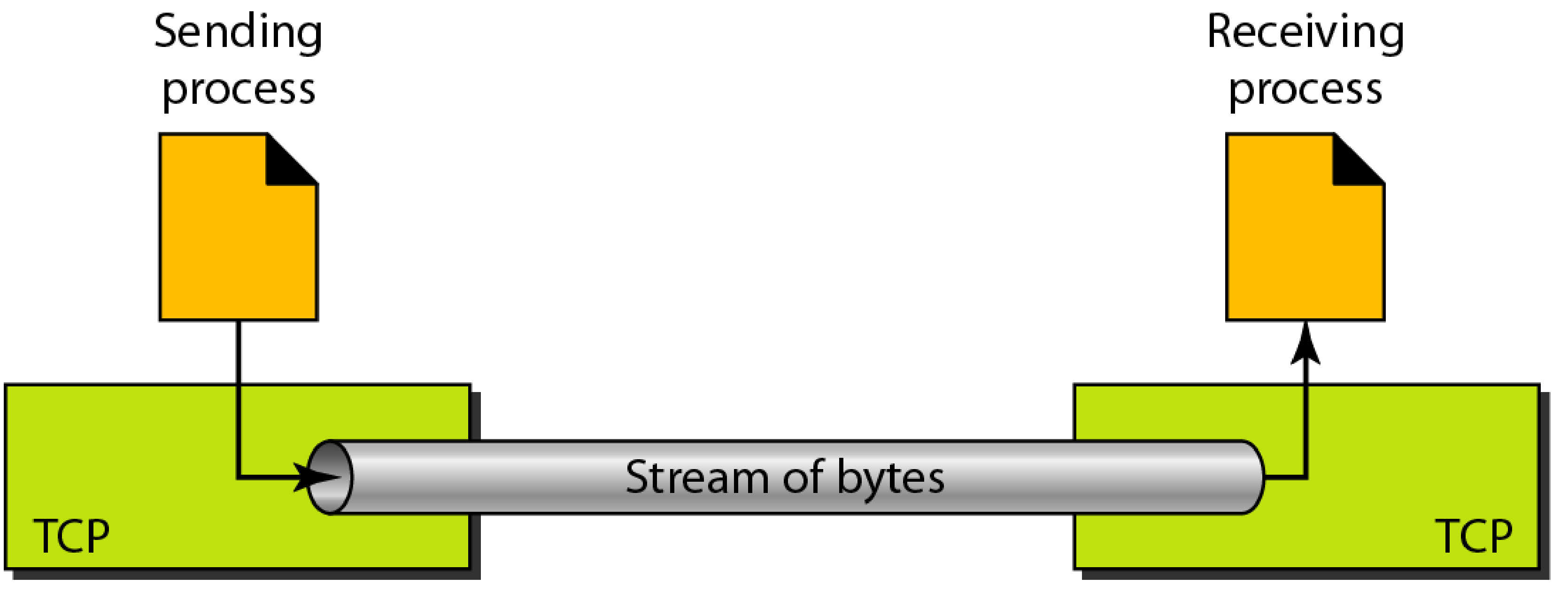
***Topics discussed in this section:***

**TCP Services**
**TCP Features**
**Segment**
**A TCP Connection**
**Flow Control**
**Error Control**

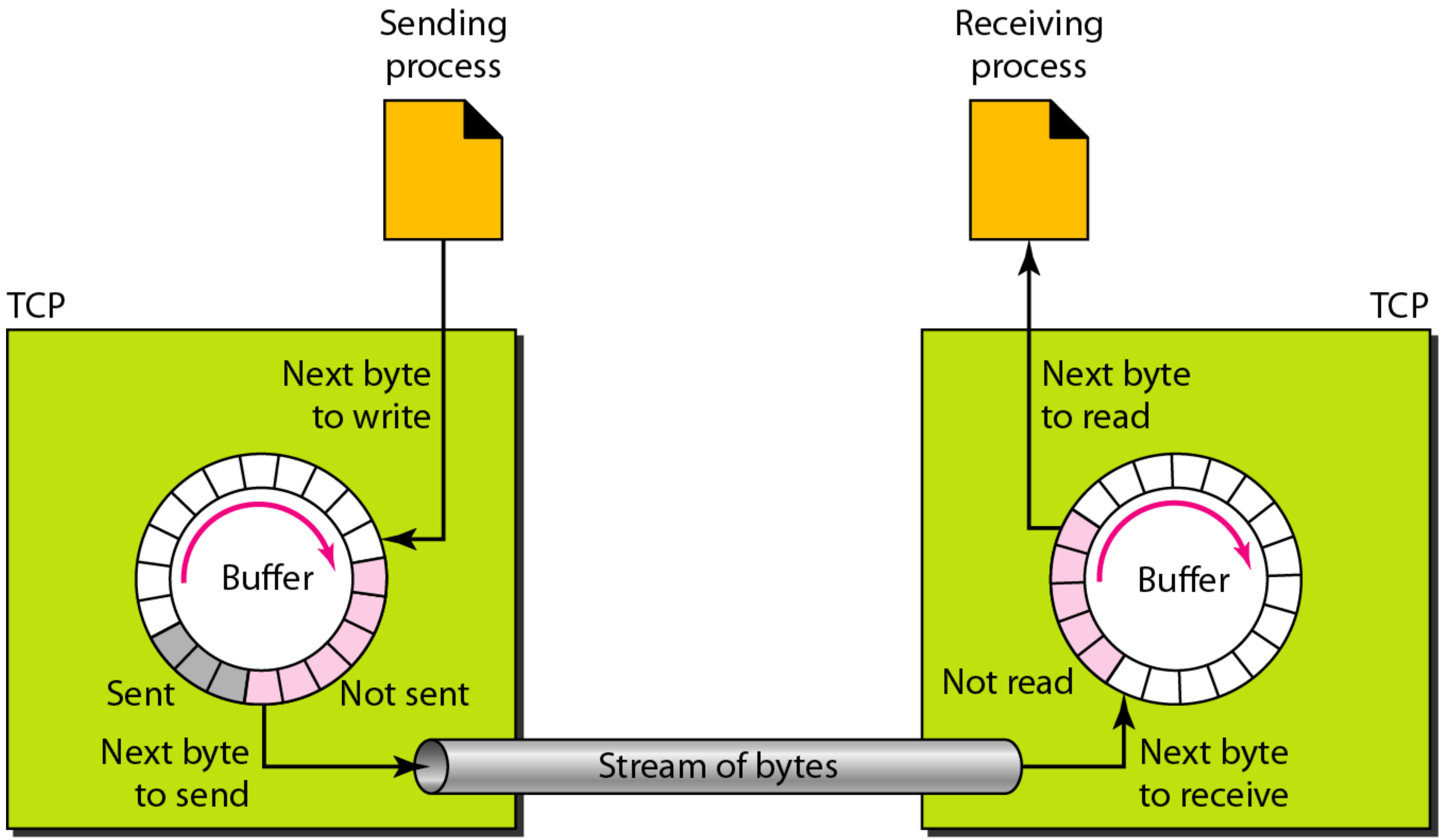
## **Table 23.2** *Well-known ports used by TCP*

| *Port* | *Protocol* | *Description* |
|---|---|---|
| 7 | Echo | Echoes a received datagram back to the sender |
| 9 | Discard | Discards any datagram that is received |
| 11 | Users | Active users |
| 13 | Daytime | Returns the date and the time |
| 17 | Quote | Returns a quote of the day |
| 19 | Chargen | Returns a string of characters |
| 20 | FTP, Data | File Transfer Protocol (data connection) |
| 21 | FTP, Control | File Transfer Protocol (control connection) |
| 23 | TELNET | Terminal Network |
| 25 | SMTP | Simple Mail Transfer Protocol |
| 53 | DNS | Domain Name Server |
| 67 | BOOTP | Bootstrap Protocol |
| 79 | Finger | Finger |
| 80 | HTTP | Hypertext Transfer Protocol |
| 111 | RPC | Remote Procedure Call |

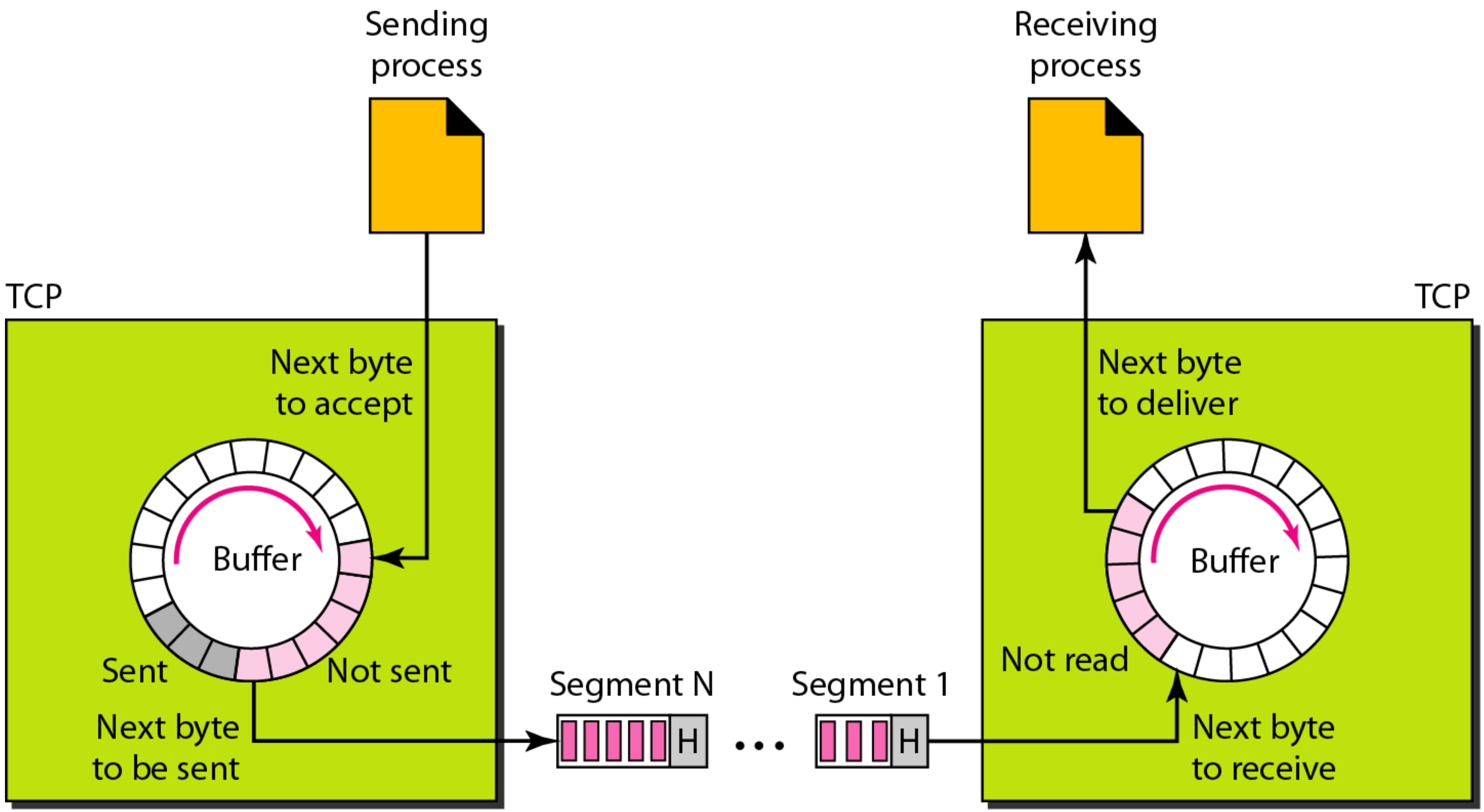
## Figure 23.13 *Stream delivery*



## **Figure 23.14** *Sending and receiving buffers*



## Figure 23.15 *TCP segments*



***Note***

**The bytes of data being transferred in each connection are numbered by TCP. The numbering starts with a randomly generated number.**

# *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) |

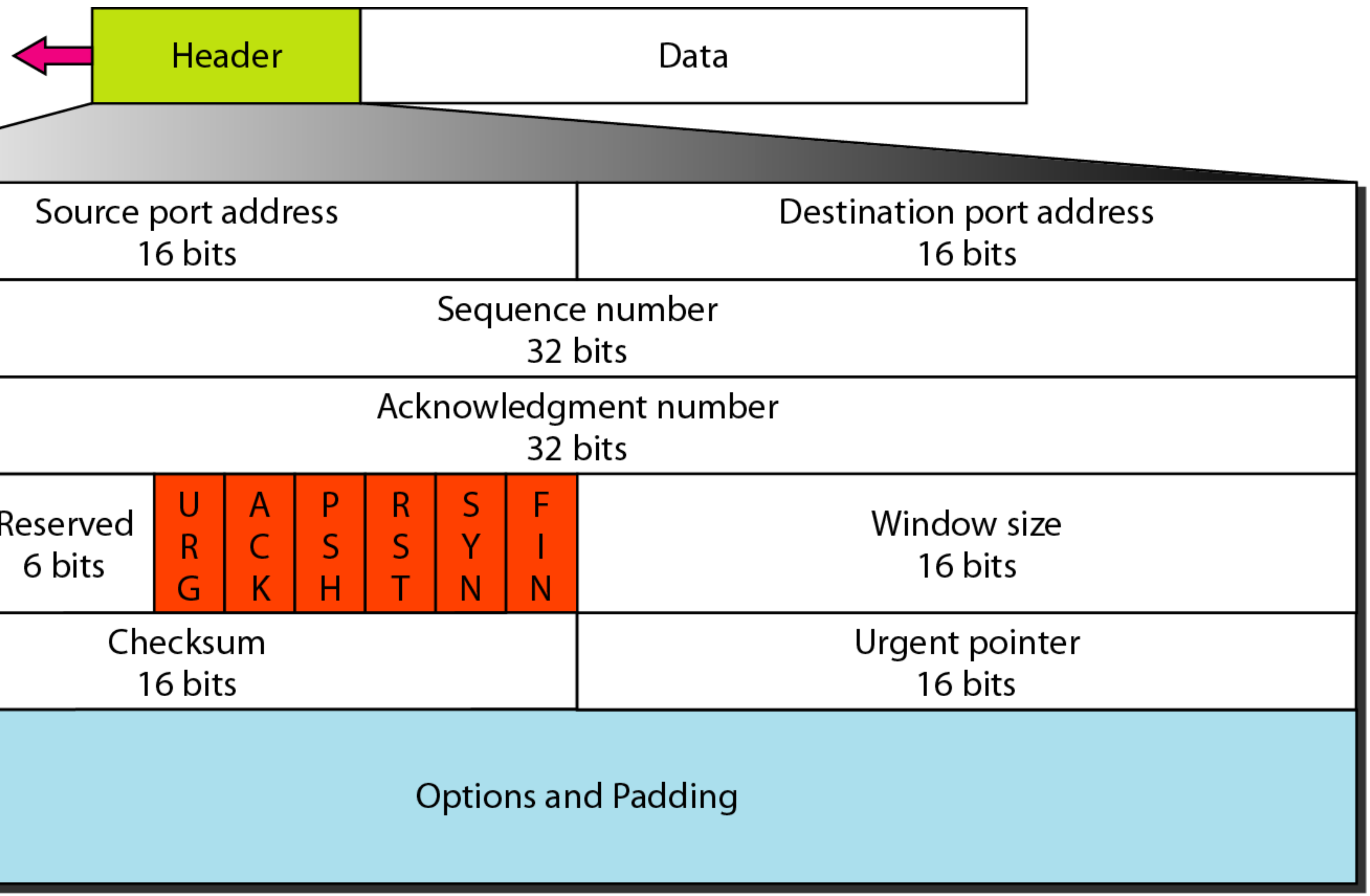
***Note***

**The value in the sequence number field of a segment defines the number of the first data byte contained in that segment.**

***Note***

**The value of the acknowledgment field in a segment defines the number of the next byte a party expects to receive.**
**The acknowledgment number is cumulative.**

## Figure 23.16 *TCP segment format*



## 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

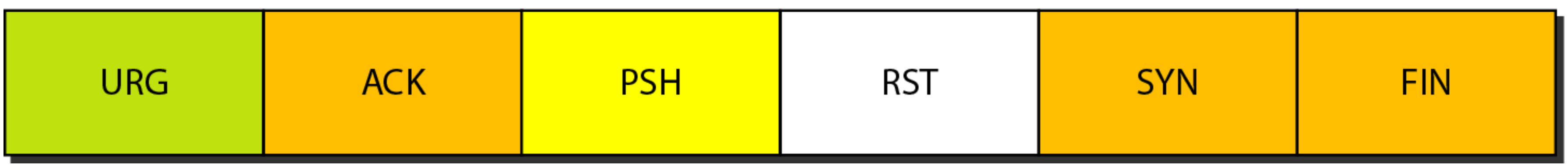
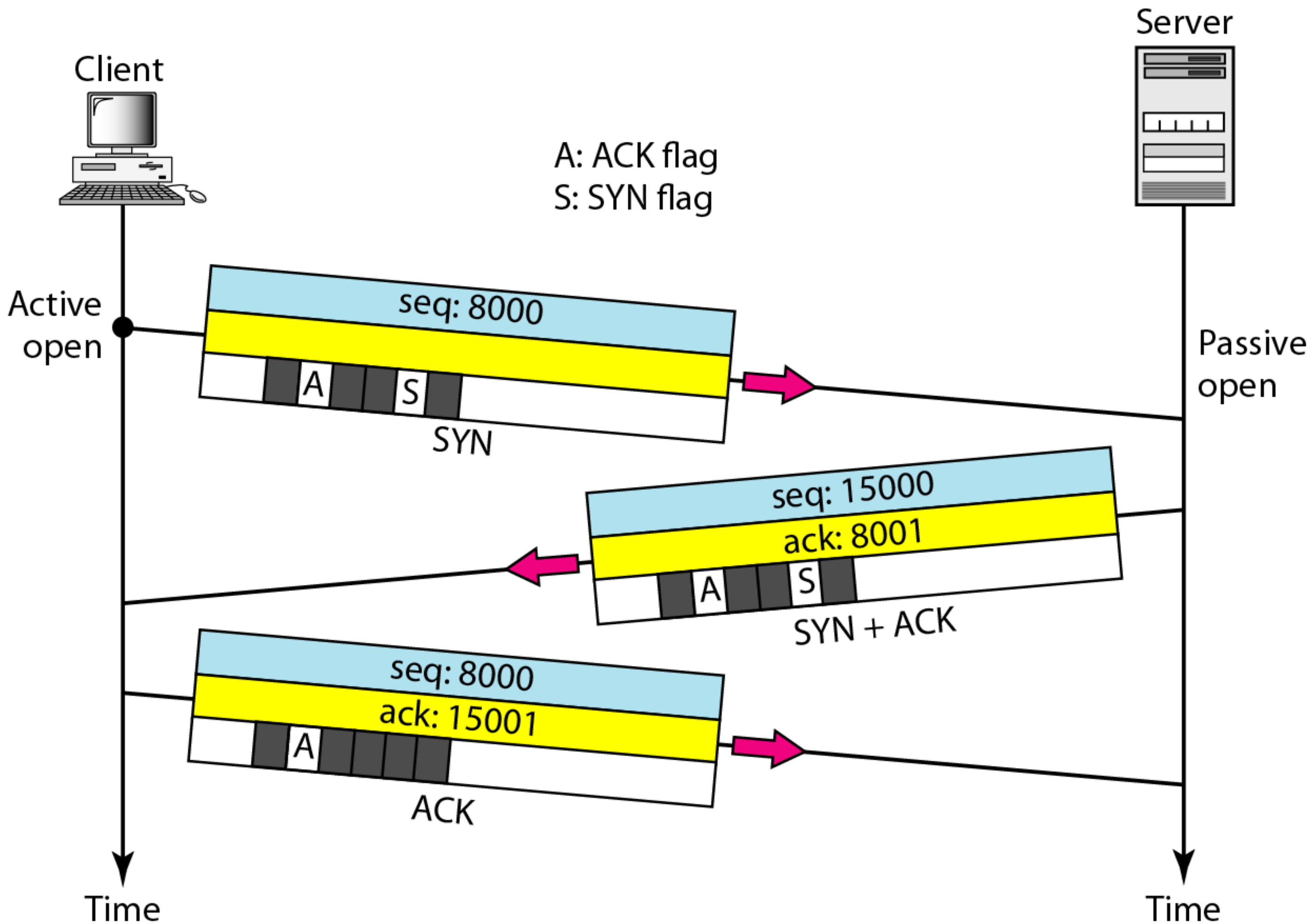


**Table 23.3** ***Description of flags in the control field***

| *Flag* | *Description* |
|---|---|
| URG | The value of the urgent pointer field is valid. |
| ACK | The value of the acknowledgment field is valid. |
| PSH | Push the data. |
| RST | Reset the connection. |
| SYN | Synchronize sequence numbers during connection. |
| FIN | Terminate the connection. |

