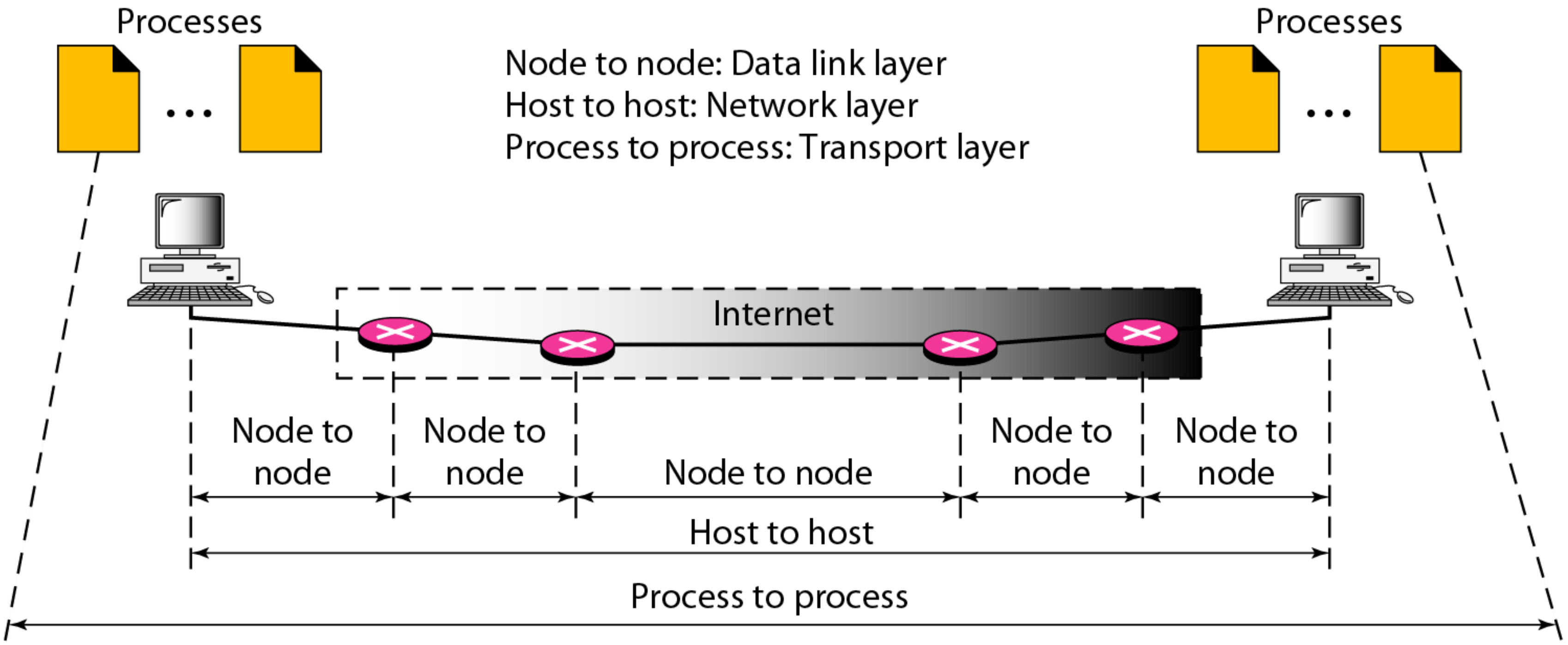
PROCESS-TO-PROCESS DELIVERY

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.

Note

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

Figure 23.1 *Types of data deliveries*



- lANA Ranges
- The lANA (Internet Assigned Number Authority) has divided the port numbers into
- three ranges: well known, registered, and dynamic (or private)
- o Well-known ports.

- The ports ranging from 0 to 1023 are assigned and controlled
- by lANA. These are the well-known ports.

- Registered ports. The ports ranging from 1024 to 49,151 are not assigned or controlled
- by lANA.

- Dynamic ports. The ports ranging from 49,152 to 65,535 are neither controlled
- nor registered. They can be used by any process. These are the ephemeral ports

