# Chapter 4

# Process-to-Process Delivery: UDP and TCP

# 4-1 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.***

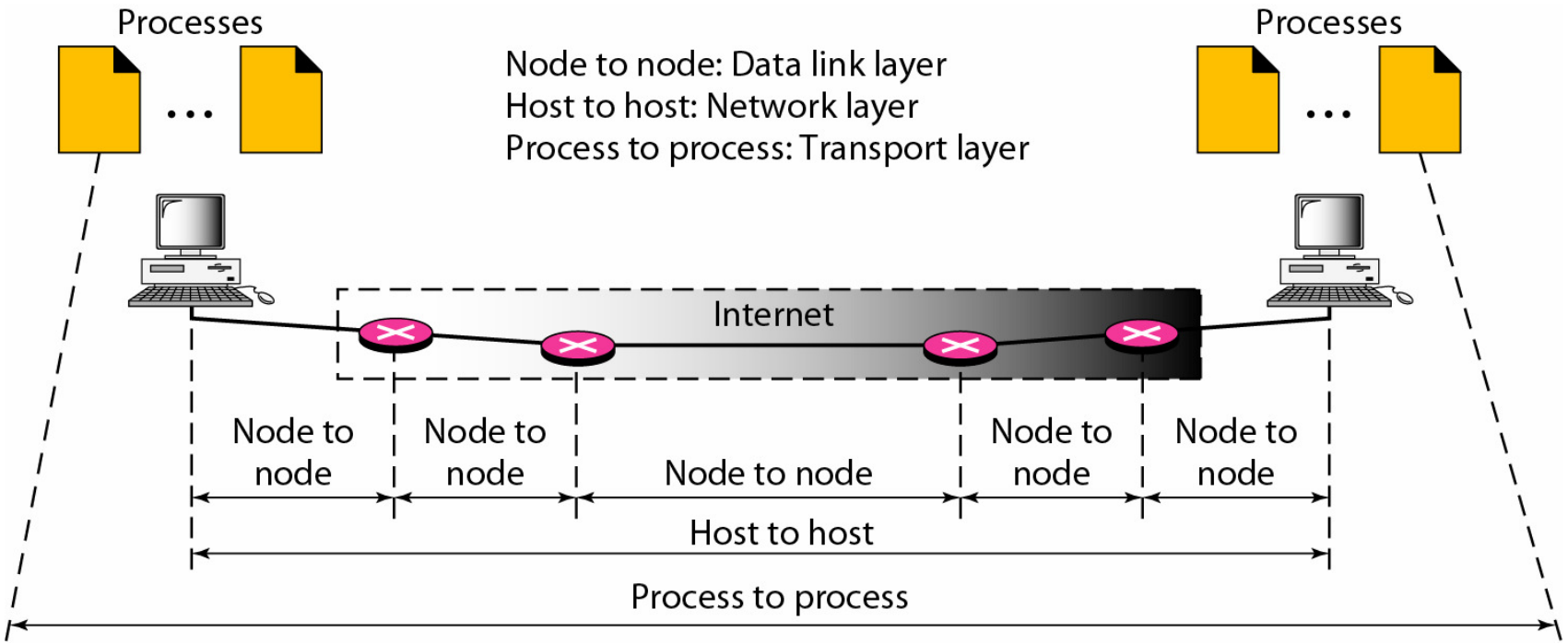
***Topics discussed in this section:***

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

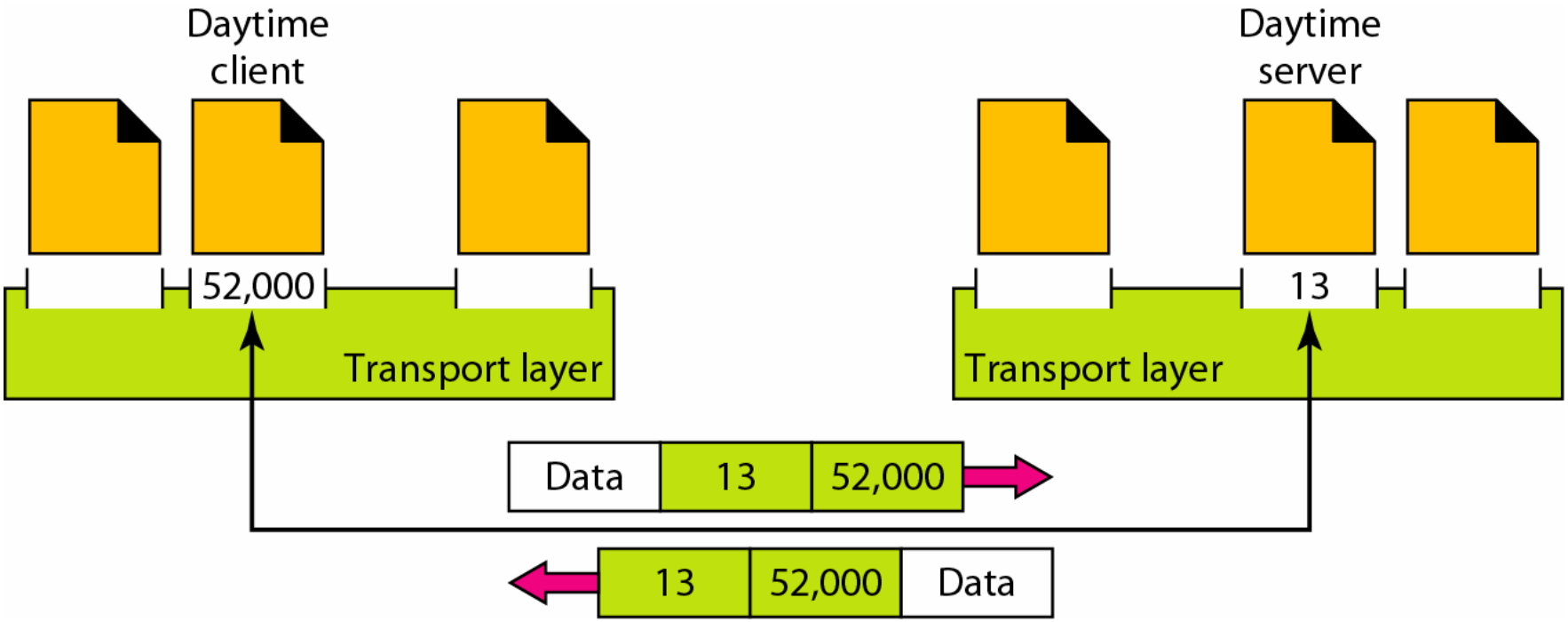
*Note*

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

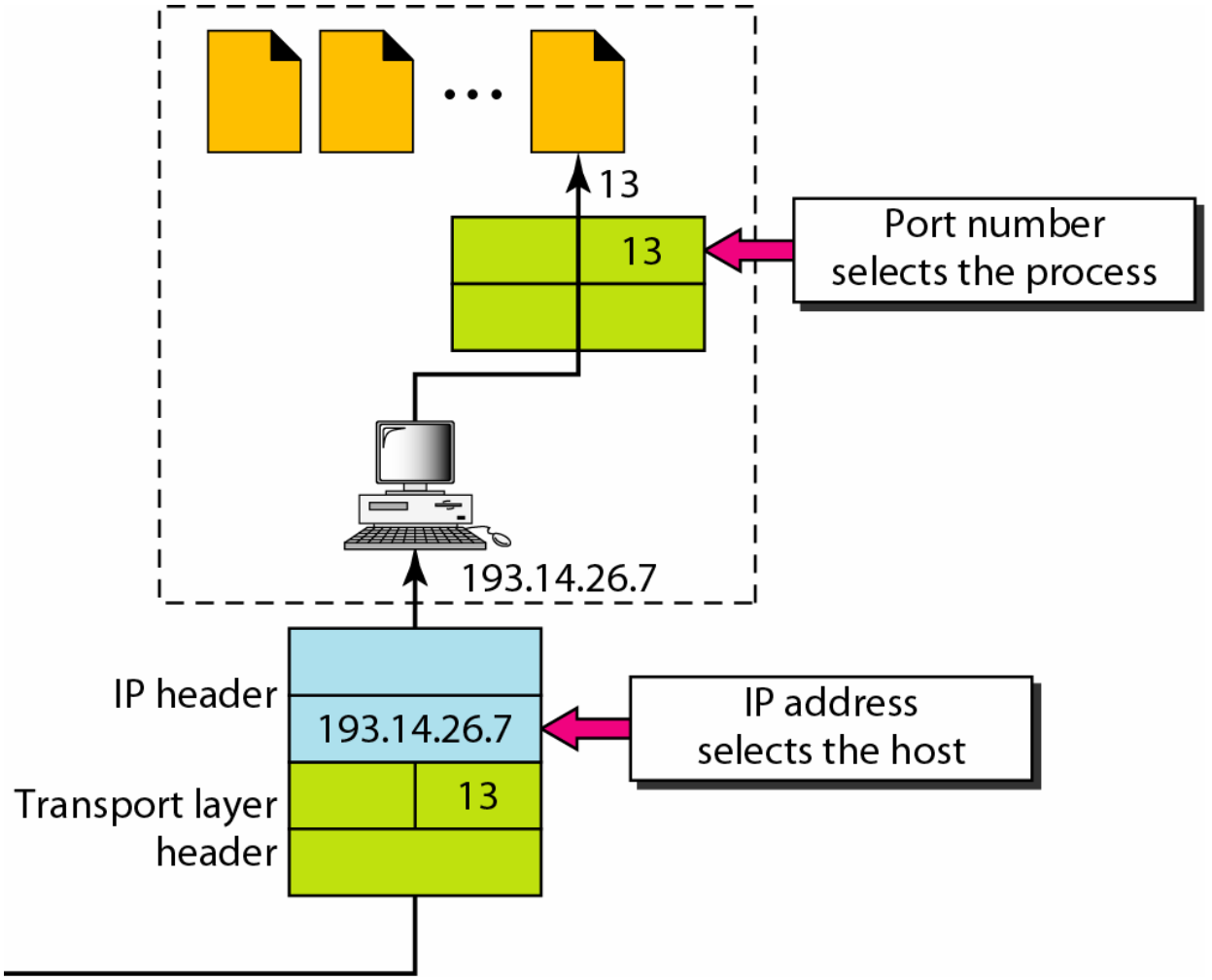
## Figure 4.1 *Types of data deliveries*