## Figure 23.18 *Connection establishment using three-way handshaking*



***Note***

**A SYN segment cannot carry data, but it consumes one sequence number.**

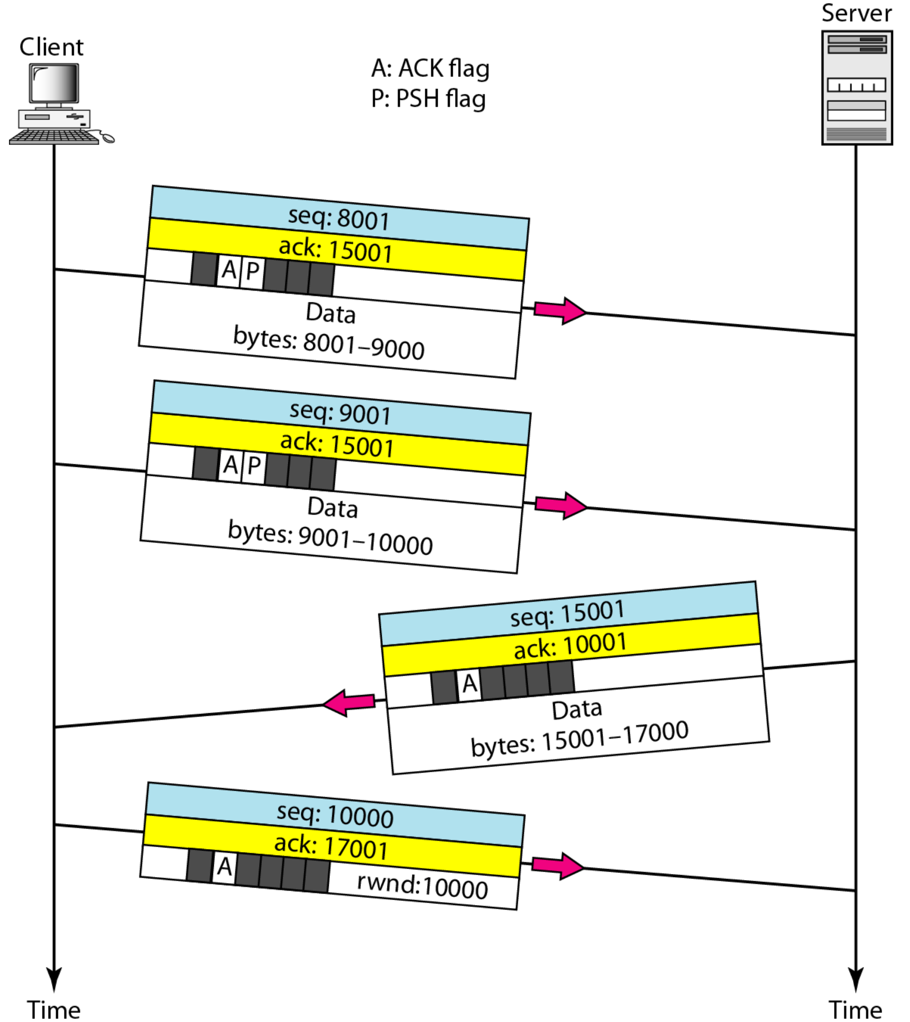
***Note***

**A SYN + ACK segment cannot carry data, but does consume one sequence number.**

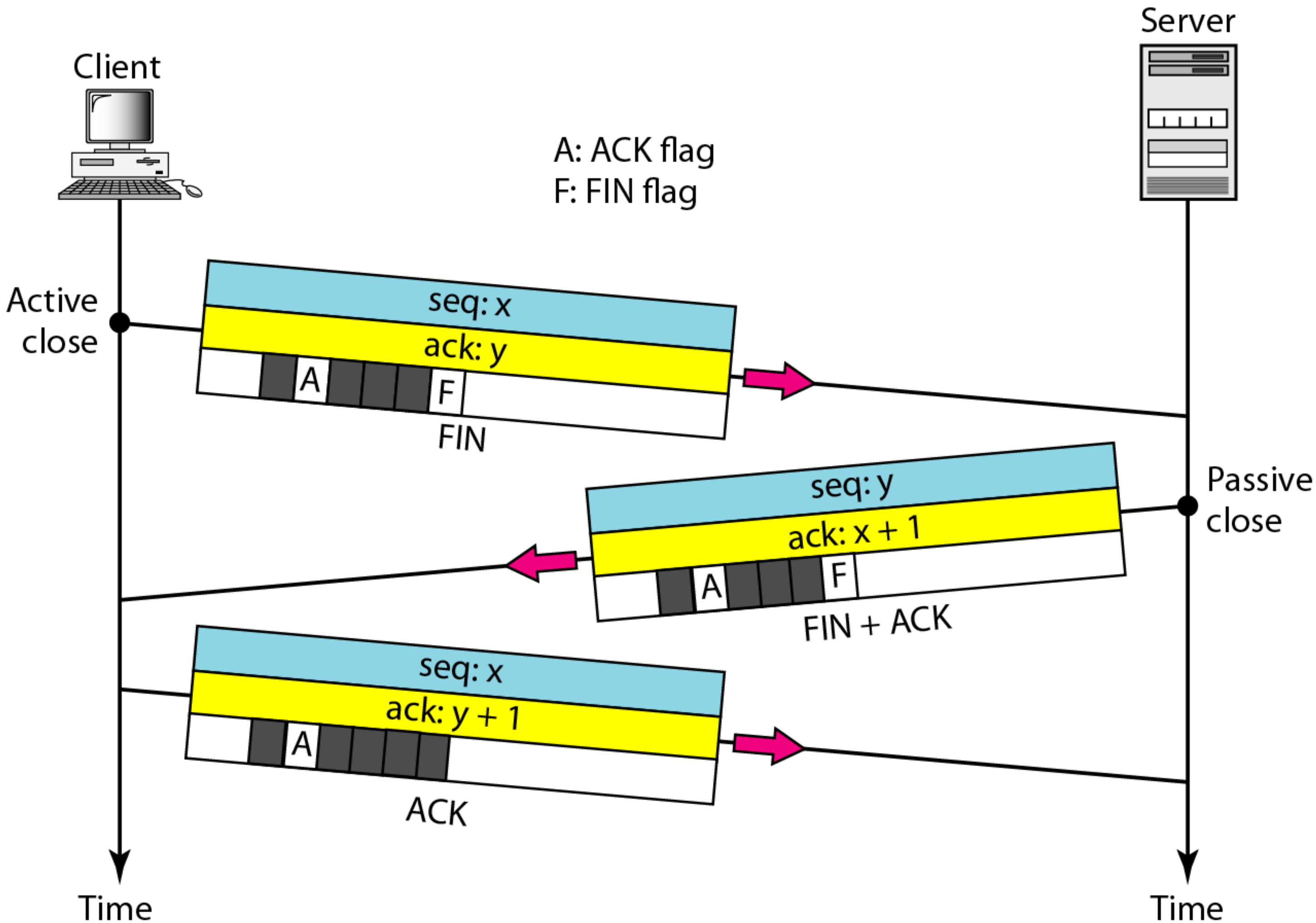
***Note***

**An ACK segment, if carrying no data, consumes no sequence number.**

# Figure 23.19 *Data transfer*



## **Figure 23.20** *Connection termination using three-way handshaking*



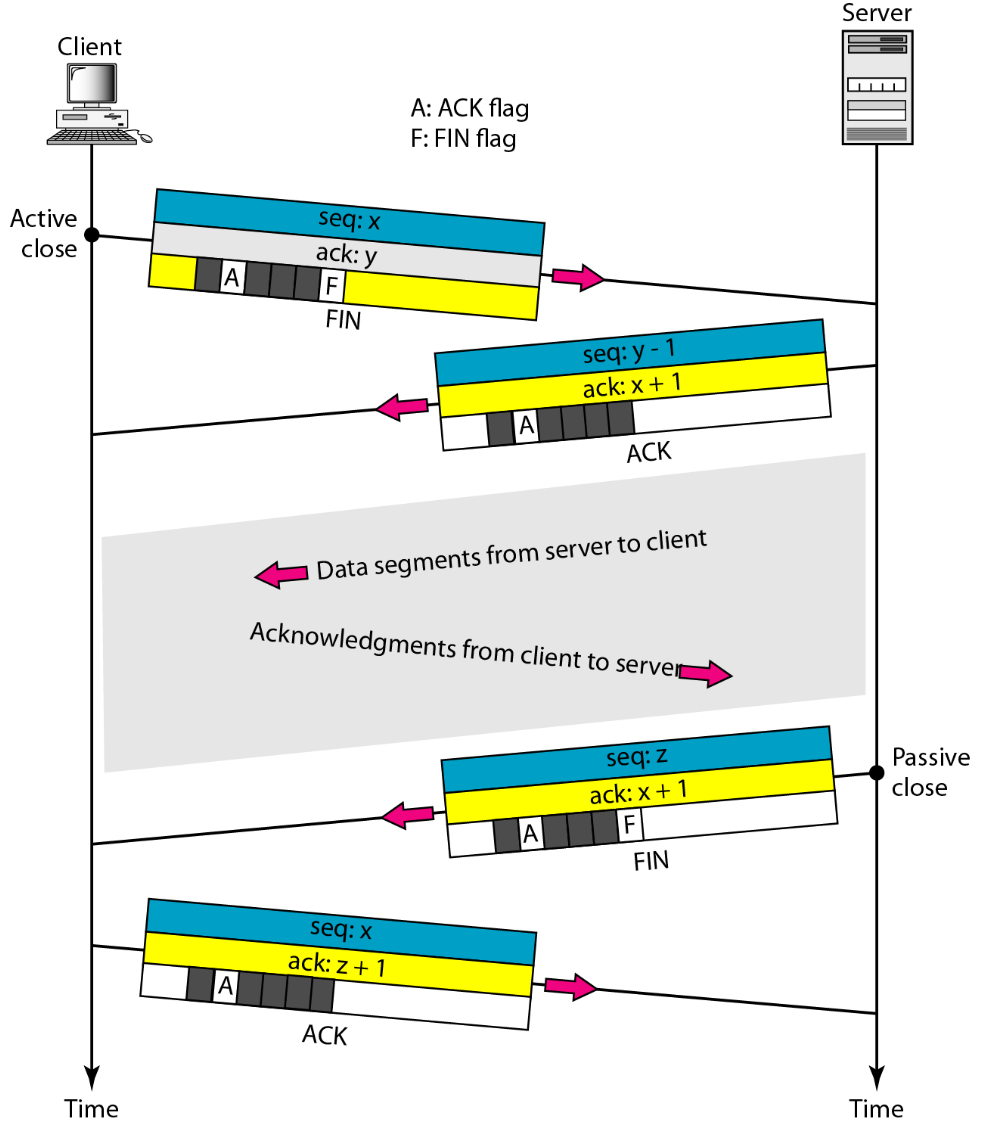
***Note***

**The FIN segment consumes one sequence number if it does not carry data.**

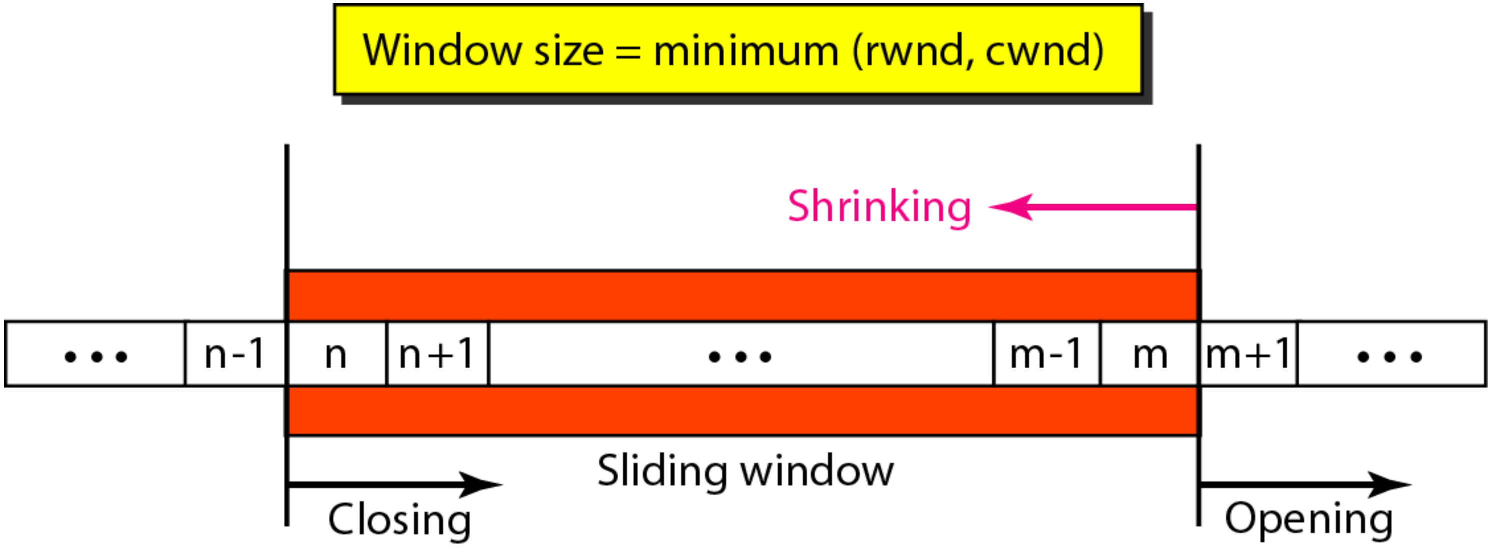
***Note***

**The FIN + ACK segment consumes one sequence number if it does not carry data.**

# Figure 23.21 *Half-close*



## **Figure 23.22** ***Sliding window***



***Note***

**A sliding window is used to make transmission more efficient as well as to control the flow of data so that the destination does not become overwhelmed with data.**
**TCP sliding windows are byte-oriented.**

## Example 23.4

***What is the value of the receiver window (rwnd) for host A if the receiver, host B, has a buffer size of 5000 bytes and 1000 bytes of received and unprocessed data?***

### Solution

***The value of rwnd = 5000 − 1000 = 4000. Host B can receive only 4000 bytes of data before overflowing its buffer. Host B advertises this value in its next segment to A.***

# *Example 23.5*

***What is the size of the window for host A if the value of rwnd is 3000 bytes and the value of cwnd is 3500 bytes?***

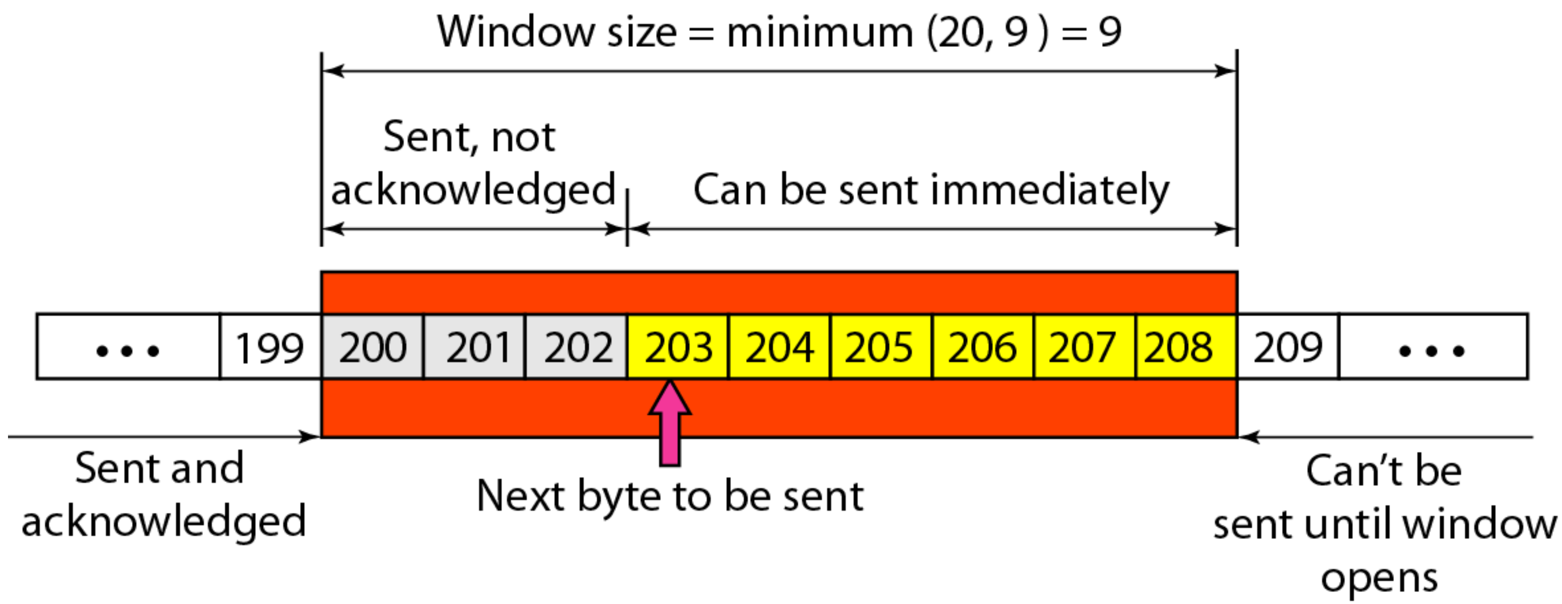
## *Solution*

***The size of the window is the smaller of rwnd and cwnd, which is 3000 bytes.***

# Example 23.6

***Figure 23.23 shows an unrealistic example of a sliding window. The sender has sent bytes up to 202. We assume that cwnd is 20 (in reality this value is thousands of bytes). The receiver has sent an acknowledgment number of 200 with an rwnd of 9 bytes (in reality this value is thousands of bytes). The size of the sender window is the minimum of rwnd and cwnd, or 9 bytes. Bytes 200 to 202 are sent, but not acknowledged. Bytes 203 to 208 can be sent without worrying about acknowledgment. Bytes 209 and above cannot be sent.***

## Figure 23.23 *Example 23.6*



*Note*

**Some points about TCP sliding windows:**

- The size of the window is the lesser of rwnd and cwnd.
- The source does not have to send a full window's worth of data.
- The window can be opened or closed by the receiver, but should not be shrunk.
- The destination can send an acknowledgment at any time as long as it does not result in a shrinking window.
- The receiver can temporarily shut down the window; the sender, however, can always send a segment of 1 byte after the window is shut down.

**Note**

**ACK segments do not consume sequence numbers and are not acknowledged.**

***Note***

**In modern implementations, a retransmission occurs if the retransmission timer expires or three duplicate ACK segments have arrived.**

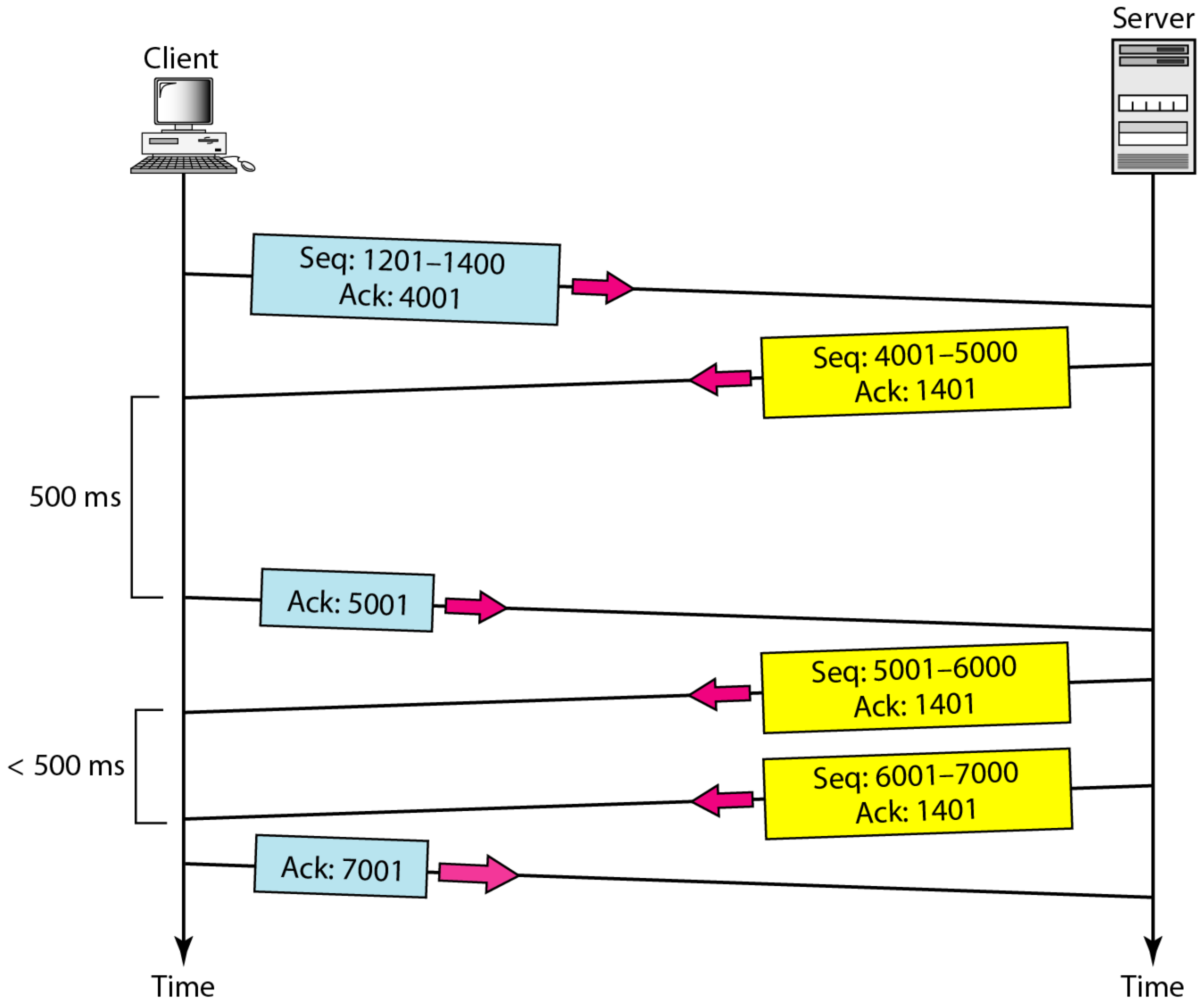
***Note***

**No retransmission timer is set for an ACK segment.**

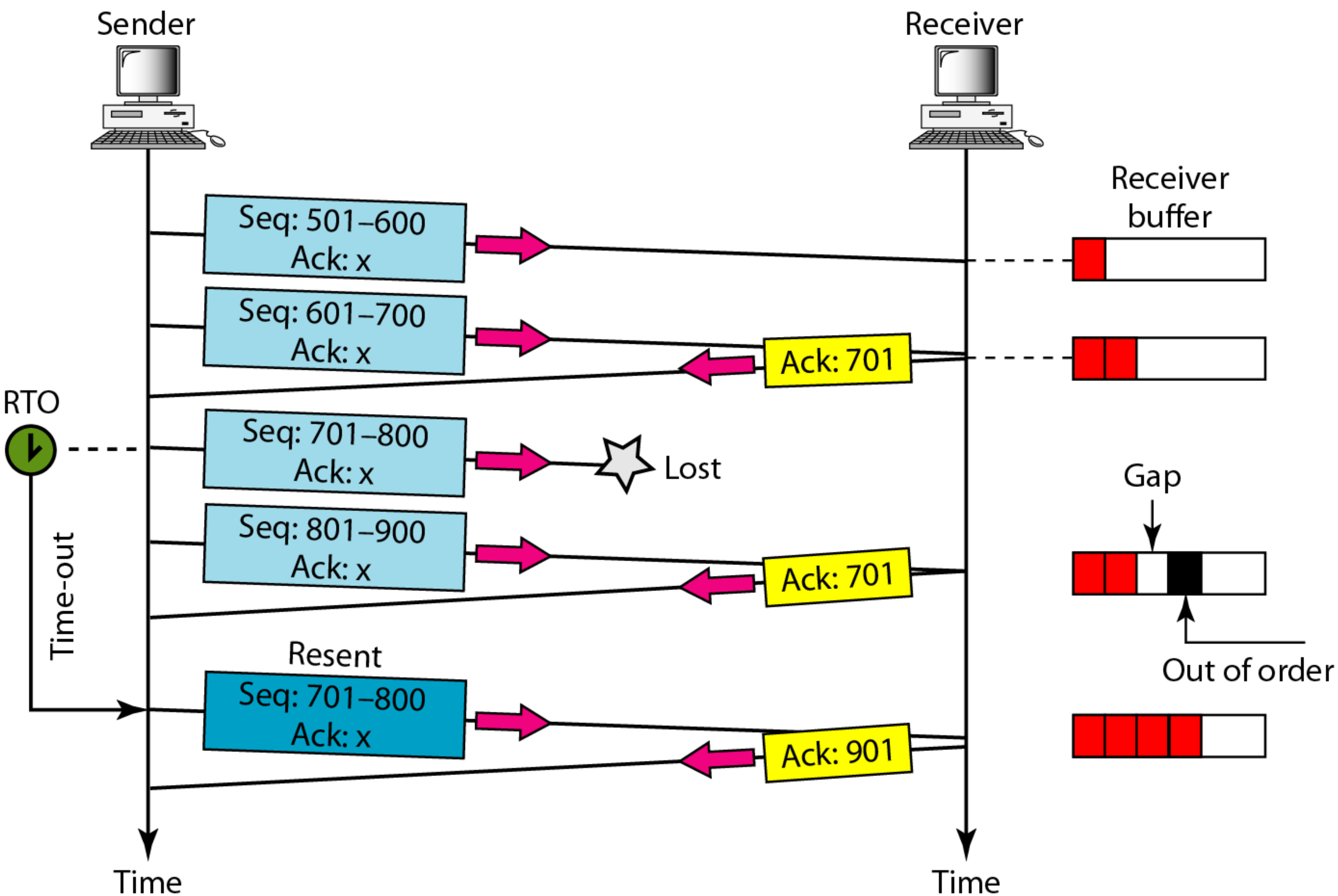
***Note***

**Data may arrive out of order and be temporarily stored by the receiving TCP, but TCP guarantees that no out-of-order segment is delivered to the process.**