Figure 23.2 *Port numbers*

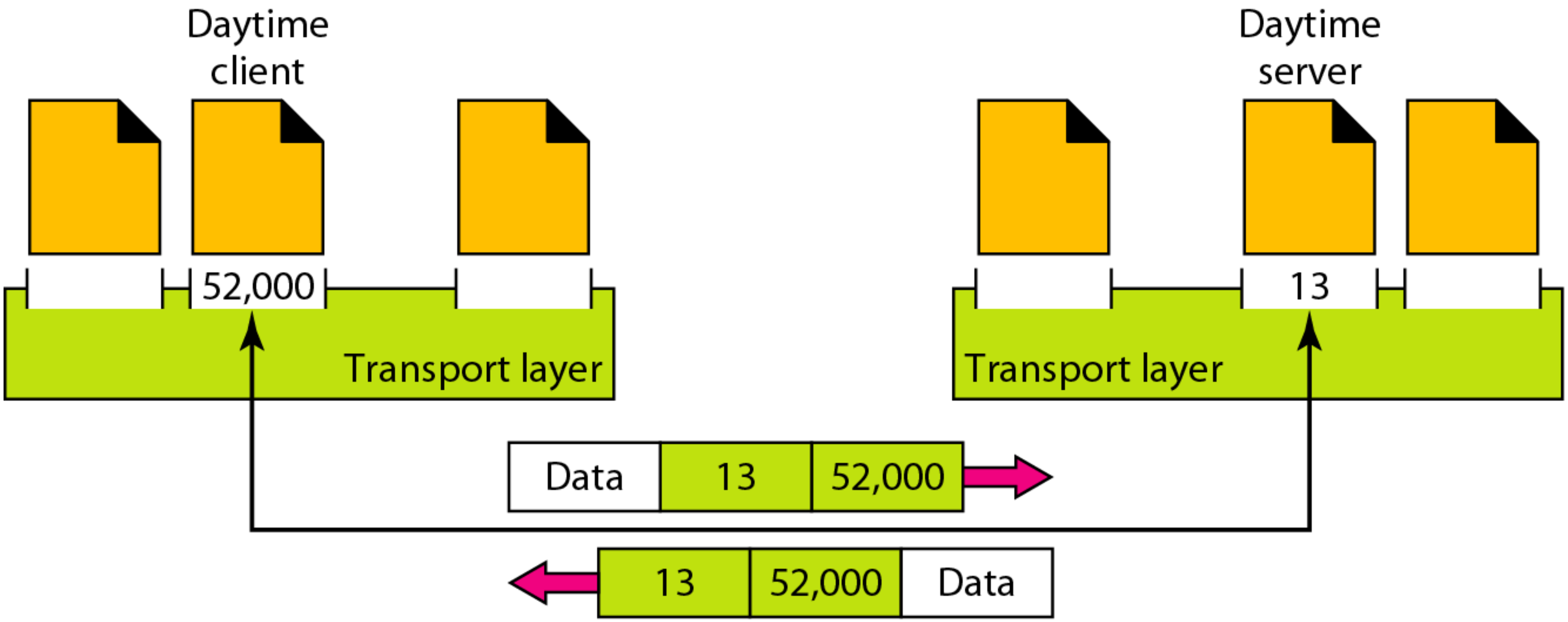


Figure 23.3 *IP addresses versus port numbers*

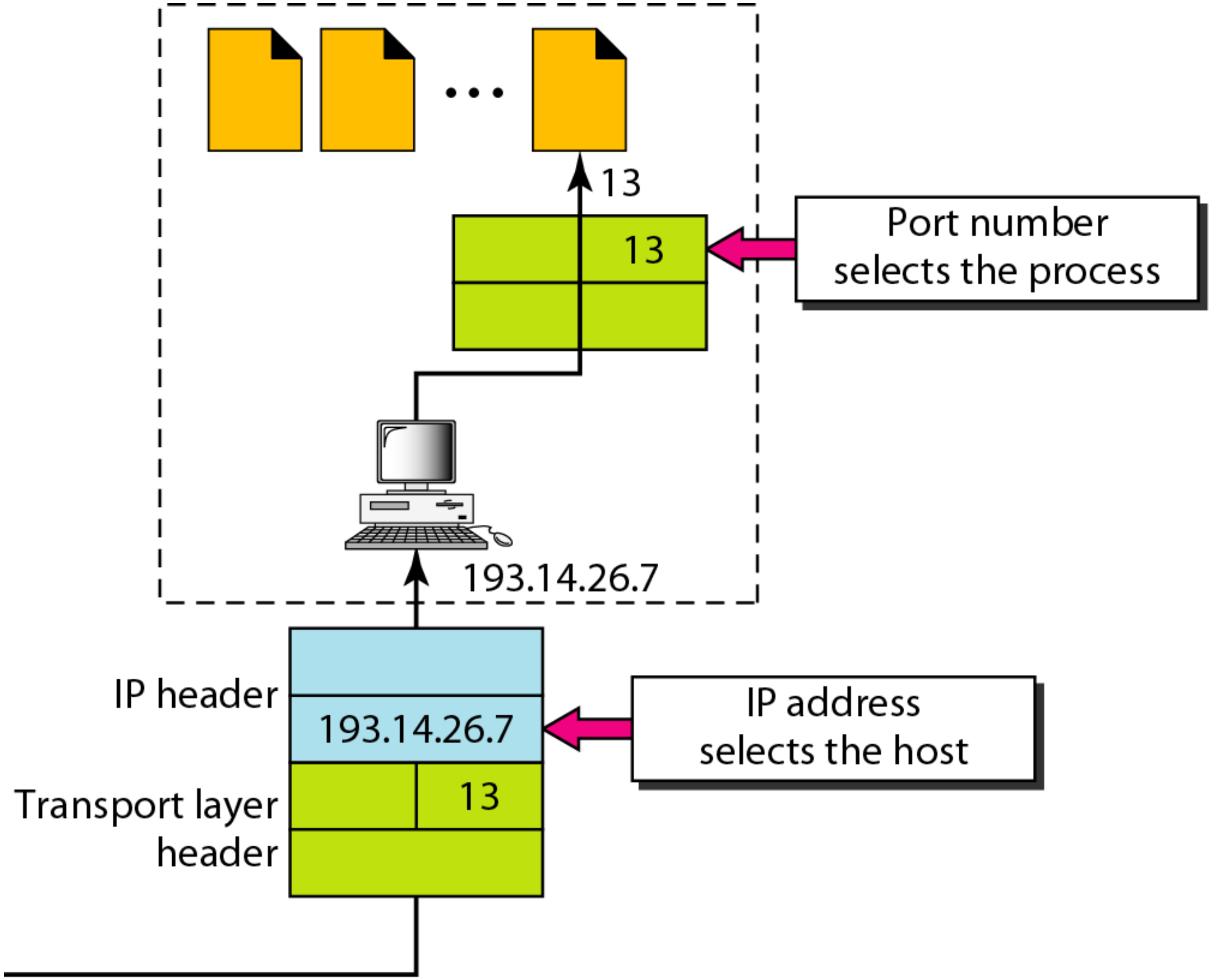
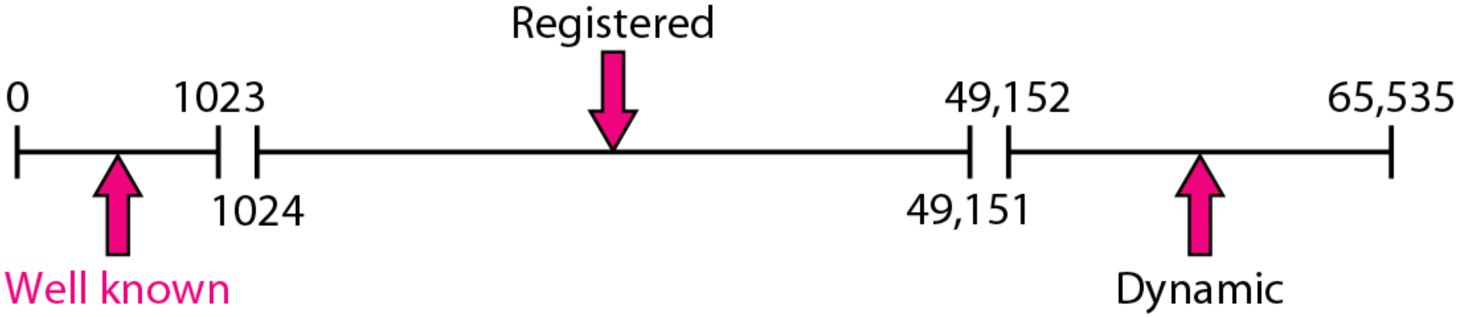
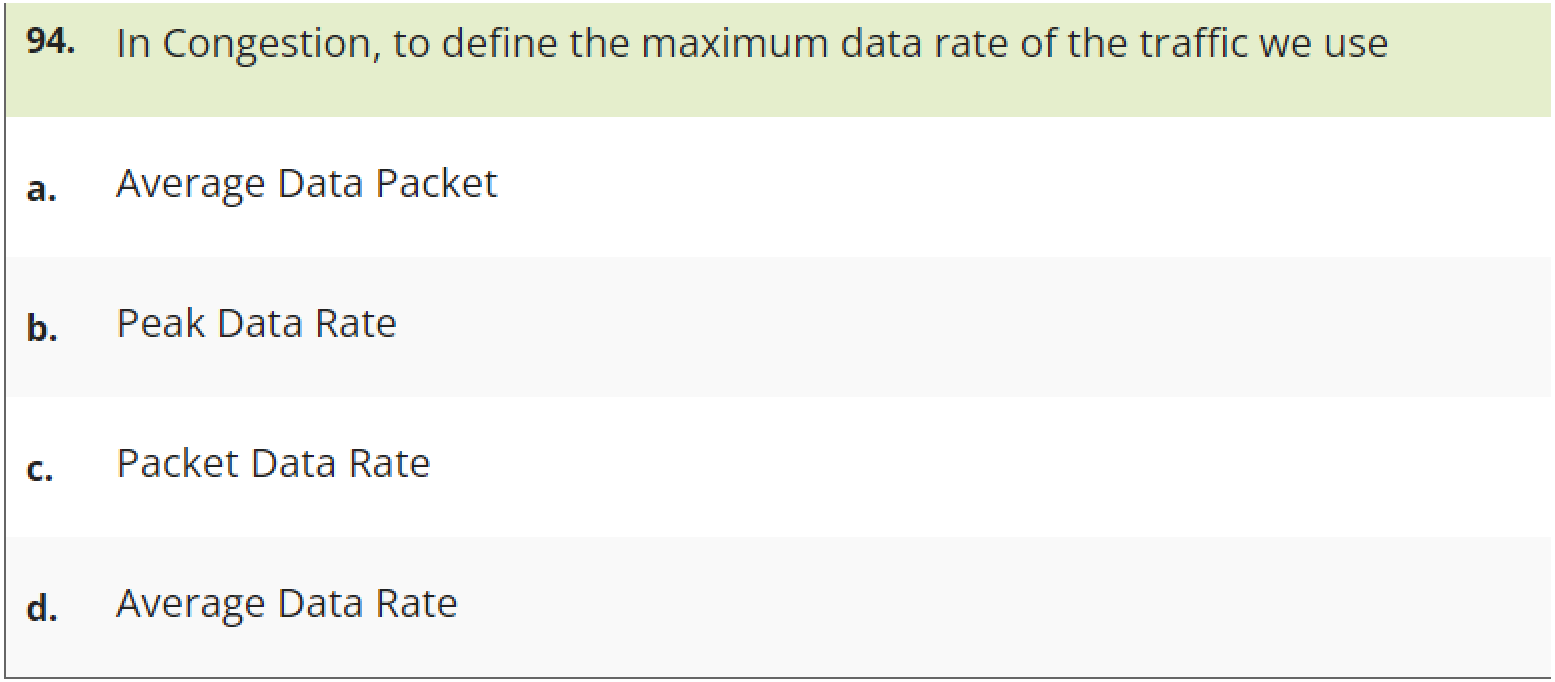


Figure 23.4 *IANA ranges*





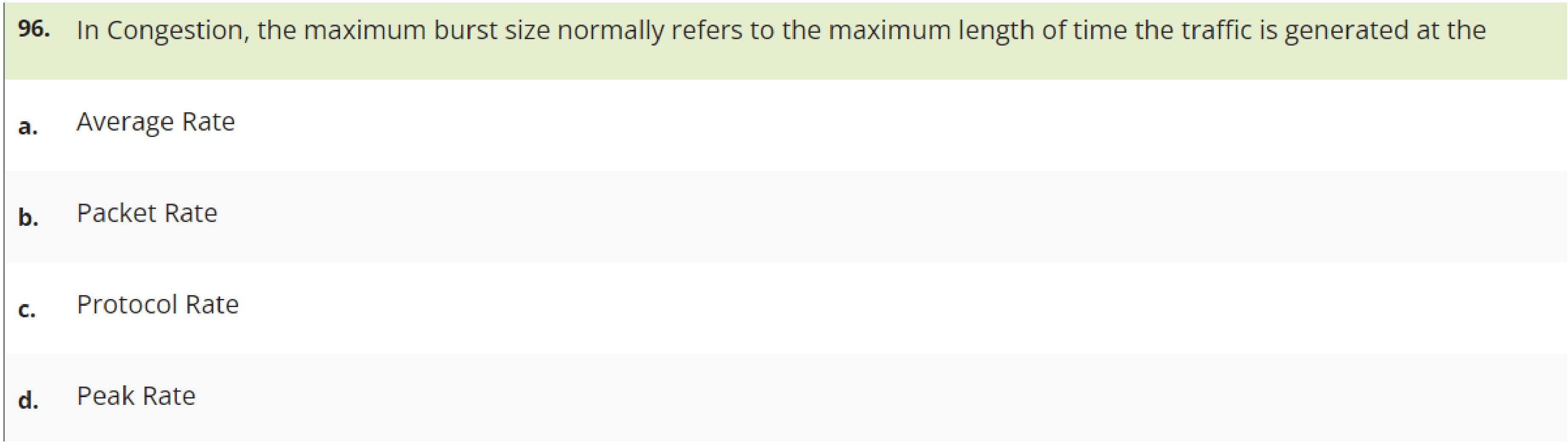
94. In Congestion, to define the maximum data rate of the traffic we use

a. Average Data Packet

b. Peak Data Rate

c. Packet Data Rate

d. Average Data Rate



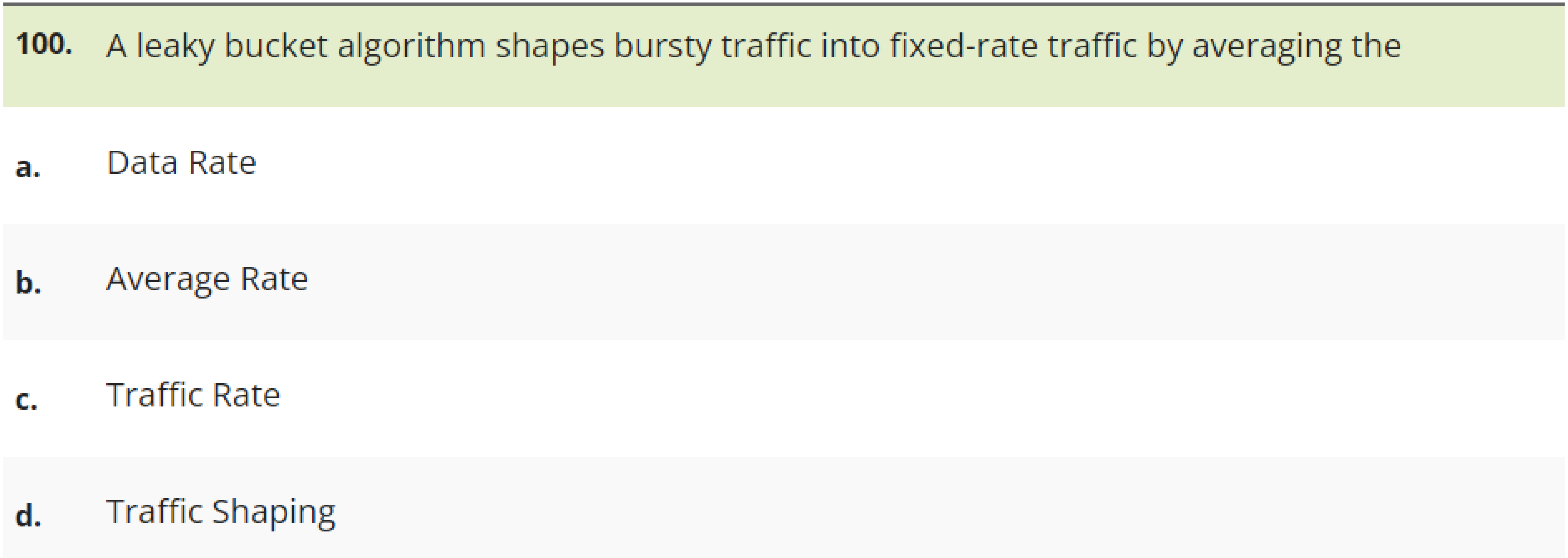
96. In Congestion, the maximum burst size normally refers to the maximum length of time the traffic is generated at the

