Data Communications and Networking Fourth Edition

Forouzan

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*

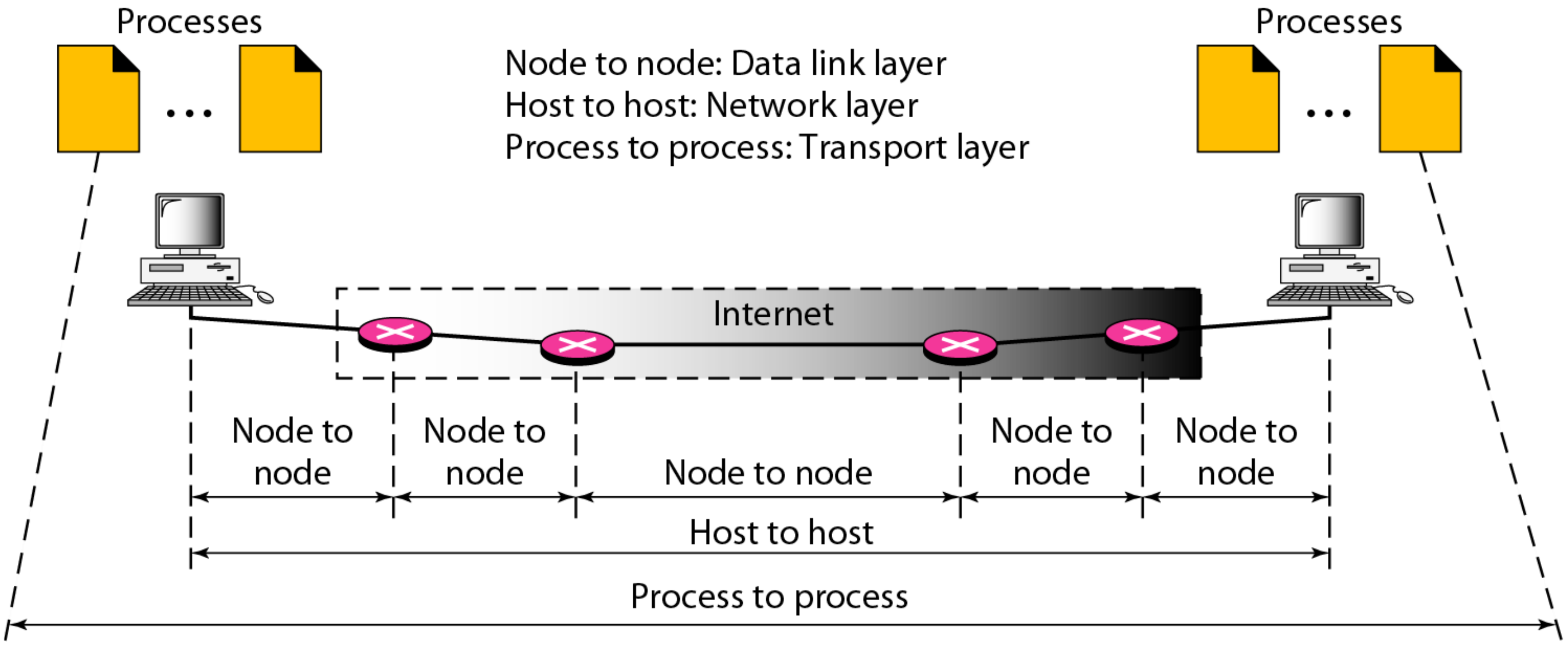


Figure 23.2 *Port numbers*

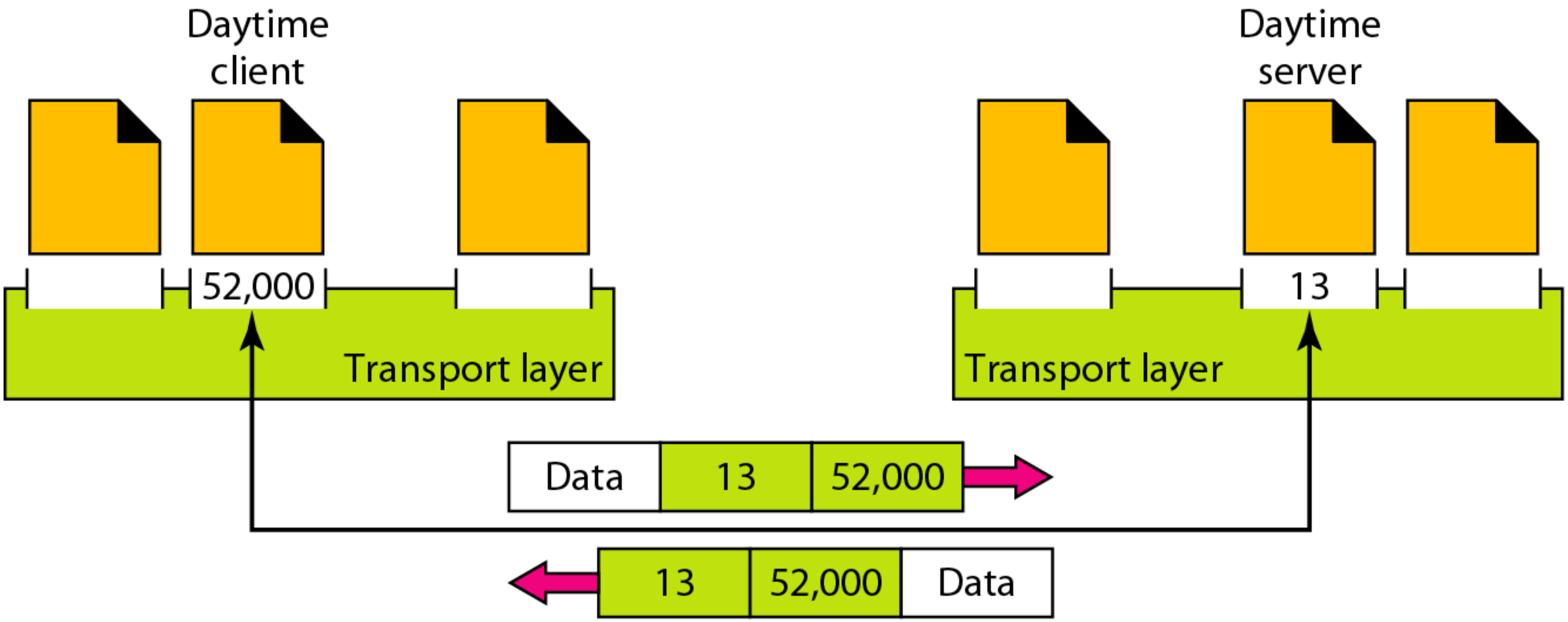


Figure 23.3 *IP addresses versus port numbers*