## Figure 23.24 *Normal operation*



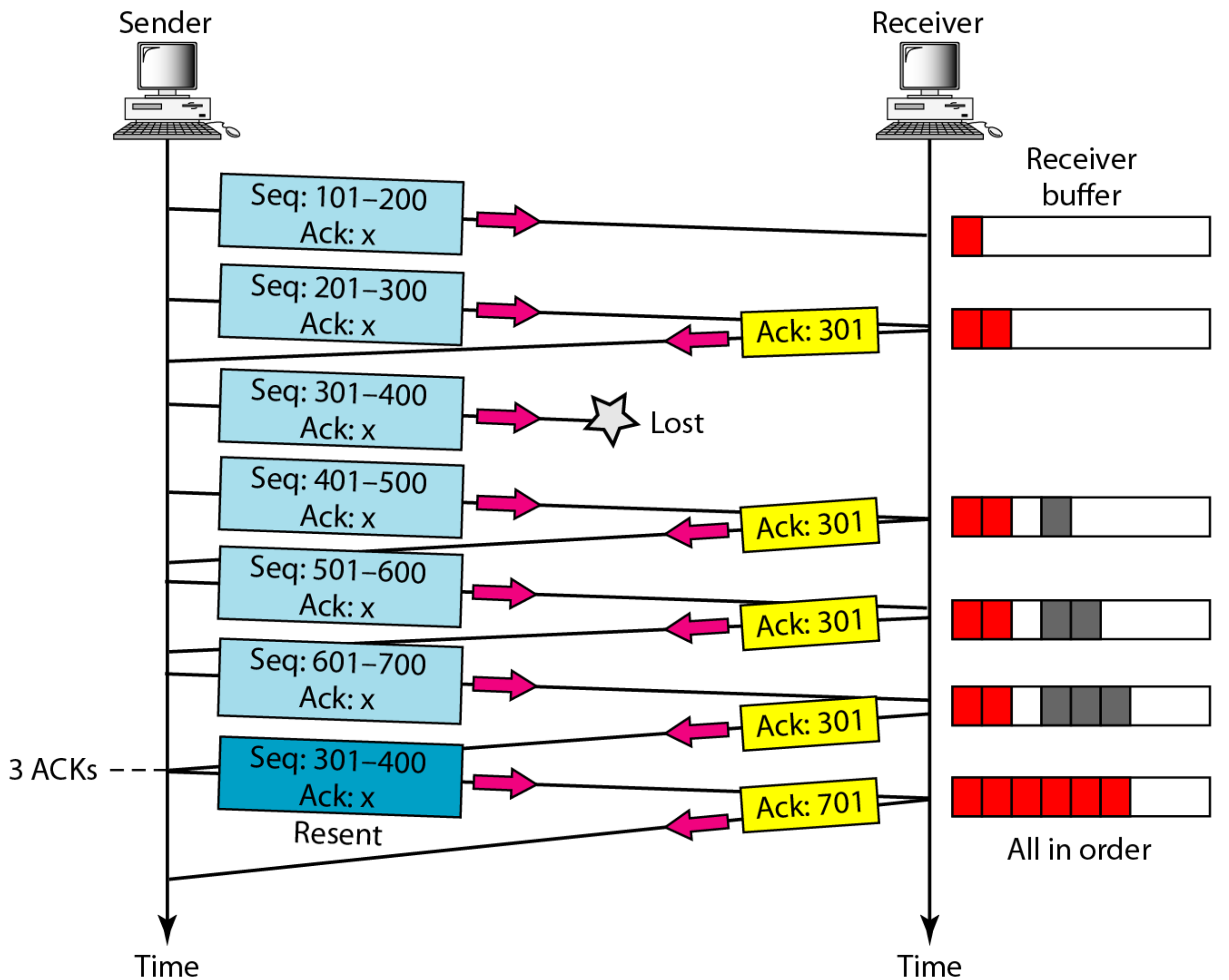
## **Figure 23.25** *Lost segment*



***Note***

**The receiver TCP delivers only ordered data to the process.**

## Figure 23.26 *Fast retransmission*



# 23-4 SCTP

***Stream Control Transmission Protocol (SCTP) is a new reliable, message-oriented transport layer protocol. SCTP, however, is mostly designed for Internet applications that have recently been introduced. These new applications need a more sophisticated service than TCP can provide.***

***Topics discussed in this section:***

**SCTP Services and Features**
**Packet Format**
**An SCTP Association**
**Flow Control and Error Control**

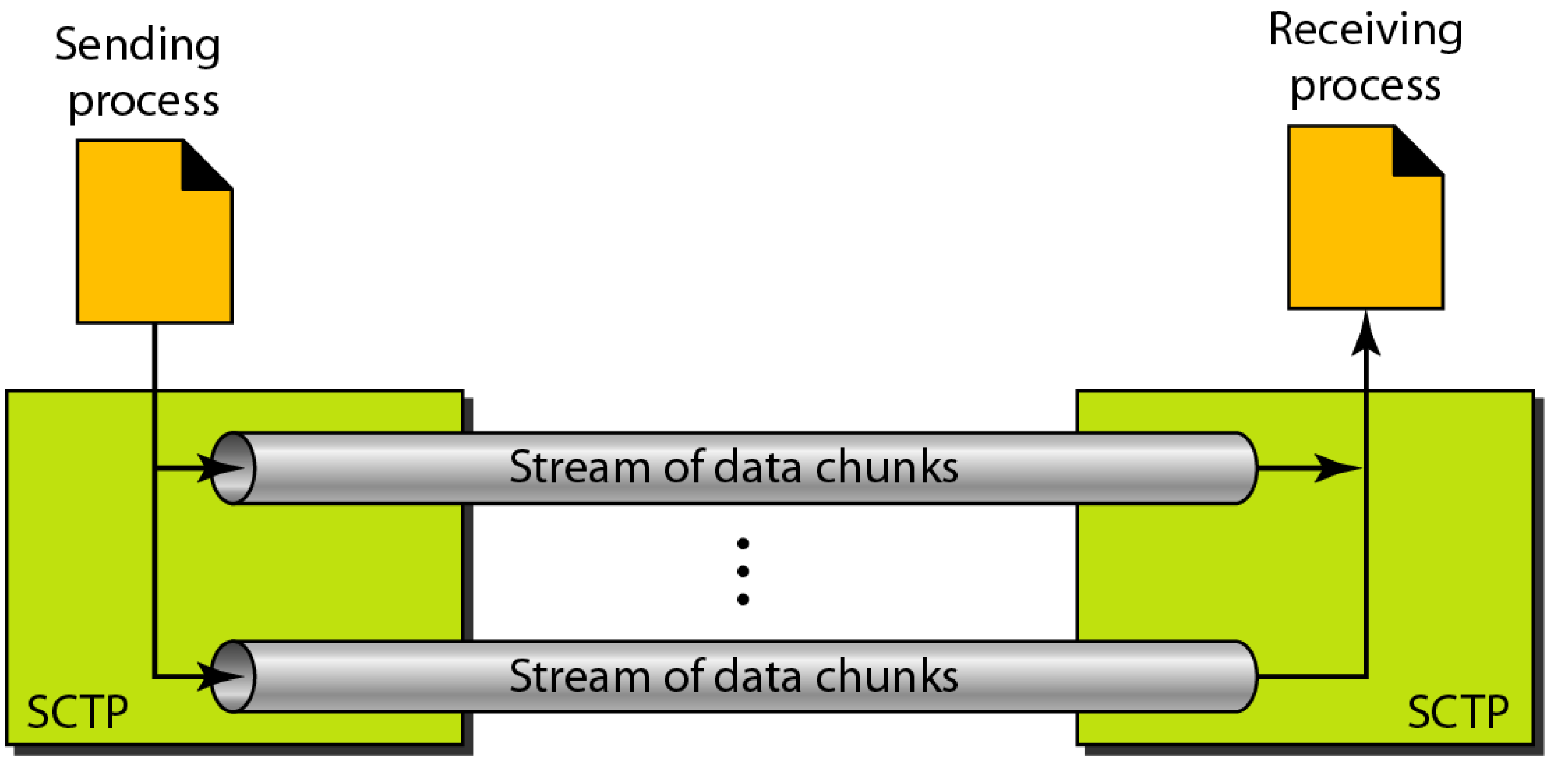
***Note***

**SCTP is a message-oriented, reliable protocol that combines the best features of UDP and TCP.**

## Table 23.4 *Some SCTP applications*

| *Protocol* | *Port Number* | *Description* |
|---|---|---|
| IUA | 9990 | ISDN over IP |
| M2UA | 2904 | SS7 telephony signaling |
| M3UA | 2905 | SS7 telephony signaling |
| H.248 | 2945 | Media gateway control |
| H.323 | 1718, 1719, 1720, 11720 | IP telephony |
| SIP | 5060 | IP telephony |

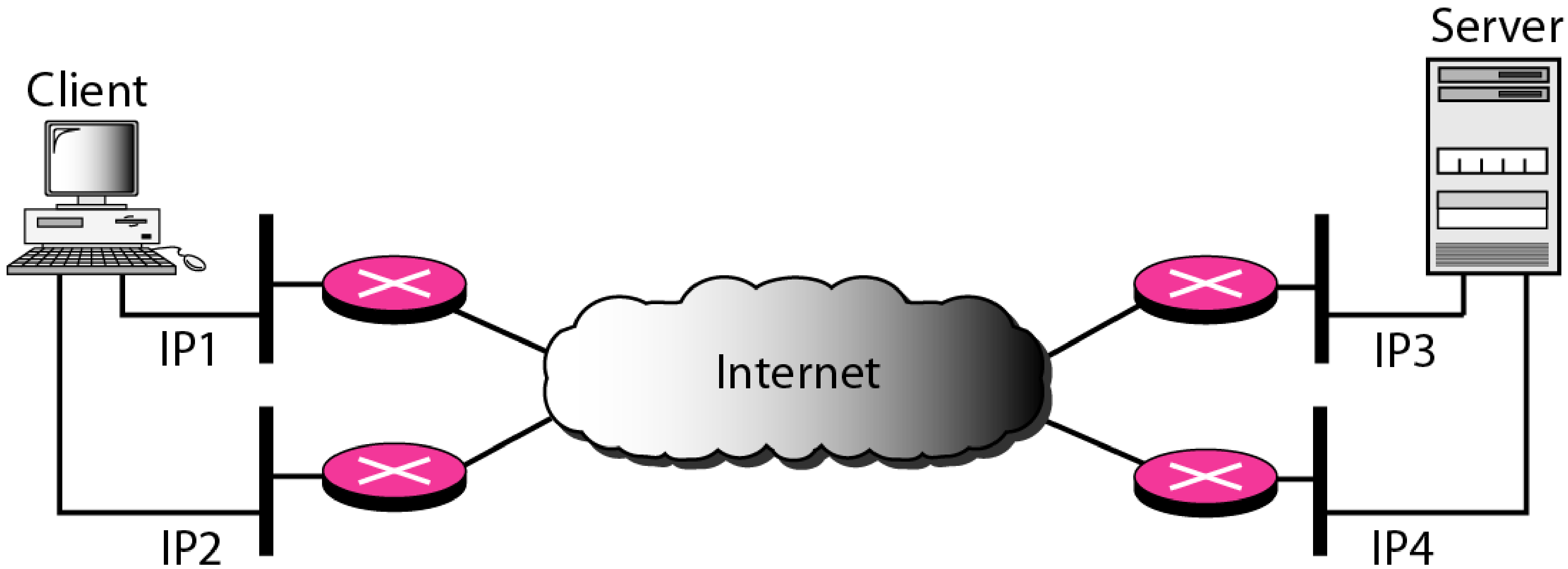
## Figure 23.27 *Multiple-stream concept*



***Note***

**An association in SCTP can involve multiple streams.**

## **Figure 23.28** ***Multihoming concept***



***Note***

**SCTP association allows multiple IP addresses for each end.**

***Note***

**In SCTP, a data chunk is numbered using a TSN.**

***Note***

**To distinguish between different streams, SCTP uses an SI.**

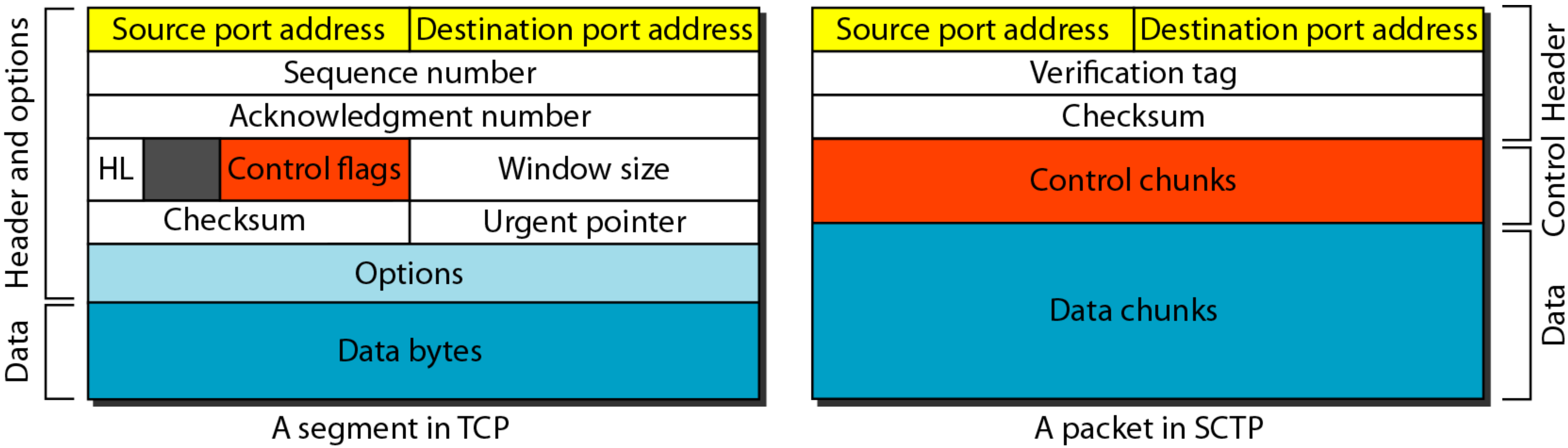
***Note***

**To distinguish between different data chunks belonging to the same stream, SCTP uses SSNs.**

***Note***

**TCP has segments; SCTP has packets.**

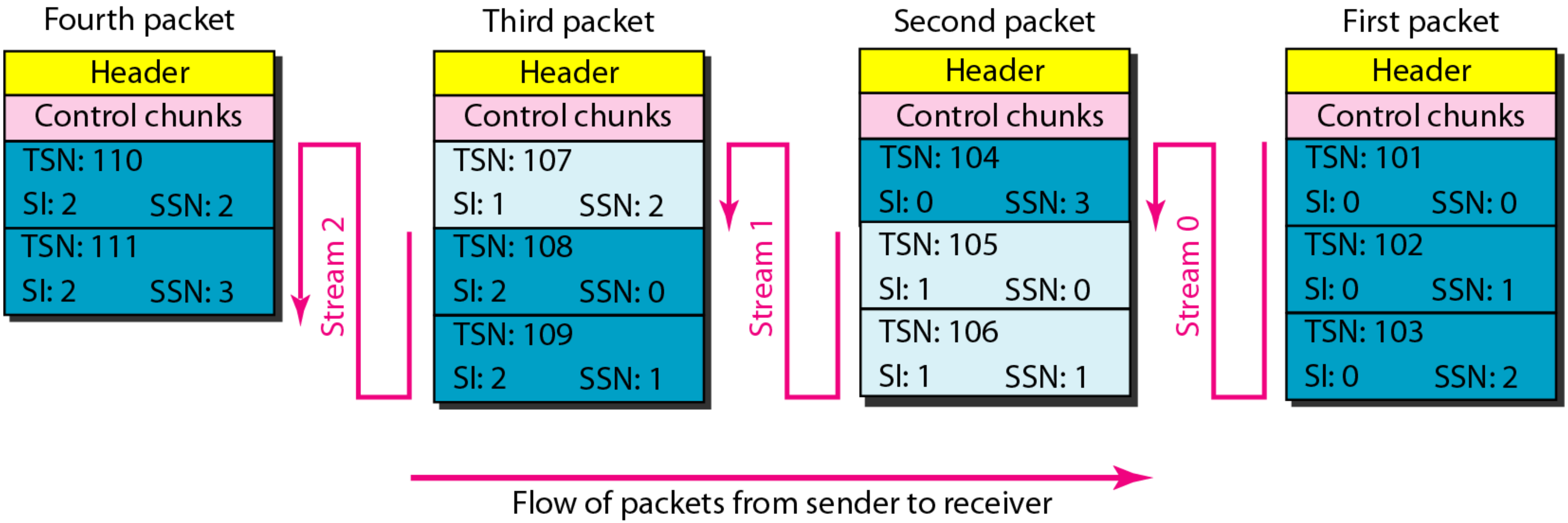
## Figure 23.29 *Comparison between a TCP segment and an SCTP packet*



***Note***

**In SCTP, control information and data information are carried in separate chunks.**

## Figure 23.30 *Packet, data chunks, and streams*



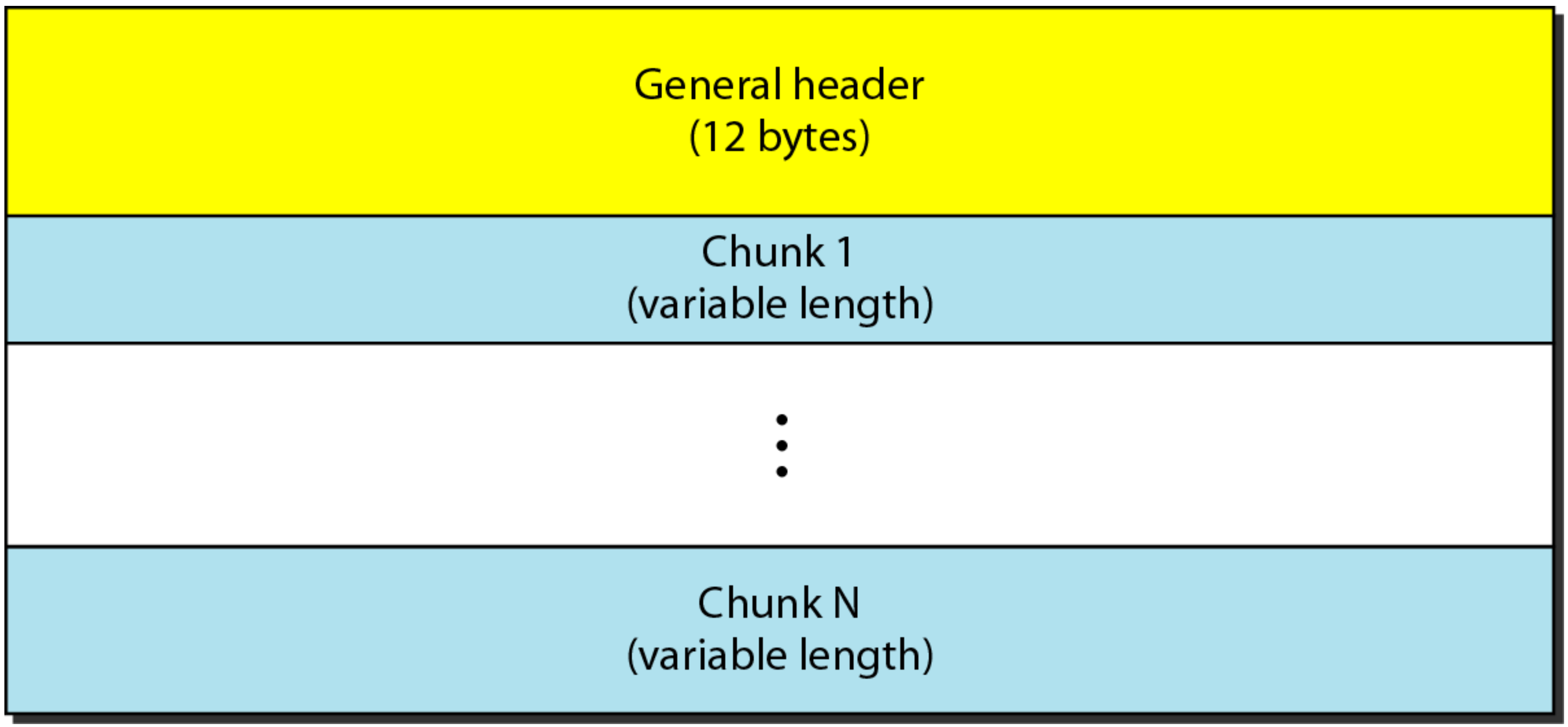
***Note***

**Data chunks are identified by three items: TSN, SI, and SSN. TSN is a cumulative number identifying the association; SI defines the stream; SSN defines the chunk in a stream.**

*Note*

**In SCTP, acknowledgment numbers are used to acknowledge only data chunks; control chunks are acknowledged by other control chunks if necessary.**

## **Figure 23.31** *SCTP packet format*



***Note***

**In an SCTP packet, control chunks come before data chunks.**

## Figure 23.32 *General header*

| | |
|---|---|
| Source port address<br>16 bits | Destination port address<br>16 bits |
| Verification tag<br>32 bits | |
| Checksum<br>32 bits | |