## Figure 4.2 *Port numbers*



## Figure 4.3 *IP addresses versus port numbers*



## Figure 4.4 *IANA ranges*

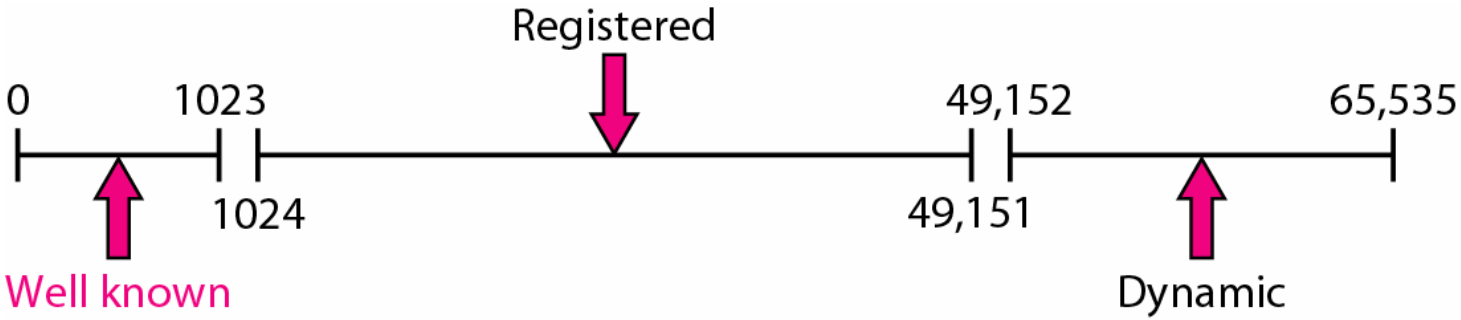


**Figure 4.5** *Socket address*

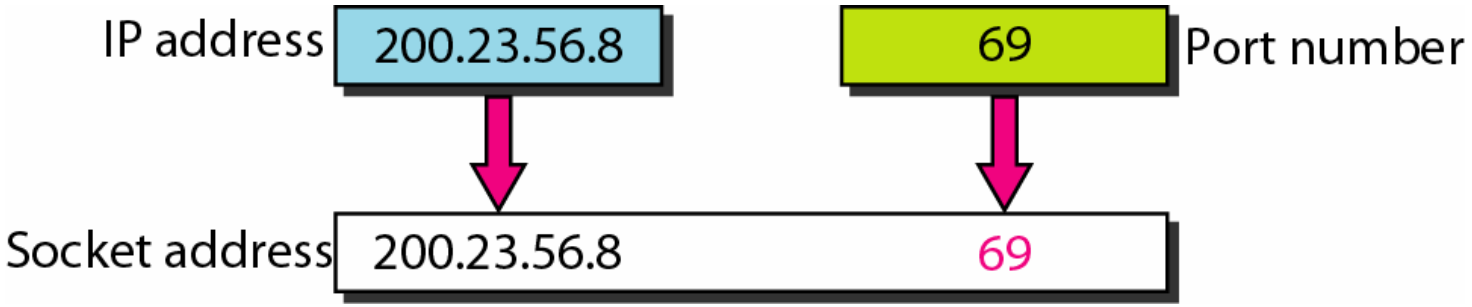
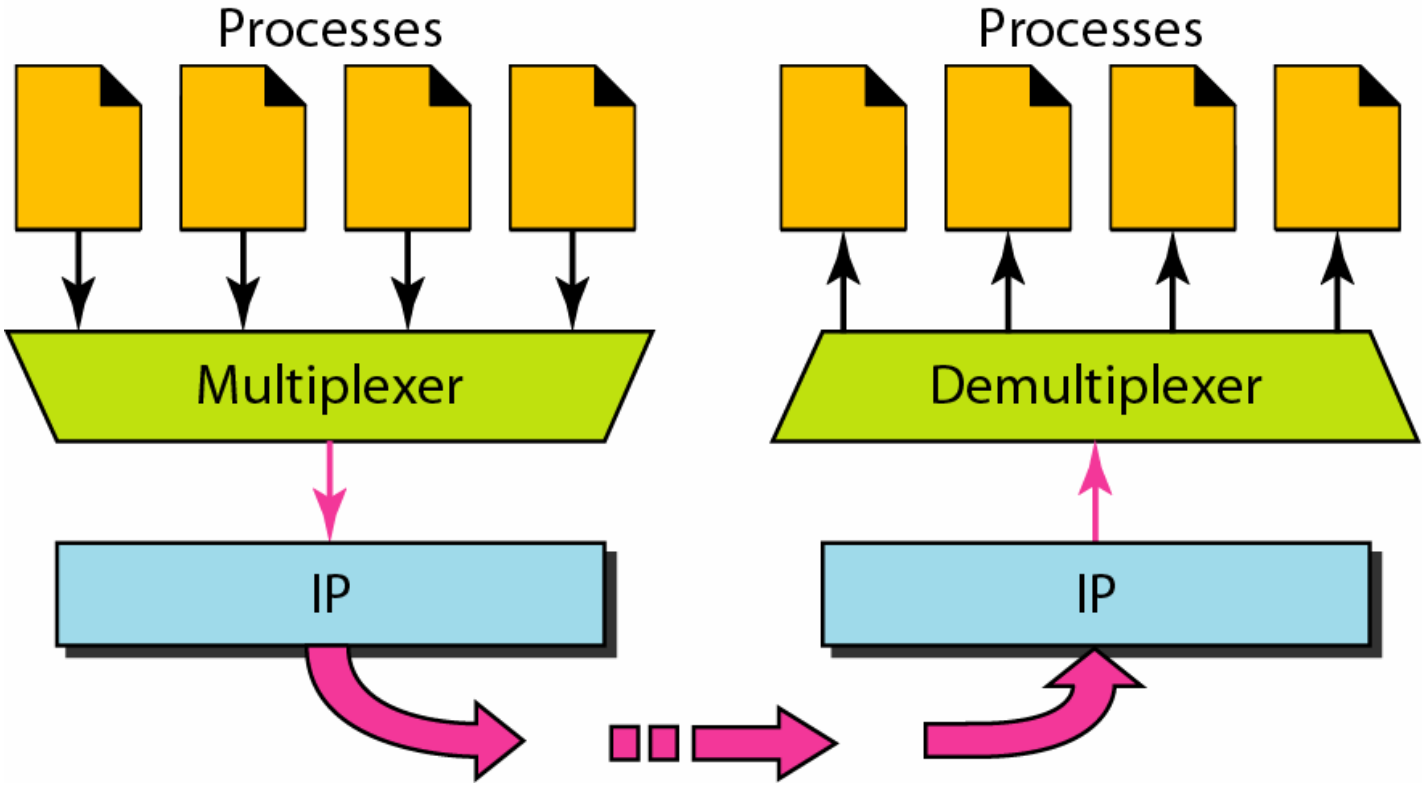


**Figure 4.6** *Multiplexing and demultiplexing*



## Figure 4.7 *Error control*

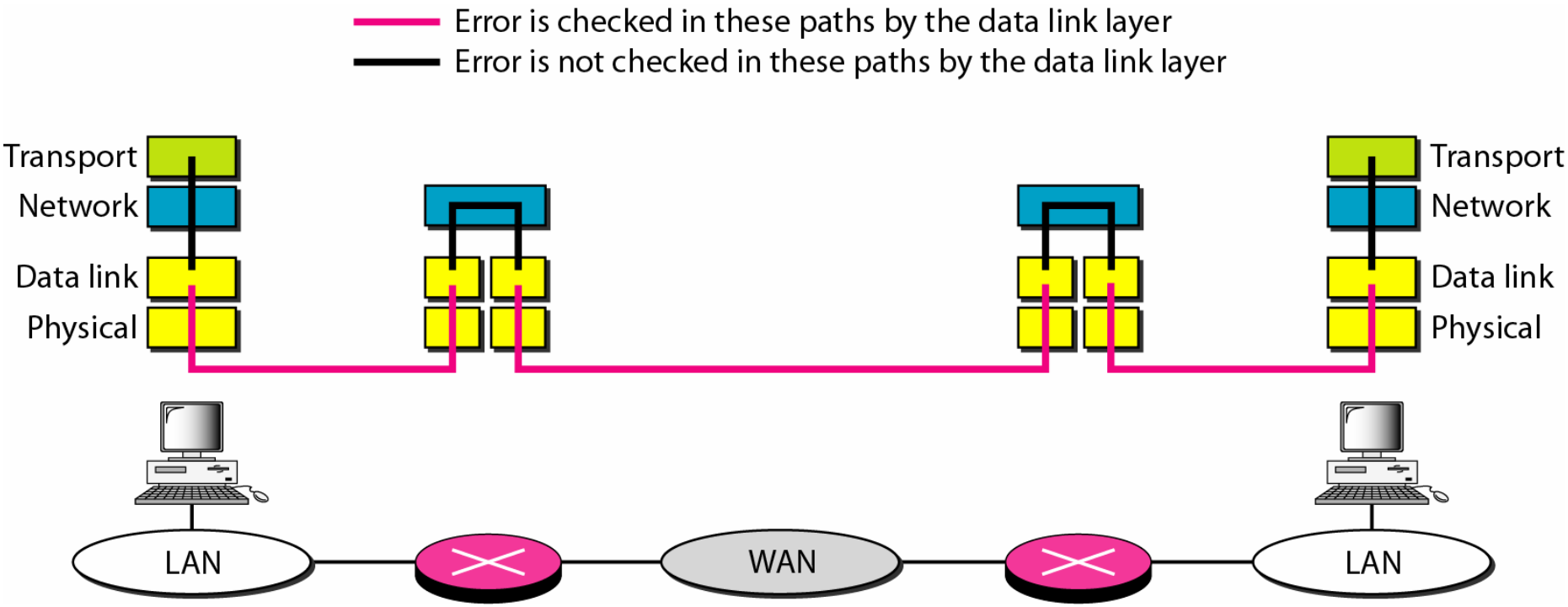
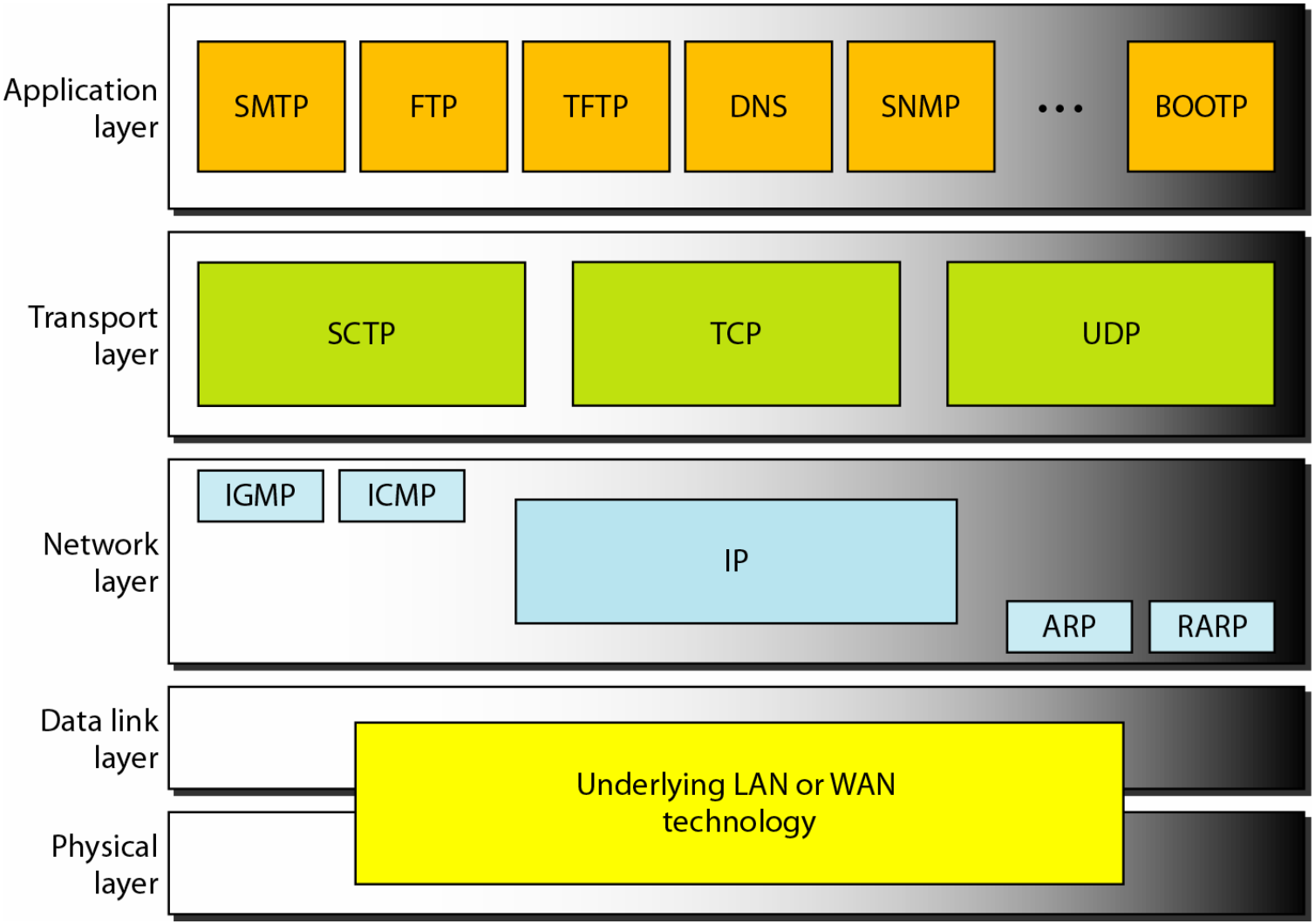


