DRONACHARYA
College of Engineering

Computer Science & Engineering

Data Communication and Computer Networks

(MTCSE-101-A)

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.

Introduction

- The transport layer is responsible for the delivery of a message from one process to another
- the transport layer header must include a service –point –address in the OSI model and port number in the TCP/IP (internet model)
- The Internet model has three protocols at the transport layer: **UDP, TCP, and SCTP**.
- **UDP**: Is the simplest of the three.
- **TCP**: A complex transport layer protocol.
- **SCTP**: The new transport layer protocol that is designed for specific applications such as multimedia.

Process-to-process Delivery

- The Data link layer is responsible for delivery of frames between nodes over a link ➔ node to node delivery
- The network layer is responsible for delivery of datagrams between two hosts ➔ host to host delivery
- Real communication takes place between two processes (application programs). We need process-to-process delivery.
- We need a mechanism to deliver data from one of processes running on the source host to the corresponding process running on the destination host.
- The transport layer is responsible for process-to-process delivery.

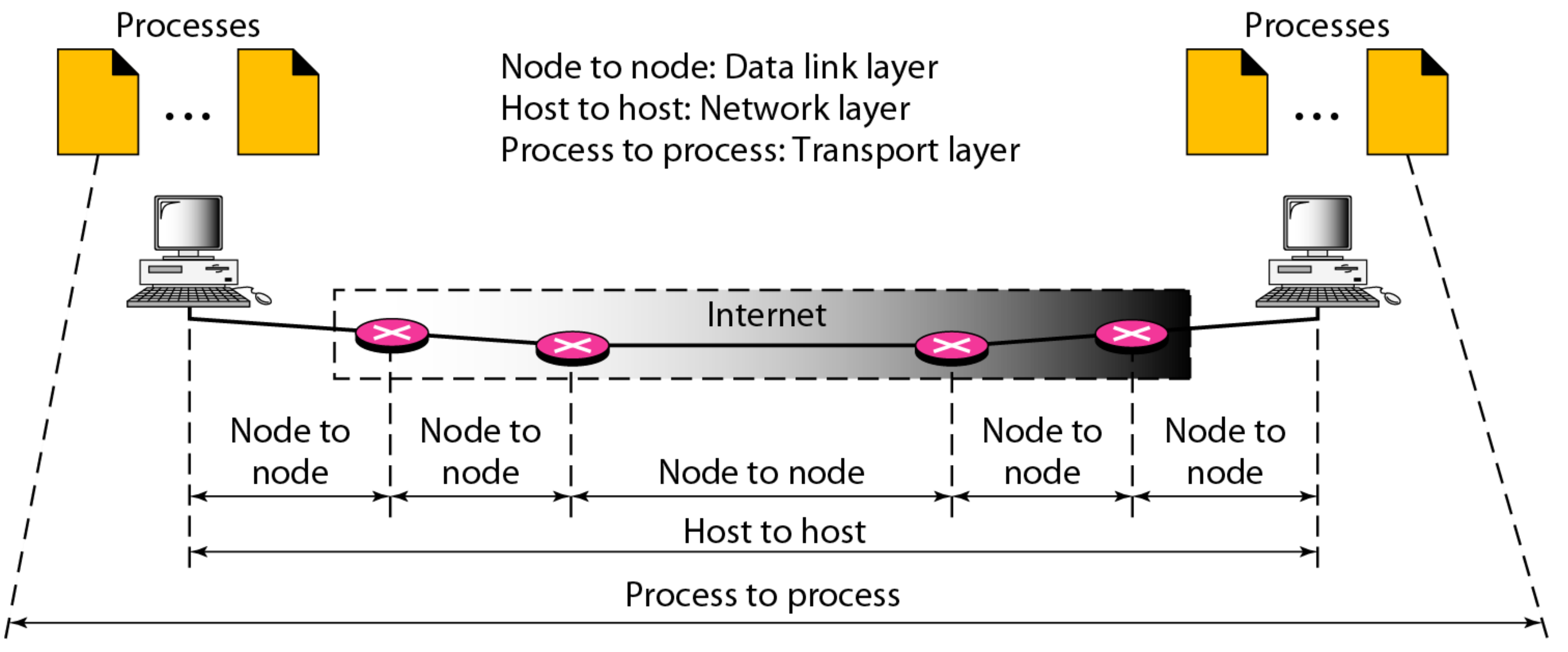
Addressing

- At the data link layer, we need a MAC address to choose one node among several
- At the network layer, we need an IP address to choose one host among millions.
- At the transport layer, we need a port number, to choose among multiple processes running on the destination host.
- The destination port number is needed for delivery; the source port number is needed for the reply.
- In the Internet model, the port numbers are 16-bit integers between 0 and 65,535.