a. Average Rate

b. Packet Rate

c. Protocol Rate

d. Peak Rate



100. A leaky bucket algorithm shapes bursty traffic into fixed-rate traffic by averaging the

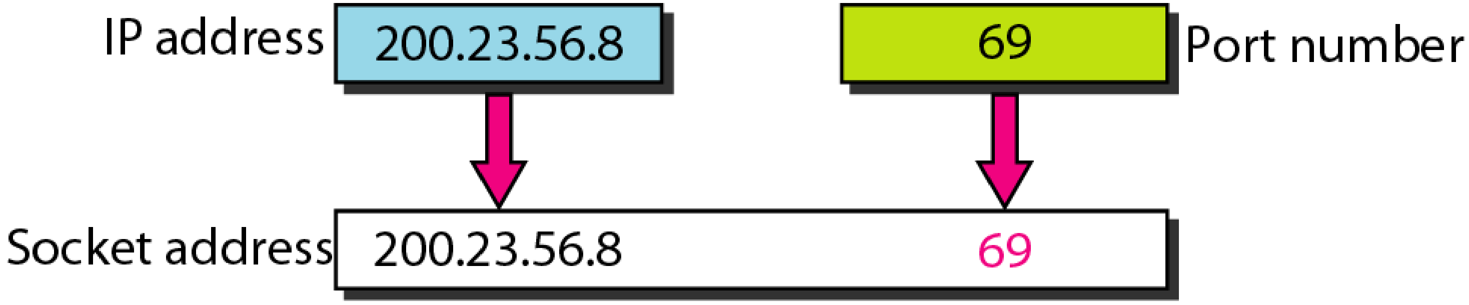
a. Data Rate

b. Average Rate

c. Traffic Rate

d. Traffic Shaping

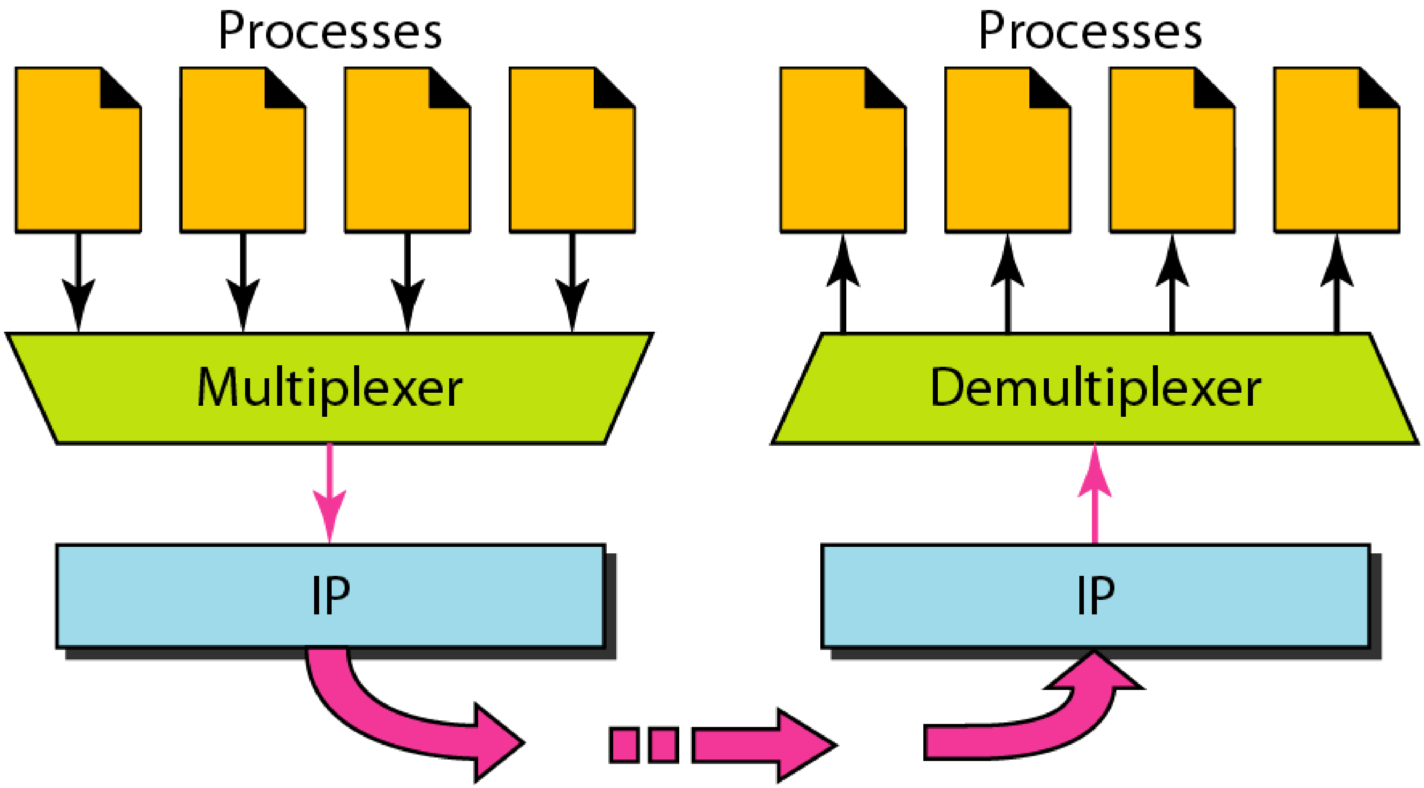
Figure 23.5 *Socket address*



powershell

Netstat -n

Figure 23.6 *Multiplexing and demultiplexing*



Connectionless Versus Connection-Oriented Service

Connectionless Service

- In a connectionless service, the packets are sent from one party to another with no need for connection establishment or connection release. The packets are not numbered; they may be delayed or lost or may arrive out of sequence. There is no acknowledgment either. We will see shortly that one of the transport layer protocols in the Internet model, UDP, is connectionless.

Connection-Oriented *Service*

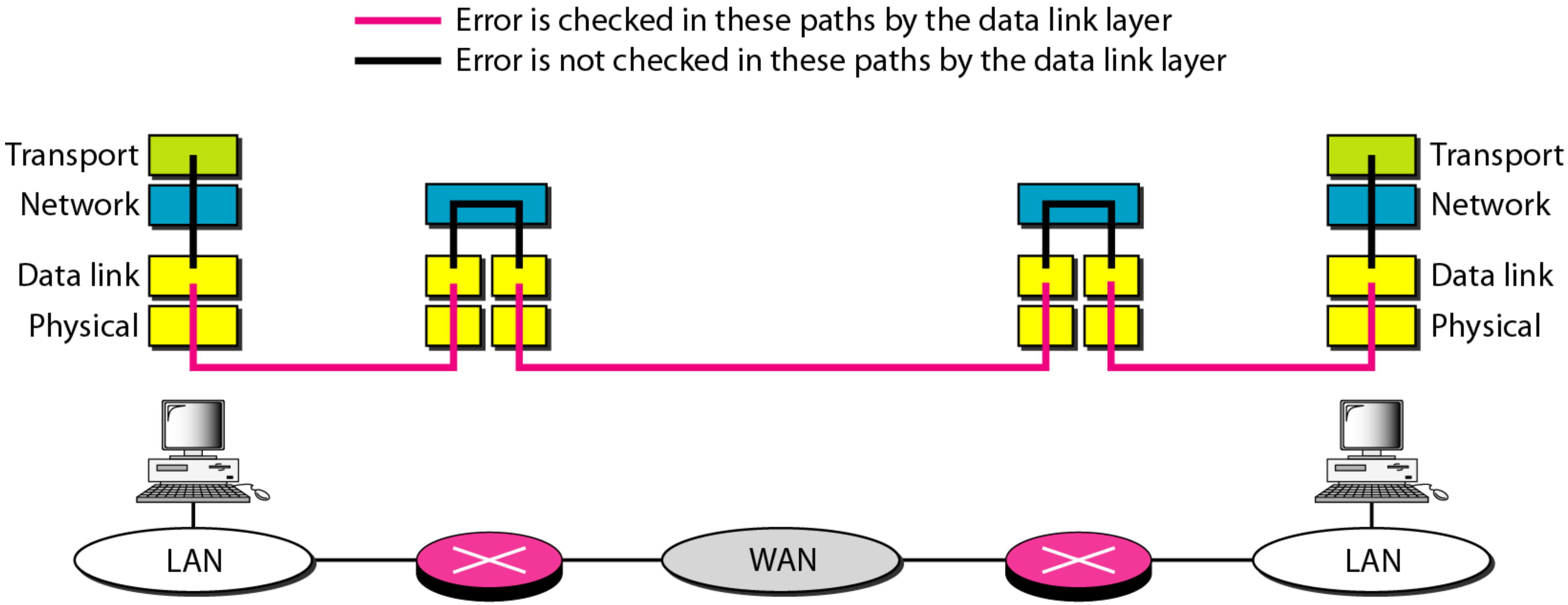
- In a connection-oriented service, a connection is first established between the sender and the receiver. Data are transferred. At the end, the connection is released. TCP and SCTP are connection-oriented protocols.

Reliable Versus Unreliable

- The transport layer service can be reliable or unreliable. If the application layer program needs reliability, we use a reliable transport layer protocol by implementing flow and error control at the transport layer. This means a slower and more complex service.
- On the other hand, if the application program does not need reliability because it uses its own flow and error control mechanism or it needs fast service or the nature of the service does not demand flow and error control (real-time applications), then an unreliable protocol can be used.
- UDP is connectionless and unreliable; TCP and SCTP are connection oriented and reliable.

- These three can respond to the demands of the application layer programs.

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

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.