**Figure 4.8** *Position of UDP, TCP, and SCTP in TCP/IP suite*



# 4-2 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.***

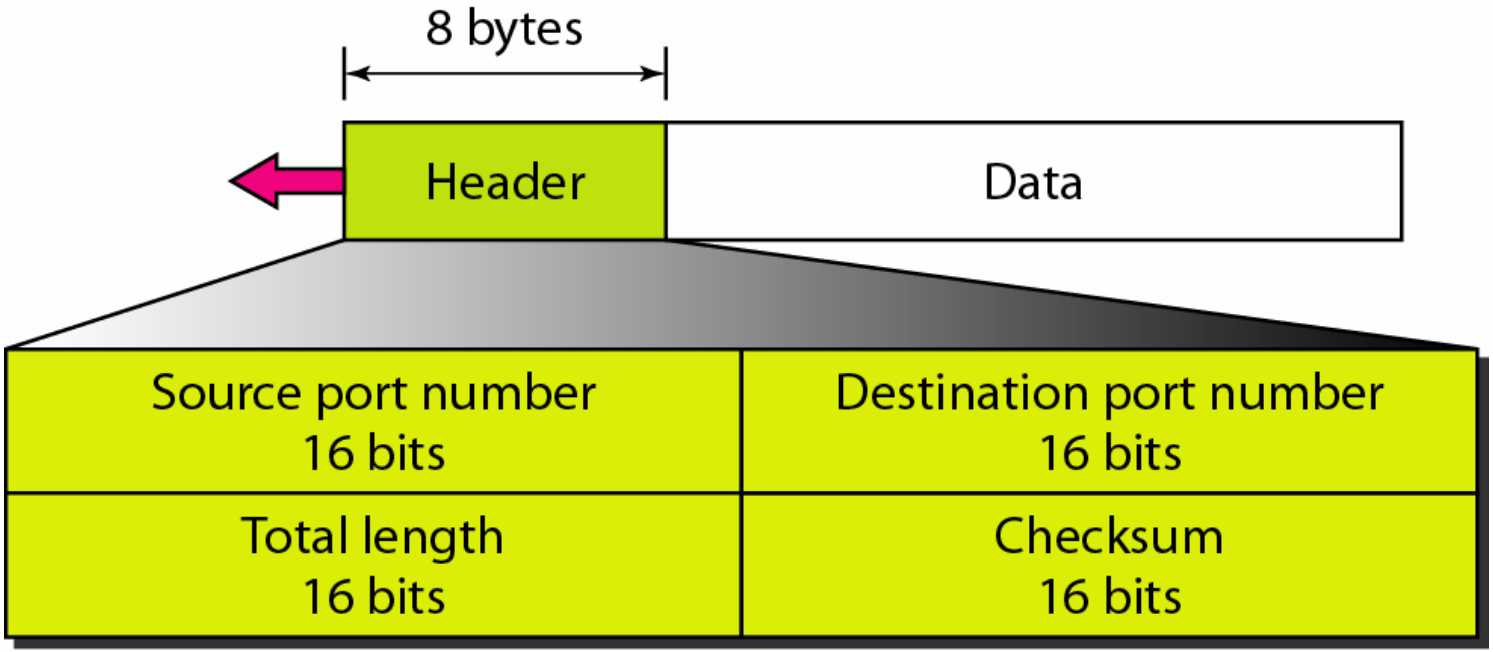
***Topics discussed in this section:***

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

## Table 4.1 *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) |

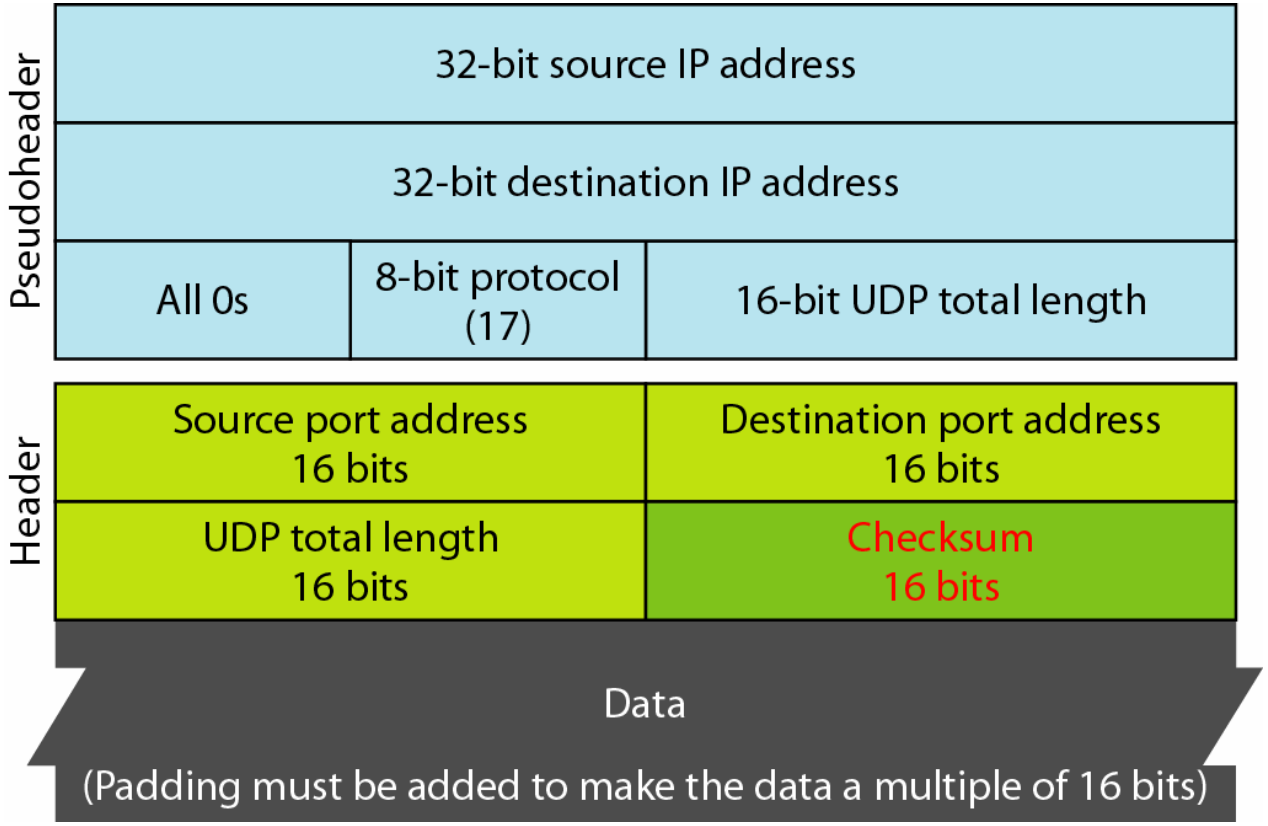
**Figure 4.9** *User datagram format*

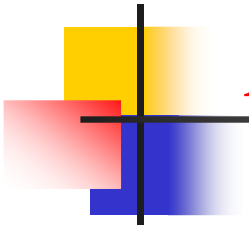


*Note*

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

## Figure 4.10 *Pseudoheader for checksum calculation*

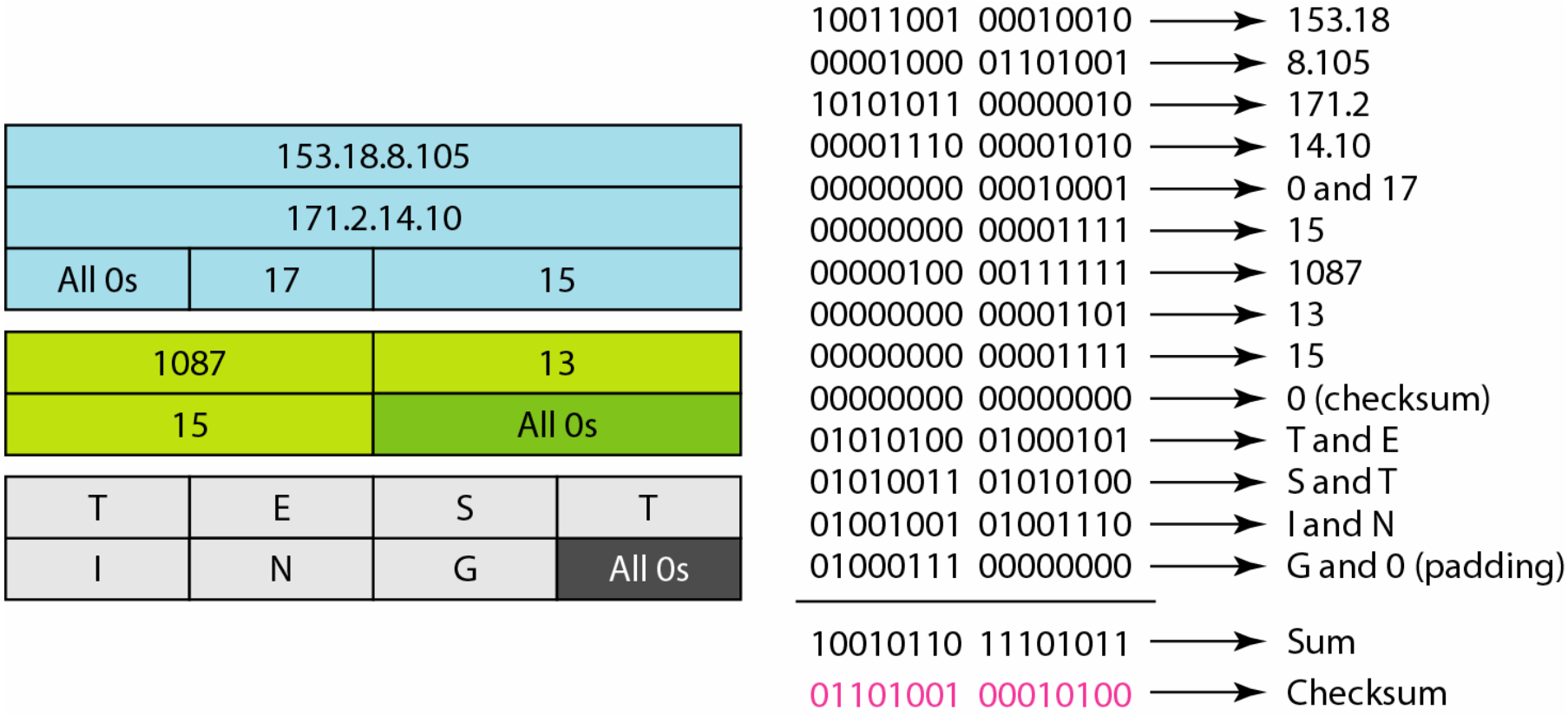




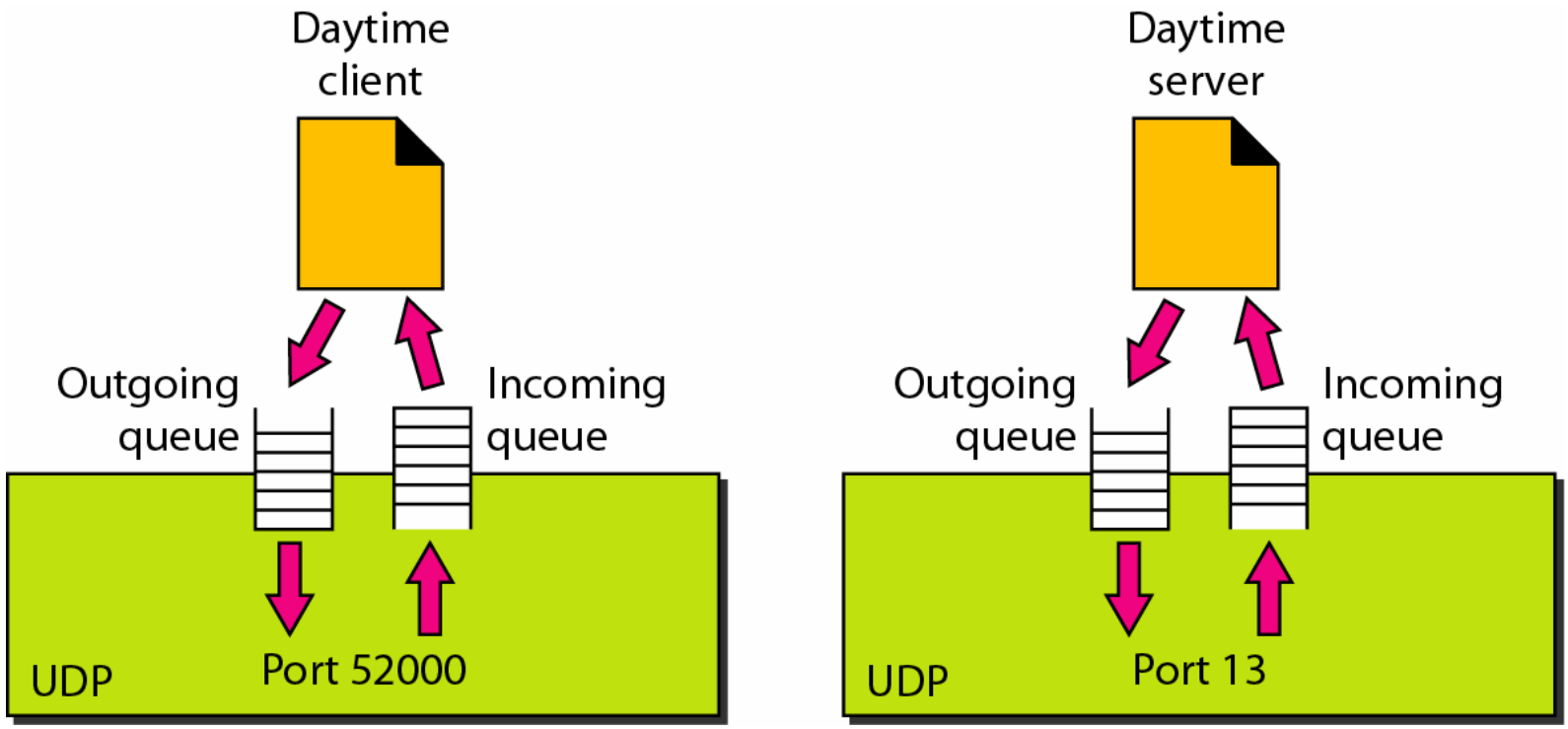
# *Example 23.2*

***Figure 4.11 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 4.11 *Checksum calculation of a simple UDP user datagram*



## Figure 4.12 *Queues in UDP*



*Note*

**Connectionless Services: UDP provides a connectionless service. This means that each user datagram sent by UDP is an independent datagram**

Note

**Flow and Error Control: UDP is a very unreliable transport protocol. There is no flow and error control.**

*Note*

**Use of UDP: UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control.**

# 4-3 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 4.2 *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 4.13** *Stream delivery*