## Table 23.5 *Chunks*

| *Type* | *Chunk* | *Description* |
|---|---|---|
| **0** | **DATA** | User data |
| **1** | **INIT** | Sets up an association |
| **2** | **INIT ACK** | Acknowledges INIT chunk |
| **3** | **SACK** | Selective acknowledgment |
| **4** | **HEARTBEAT** | Probes the peer for liveliness |
| **5** | **HEARTBEAT ACK** | Acknowledges HEARTBEAT chunk |
| **6** | **ABORT** | Aborts an association |
| **7** | **SHUTDOWN** | Terminates an association |
| **8** | **SHUTDOWN ACK** | Acknowledges SHUTDOWN chunk |
| **9** | **ERROR** | Reports errors without shutting down |
| **10** | **COOKIE ECHO** | Third packet in association establishment |
| **11** | **COOKIE ACK** | Acknowledges COOKIE ECHO chunk |
| **14** | **SHUTDOWN COMPLETE** | Third packet in association termination |
| **192** | **FORWARD TSN** | For adjusting cumulative TSN |

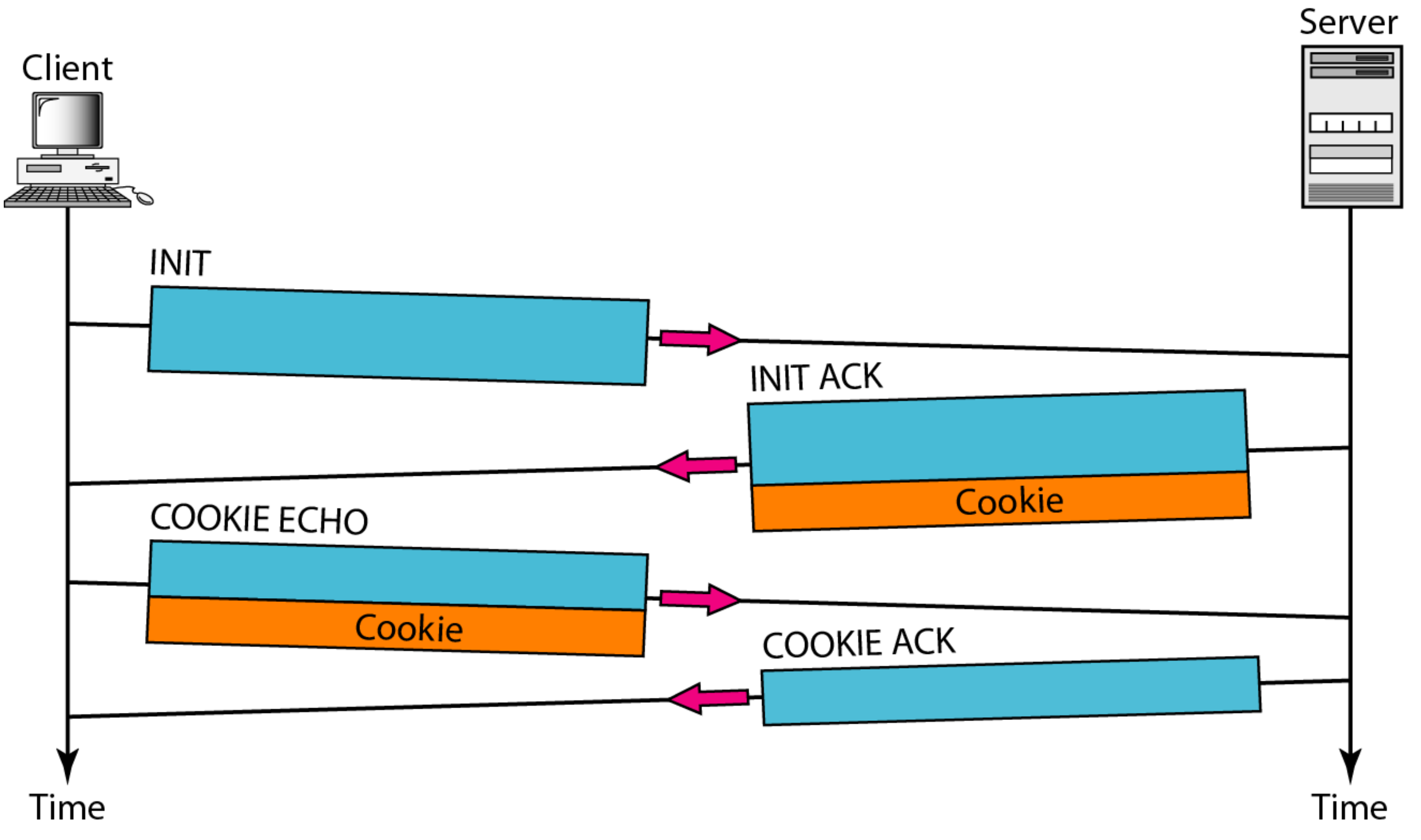
***Note***

**A connection in SCTP is called an association.**

***Note***

**No other chunk is allowed in a packet carrying an INIT or INIT ACK chunk. A COOKIE ECHO or a COOKIE ACK chunk can carry data chunks.**

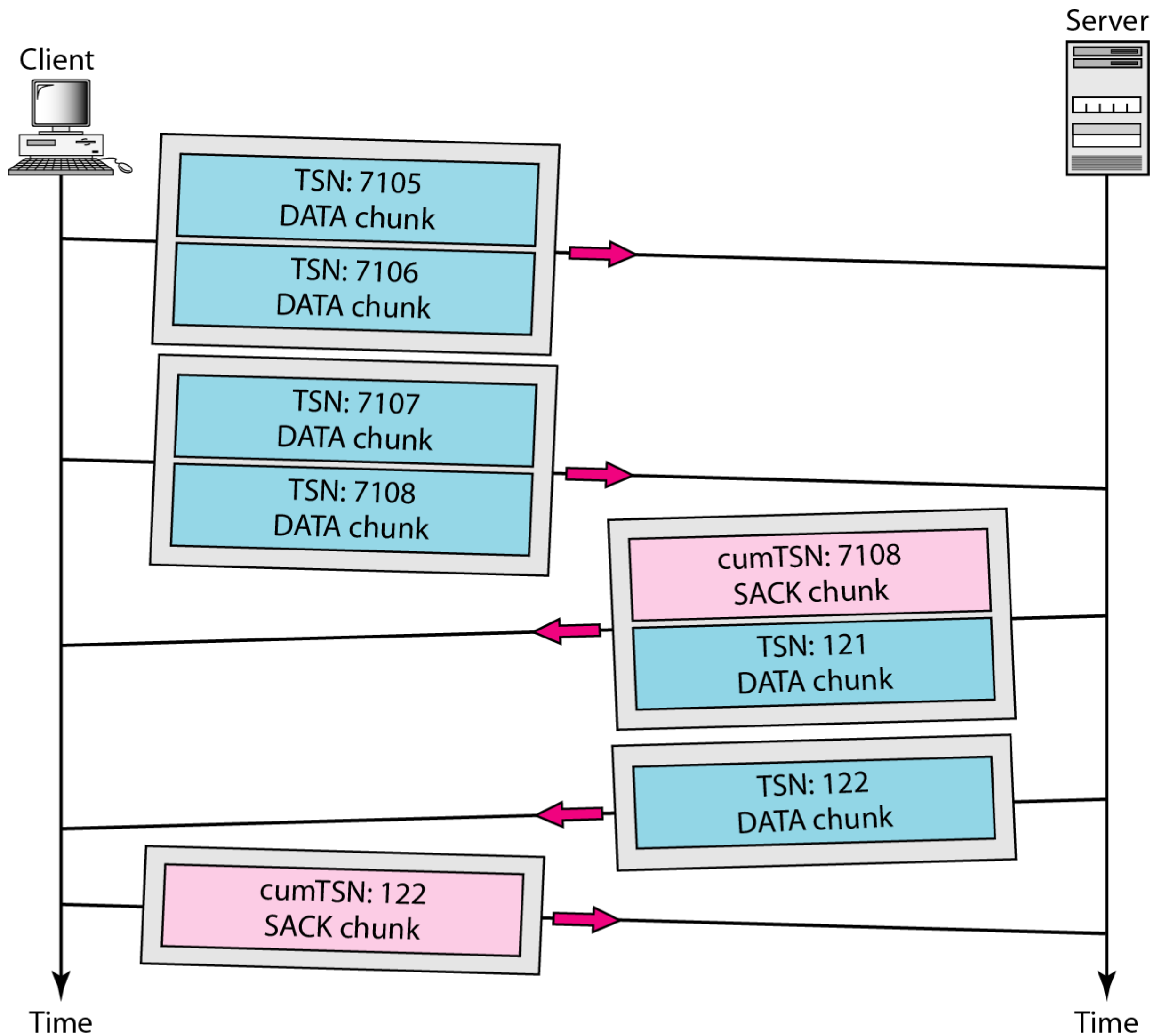
## Figure 23.33 *Four-way handshaking*



***Note***

**In SCTP, only DATA chunks consume TSNs;**
**DATA chunks are the only chunks that are acknowledged.**

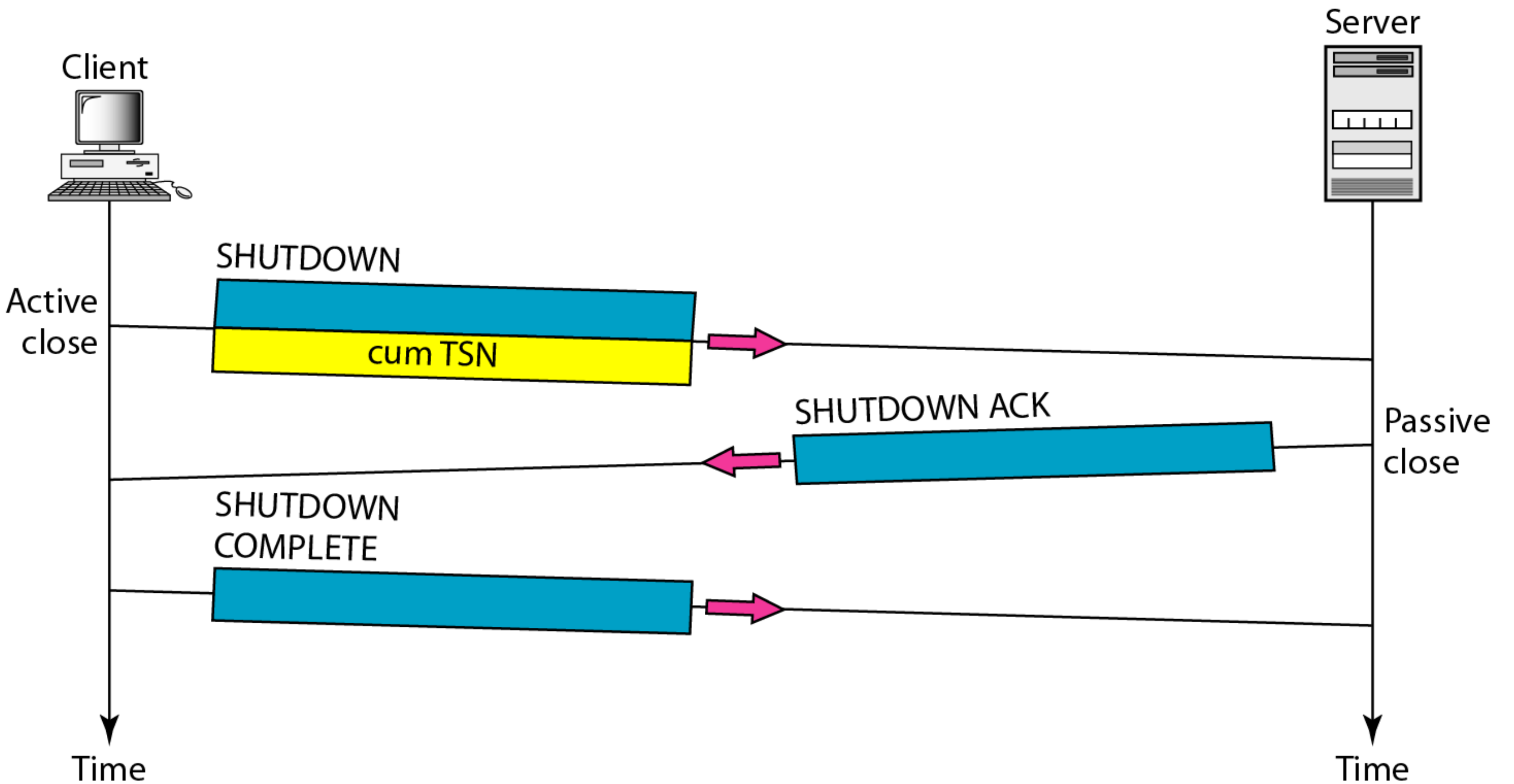
## **Figure 23.34** *Simple data transfer*



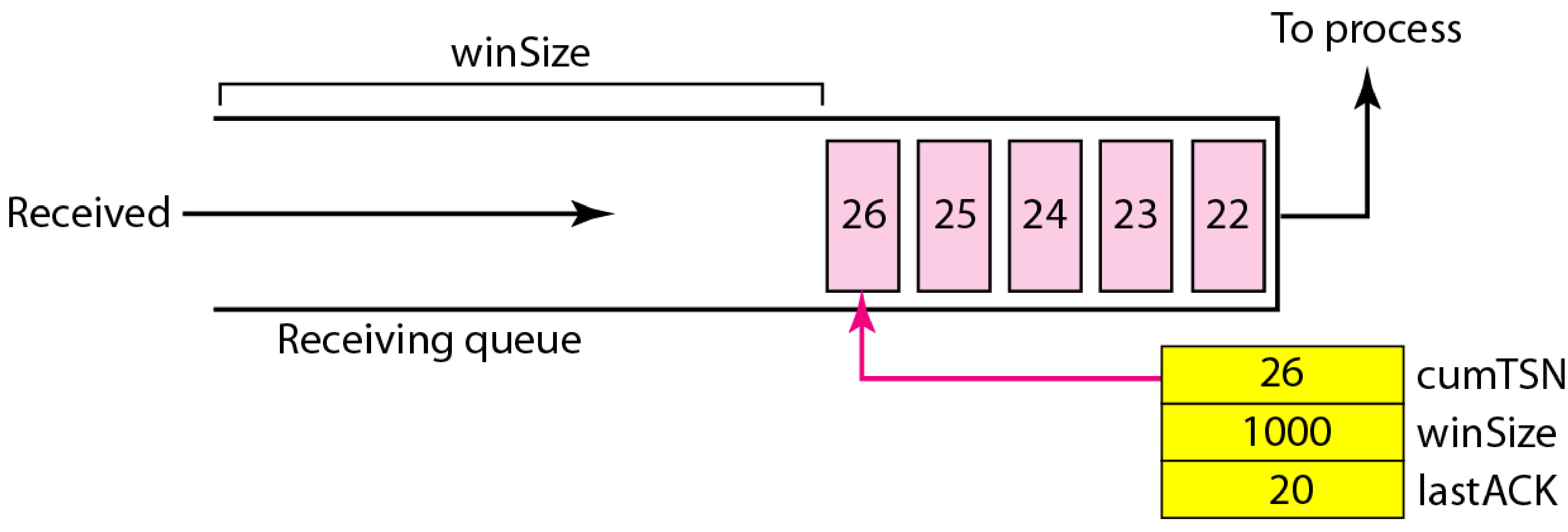
***Note***

**The acknowledgment in SCTP defines the cumulative TSN, the TSN of the last data chunk received in order.**

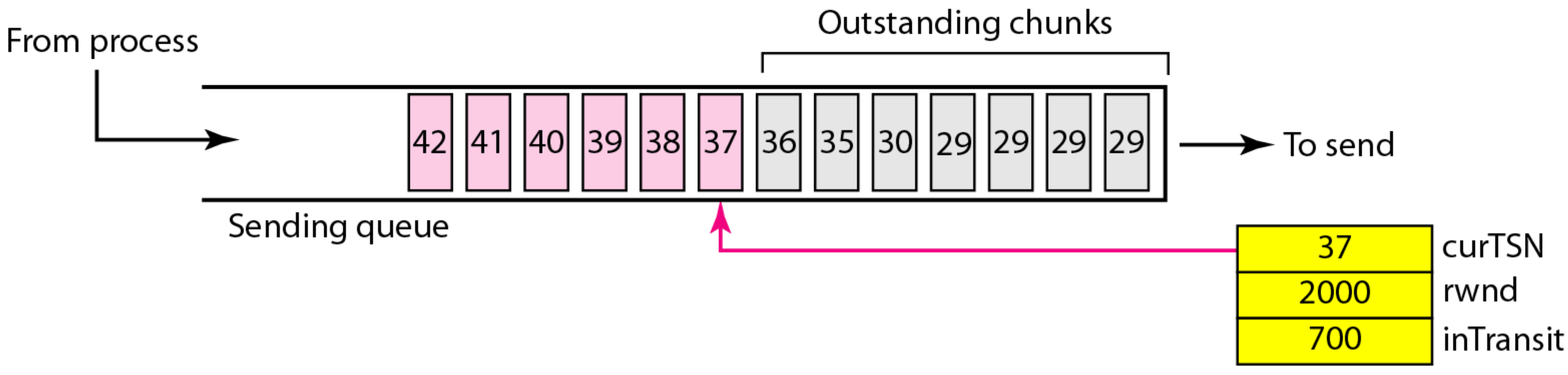
## **Figure 23.35** *Association termination*



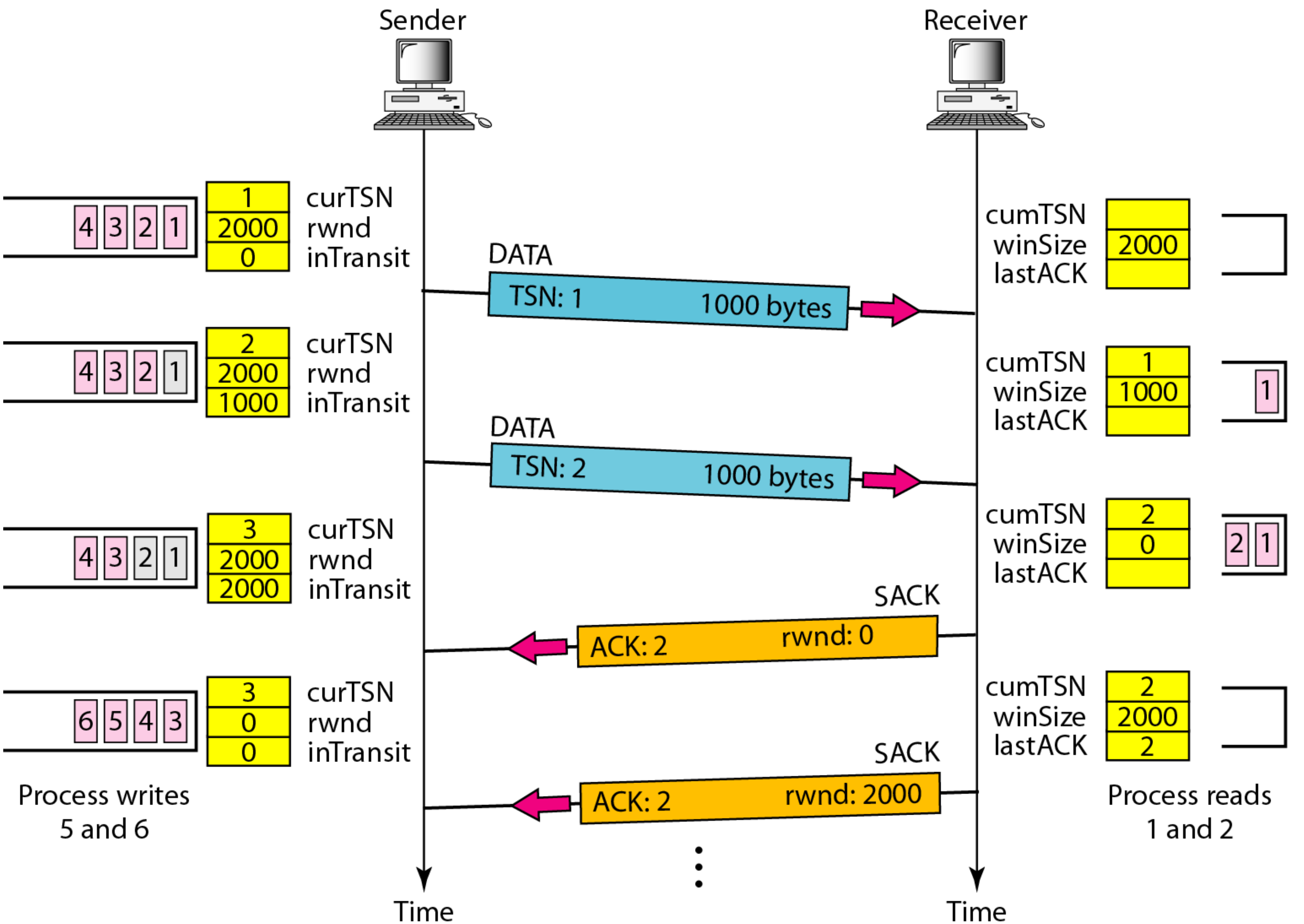
## **Figure 23.36** *Flow control, receiver site*