Figure 23.9 *User datagram format*

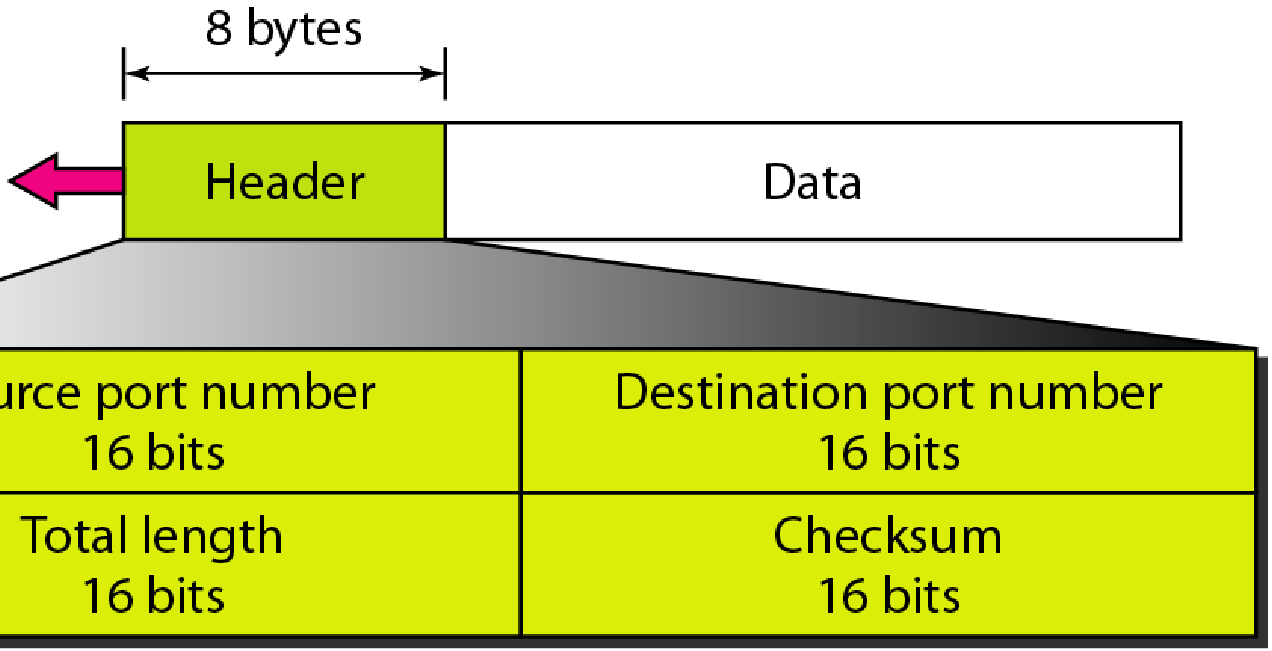
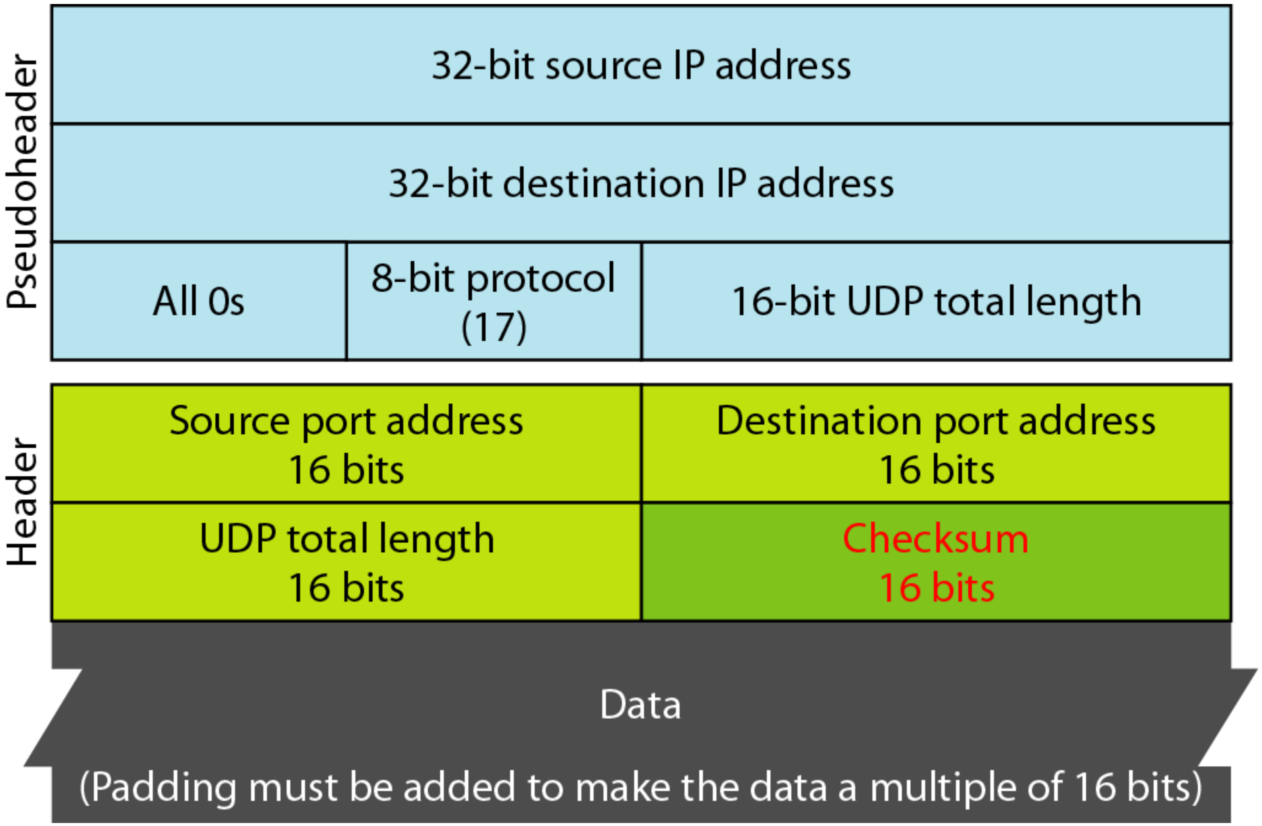


Figure 23.10 *Pseudoheader for checksum calculation*



Pseudoheader

32-bit source IP address

32-bit destination IP address

All 0s

8-bit protocol (17)

16-bit UDP total length

Header

Source port address 16 bits

Destination port address 16 bits

UDP total length 16 bits

Checksum 16 bits

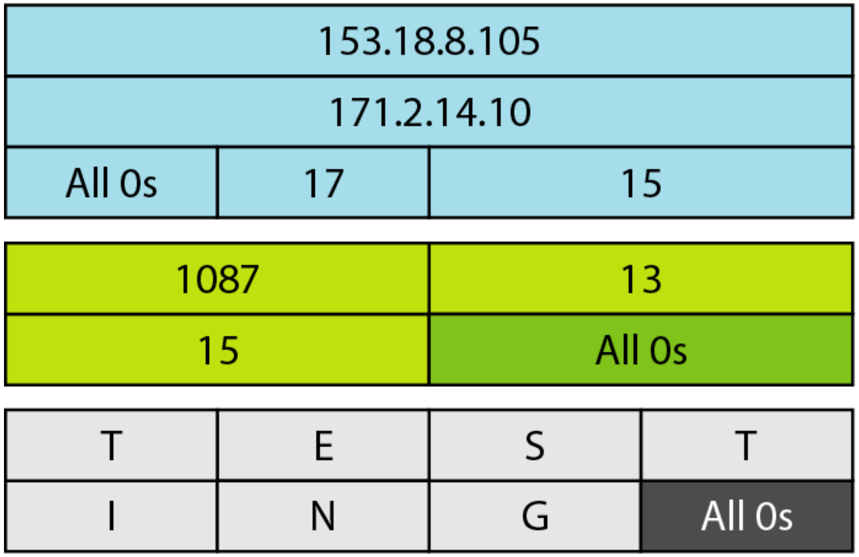
Data

(Padding must be added to make the data a multiple of 16 bits)

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

UDP Operation

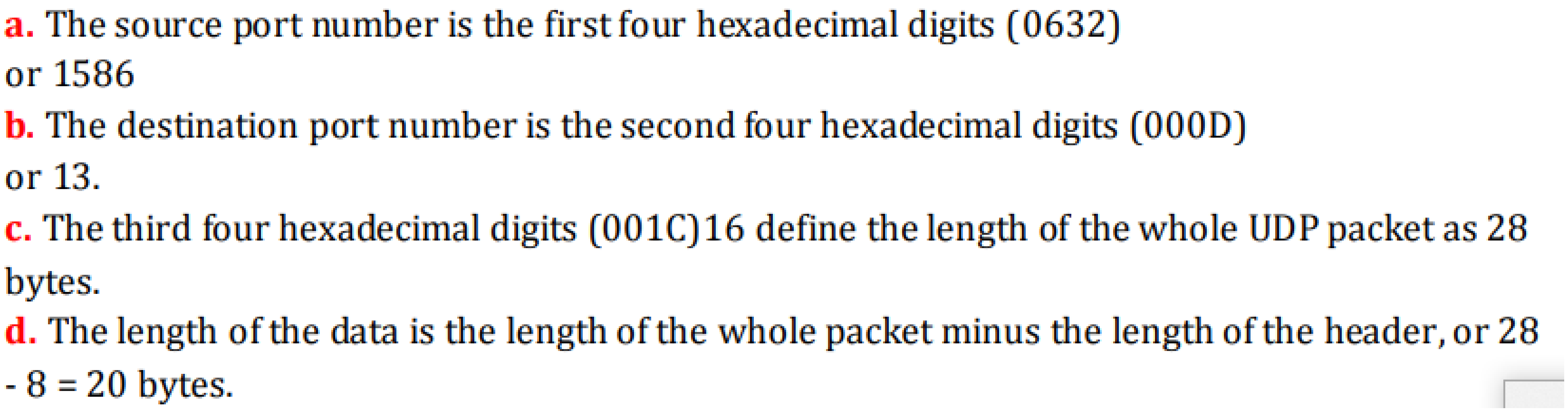
- Connectionless service- no relation between datagram, not numbered
- No Flow and error control- no flow control so no window mechanics.
- No error control except checksum (silently discard packet)
- Encapsulation and decapsulation – IP Datagrams

Example 23.2.3

A UDP header in hexadecimal format

06 32 00 0D 00 1C E2 17

What is the source port number?
What is the destination port number?
What is the total length of the user datagram?
What is the length of the data?



a. The source port number is the first four hexadecimal digits (0632) or 1586

b. The destination port number is the second four hexadecimal digits (000D) or 13.

c. The third four hexadecimal digits (001C)16 define the length of the whole UDP packet as 28 bytes.

d. The length of the data is the length of the whole packet minus the length of the header, or 28 - 8 = 20 bytes.