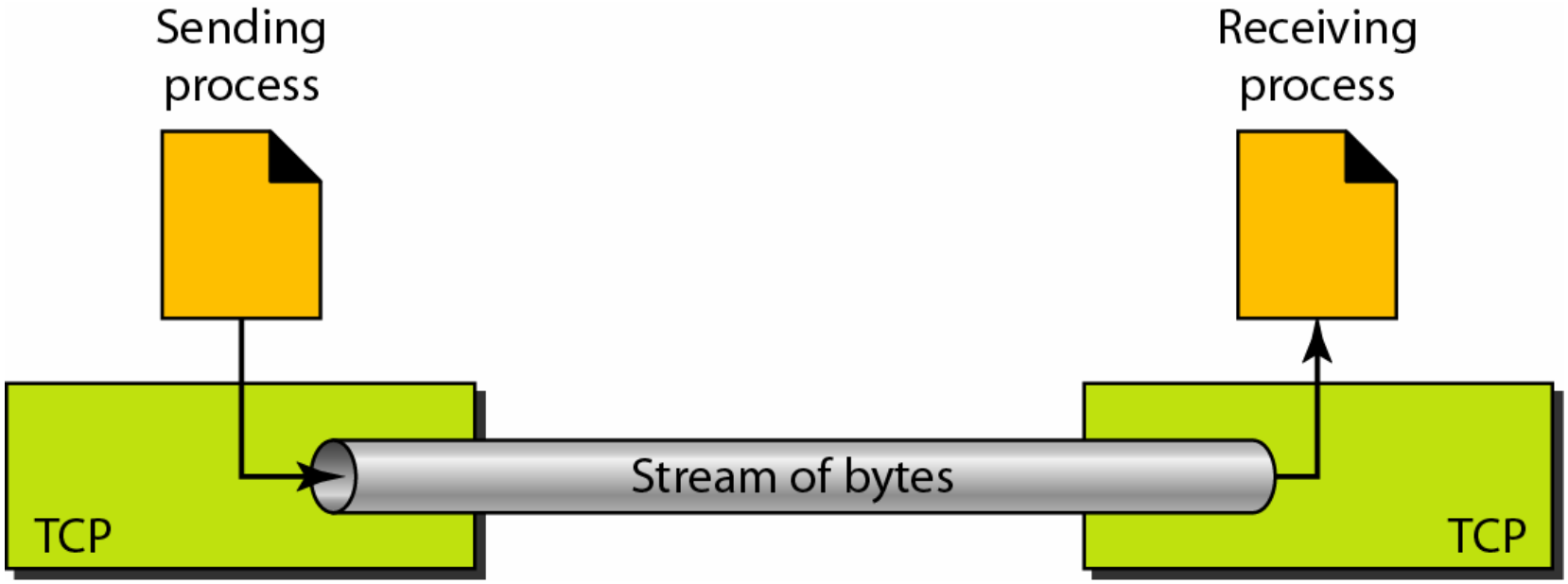
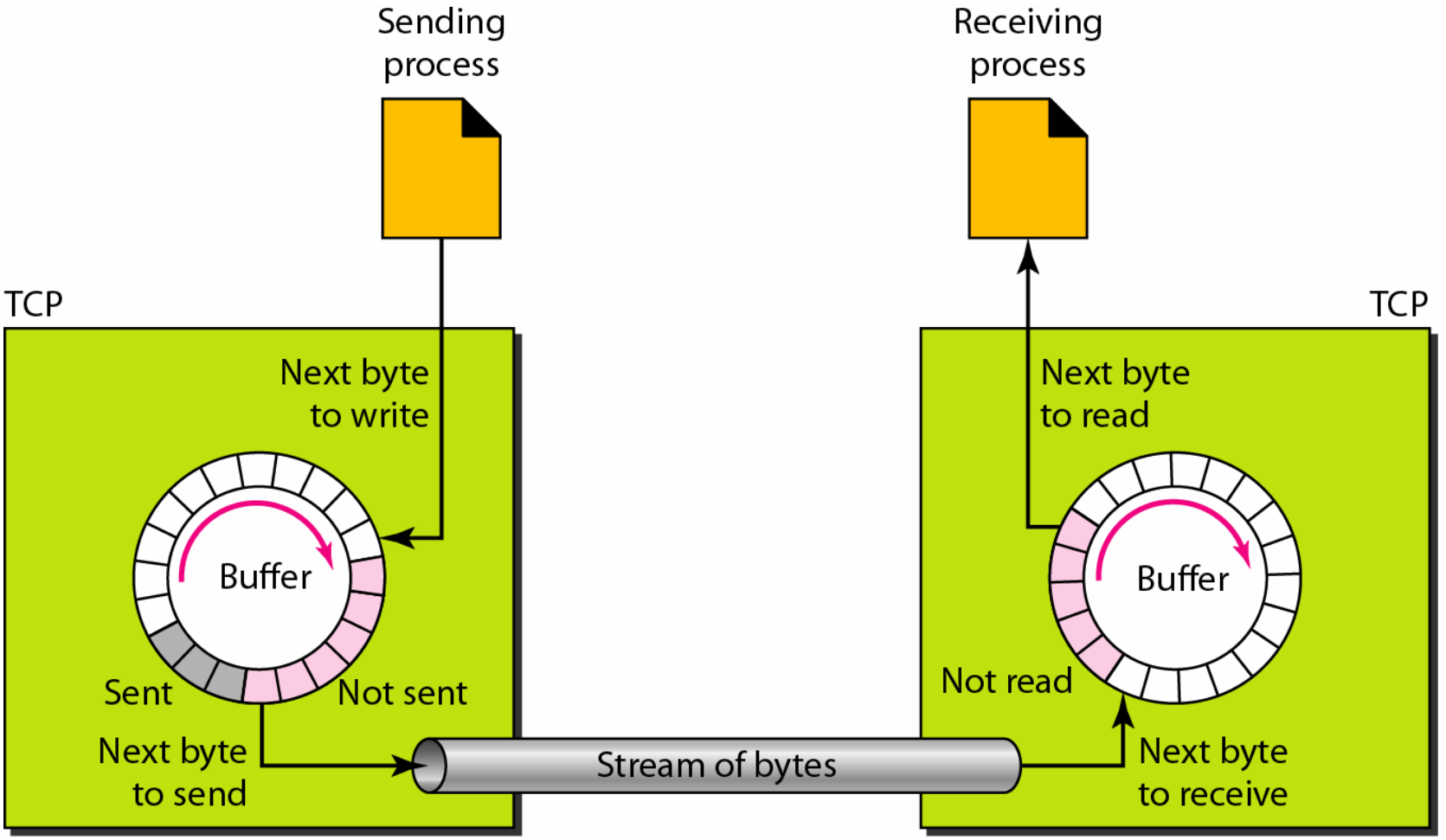
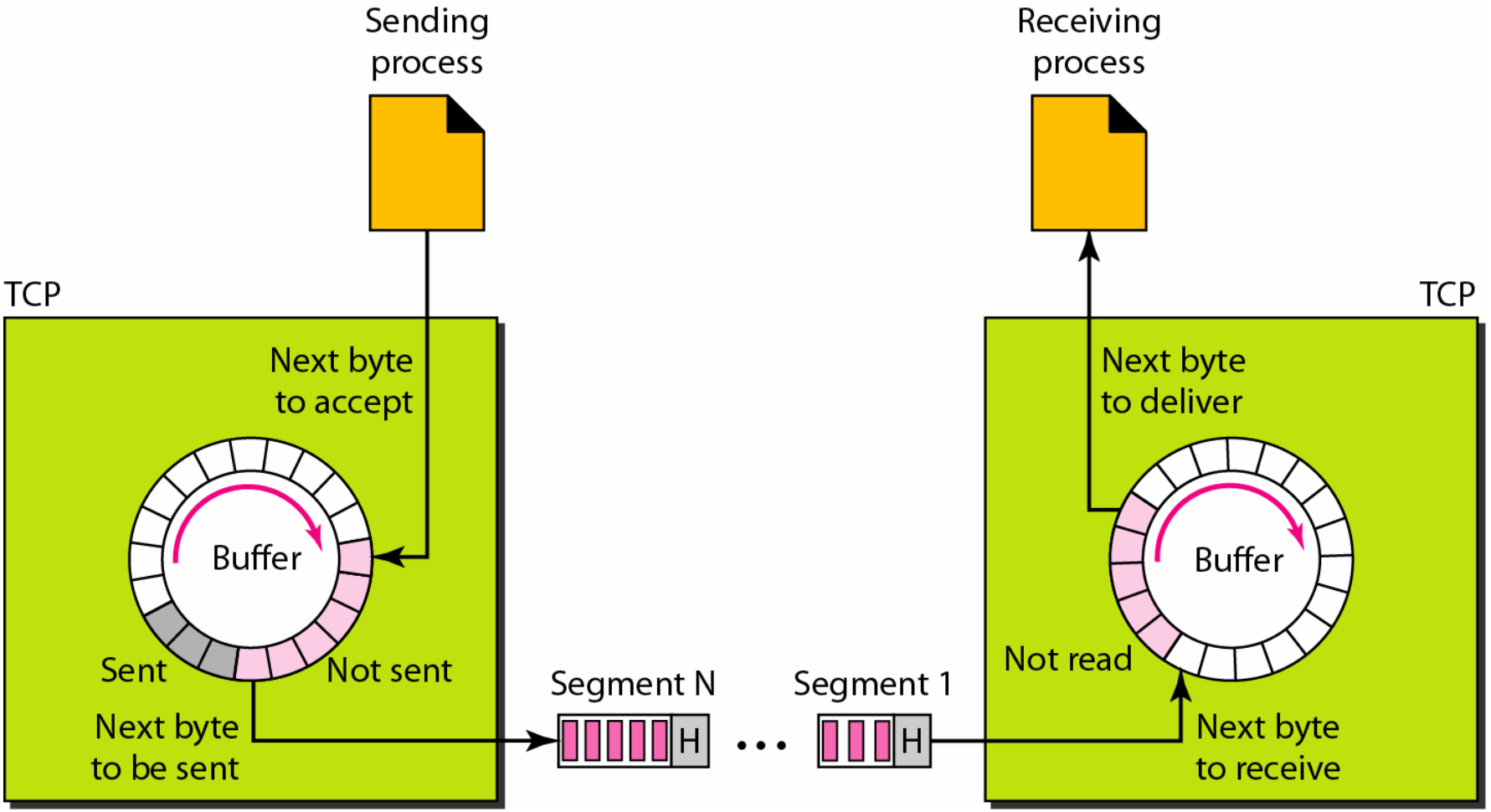


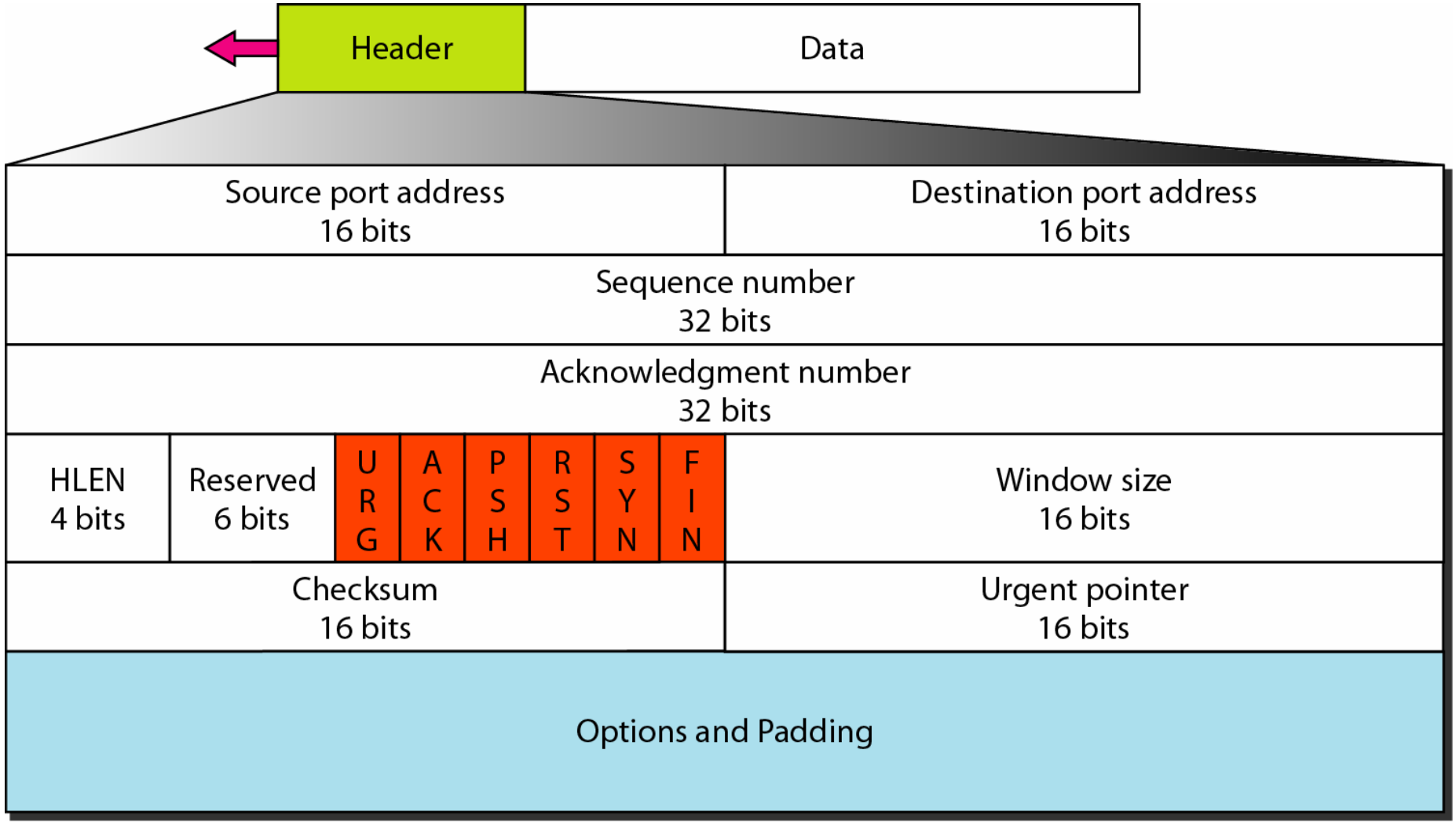
**Figure 4.14** *Sending and receiving buffers*



## Figure 4.15 *TCP segments*



## Figure 4.16 *TCP segment format*

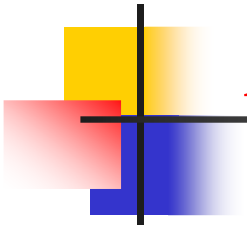


*Note*

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

*Note*

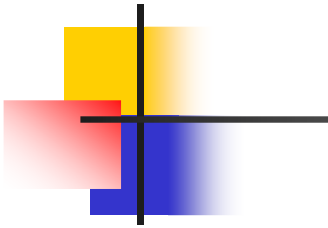
**The value in the sequence number field of a segment defines the number of the first data byte contained in that segment.**