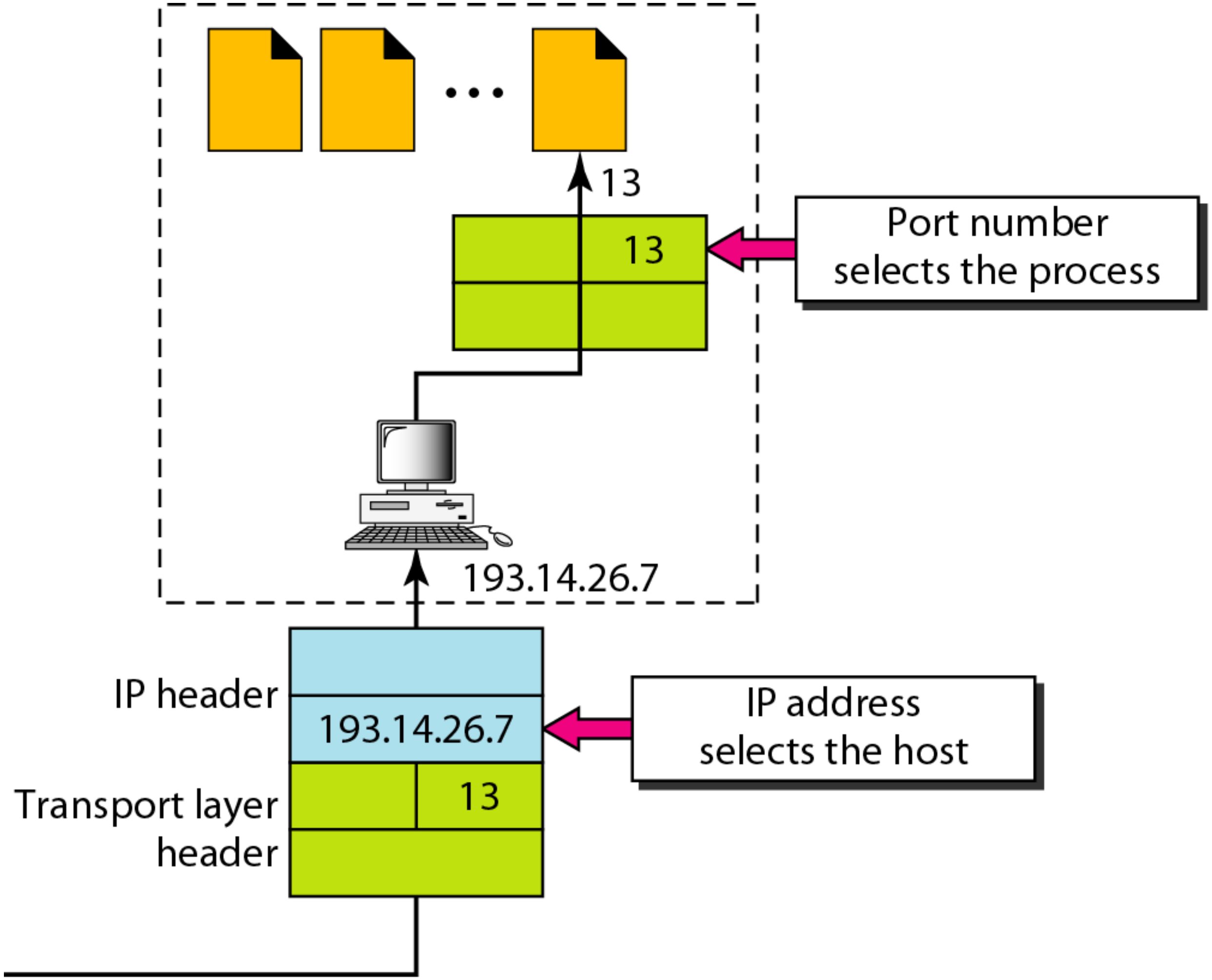


Figure 23.4 *IANA ranges*

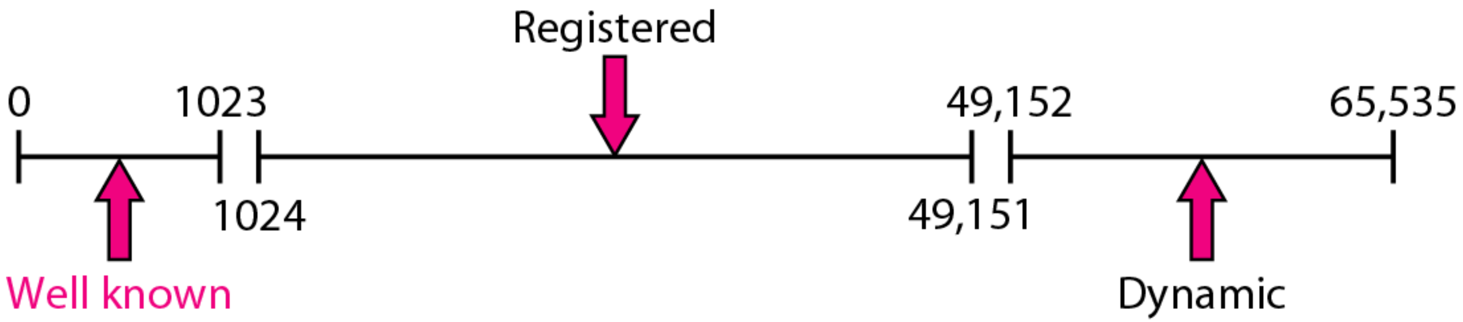


Figure 23.5 *Socket address*

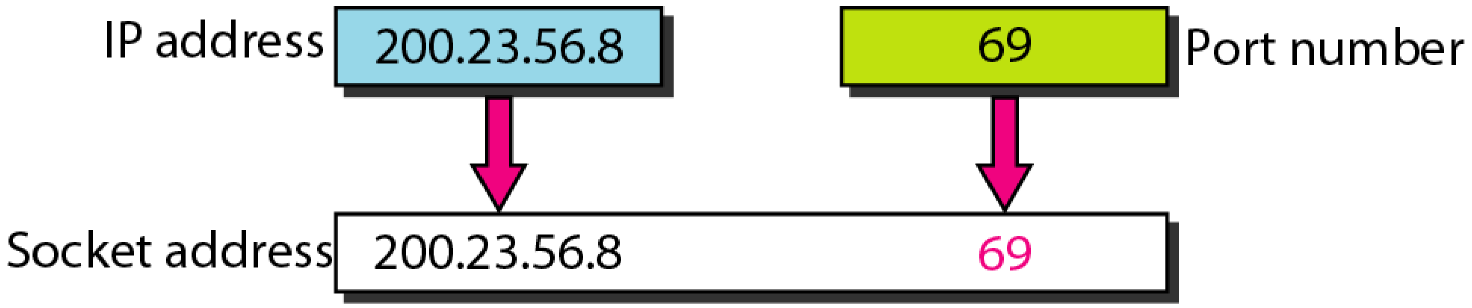


Figure 23.6 *Multiplexing and demultiplexing*

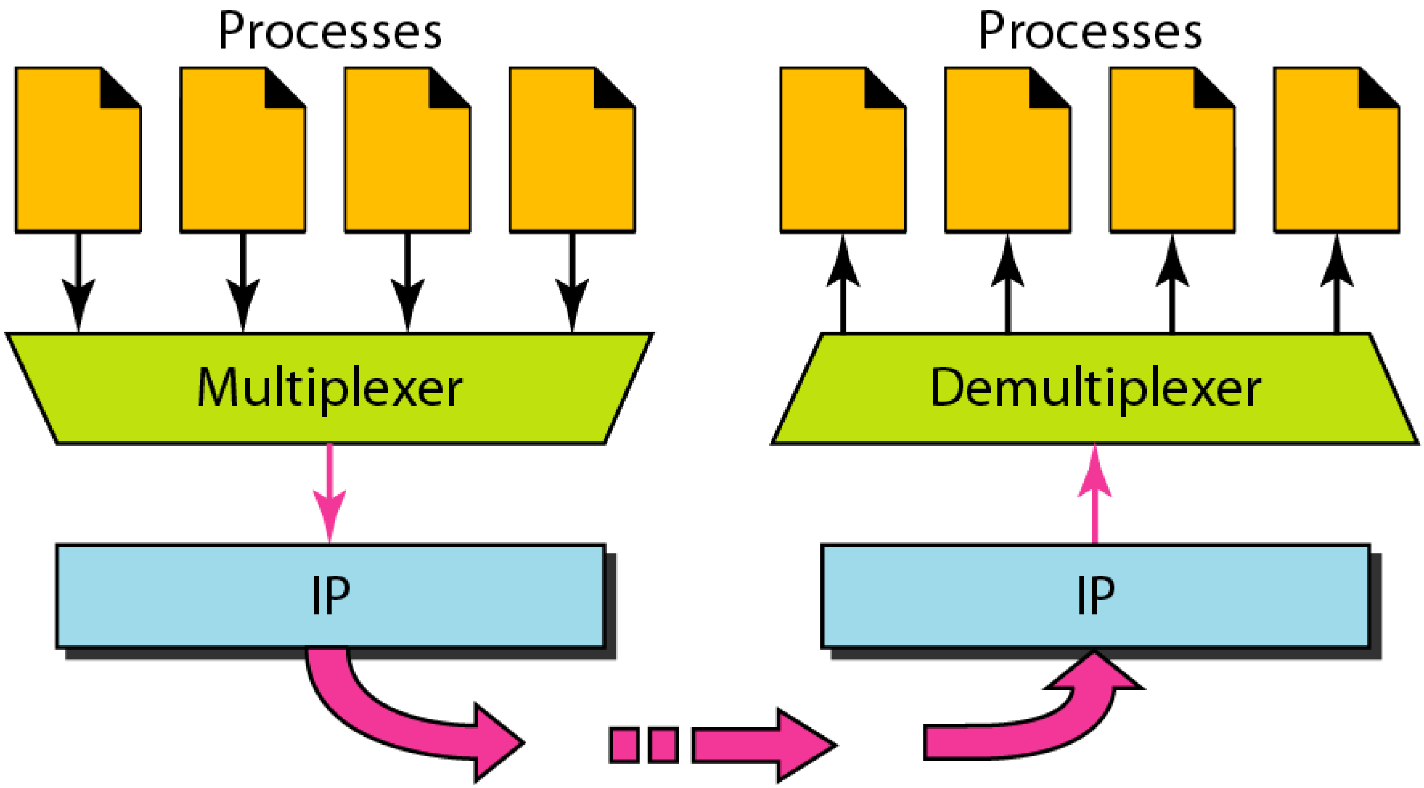


Figure 23.7 *Error control*