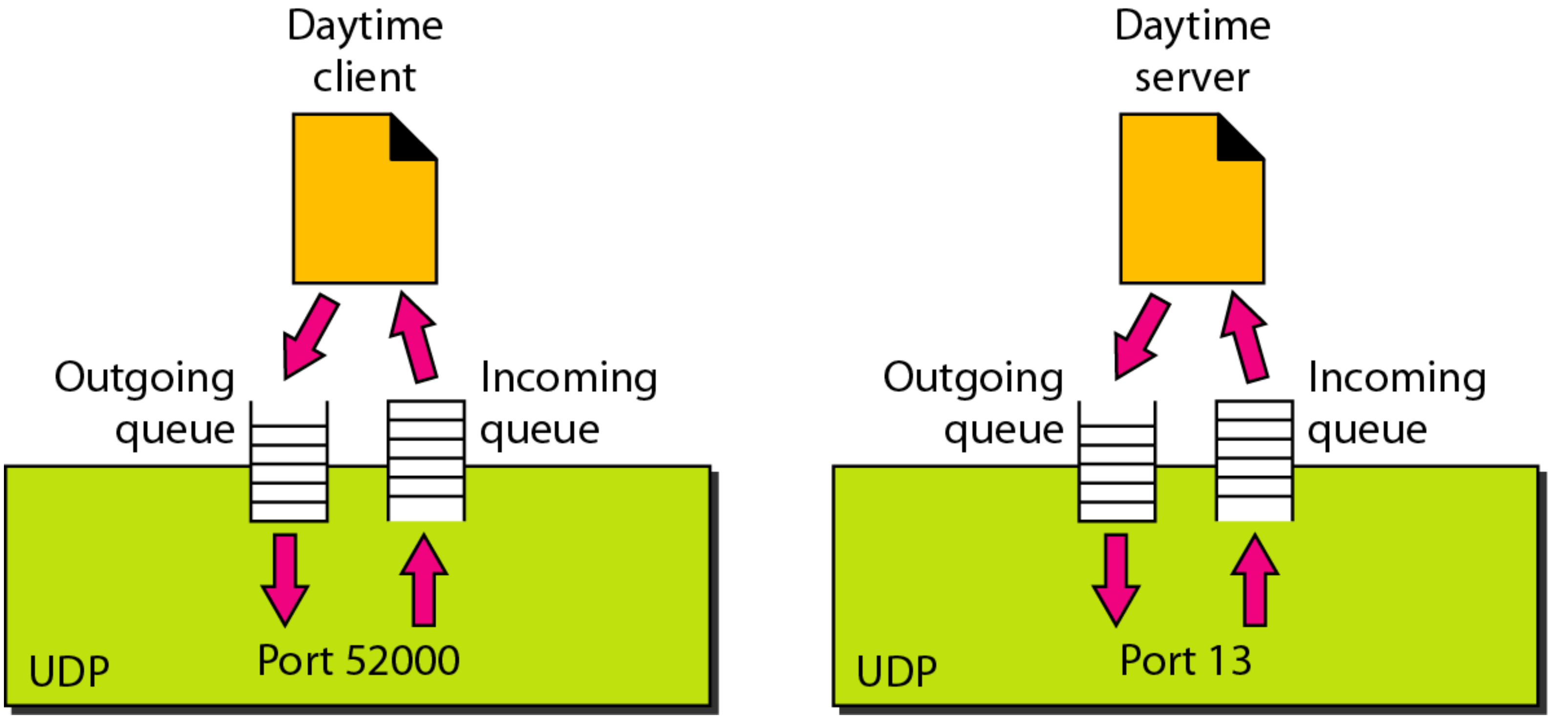
Which of the following is false with respect to UDP?
a) Connection-oriented
b) Unreliable
c) Transport layer protocol
d) Low overhead

Beyond IP, UDP provides additional services such as _______
a) Routing and switching
b) Sending and receiving of packets
c) Multiplexing and demultiplexing
d) Demultiplexing and error checking

Queuing

- Incoming and outgoing queue
- It will obtain only one port number
- Port unreachable icmp message (if queue is not created)

Figure 23.12 *Queues in UDP*



Uses of UDP

- Suitable for process that require simple request response communication with little concern for flow and error control.
- Suitable for multicasting
- Used for management process such as SNMP
- Used for routing updating protocol : RIP