# *Example 4.3*

***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 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 4.17 *Control field*

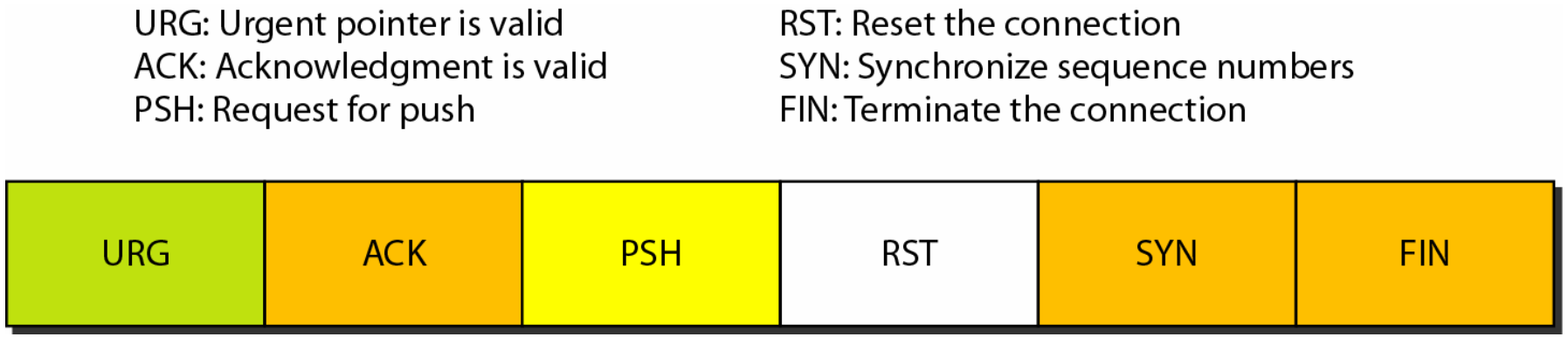
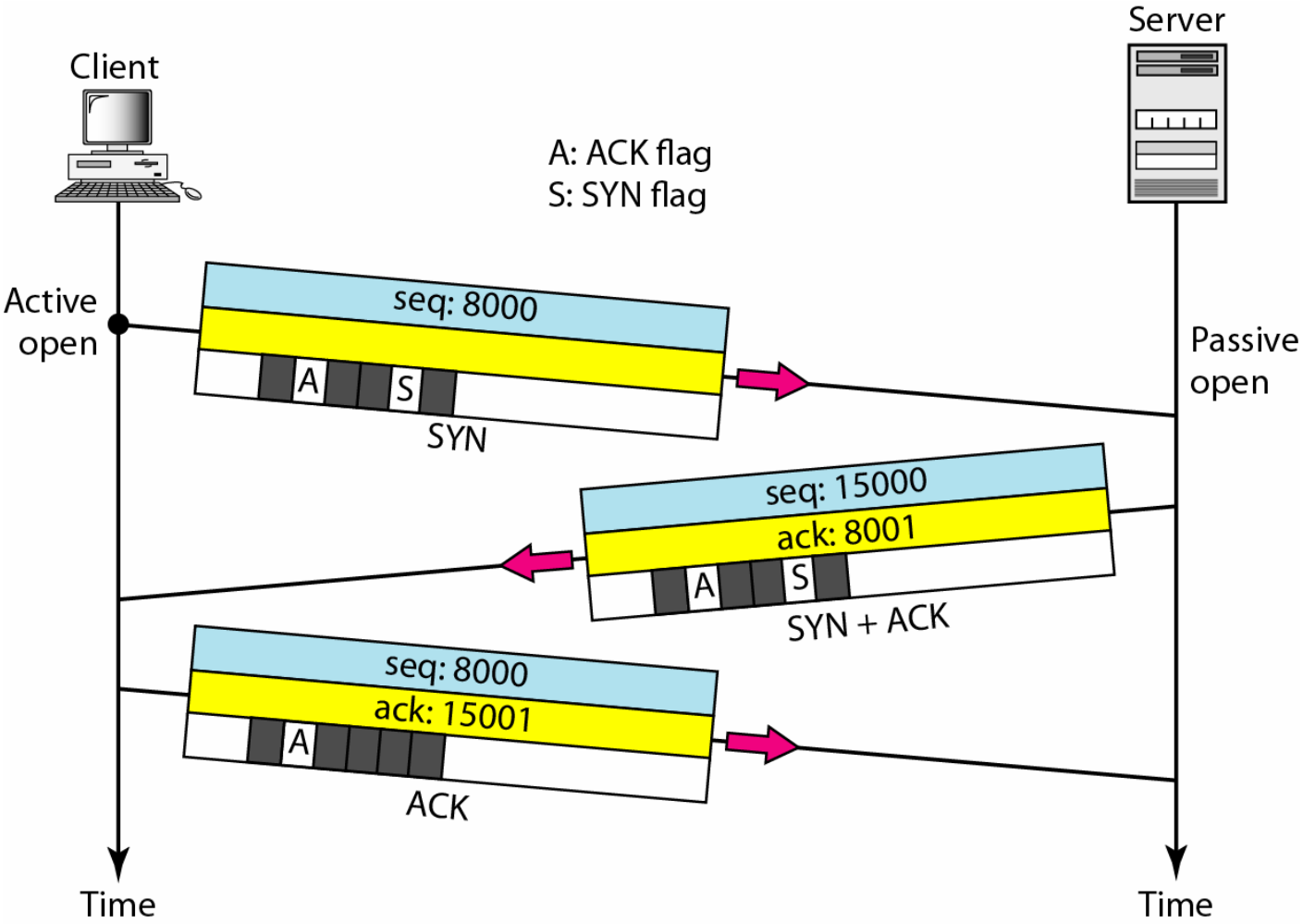


**Table 4.3** ***Description of flags in the control field***

| *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 4.18** *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 4.19 *Data transfer*

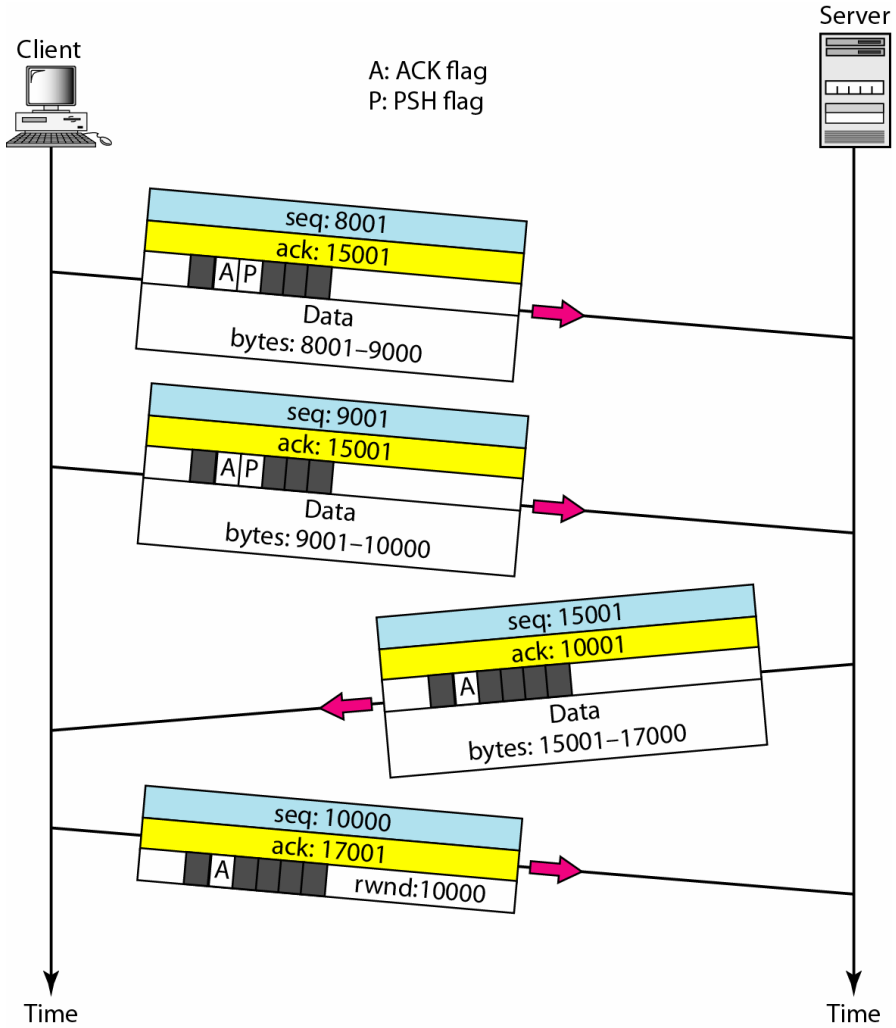
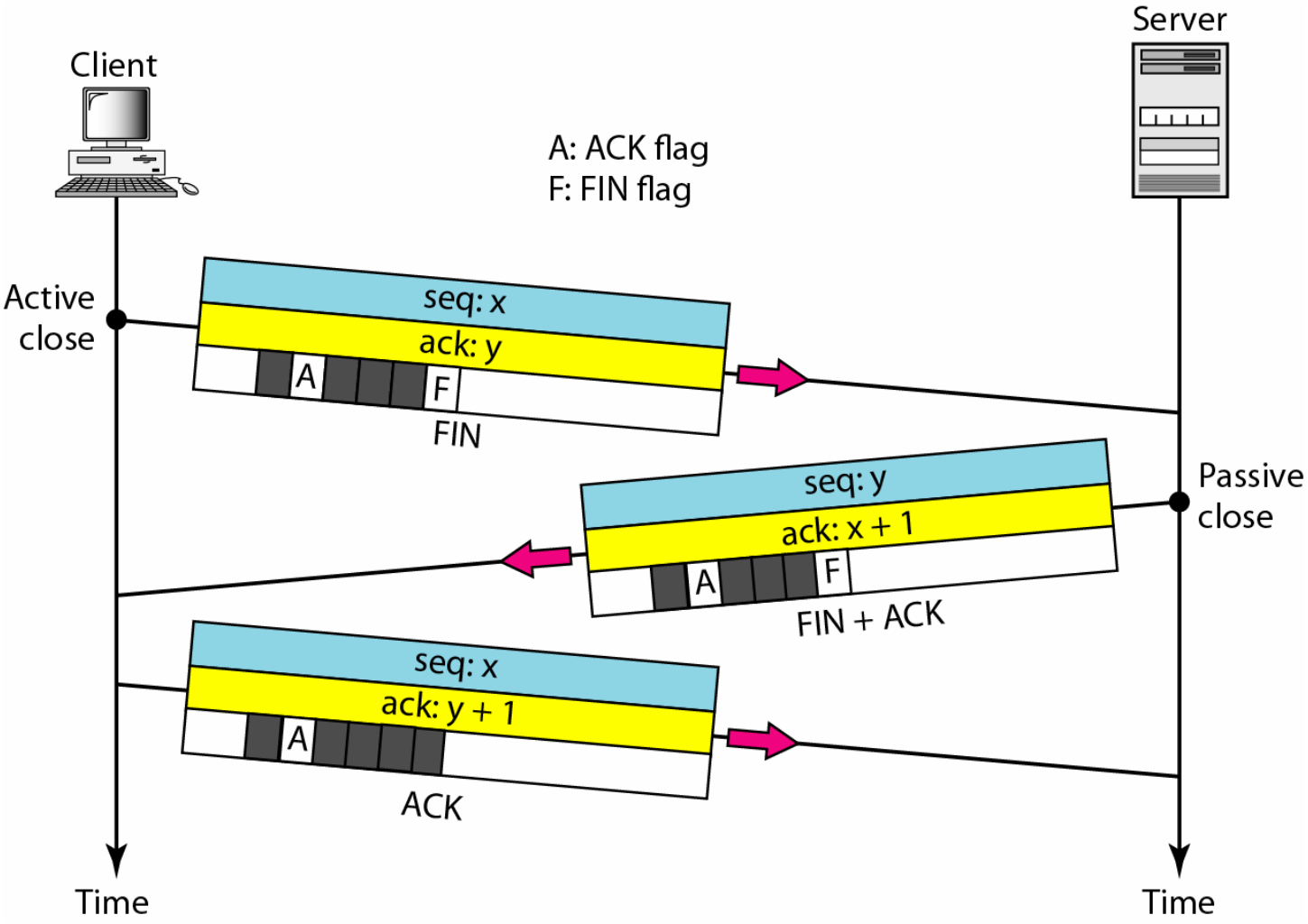