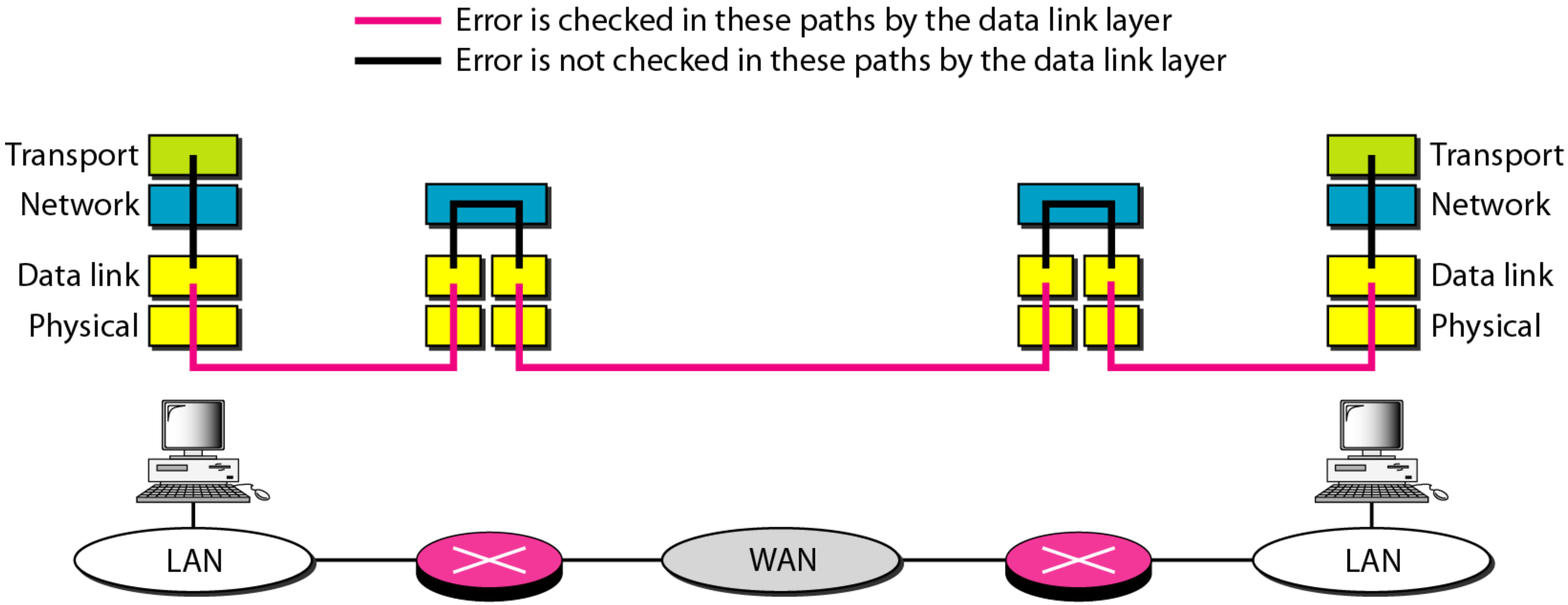
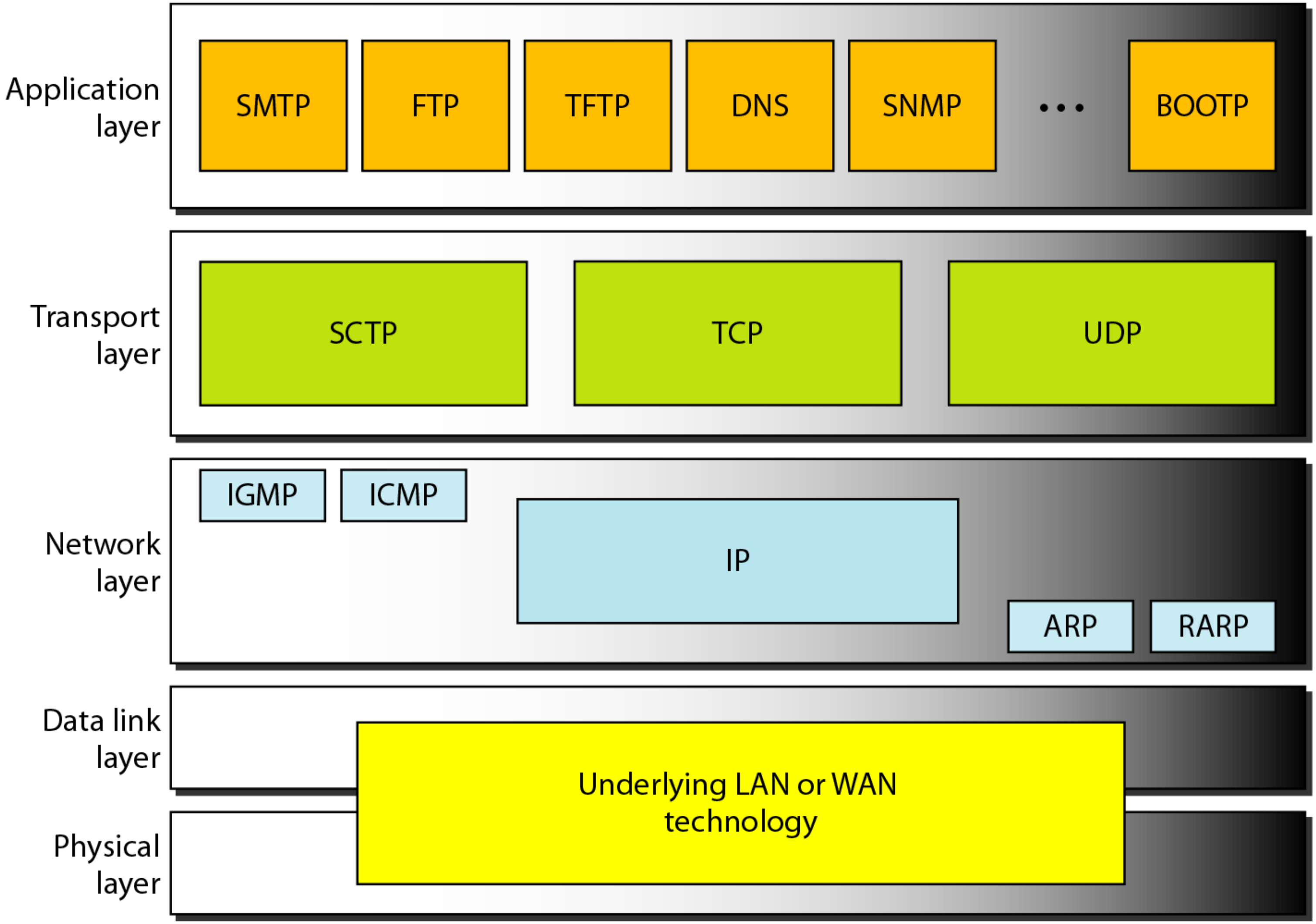

Error is checked in these paths by the data link layer
Error is not checked in these paths by the data link layer
Transport
Network
Data link
Physical
Transport
Network
Data link
Physical
LAN
WAN
LAN