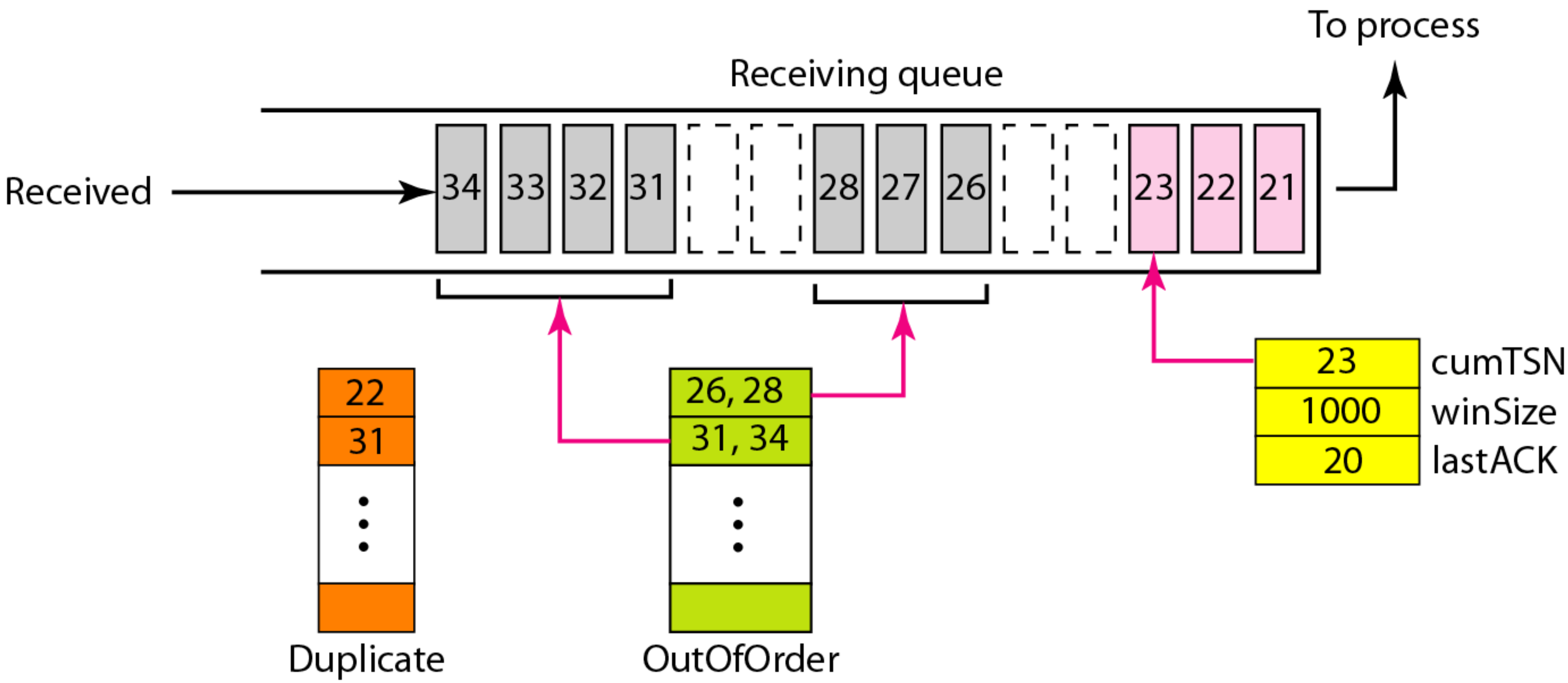
## Figure 23.37 *Flow control, sender site*



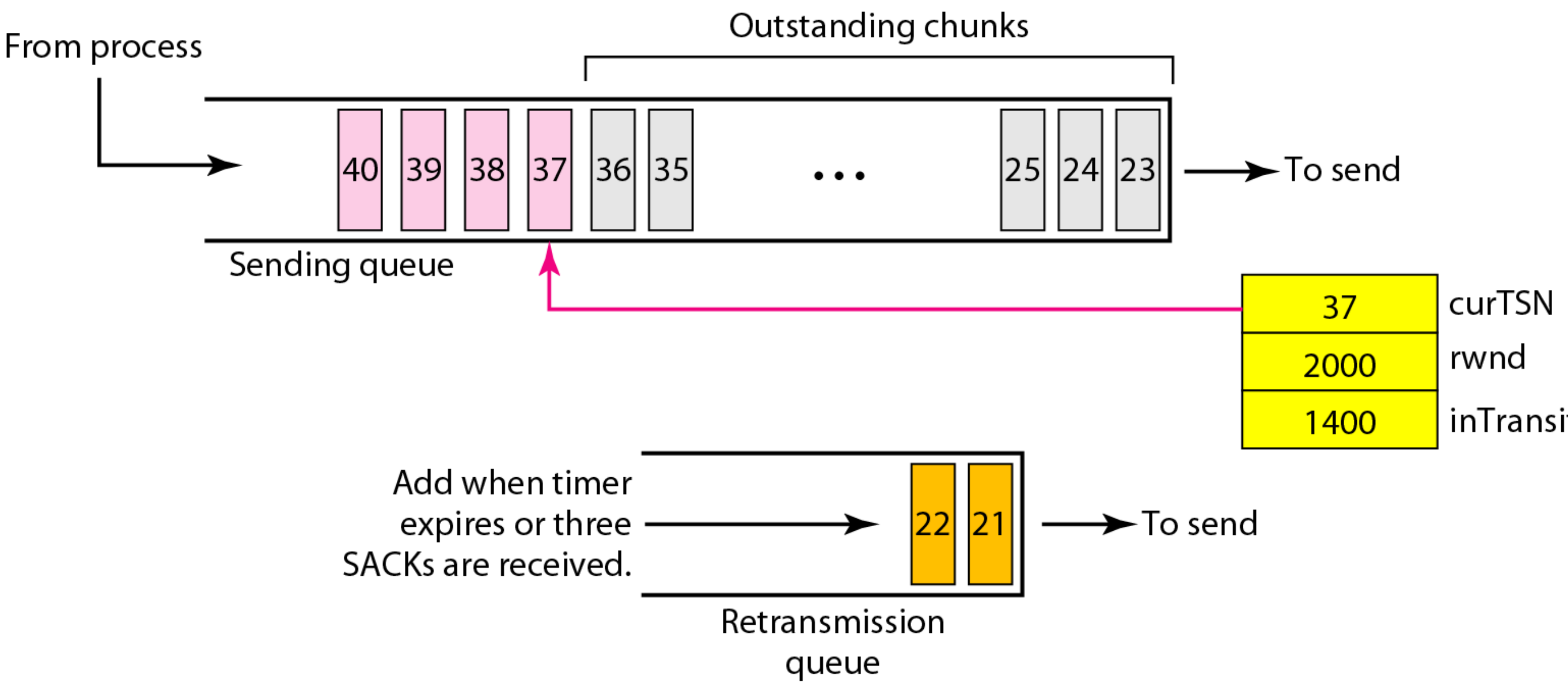
## Figure 23.38 *Flow control scenario*



## Figure 23.39 *Error control, receiver site*



## **Figure 23.40** *Error control, sender site*





# Chapter 24

# Congestion Control and Quality of Service

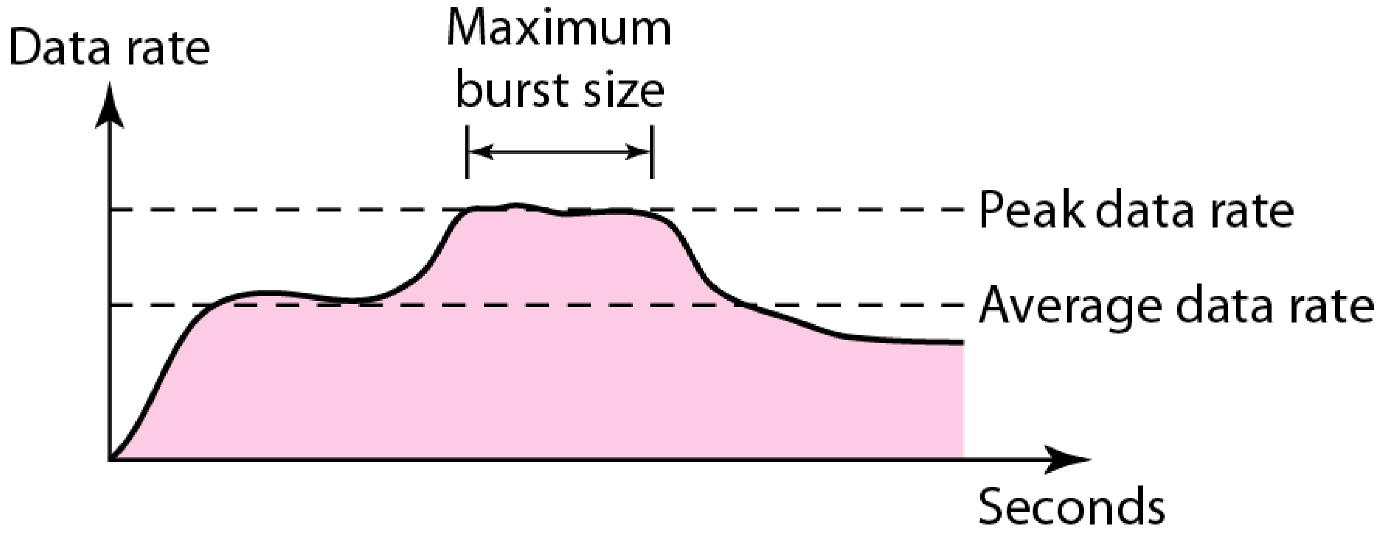
# 24-1 DATA TRAFFIC

***The main focus of congestion control and quality of service is data traffic. In congestion control we try to avoid traffic congestion. In quality of service, we try to create an appropriate environment for the traffic. So, before talking about congestion control and quality of service, we discuss the data traffic itself.***

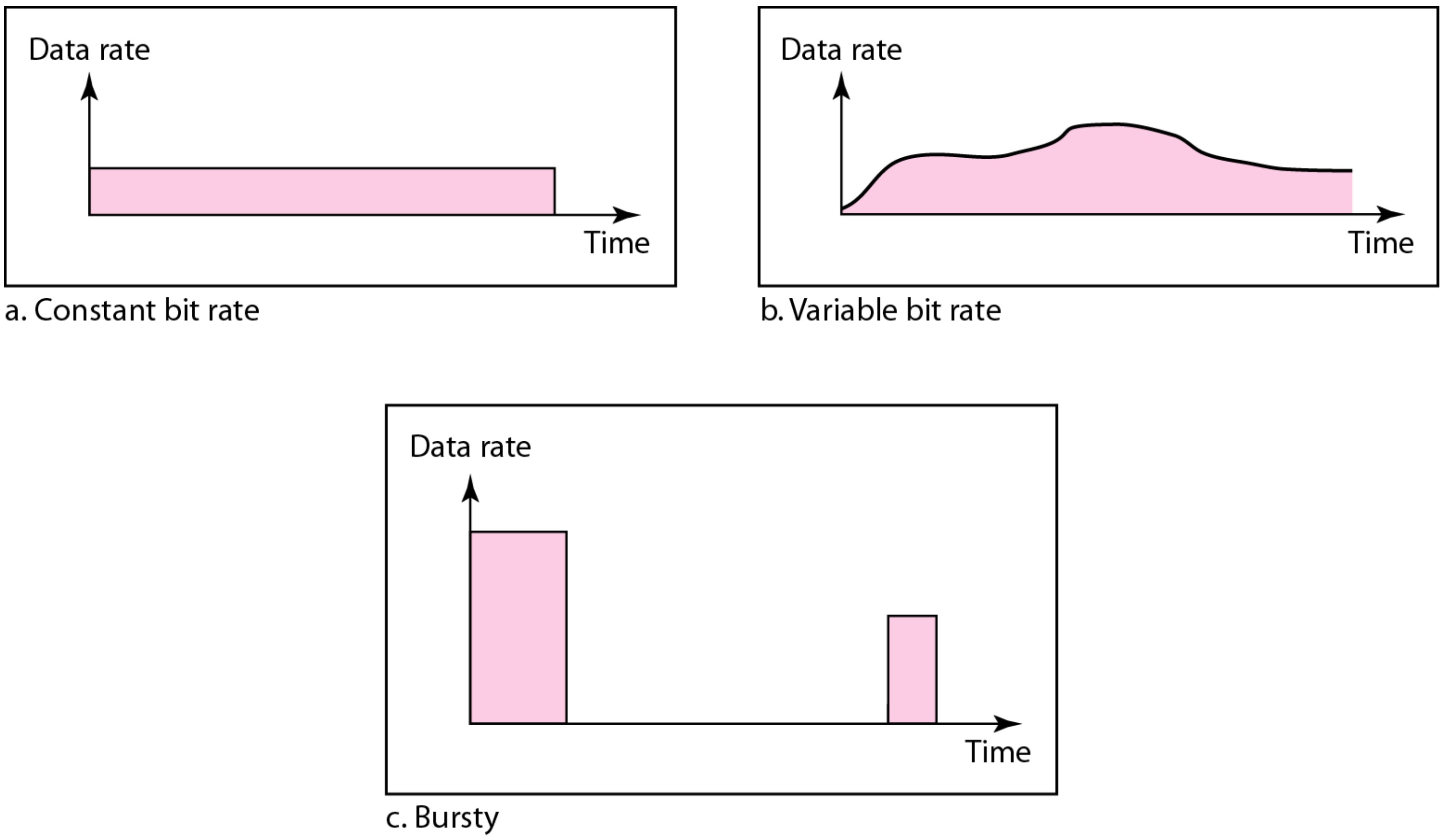
***Topics discussed in this section:***

**Traffic Descriptor**
**Traffic Profiles**

## **Figure 24.1** *Traffic descriptors*



## **Figure 24.2** *Three traffic profiles*



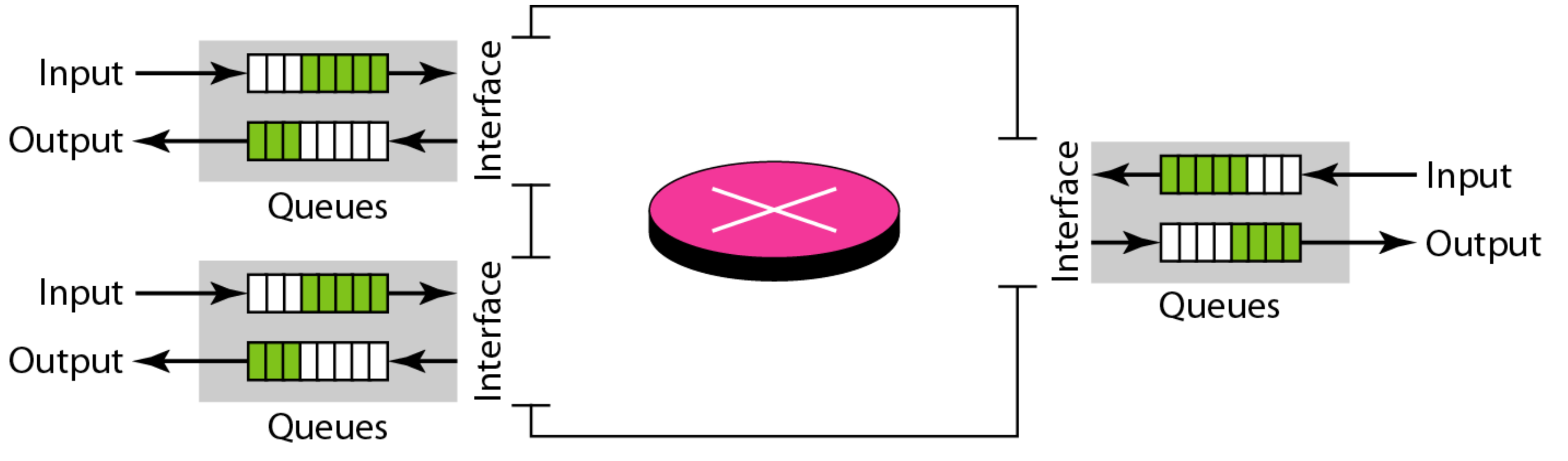
# 24-2 CONGESTION

***Congestion in a network may occur if the load on the network—the number of packets sent to the network—is greater than the capacity of the network—the number of packets a network can handle. Congestion control refers to the mechanisms and techniques to control the congestion and keep the load below the capacity.***

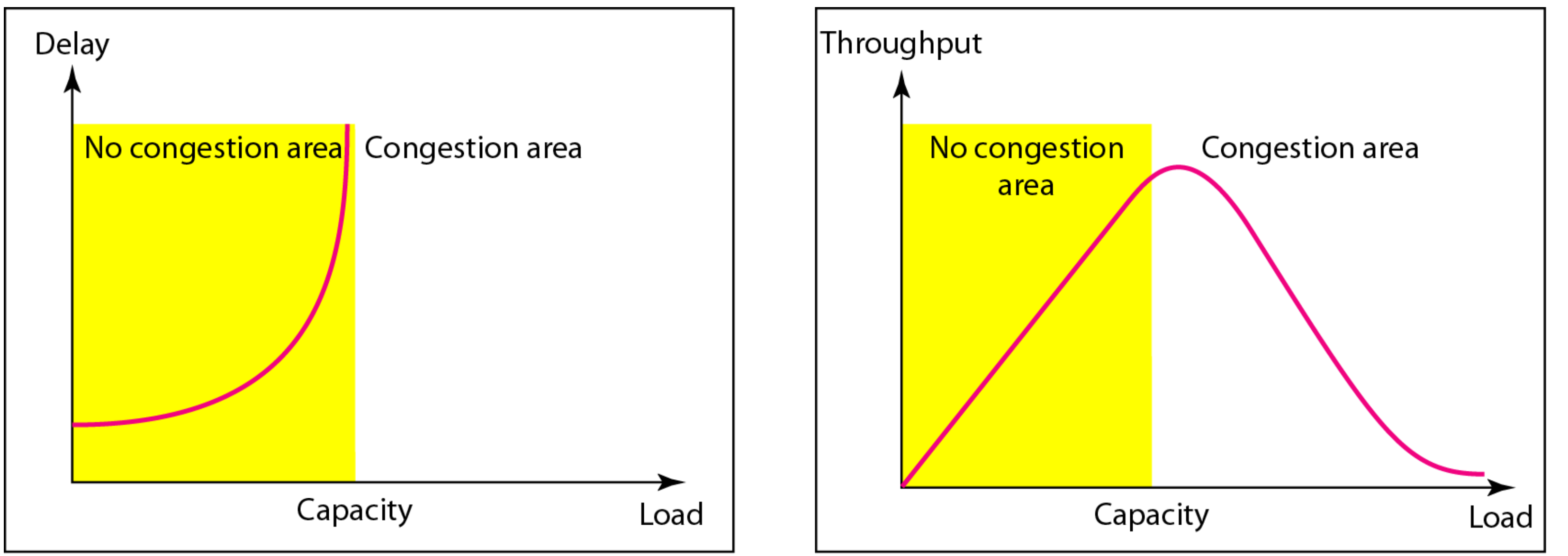
***Topics discussed in this section:***

**Network Performance**

# **Figure 24.3** ***Queues in a router***



# **Figure** *Packet delay and throughput as functions of load*




a. Delay as a function of load

b. Throughput as a function of load

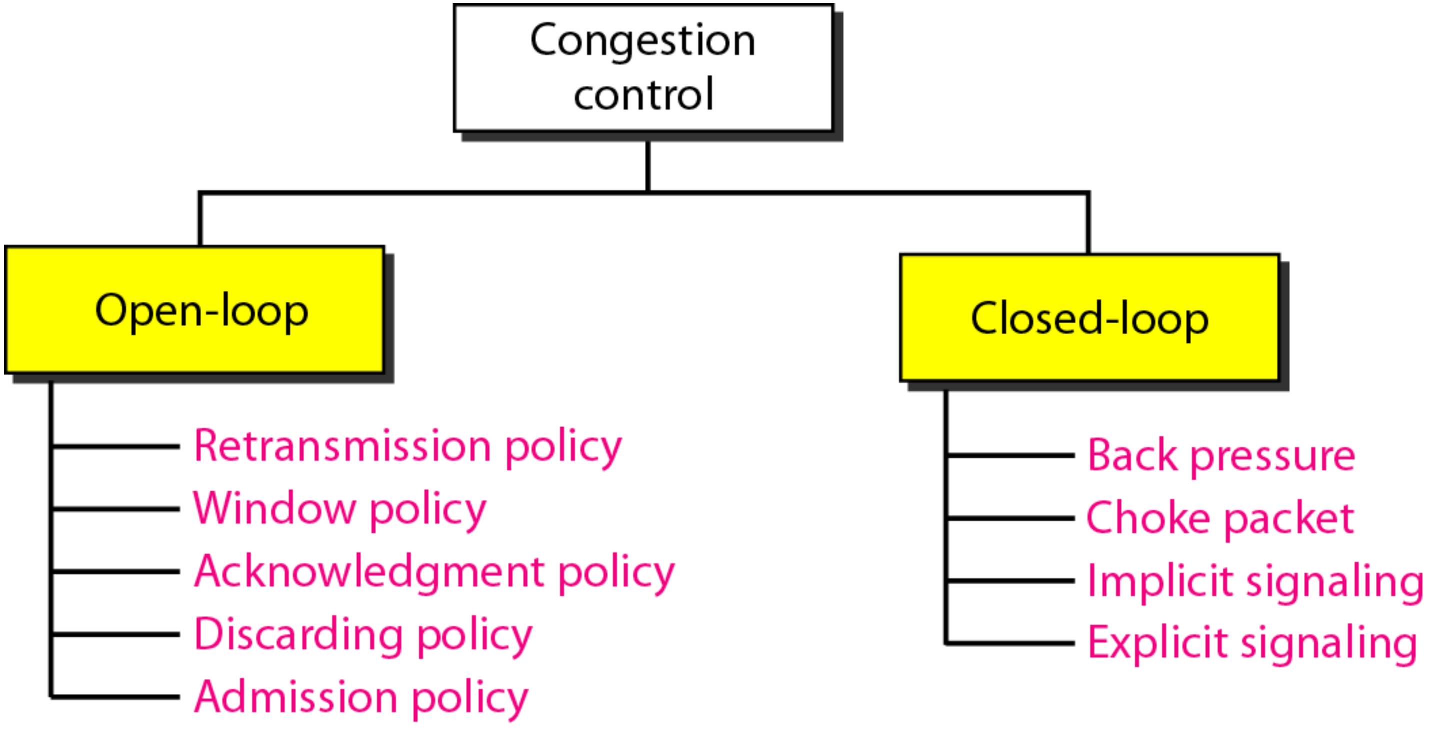
# 24-3 CONGESTION CONTROL

***Congestion control refers to techniques and mechanisms that can either prevent congestion, before it happens, or remove congestion, after it has happened. In general, we can divide congestion control mechanisms into two broad categories: open-loop congestion control (prevention) and closed-loop congestion control (removal).***