**Figure 4.20** *Connection termination using three-way handshaking*



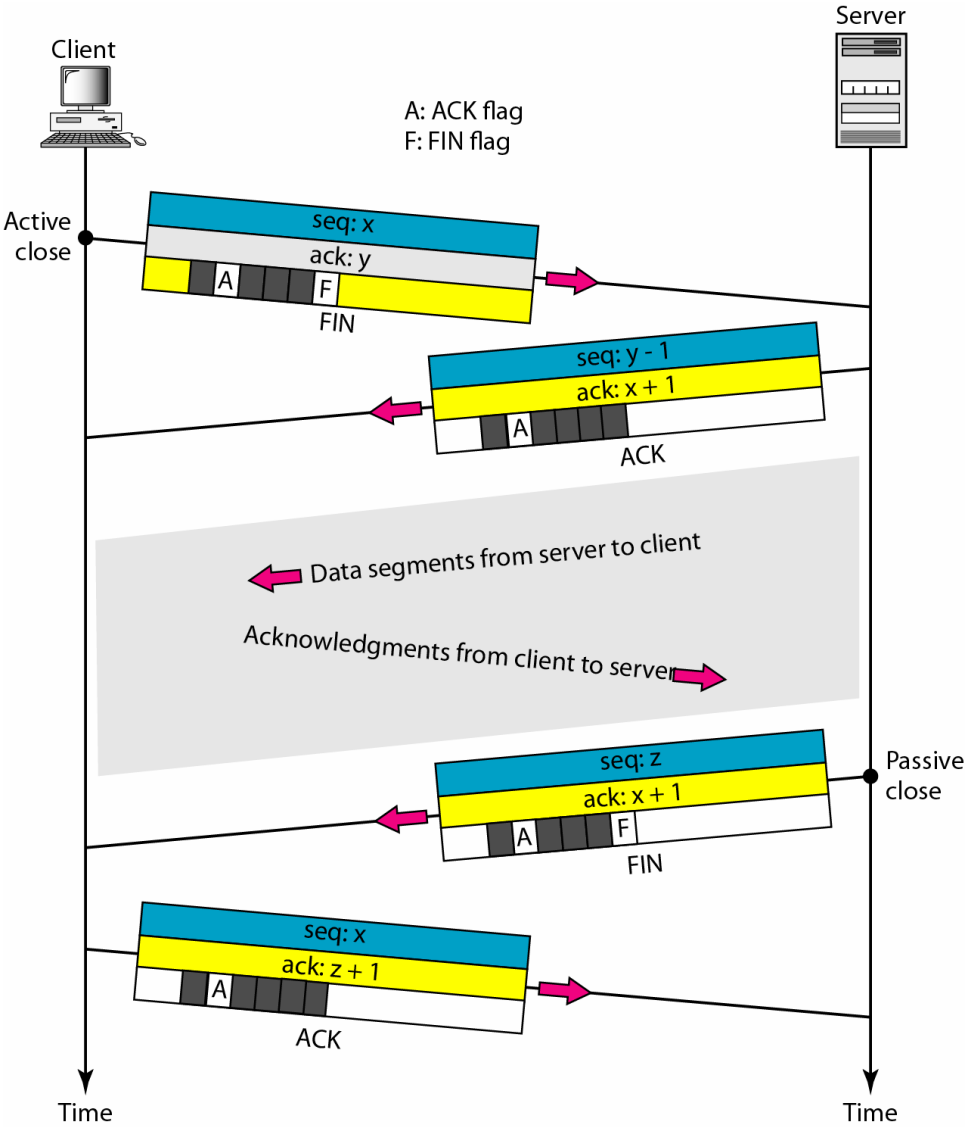
*Note*

**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 4.21 *Half-close*



Note

**Flow Control: TCP uses a sliding window to handle flow control. The sliding window protocol used by TCP, is something between the Go-Back-N and Selective Repeat 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.**

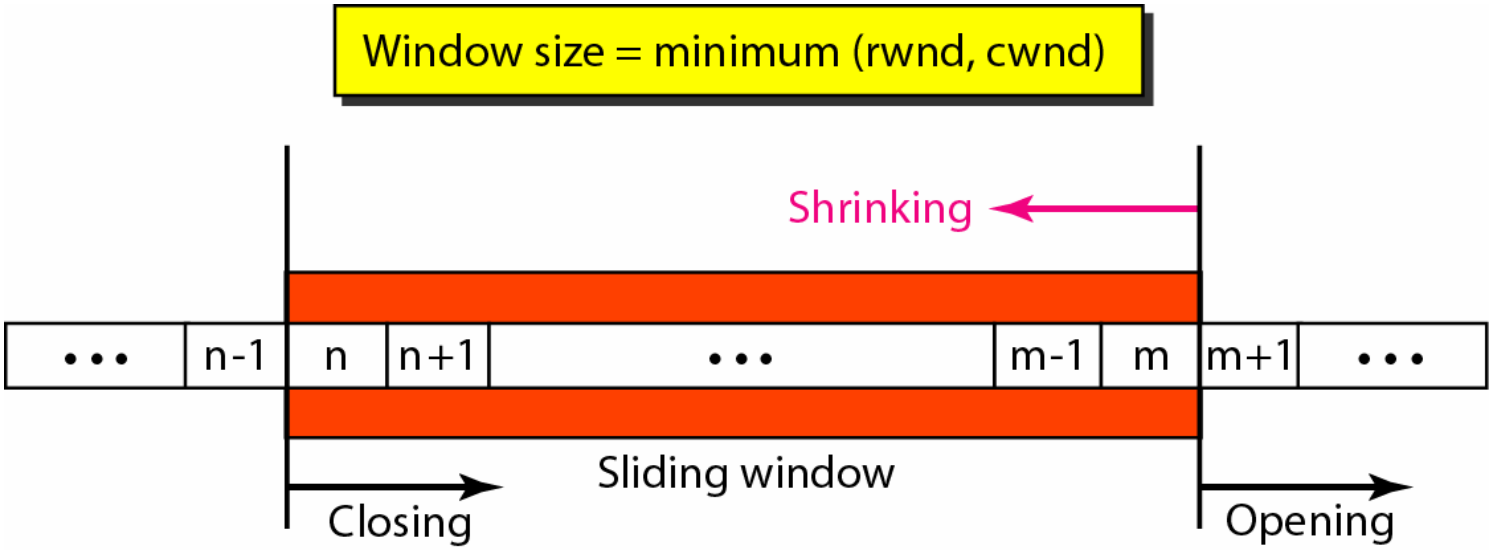
*Note*

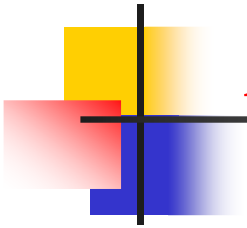
**- TCP sliding windows are byte-oriented; the data link layer is frame-oriented.**
**- The TCP's sliding window is of variable size; the data link layer was of fixed size.**

Note

**The size of the window at one end is determined by the lesser of two values: receiver window (rwnd) or congestion window (cwnd).**

## Figure 4.22 *Sliding window*





# *Example 4.5*

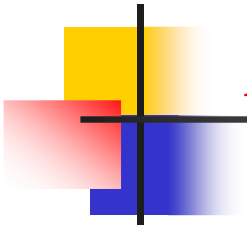
***What is the size of the window for host A if the value of rwnd is 3000 bytes and the value of cwnd is 3500 bytes?***

***Solution***

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

Note

**The receiver window is the value advertised by the opposite end in a segment containing acknowledgment. It is the number of bytes the other end can accept before its buffer overflows**



# *Example 4.4*

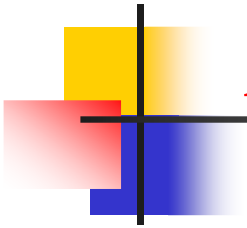
***What is the value of the receiver window (rwnd) for host A if the receiver, host B, has a buffer size of 5000 bytes and 1000 bytes of received and unprocessed data?***