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



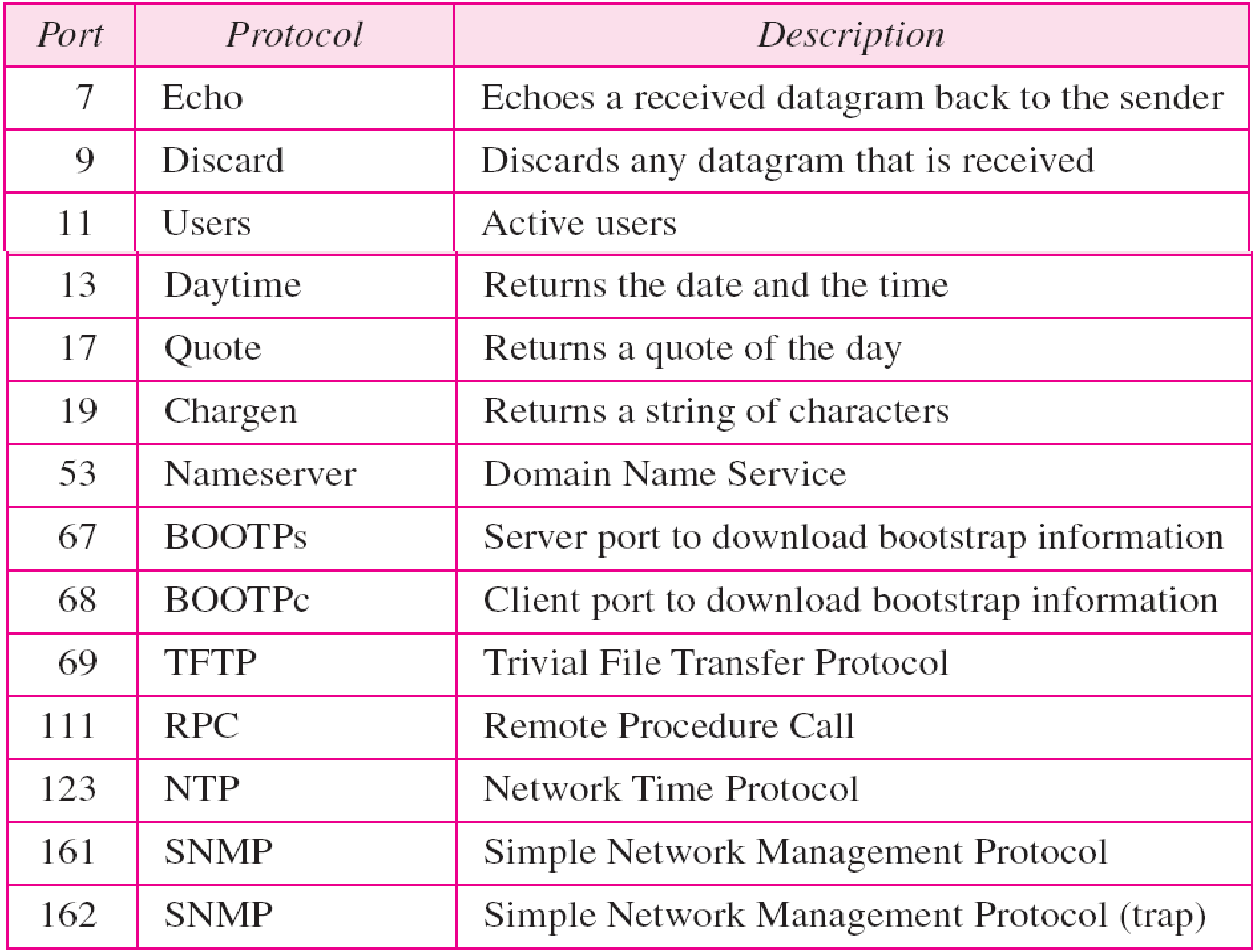
23-2 USER DATAGRAM PROTOCOL (UDP)

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:

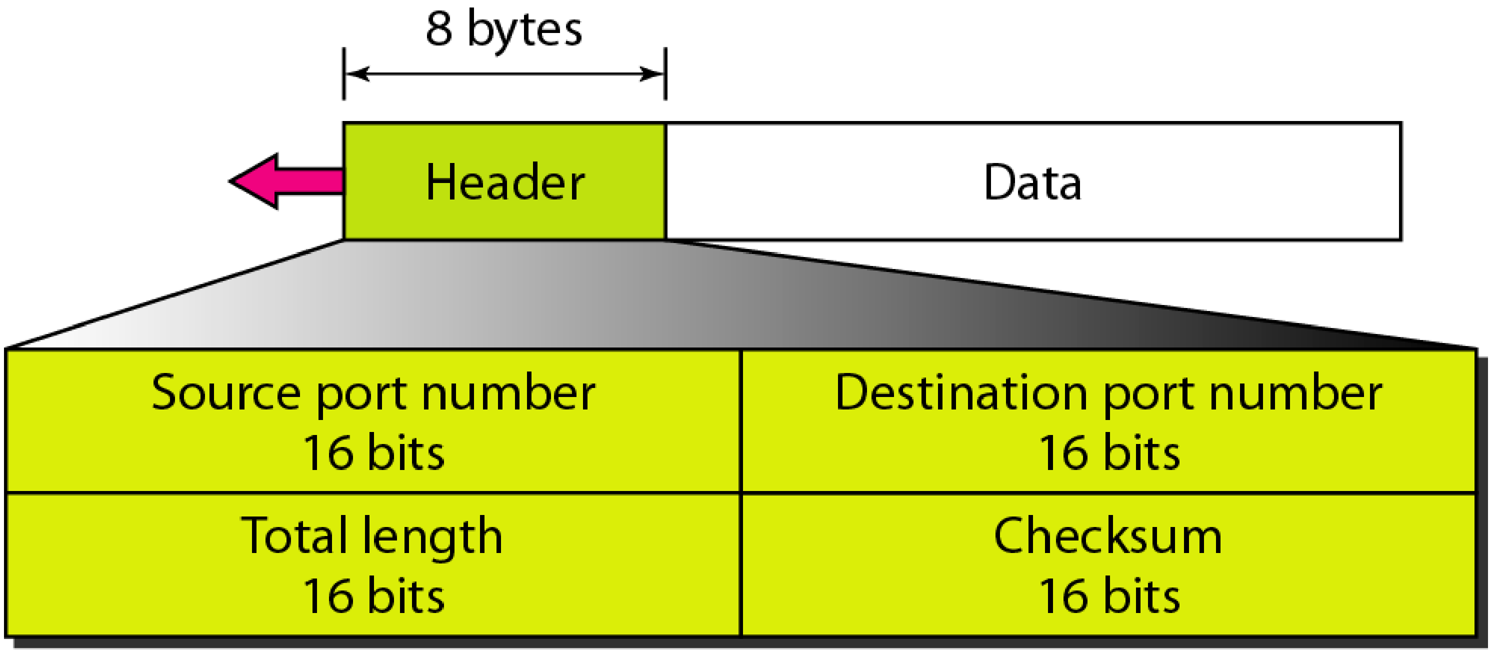
Well-Known Ports for UDP
User Datagram
Checksum
UDP Operation
Use of UDP

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)

Figure 23.9 *User datagram format*



8 bytes

Header

Data

Source port number 16 bits	Destination port number 16 bits
Total length 16 bits	Checksum 16 bits

Note

UDP length
= IP length – IP header’s length

Use of UDP

UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control. It is not usually used for a process that needs to send bulk data.

UDP is suitable for a process with internal flow and error control mechanisms.

UDP is a suitable transport protocol for multicasting. Multicasting capability is embedded in the UDP software but not in the TCP software.

UDP is used for management processes such as SNMP

UDP is used for some route updating protocols such as Routing Information Protocol.