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

Figure 23.13 *Stream delivery*

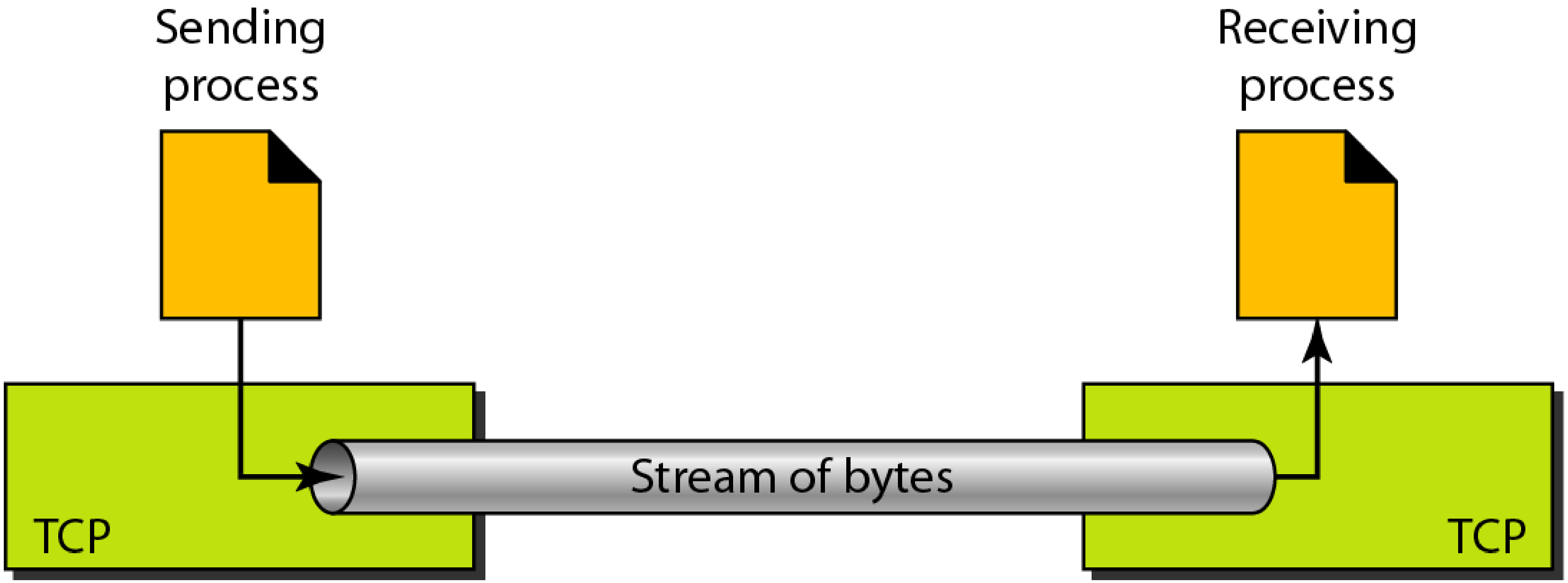


Figure 23.14 *Sending and receiving buffers*

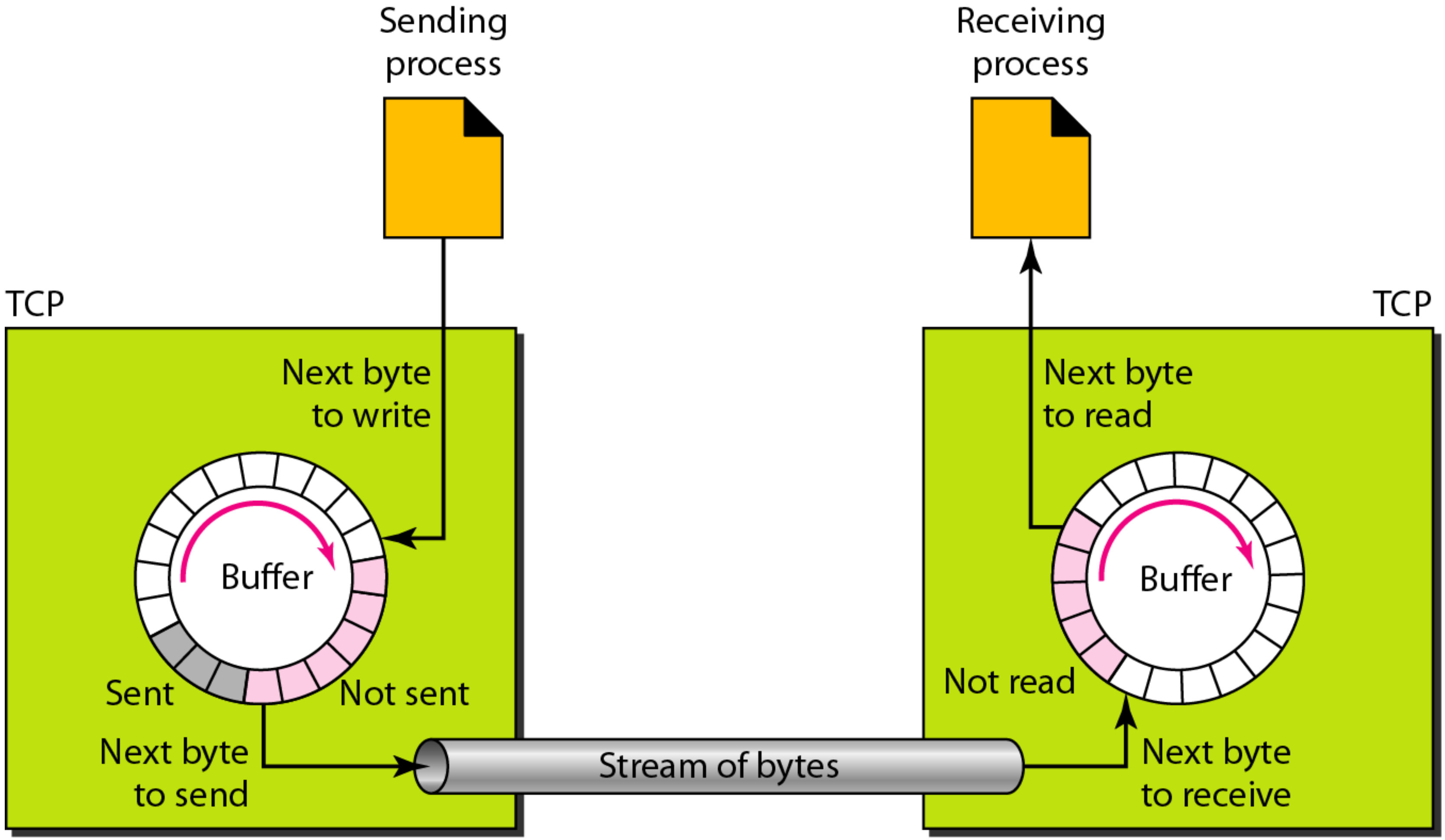
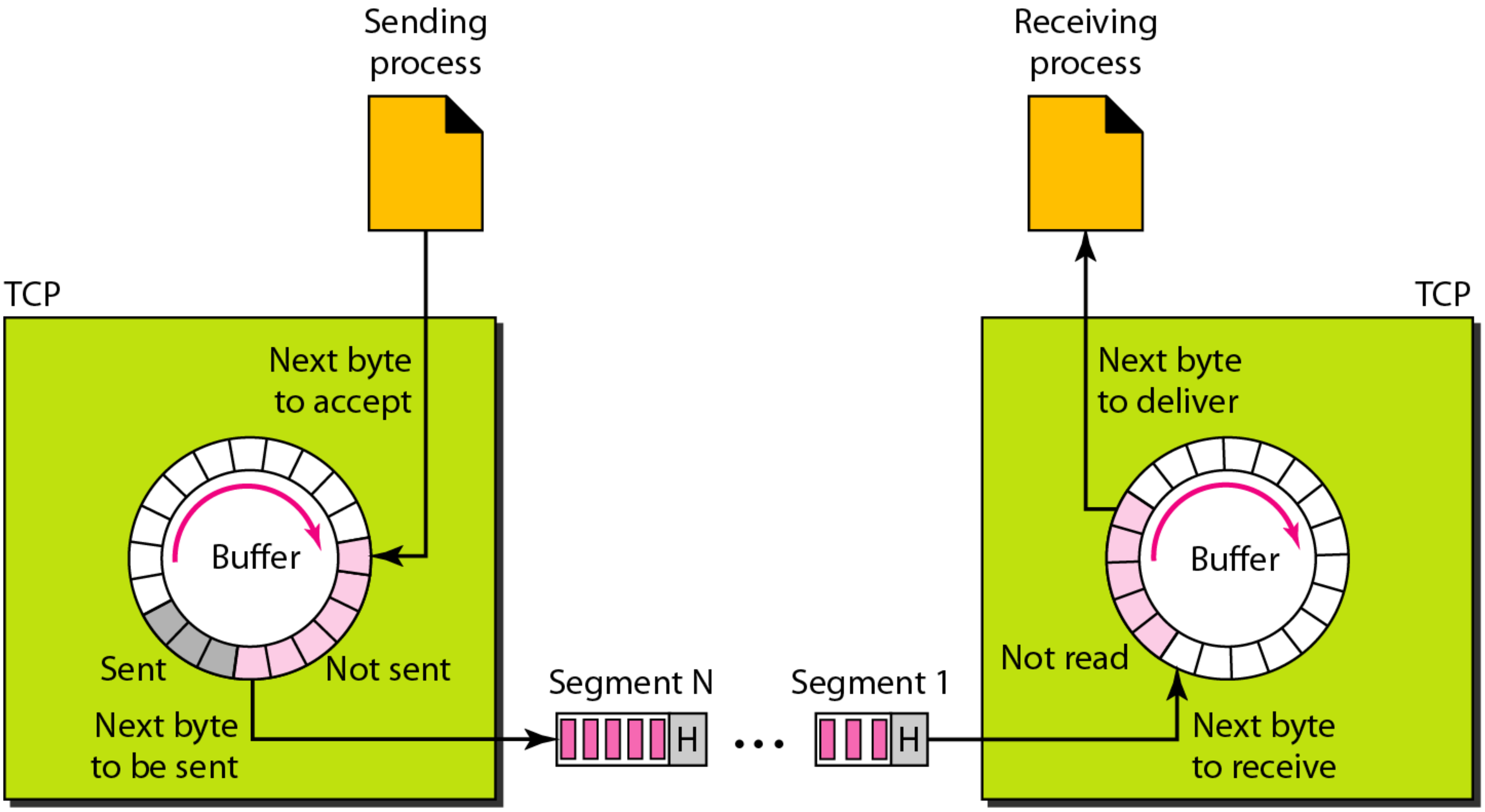


Figure 23.15 *TCP segments*



TCP

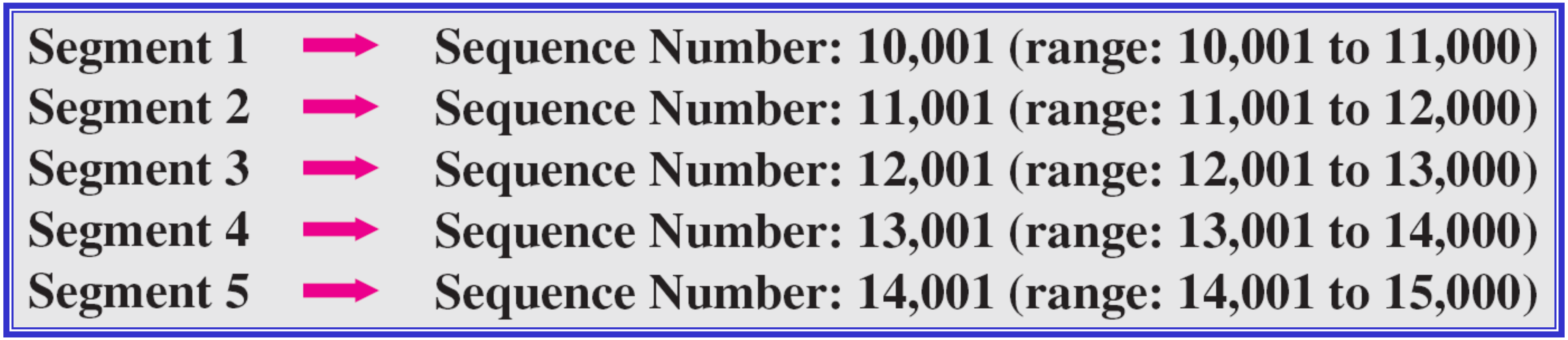
- Connection oriented phase-
- Reliable
- Features
    - Numbering system
        - No segment number – use byte number – sequence number, ack number
        - 0- 2^32 -1
- Flow control
- Error control
- Congestion control

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)