***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.***

*Note*

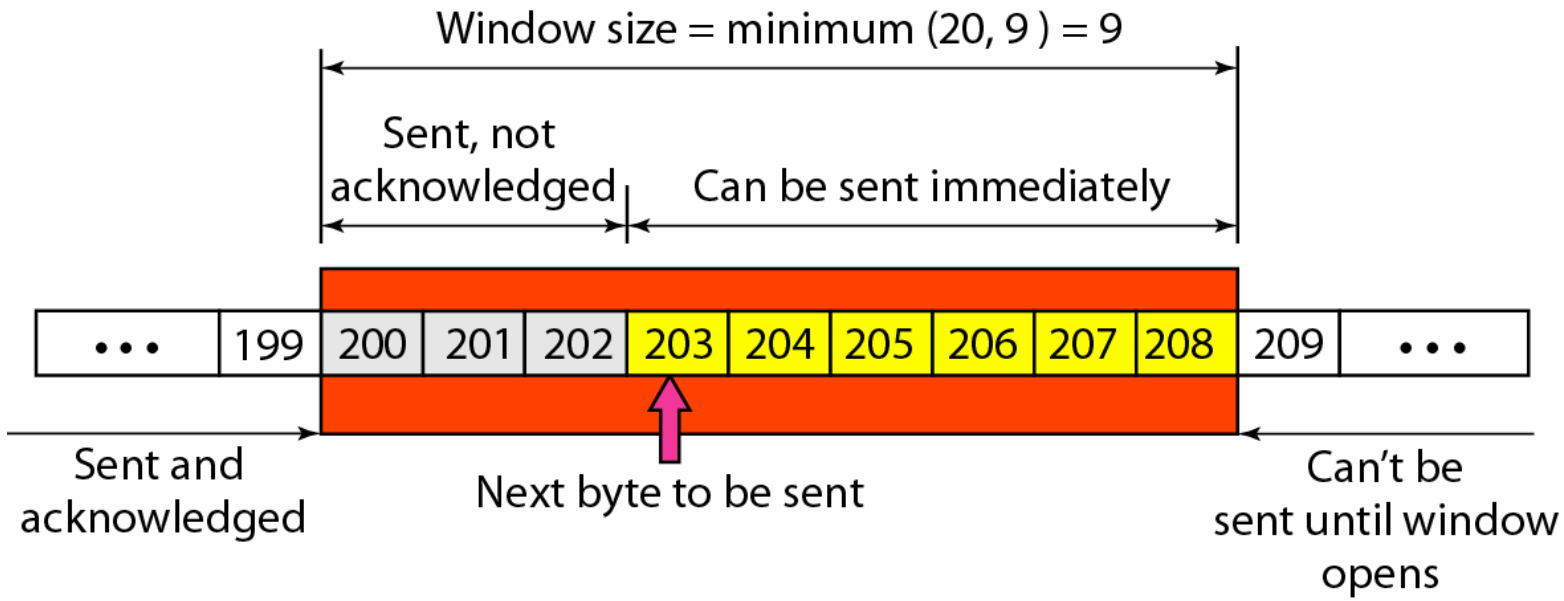
**The congestion window is a value determined by the network to avoid congestion.**



## *Example 4.6*

***Figure 4.23 shows an example of a sliding window. The sender has sent bytes up to 202. We assume that cwnd is 20. The receiver has sent an acknowledgment number of 200 with an rwnd of 9 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 4.23 *Example 4.6*



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*

**Error Control: Error detection and correction in TCP is achieved through the use of three simple tools: checksum, acknowledgment, and time-out.**

*Note*

**Checksum:**
**Each segment includes a checksum field which is used to check for a corrupted segment.**

*Note*

**Acknowledgment (ACK)**
**TCP uses acknowledgments to confirm the receipt of data segments. ACK segments do not consume sequence numbers and are not acknowledged.**

*Note*

**Retransmission:**
**The heart of the error control mechanism is the retransmission of segments.**

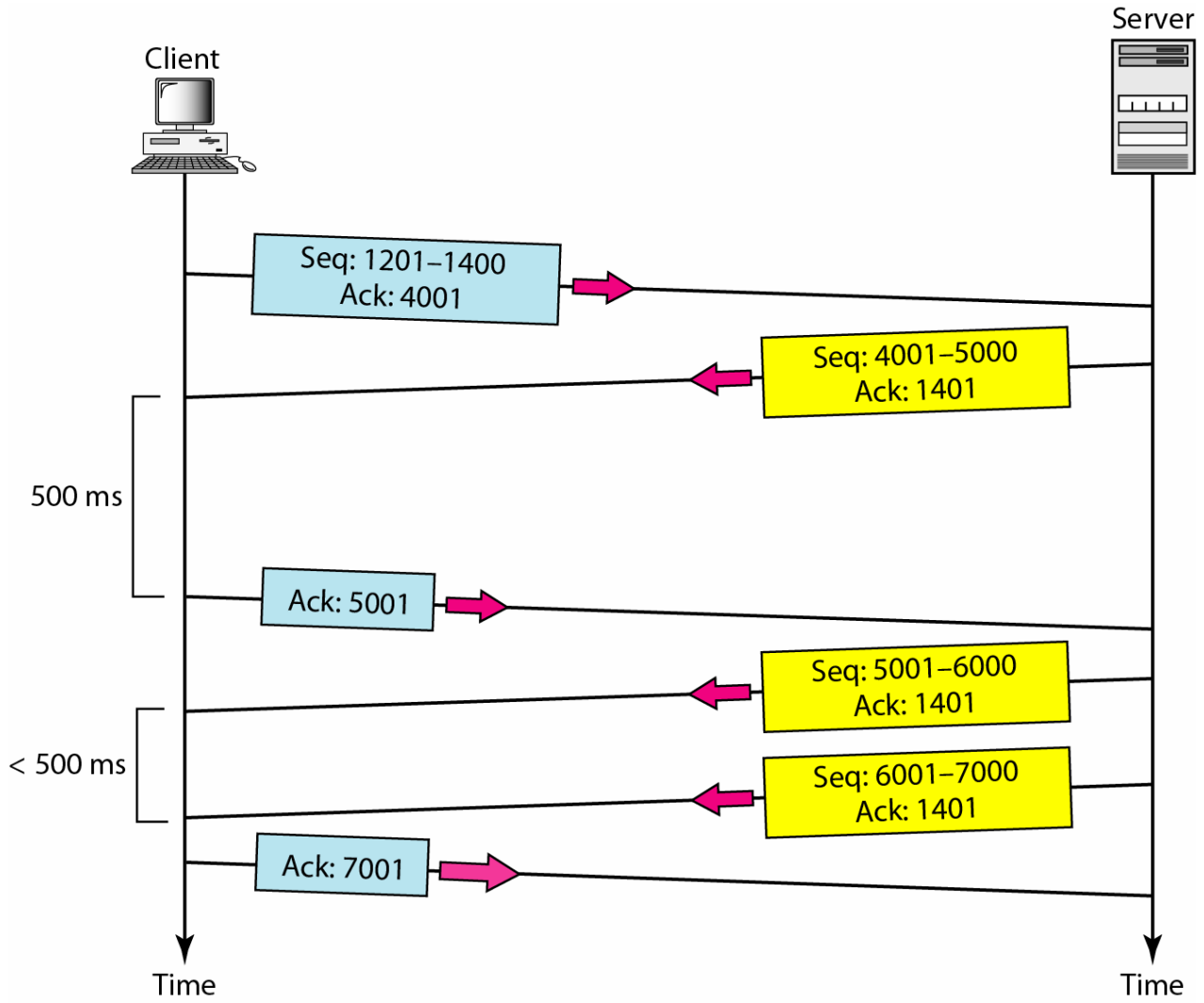
*Note*

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

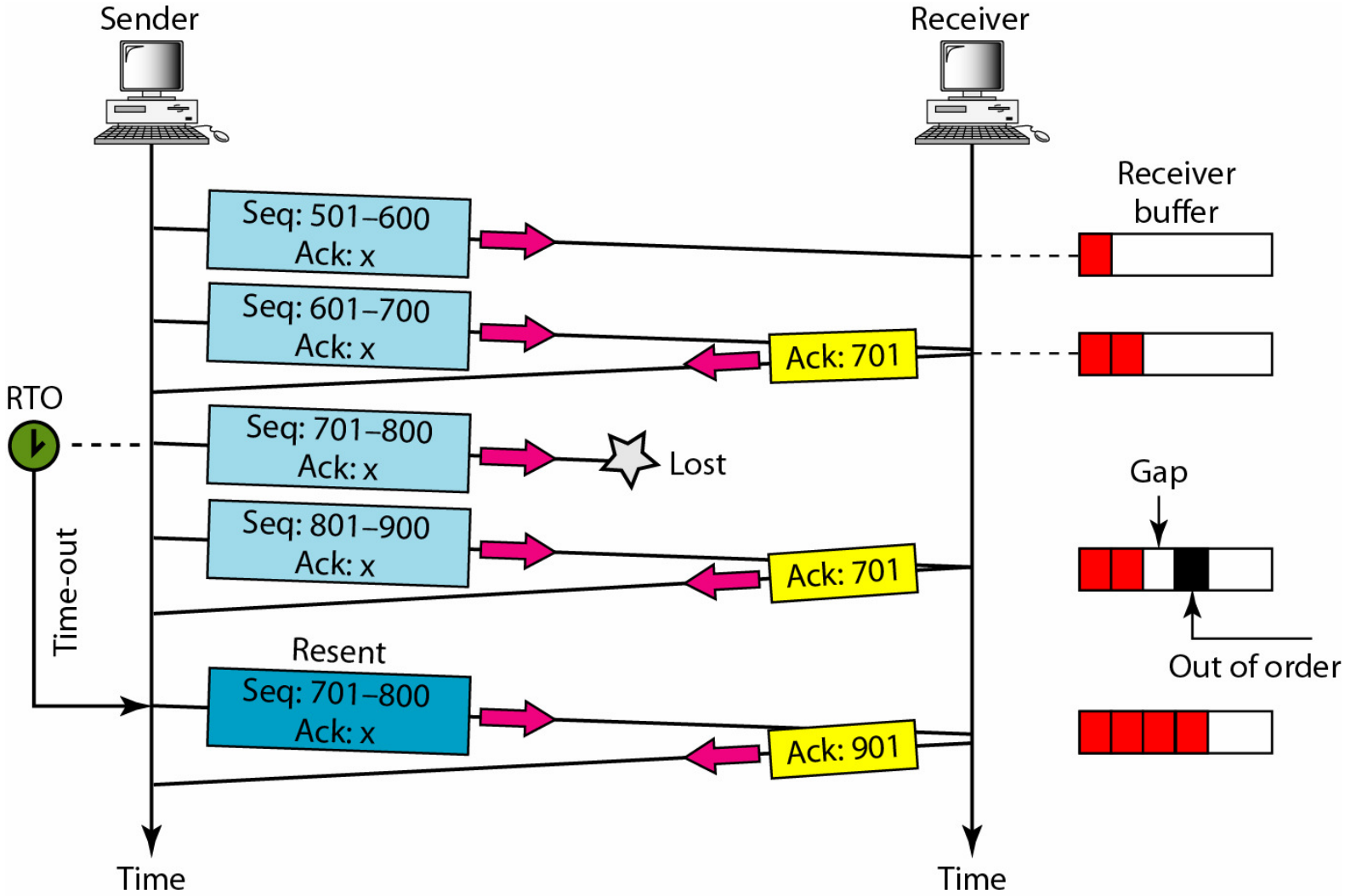
*Note*

**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 4.24 *Normal operation*



## Figure 4.25 *Lost segment*



*Note*

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

## Figure 4.26 *Fast retransmission*