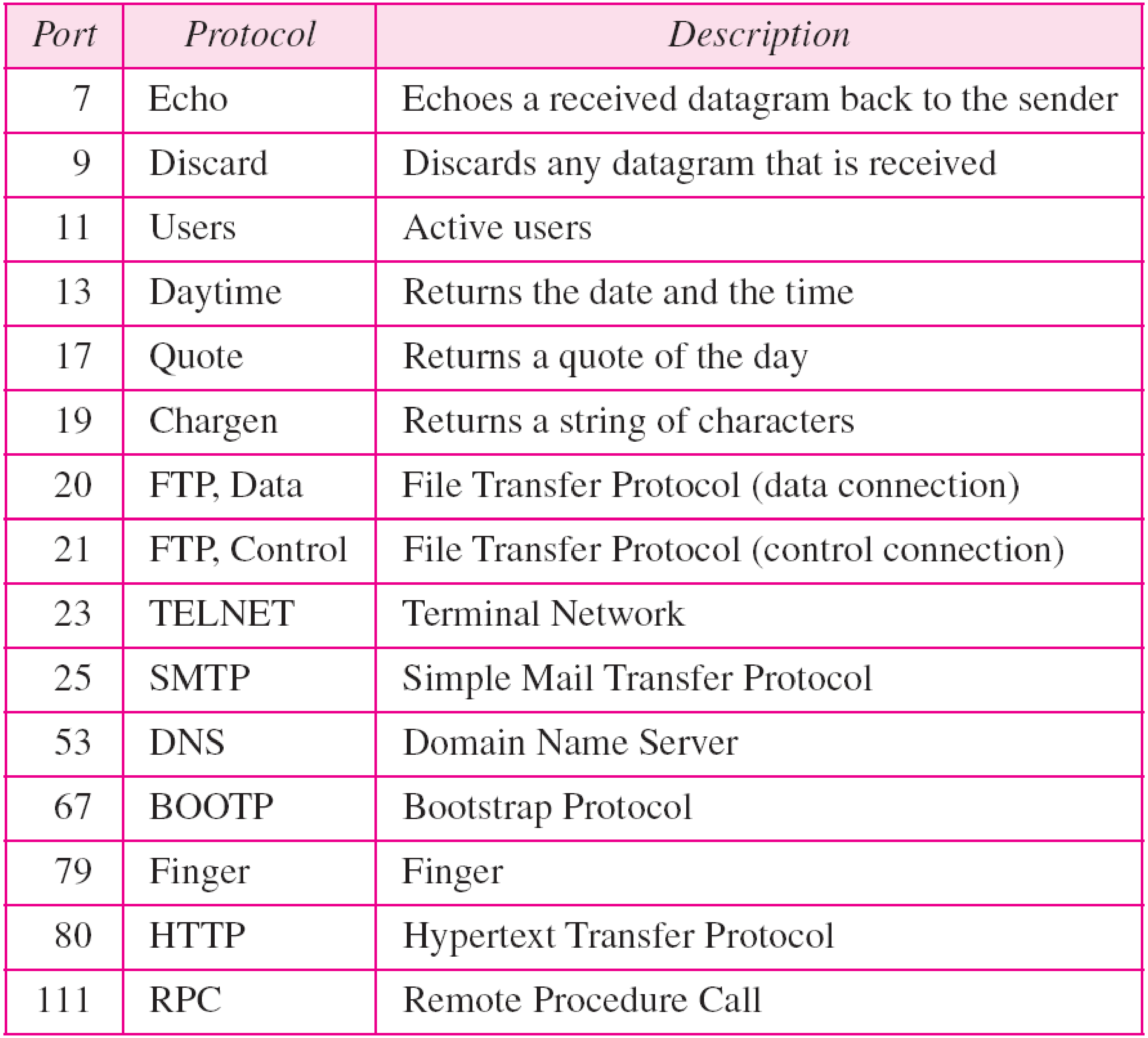
23-3 TCP

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*

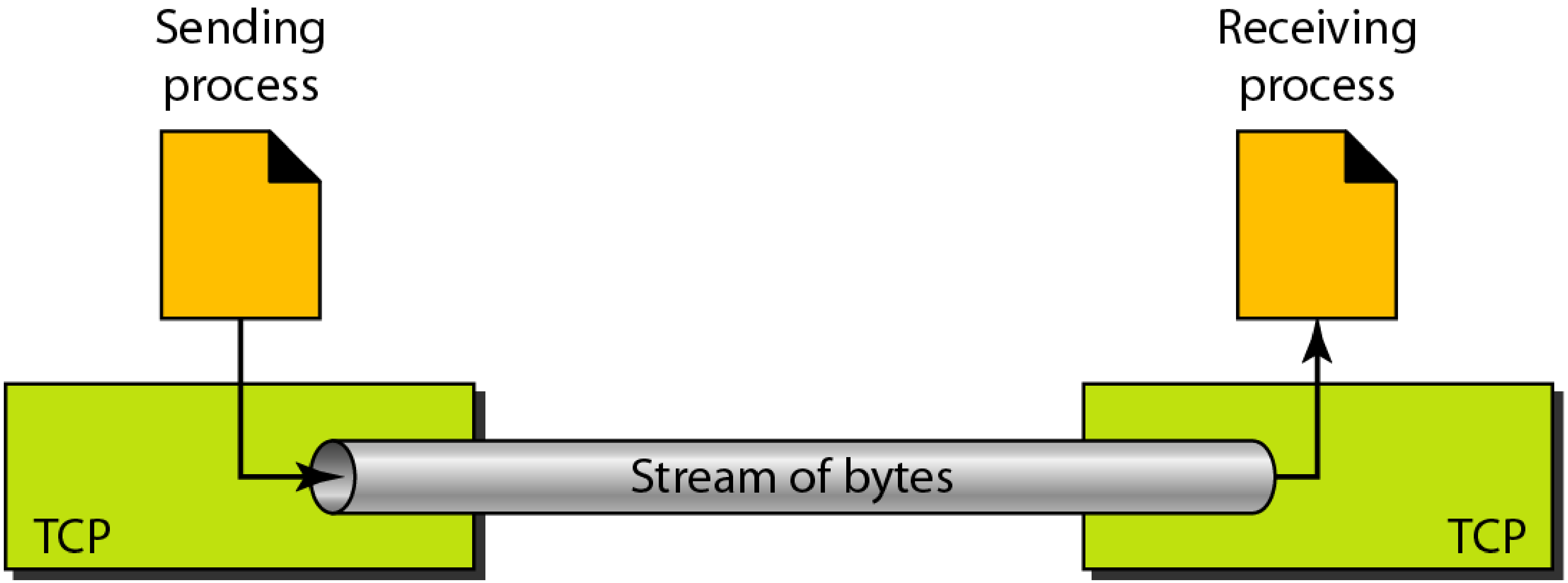


Figure 23.14 *Sending and receiving buffers*

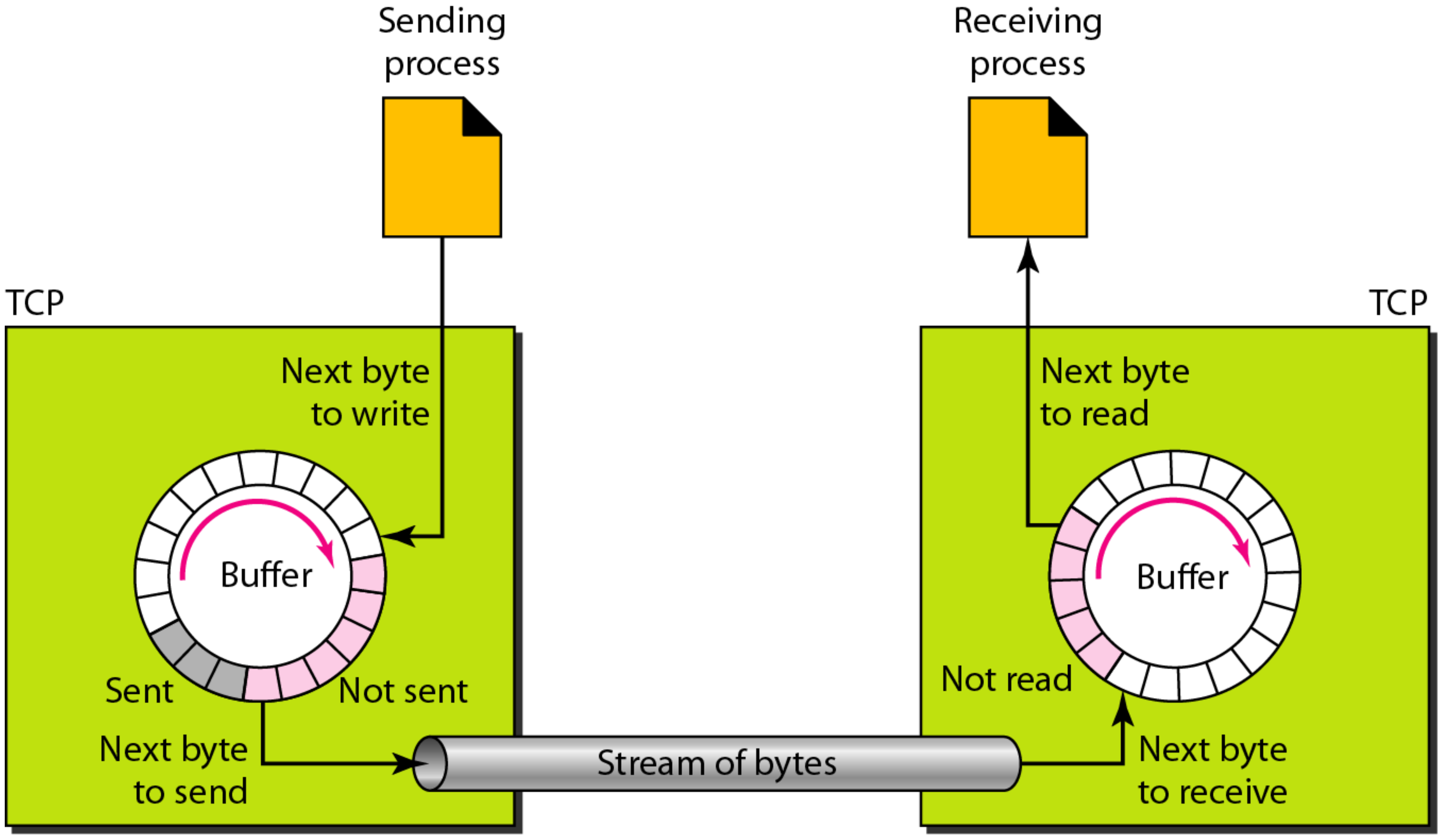