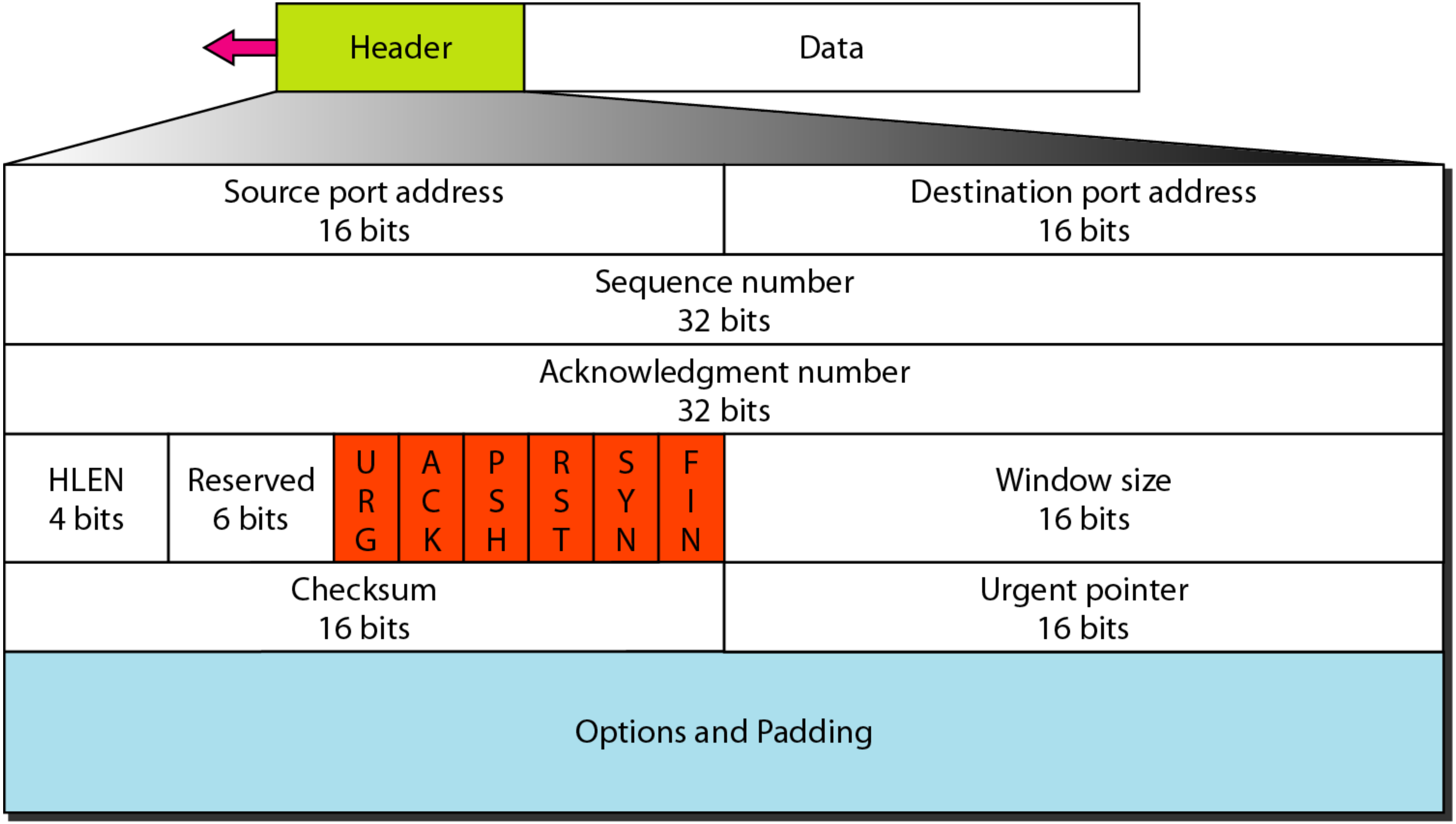
Note

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*



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

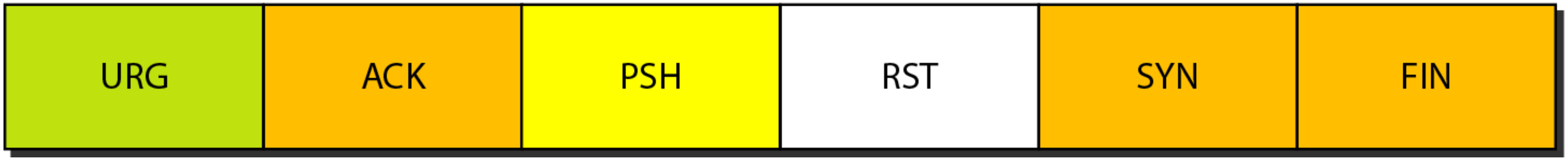
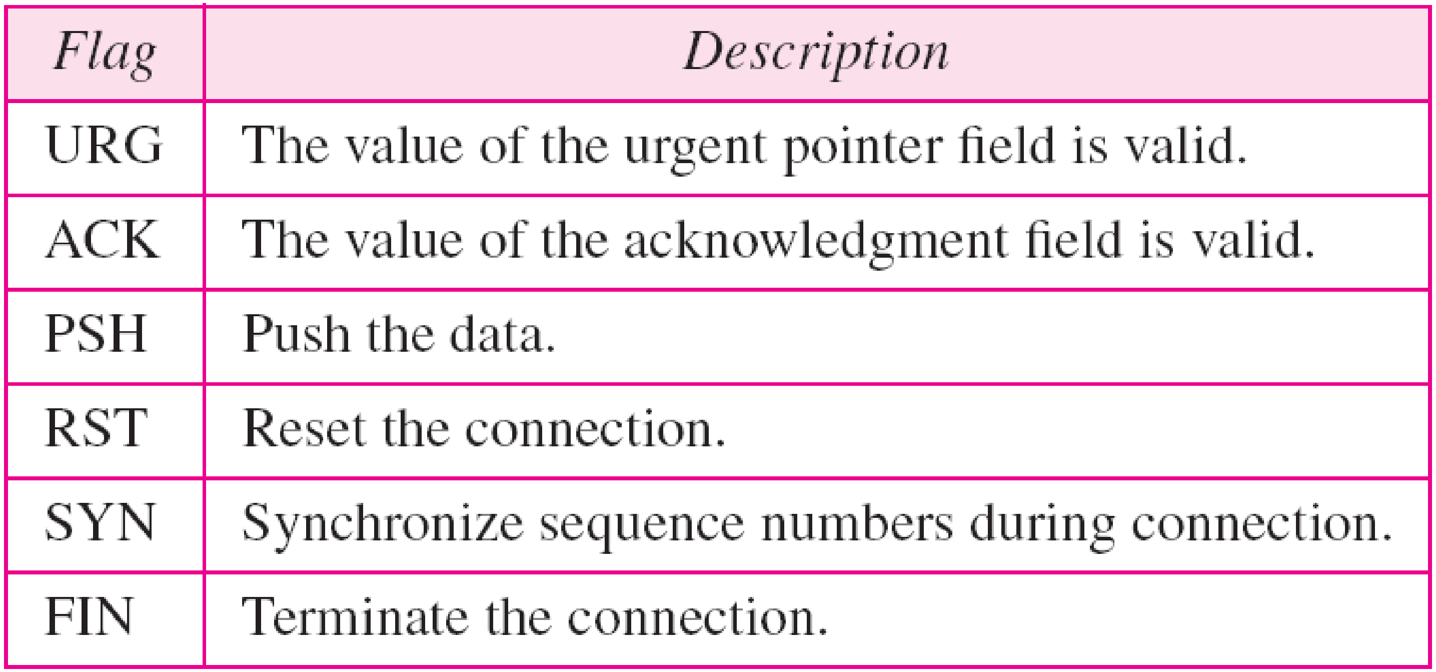


Table 23.3 *Description of flags in the control field*



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.

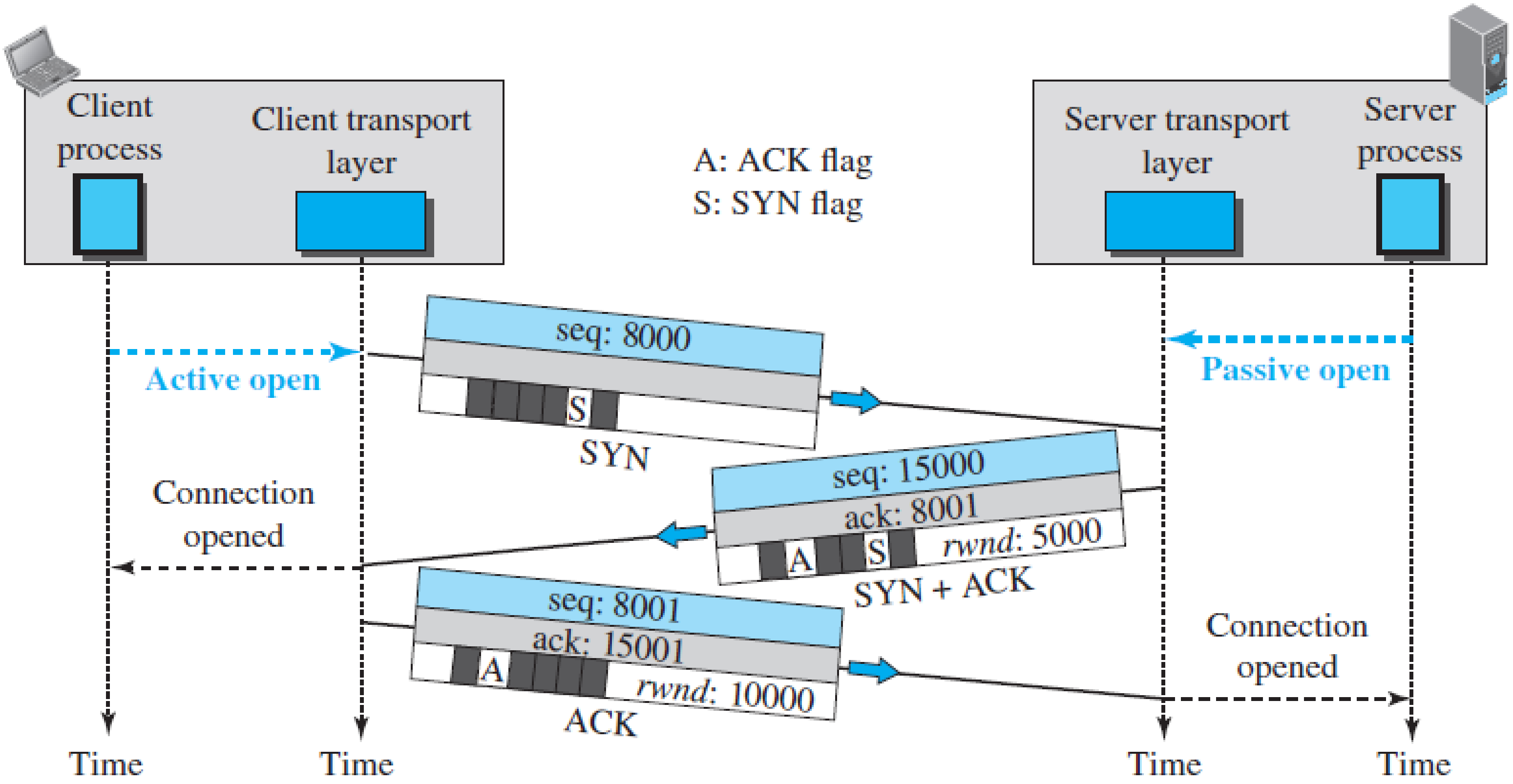
Example 23.2.4

The following is a dump of a TCP header in hexadecimal format

05320017 00000001 00000000 500207FF 00000000

What is the source port number?
What is the destination port number?
What is sequence number?
What is the acknowledgment number?
What is the length of the header?
What is the type of the segment?
What is the window size?

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.