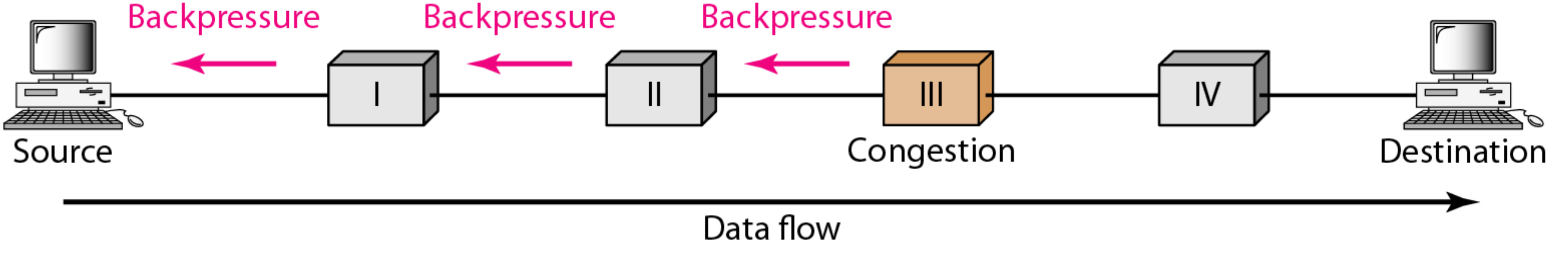
***Topics discussed in this section:***

**Open-Loop Congestion Control**
**Closed-Loop Congestion Control**

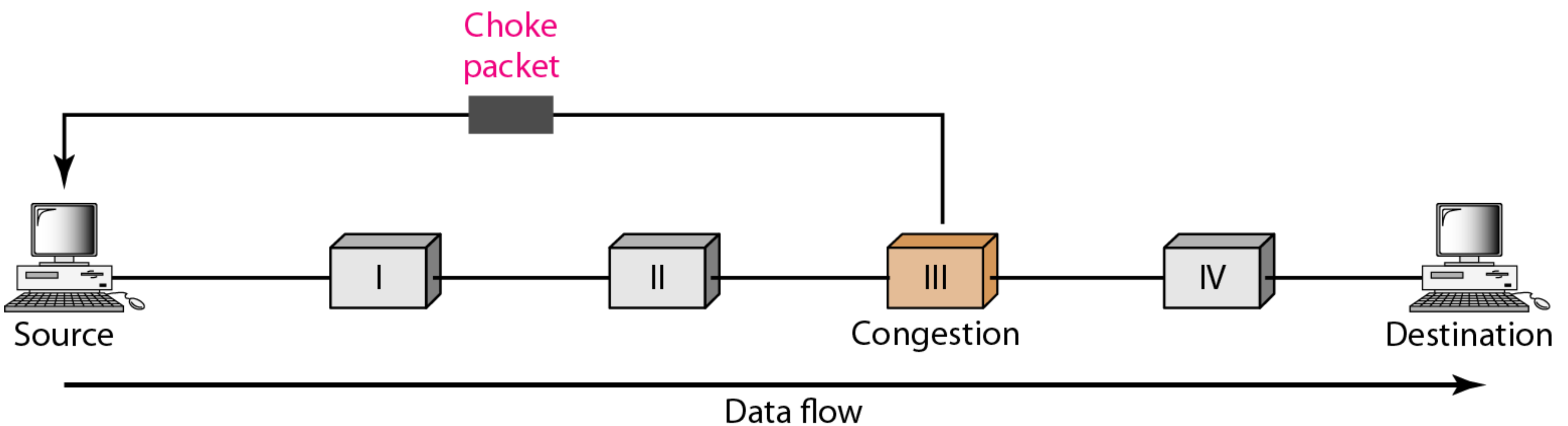
## **Figure 24.5** *Congestion control categories*



## **Figure 24.6** ***Backpressure method for alleviating congestion***



## Figure 24.7 *Choke packet*



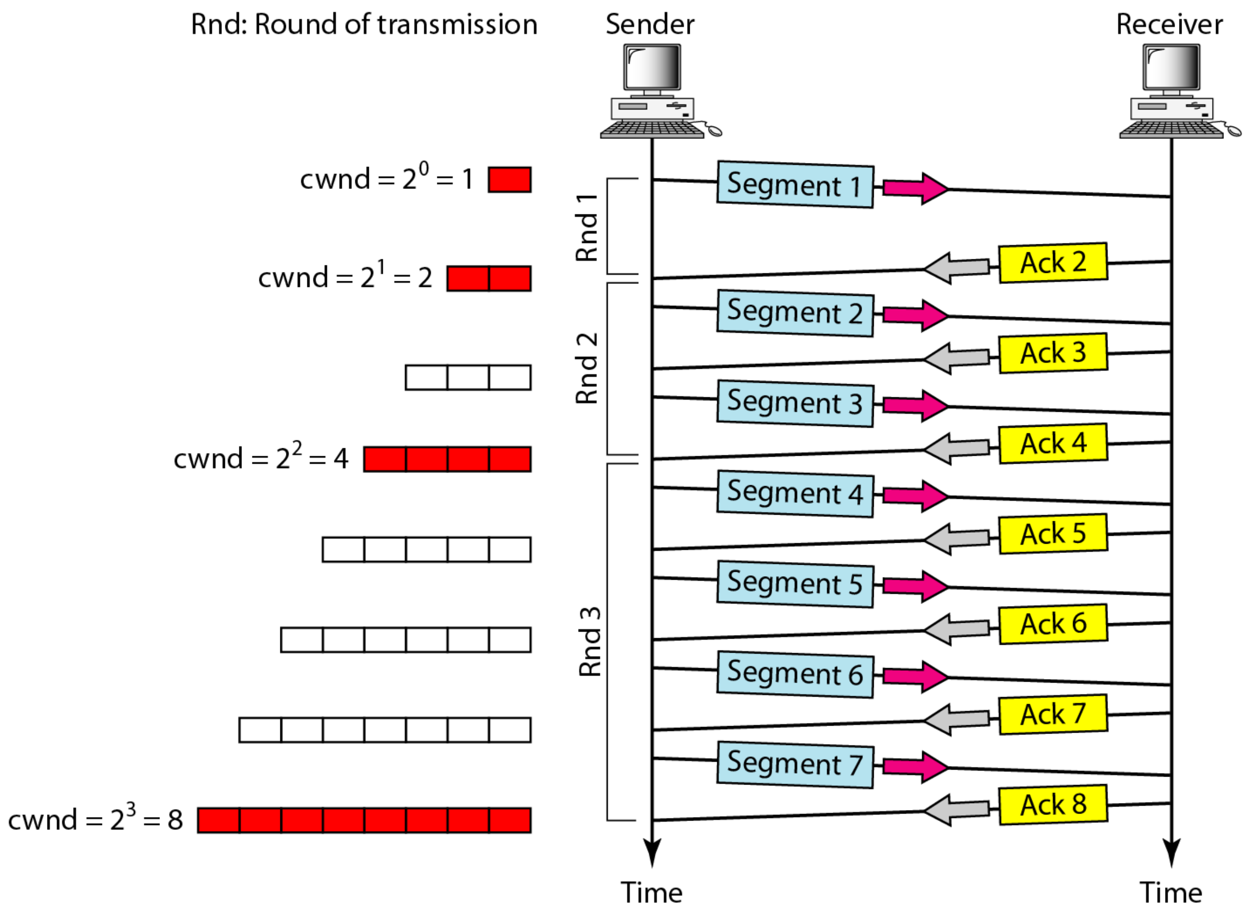
# 24-4 TWO EXAMPLES

***To better understand the concept of congestion control, let us give two examples: one in TCP and the other in Frame Relay.***

***Topics discussed in this section:***

**Congestion Control in TCP**
**Congestion Control in Frame Relay**

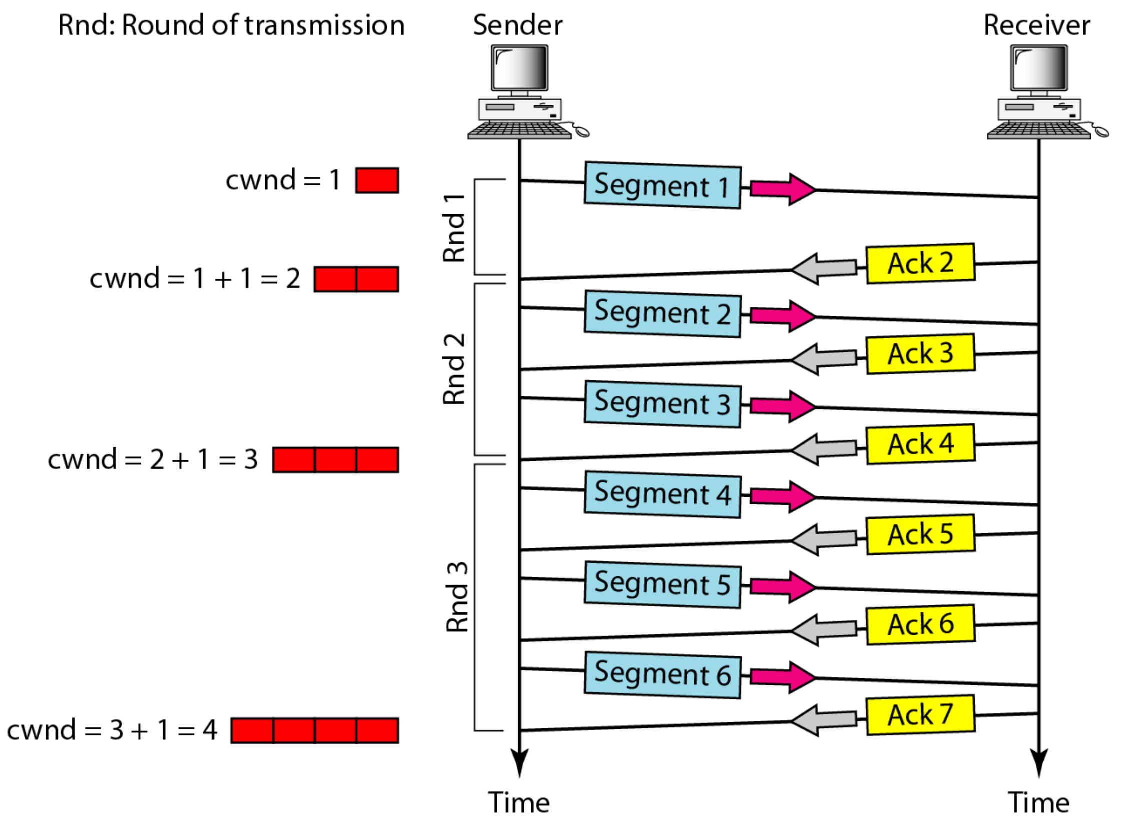
## **Figure 24.8** *Slow start, exponential increase*



***Note***

**In the slow-start algorithm, the size of the congestion window increases exponentially until it reaches a threshold.**

## Figure 24.9 *Congestion avoidance, additive increase*



***Note***

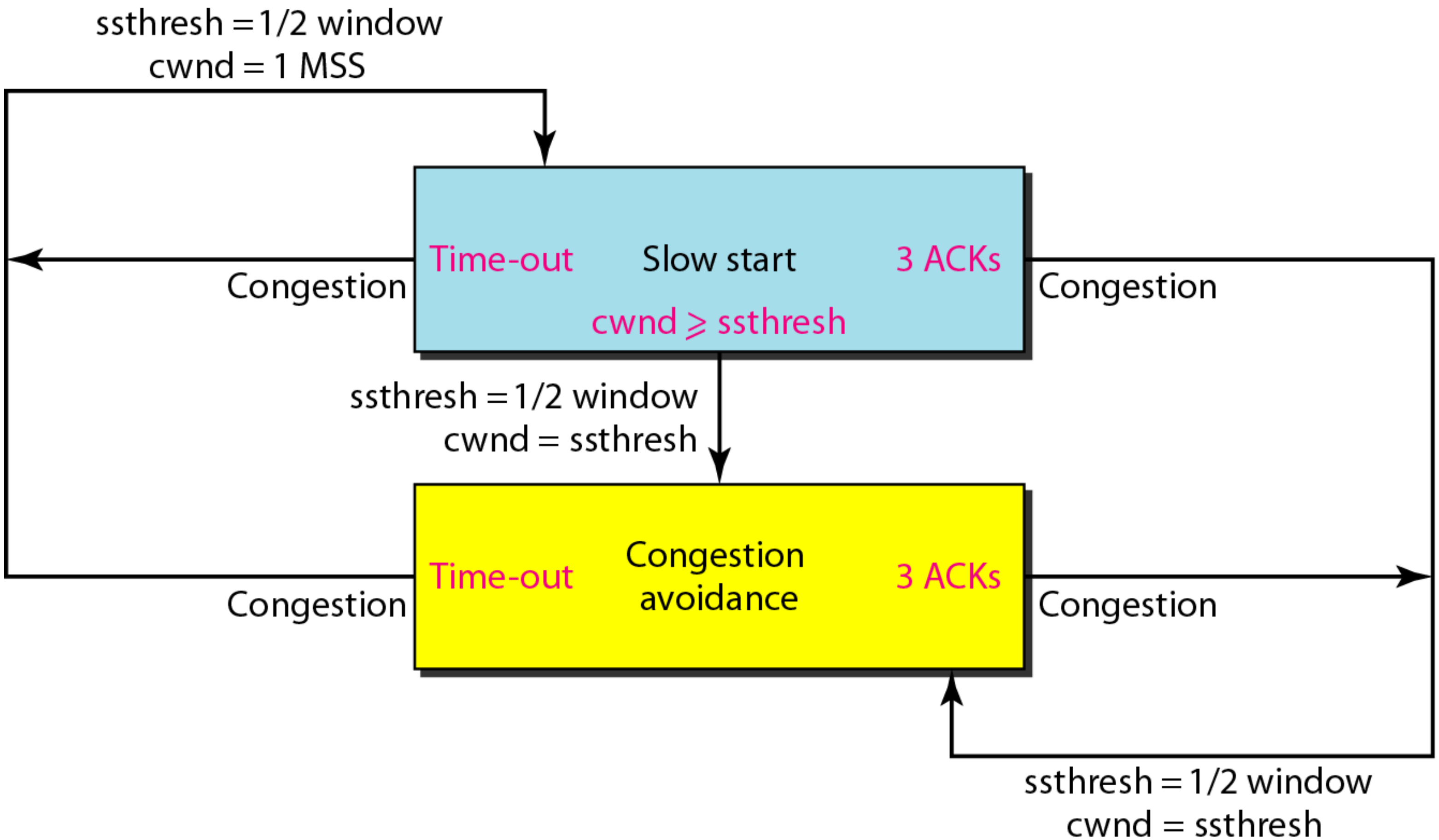
**In the congestion avoidance algorithm, the size of the congestion window increases additively until congestion is detected.**

***Note***

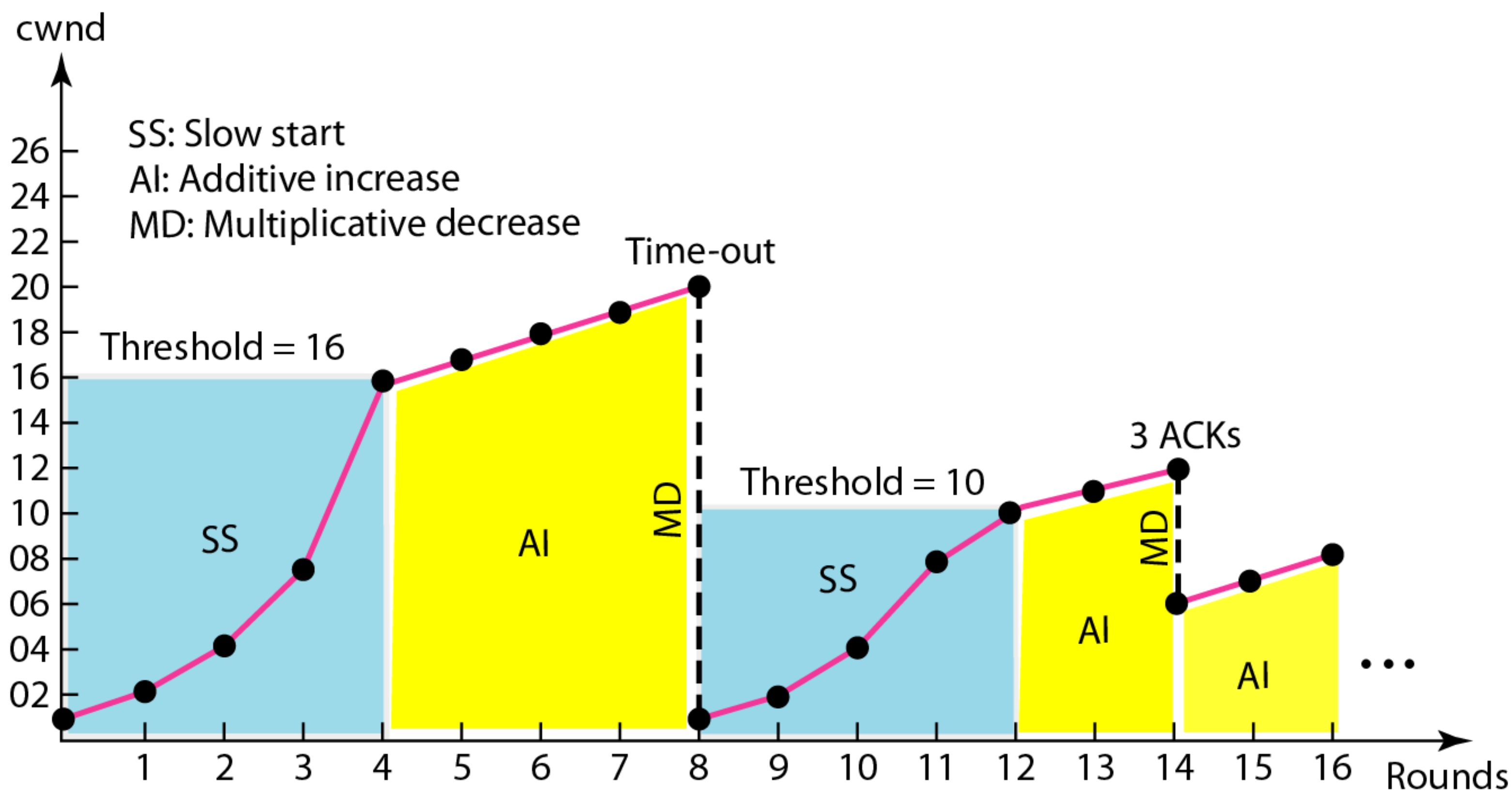
**An implementation reacts to congestion detection in one of the following ways:**

- **If detection is by time-out, a new slow start phase starts.**
- **If detection is by three ACKs, a new congestion avoidance phase starts.**

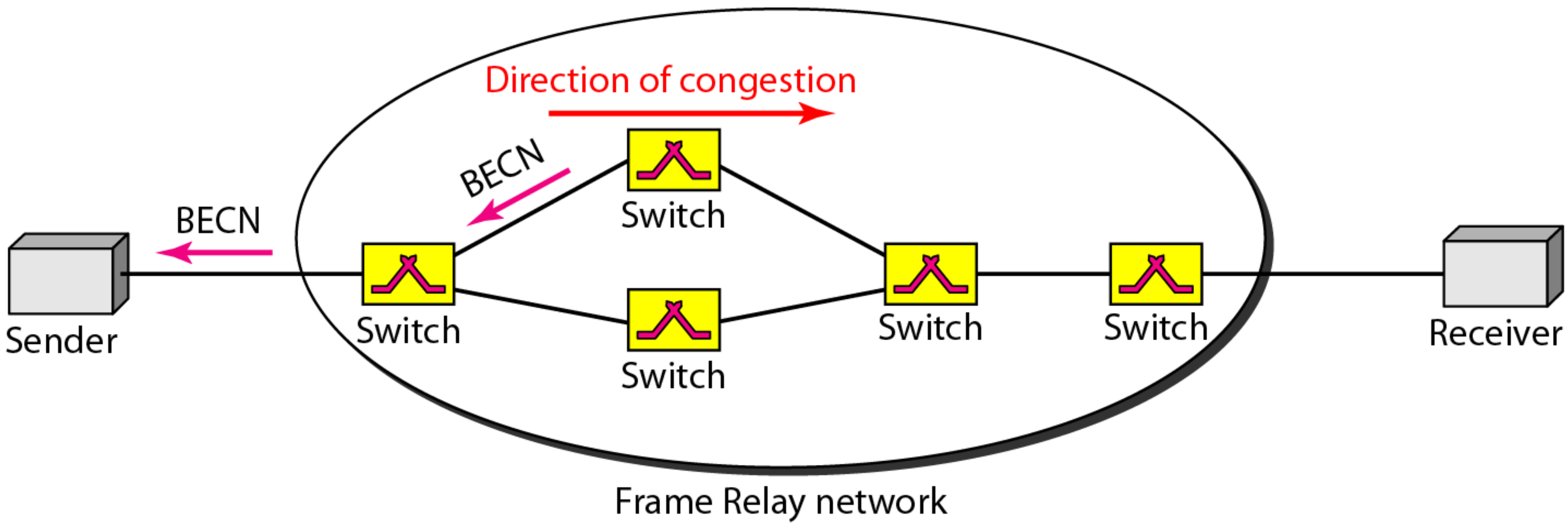
# Figure 24.10 *TCP congestion policy summary*