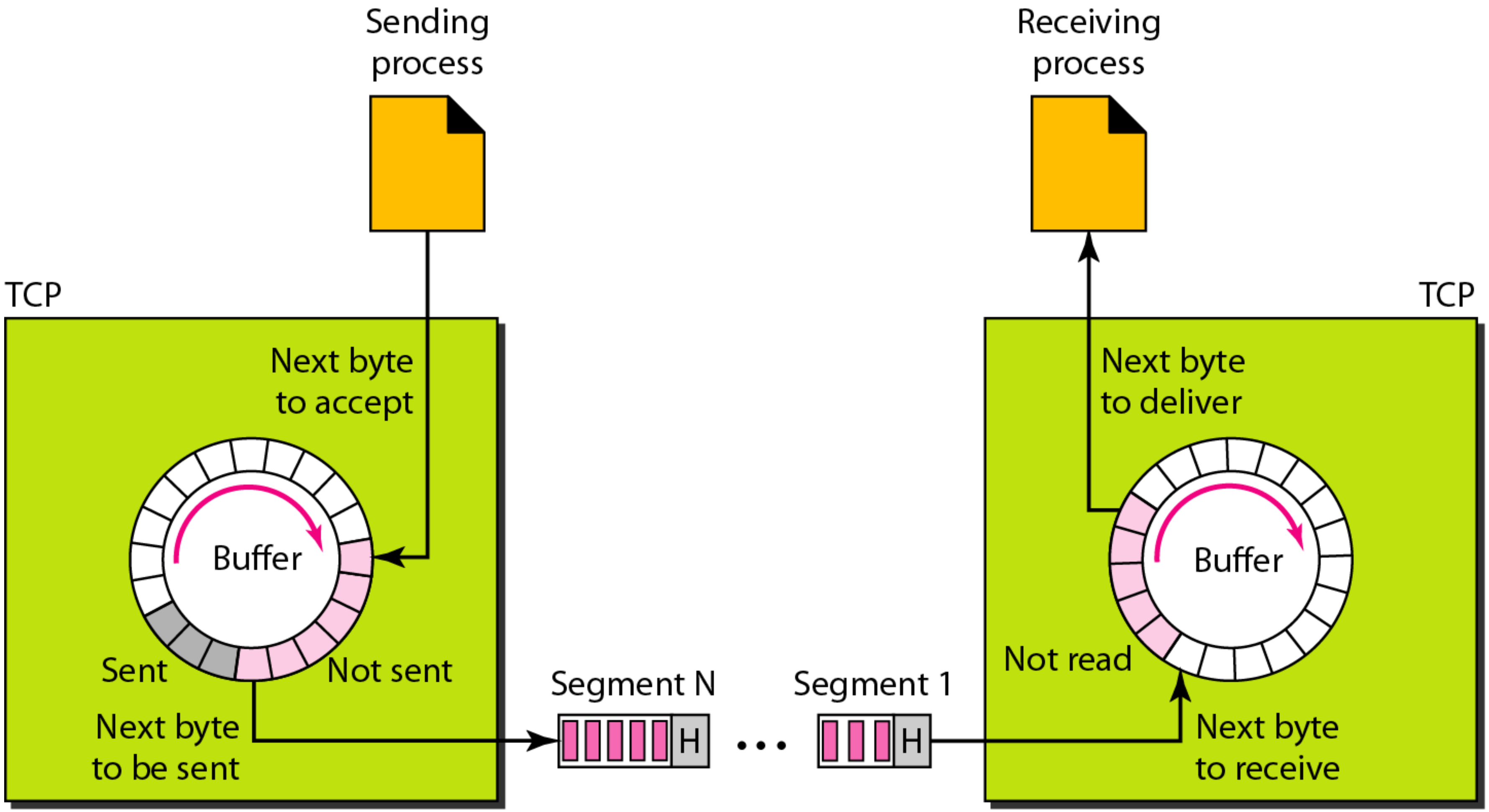


Figure 23.15 *TCP segments*

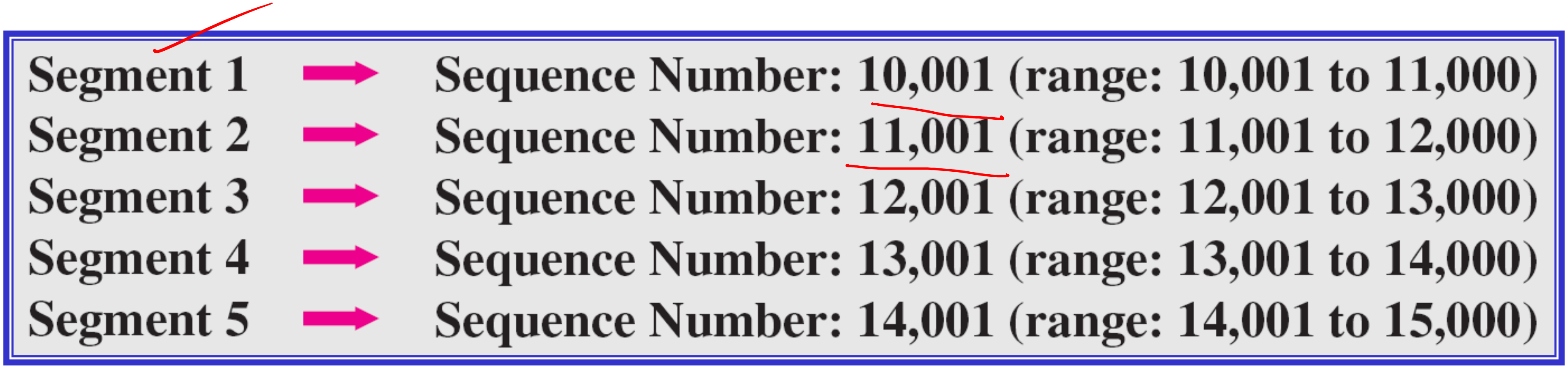


Note

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)

Segment1 → 10,001 Size of segment is 1000 bytes

segment 2 → 11,001 1500 ;