- Syn flooding attack
- Simultaneous open

Figure 23.19 *Data transfer*

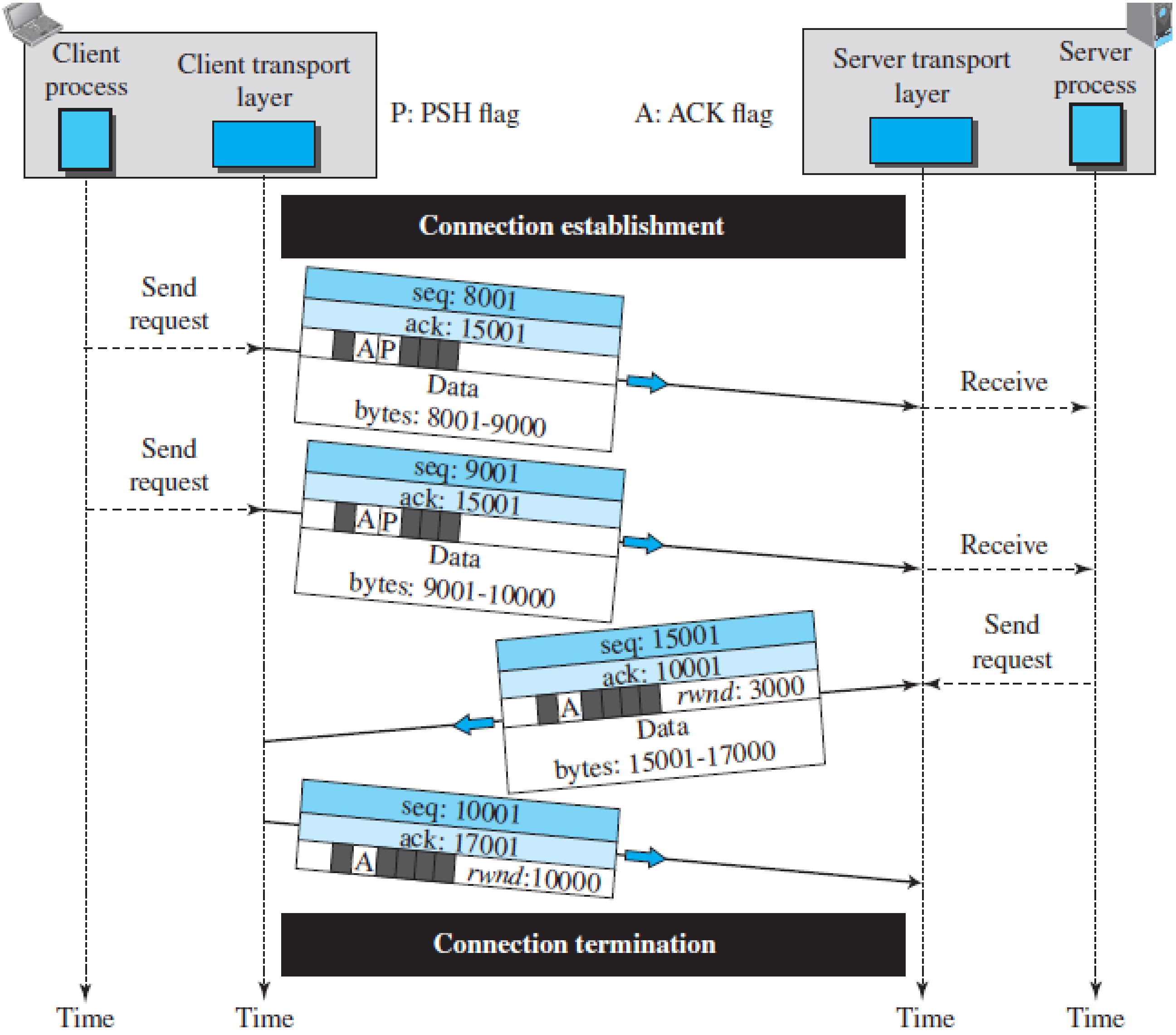
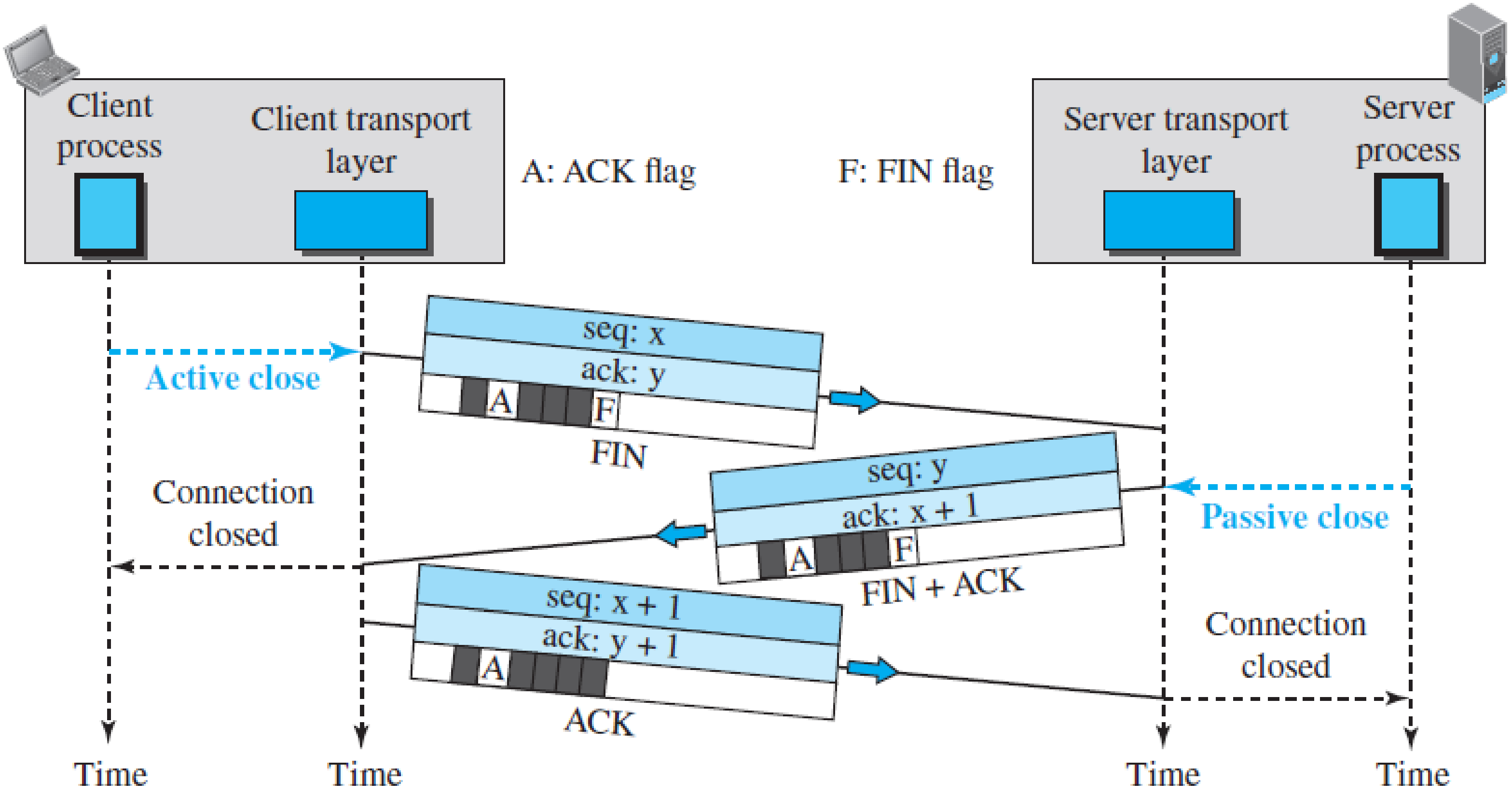


Figure 23.20 *Connection termination using three-way handshaking*



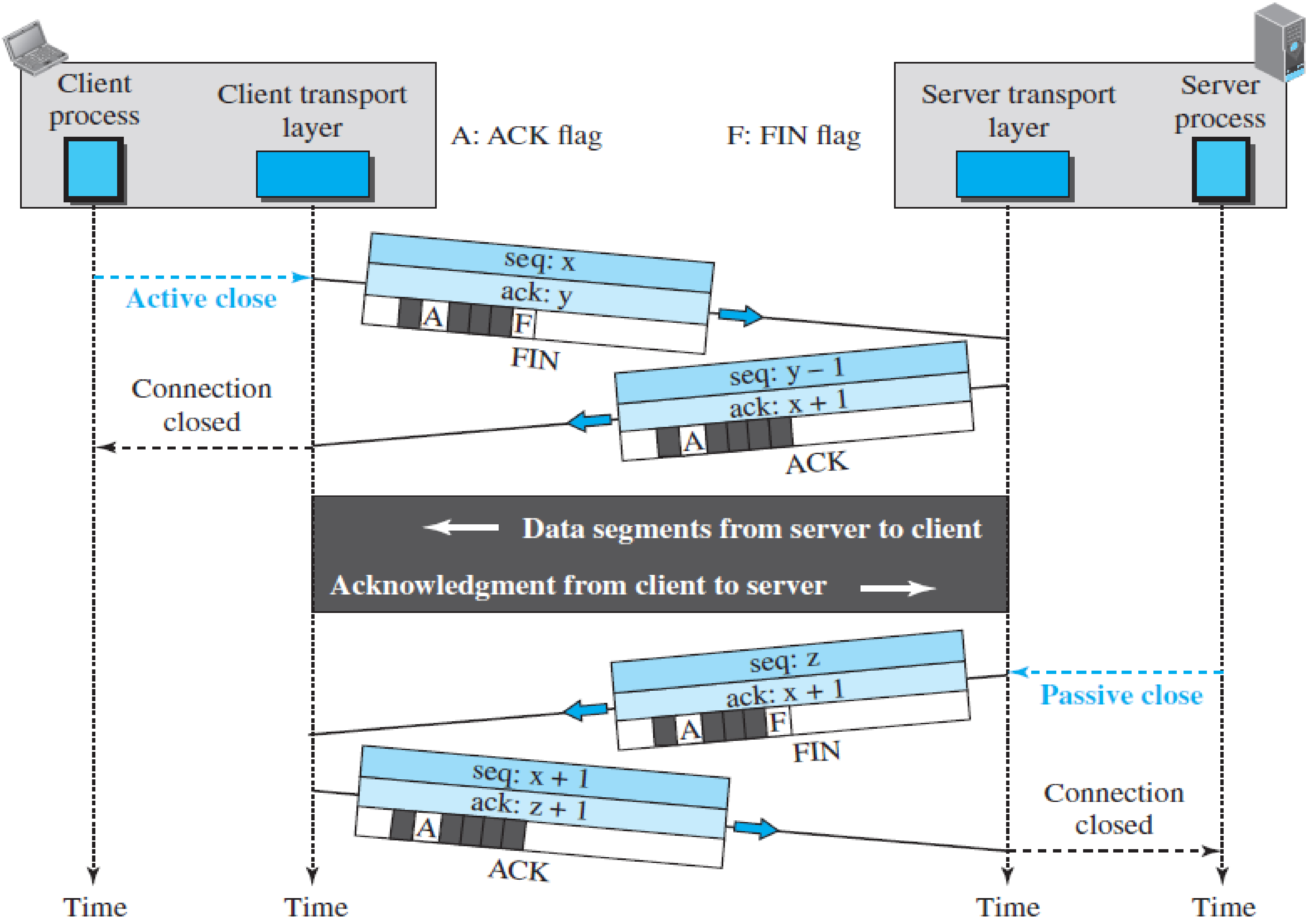
Note

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*


Client process
Client transport layer
A: ACK flag
F: FIN flag
Server transport layer
Server process
seq: x
ack: y
A
F
FIN
Active close
Connection closed
seq: y − 1
ack: x + 1
A
ACK
Data segments from server to client
Acknowledgment from client to server
seq: z
ack: x + 1
A
F
FIN
Passive close
seq: x + 1
ack: z + 1
A
ACK
Connection closed
Time
Time
Time
Time