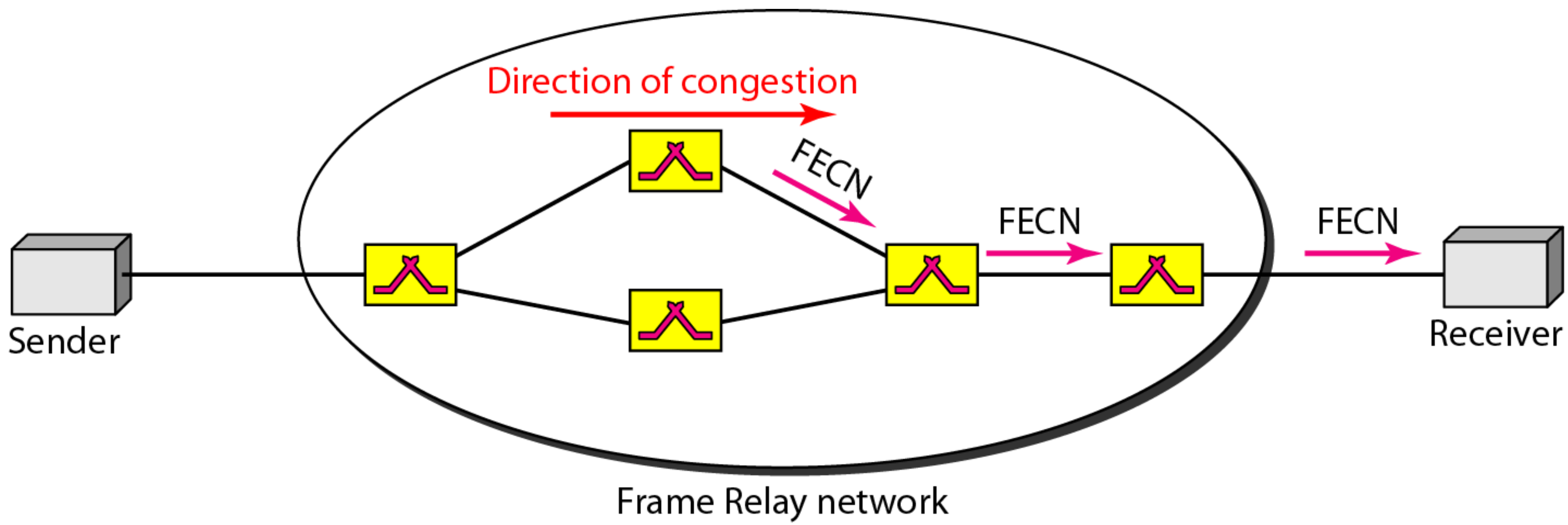
## Figure 24.11 *Congestion example*



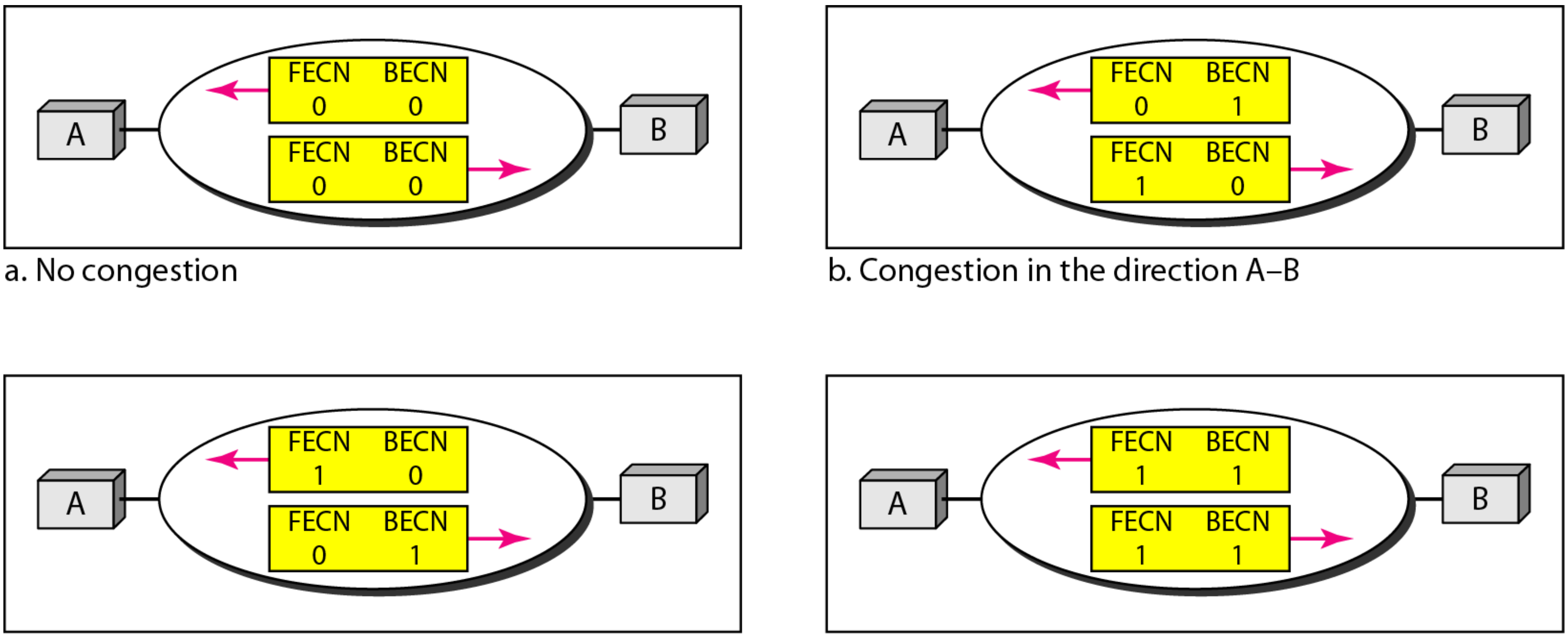
## Figure 24.12 *BECN*



# Figure 24.13 *FECN*



## **Figure 24.14** *Four cases of congestion*




a. No congestion

b. Congestion in the direction A–B

c. Congestion in the direction B–A

d. Congestion in both directions

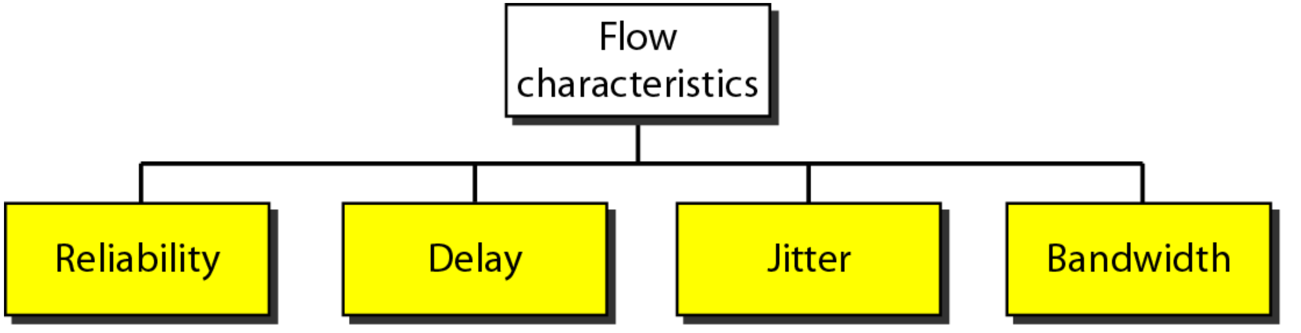
# 24-5 QUALITY OF SERVICE

***Quality of service (QoS) is an internetworking issue that has been discussed more than defined. We can informally define quality of service as something a flow seeks to attain.***

***Topics discussed in this section:***

**Flow Characteristics**
**Flow Classes**

## Figure 24.15 *Flow characteristics*



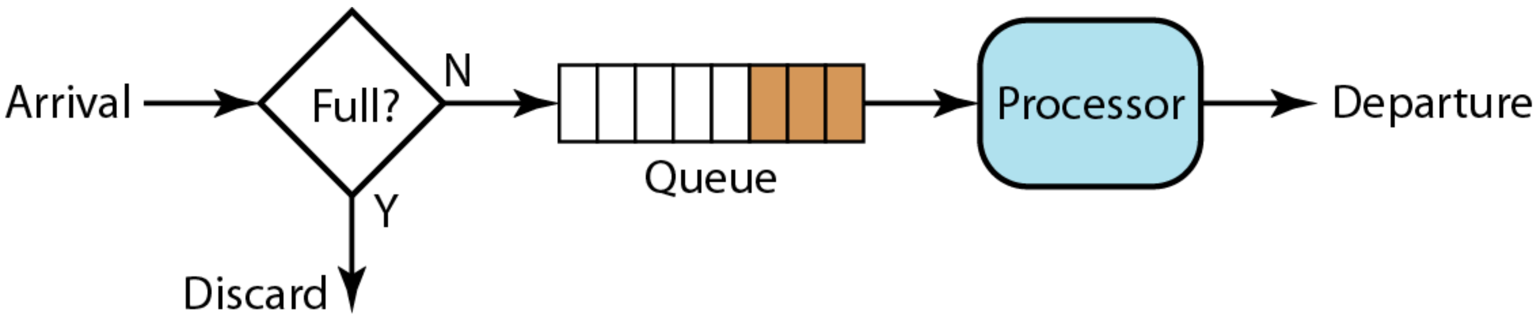
# 24-6 TECHNIQUES TO IMPROVE QoS

***In Section 24.5 we tried to define QoS in terms of its characteristics. In this section, we discuss some techniques that can be used to improve the quality of service. We briefly discuss four common methods: scheduling, traffic shaping, admission control, and resource reservation.***

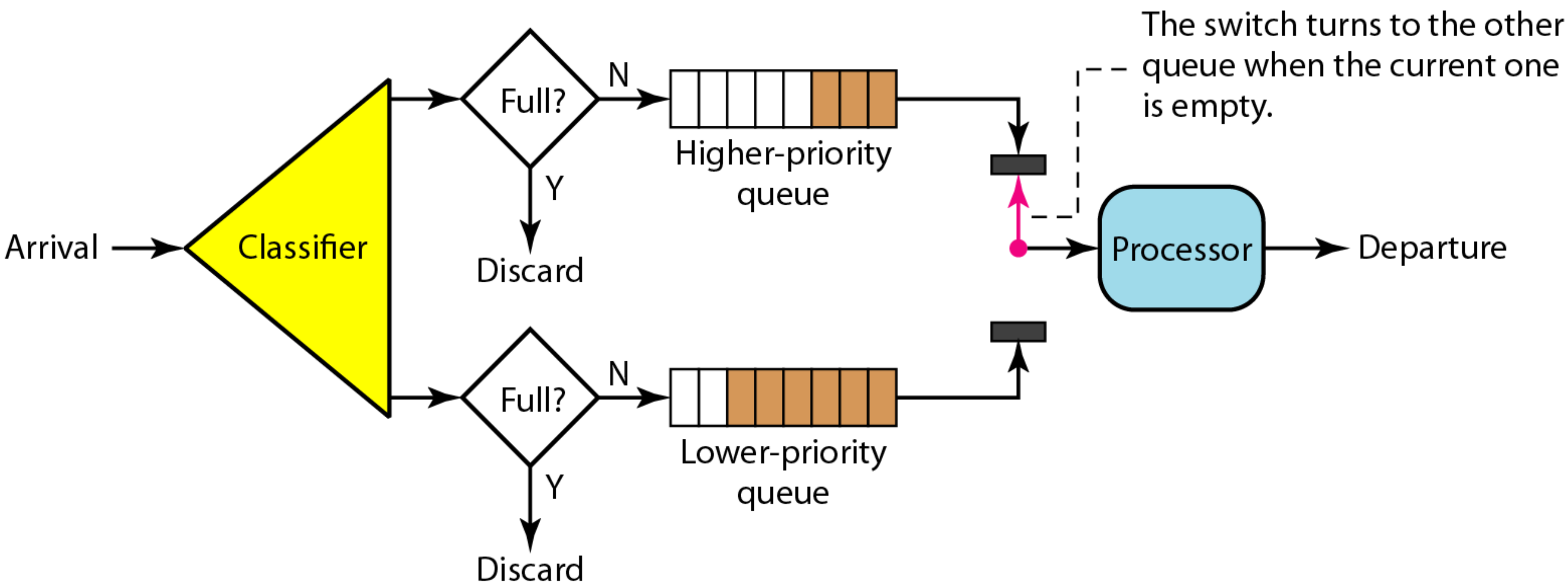
***Topics discussed in this section:***

**Scheduling**
**Traffic Shaping**
**Resource Reservation**
**Admission Control**

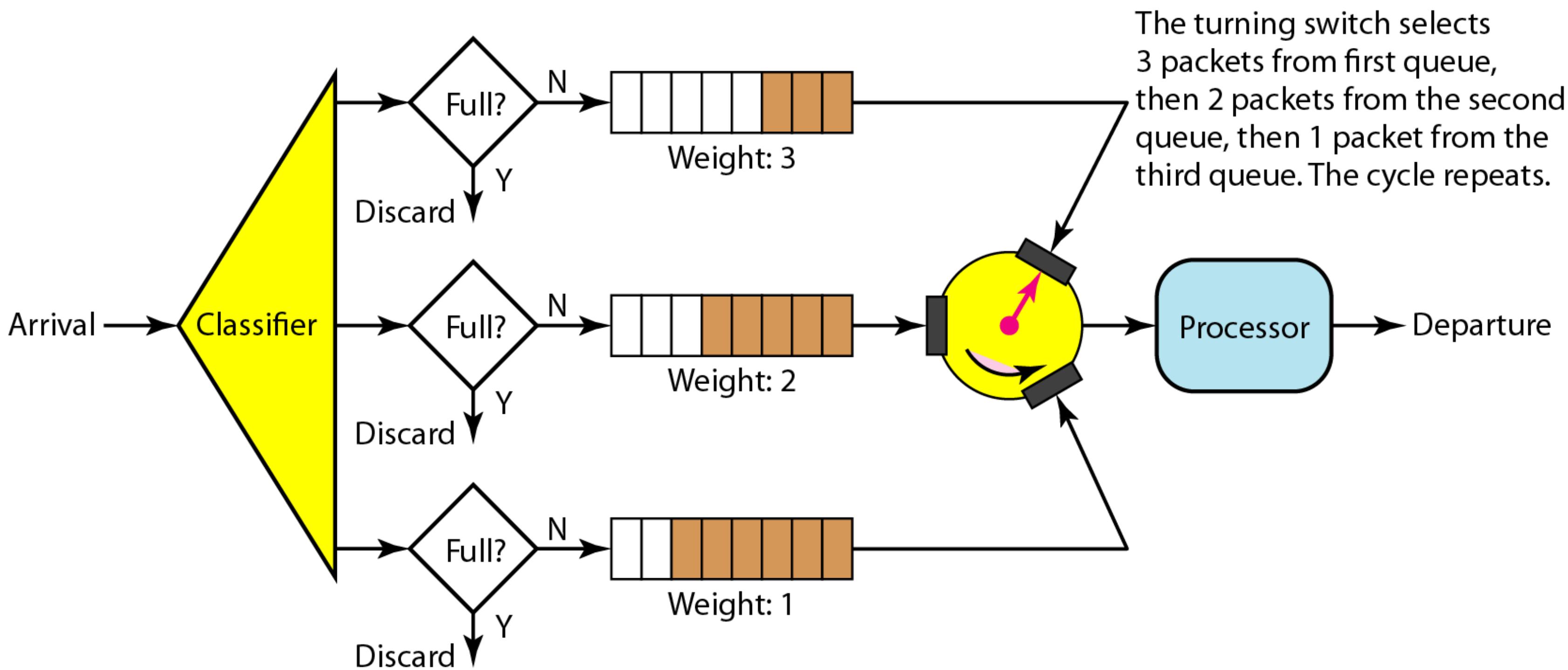
## Figure 24.16 *FIFO queue*



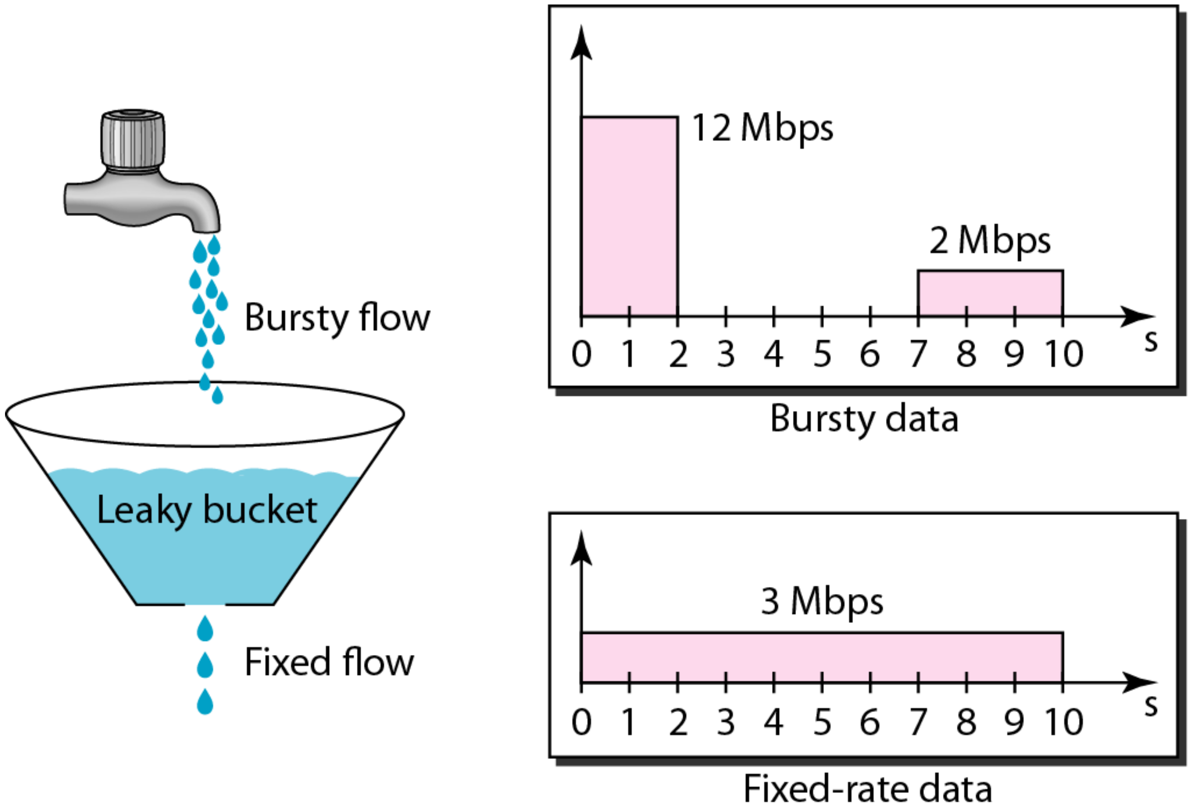
## Figure 24.17 *Priority queuing*



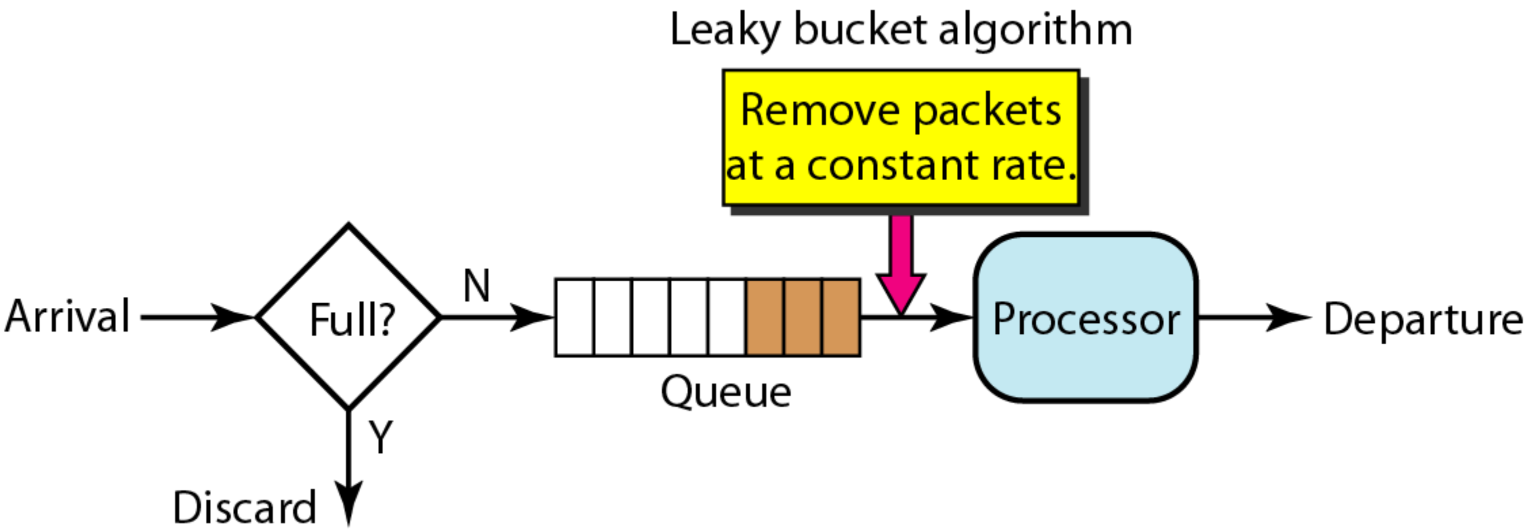
## Figure 24.18 *Weighted fair queuing*



## Figure 24.19 *Leaky bucket*



## Figure 24.20 *Leaky bucket implementation*



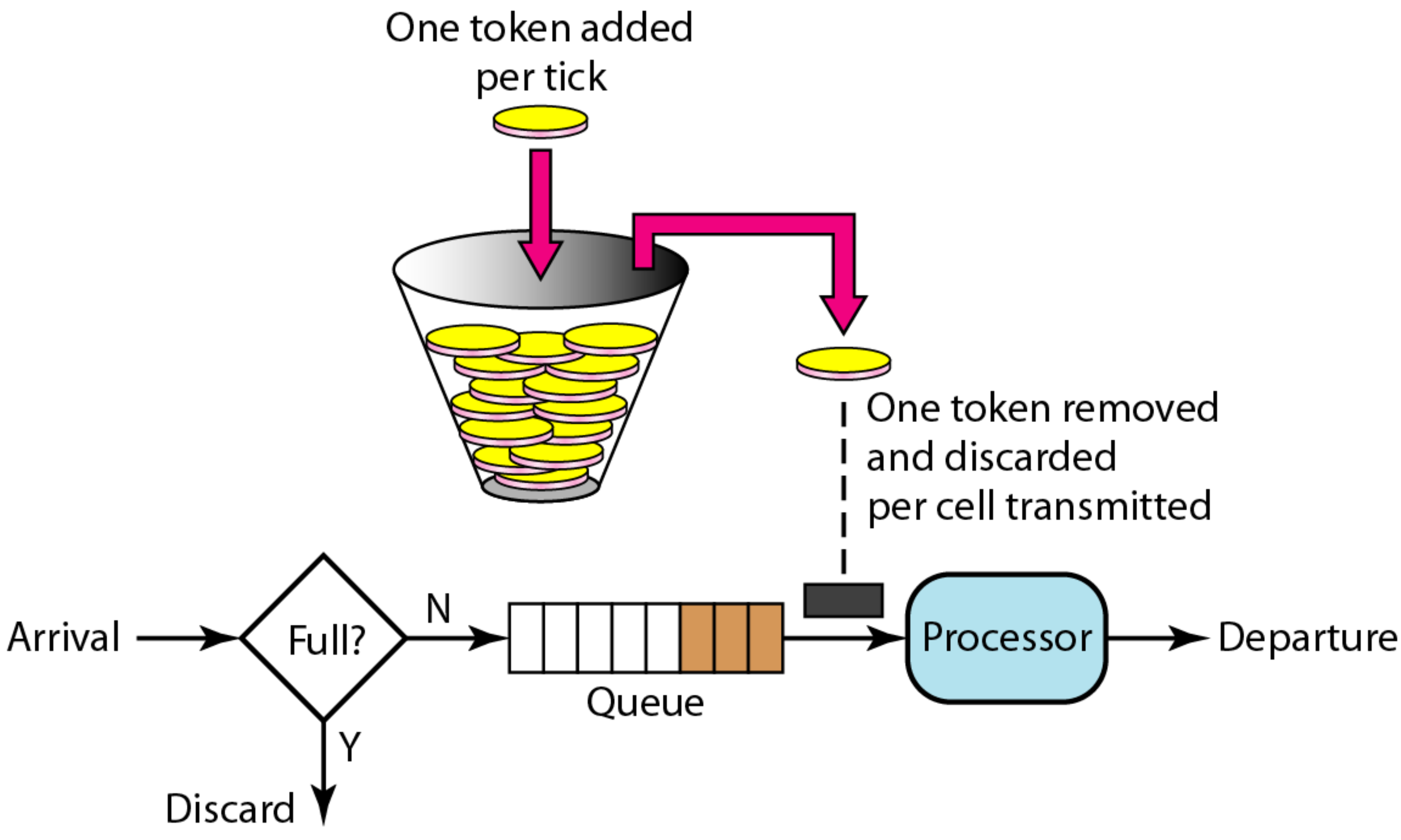
***Note***

**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.**

***Note***

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

## Figure 24.21 *Token bucket*



# 24-7 INTEGRATED SERVICES

***Two models have been designed to provide quality of service in the Internet: Integrated Services and Differentiated Services. We discuss the first model here.***