segment 3 → 125001 ✓

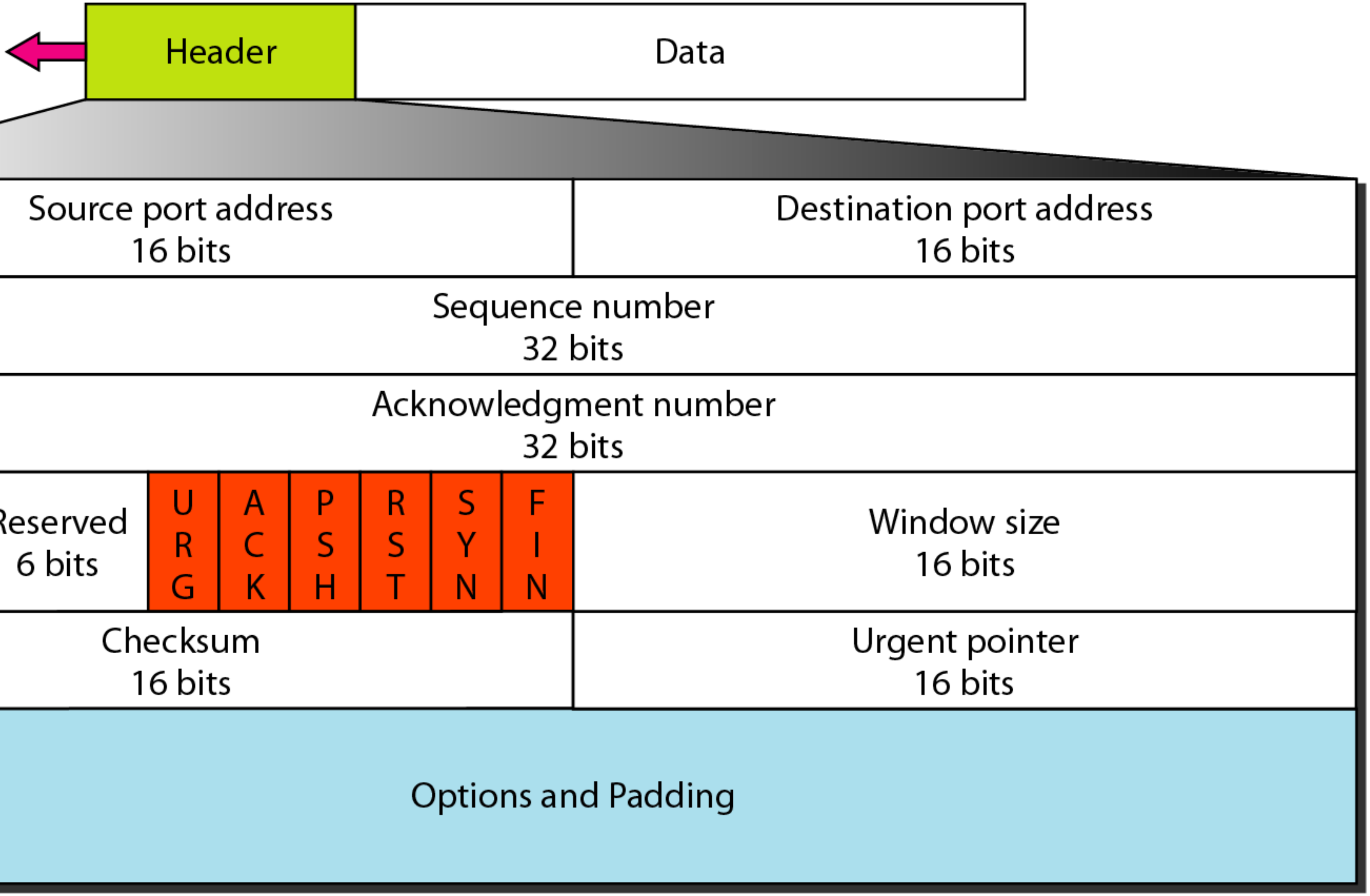
Note

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*



Header

Data

Source port address
16 bits

Destination port address
16 bits

Sequence number
32 bits

Acknowledgment number
32 bits

HLEN
4 bits

Reserved
6 bits

URG

ACK

PSH

RST

SYN

FIN

Window size
16 bits

Checksum
16 bits

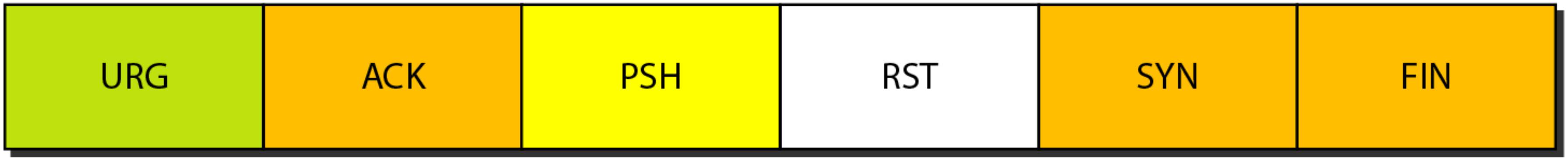
Urgent pointer
16 bits

Options and Padding

Figure 23.17 *Control field*

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

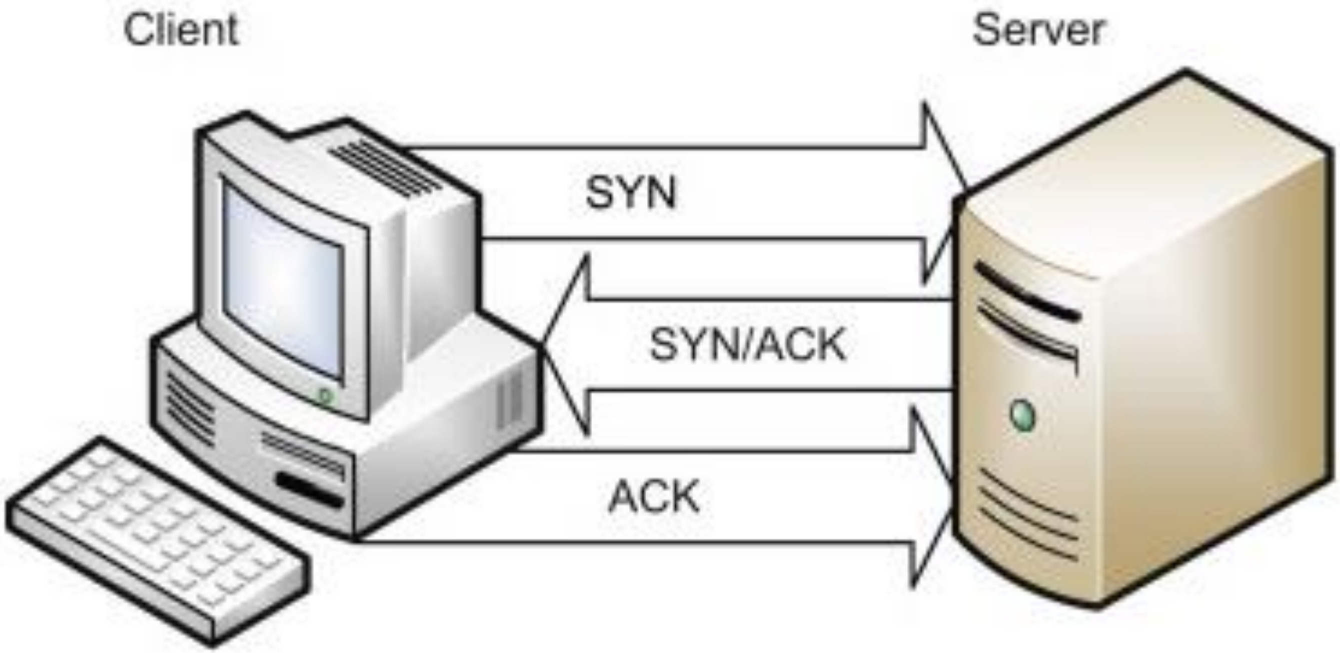
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection



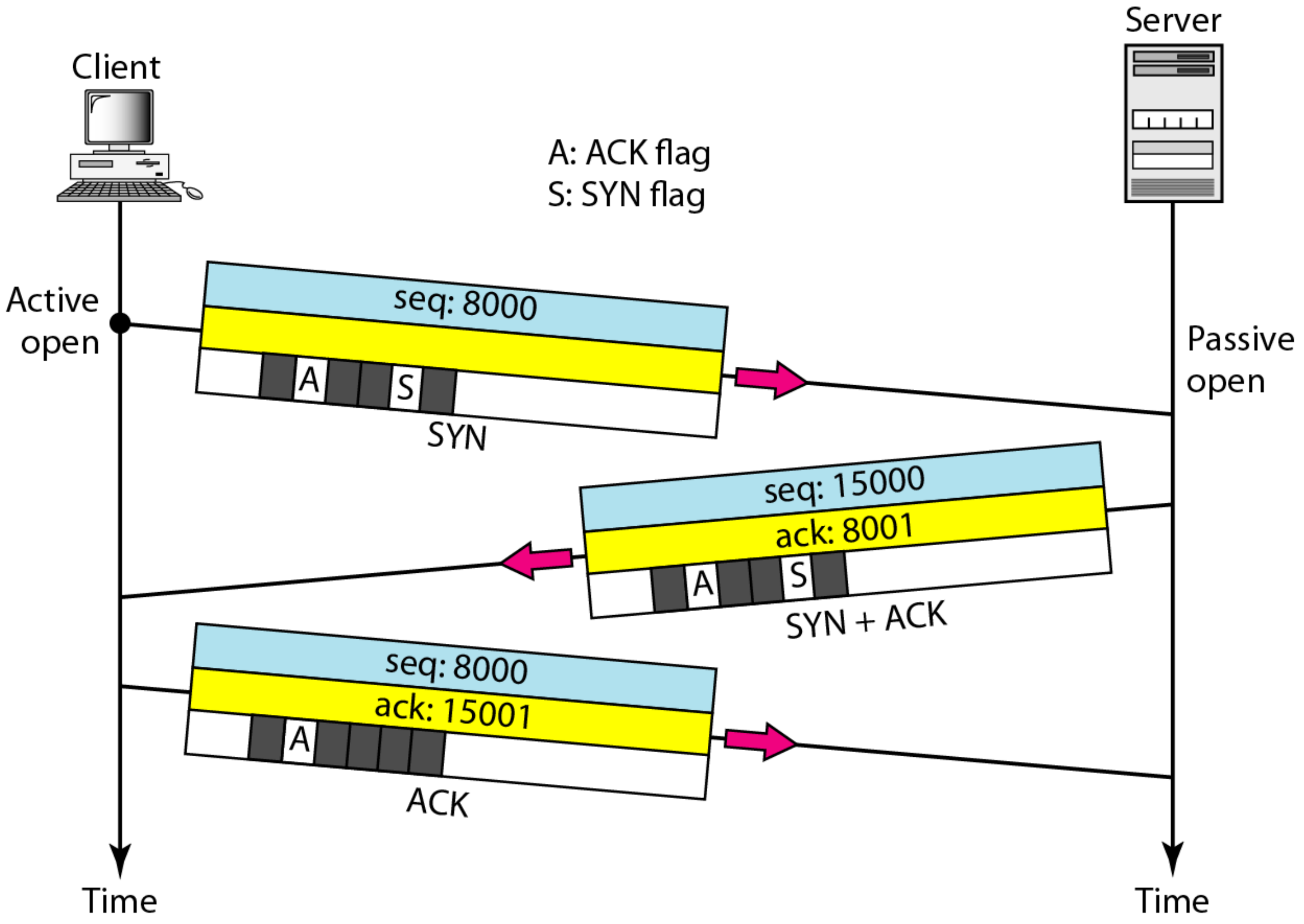
URG ACK PSH RST SYN FIN

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.

Three way handshake in TCP



Connection establishment using three-way handshaking



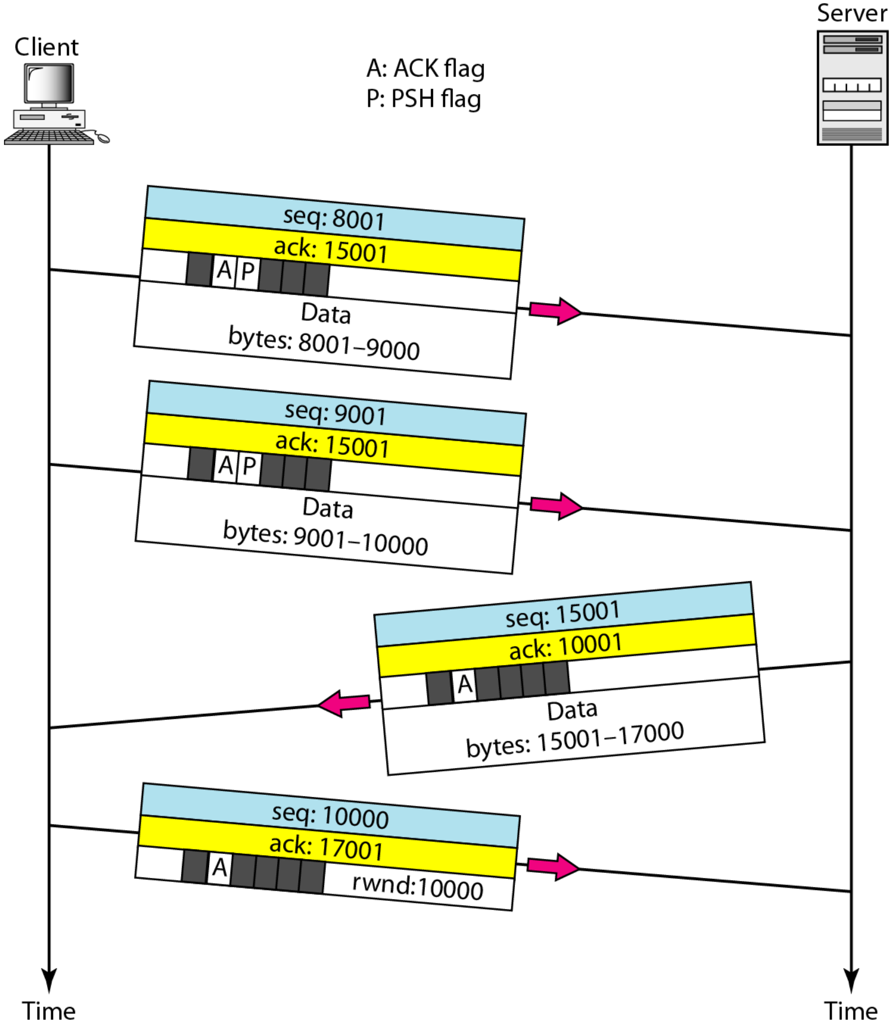
Note

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

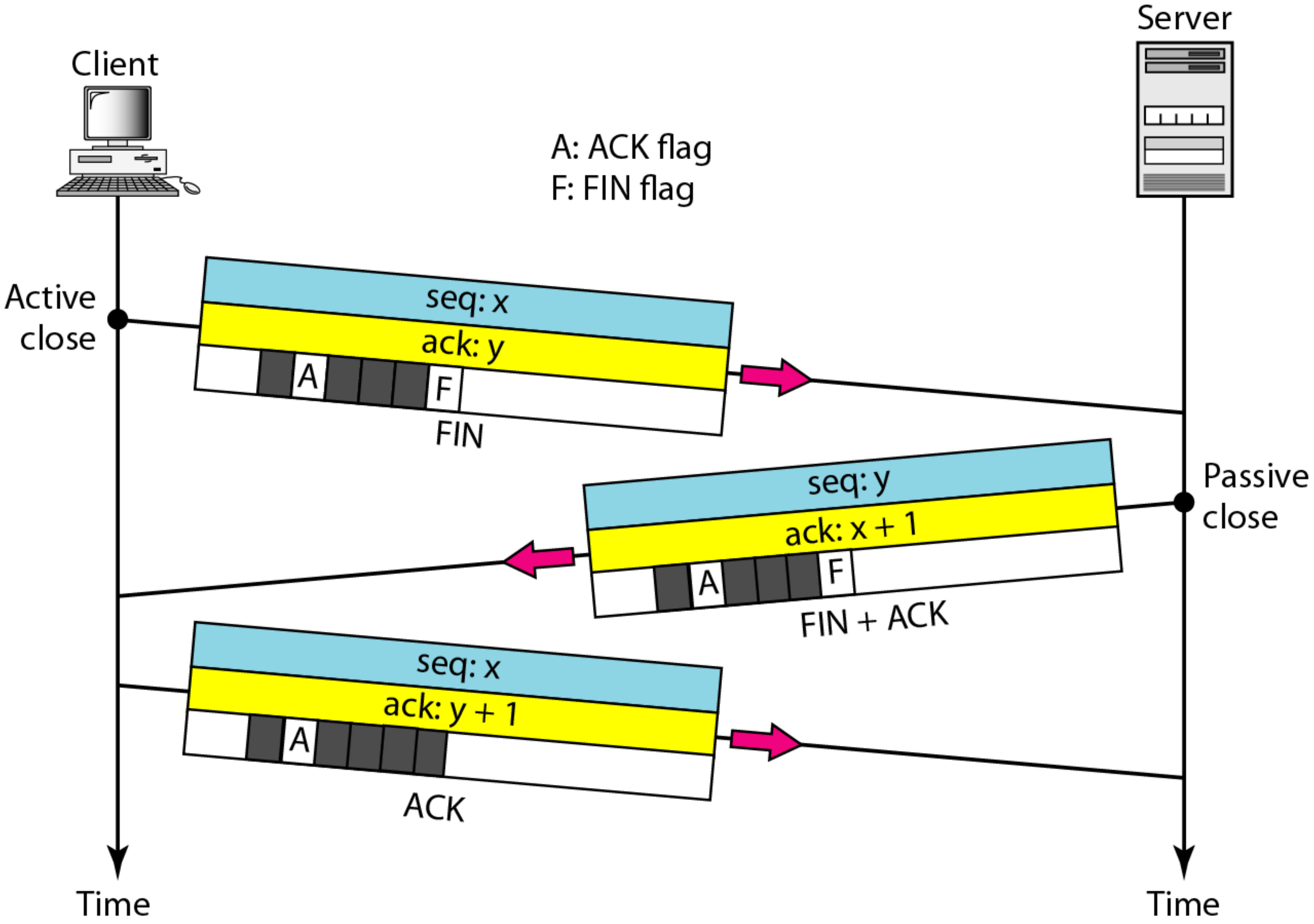
A SYN + ACK segment cannot carry data, but does consume one sequence number.

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

Data transfer



Connection termination using three-way handshakin


Client
Server
A: ACK flag
F: FIN flag
Active close
seq: x
ack: y
A
F
FIN
Passive close
seq: y
ack: x + 1
A
F
FIN + ACK
seq: x
ack: y + 1
A
ACK
Time
Time