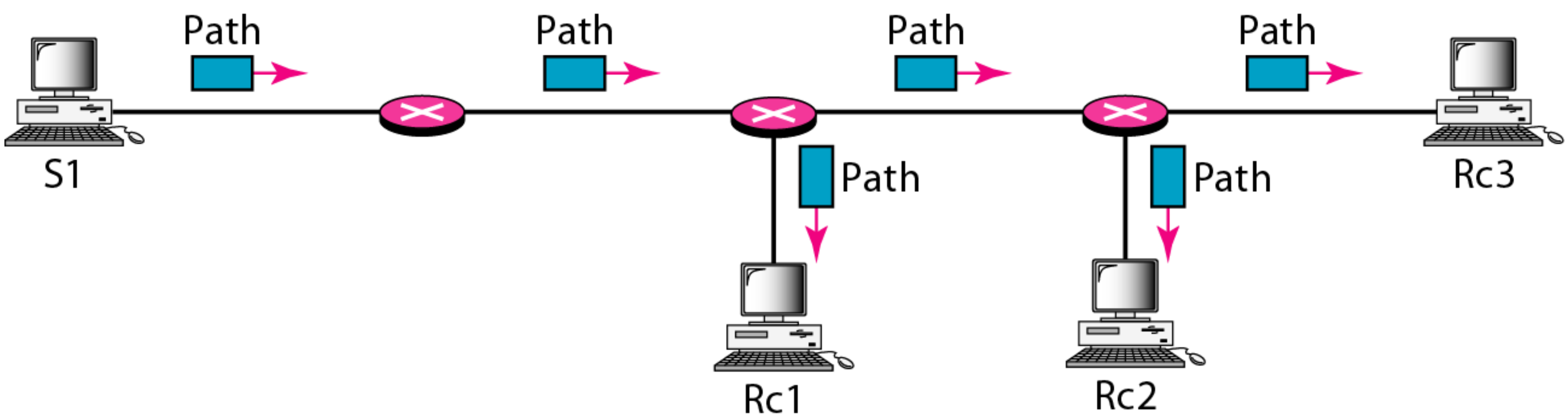
***Topics discussed in this section:***

**Signaling**
**Flow Specification**
**Admission**
**Service Classes**
**RSVP**
**Problems with Integrated Services**

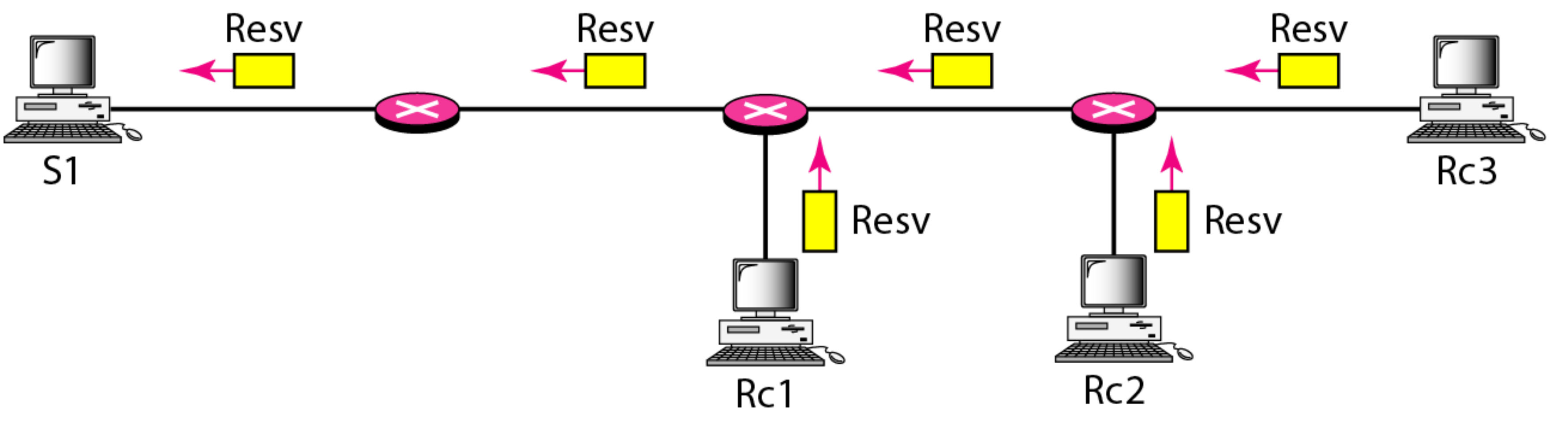
***Note***

**Integrated Services is a flow-based QoS model designed for IP.**

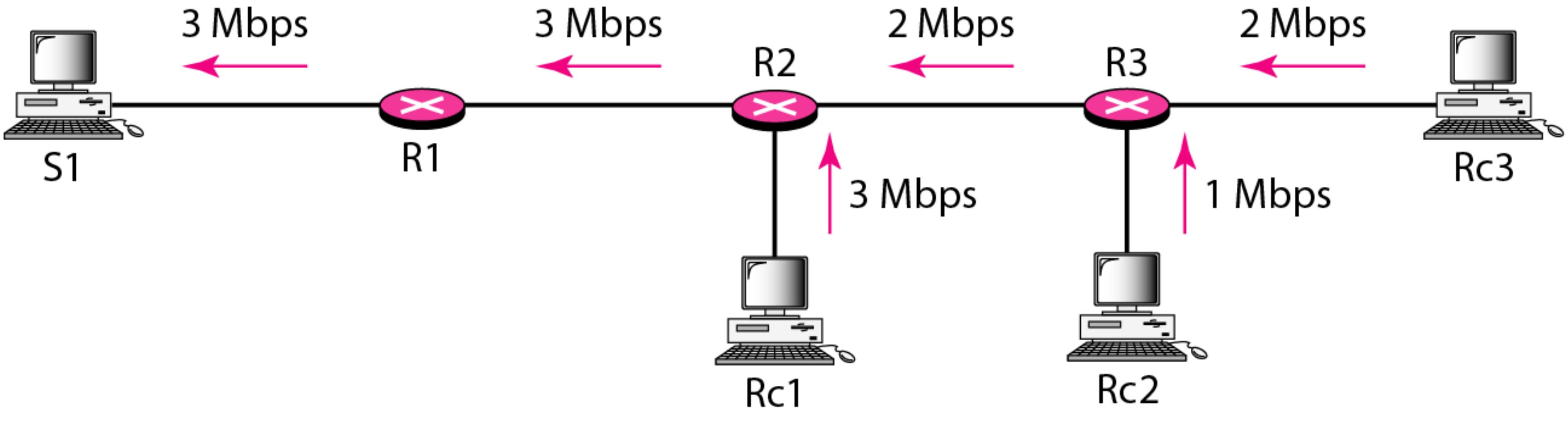
## Figure 24.22 *Path messages*



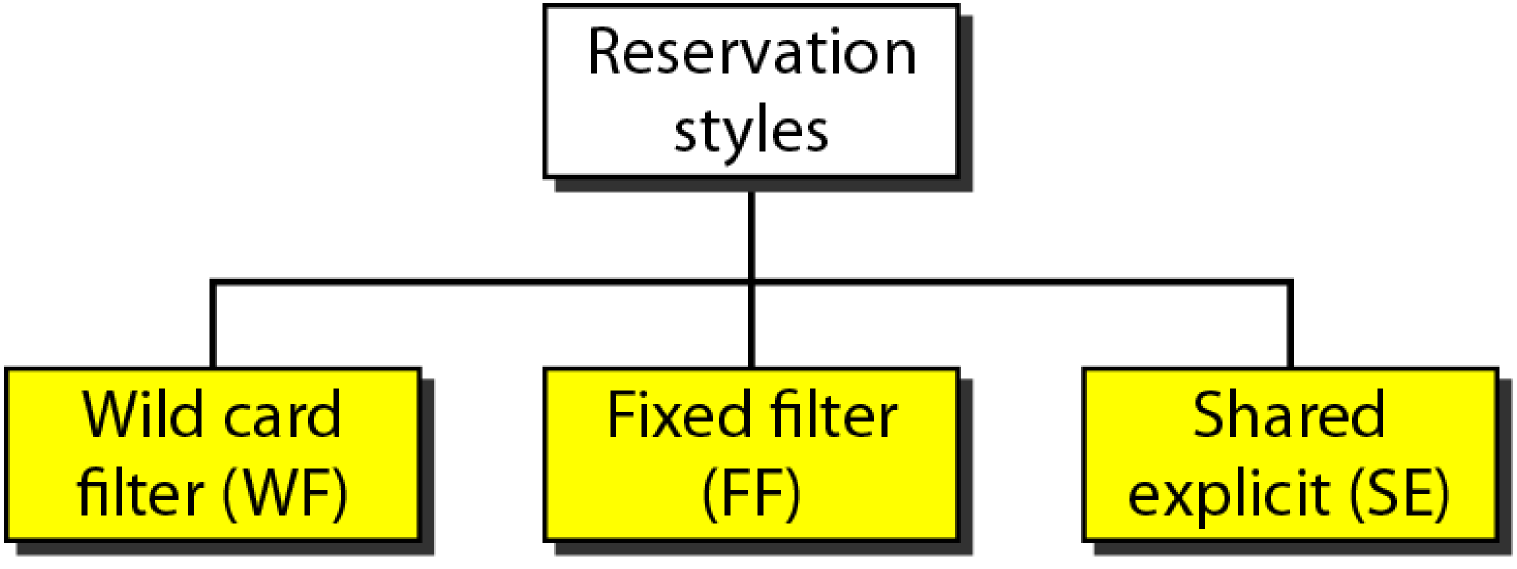
## Figure 24.23 *Resv messages*



## Figure 24.24 *Reservation merging*



## **Figure 24.25** *Reservation styles*



# 24-8 DIFFERENTIATED SERVICES

***Differentiated Services (DS or Diffserv) was introduced by the IETF (Internet Engineering Task Force) to handle the shortcomings of Integrated Services.***

***Topics discussed in this section:***

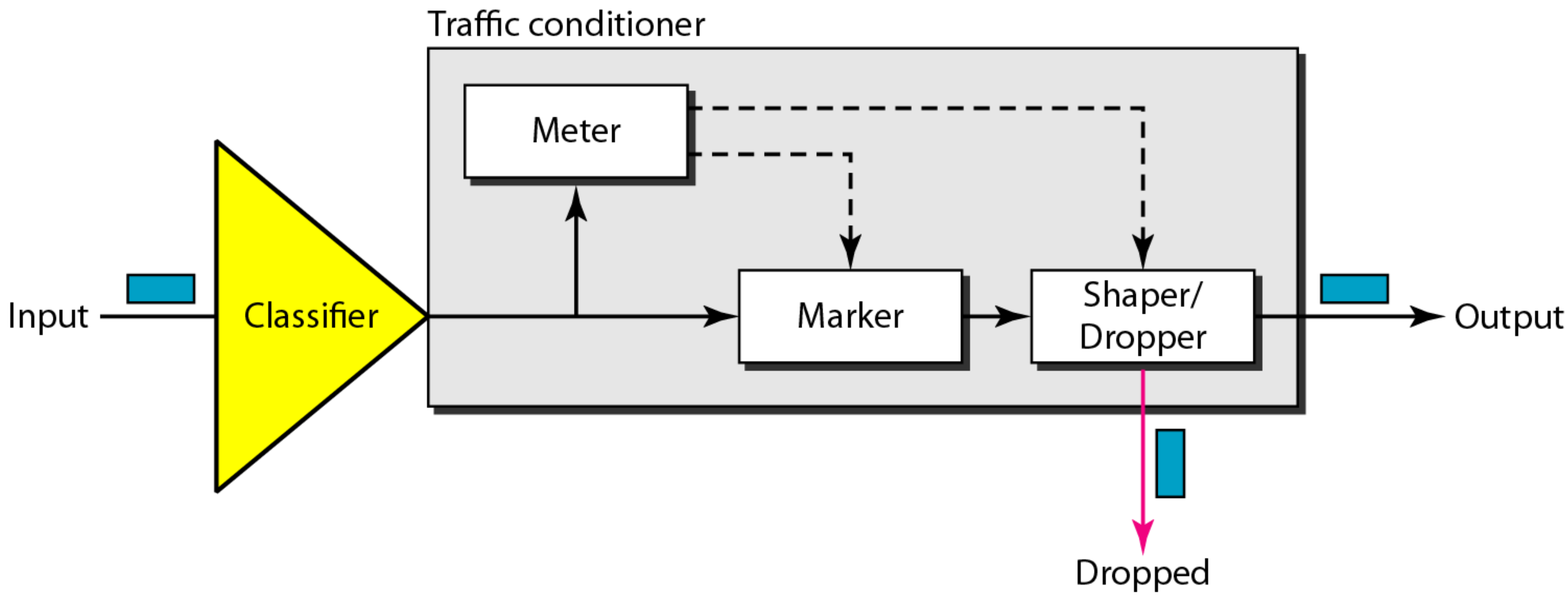
**DS Field**

***Note***

**Differentiated Services is a class-based QoS model designed for IP.**

## Figure 24.26 *DS field*

## **Figure 24.27** ***Traffic conditioner***



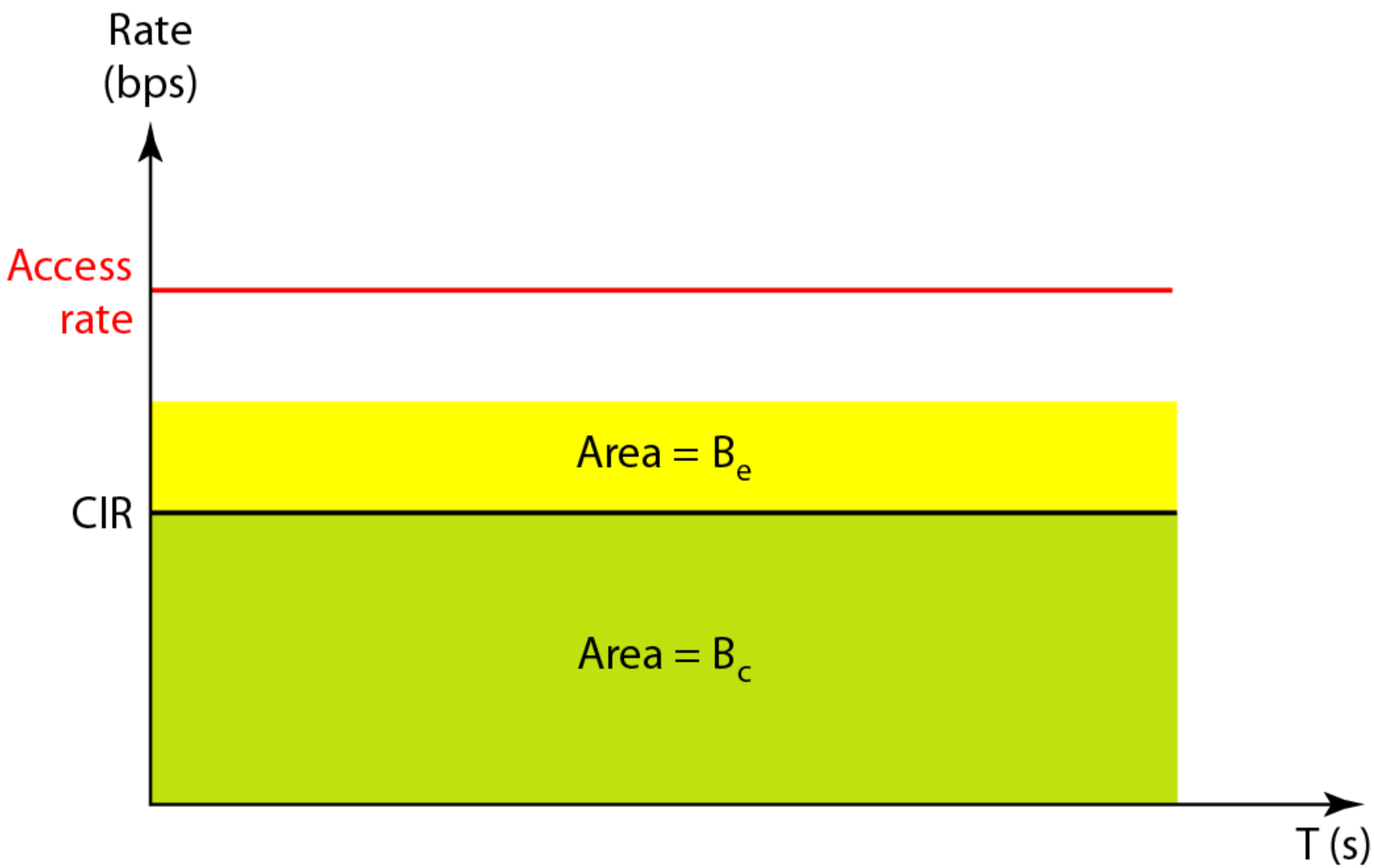
# 24-9 QoS IN SWITCHED NETWORKS

***Let us now discuss QoS as used in two switched networks: Frame Relay and ATM. These two networks are virtual-circuit networks that need a signaling protocol such as RSVP.***

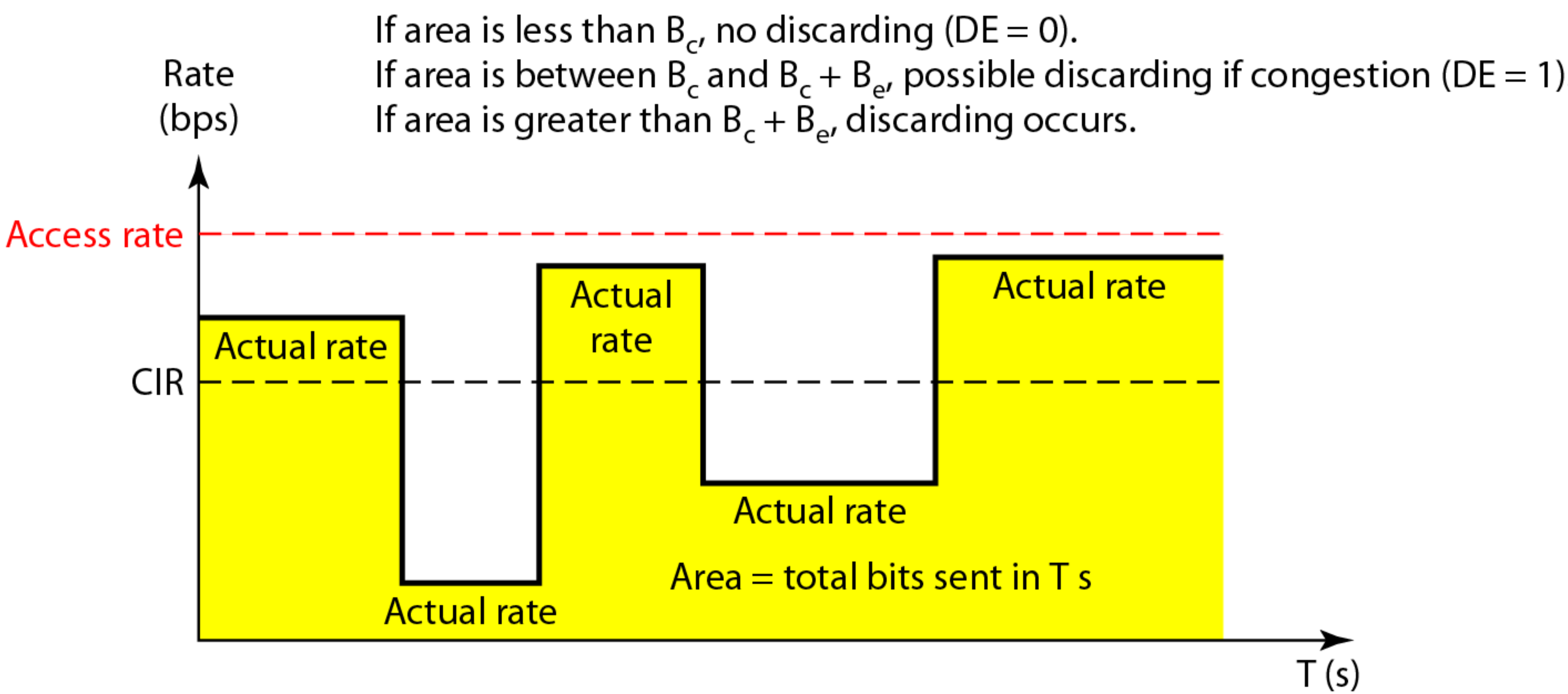
***Topics discussed in this section:***

**QoS in Frame Relay**
**QoS in ATM**

## **Figure 24.28** *Relationship between traffic control attributes*



## Figure 24.29 *User rate in relation to Bc and Bc + Be*



## **Figure 24.30** ***Service classes***

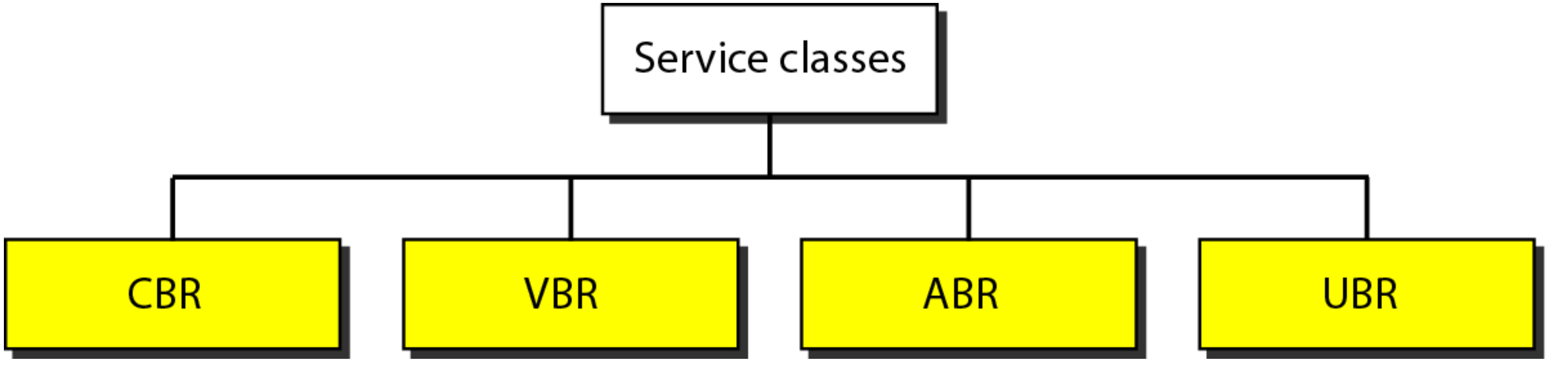


**Figure 24.31** ***Relationship of service classes to the total capacity of the network***