Note

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

Figure 23.1 *Types of data deliveries*



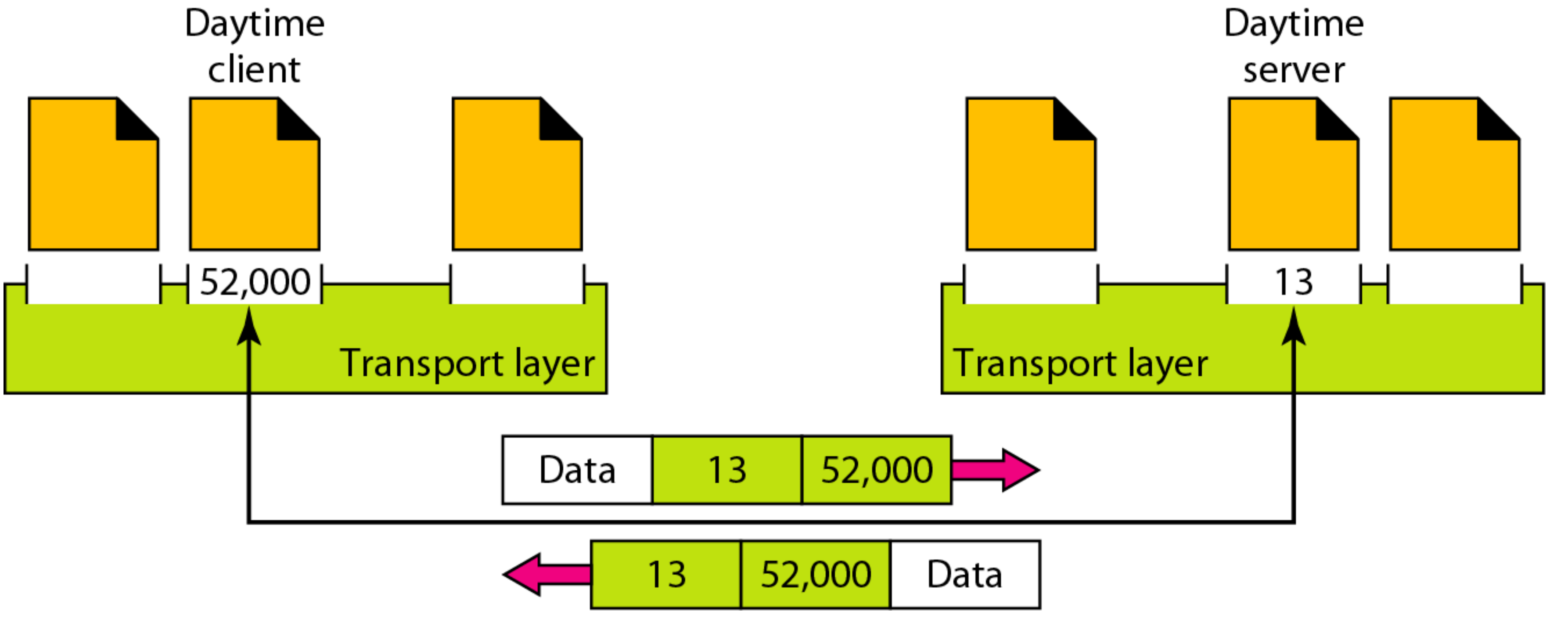
Port number

- The client program defines itself with a port number, chosen randomly by the transport layer software running on the client host.
- The server process must also define itself with a port number This port number, however, cannot be chosen randomly
- The Internet uses port numbers for servers called well-known port numbers.

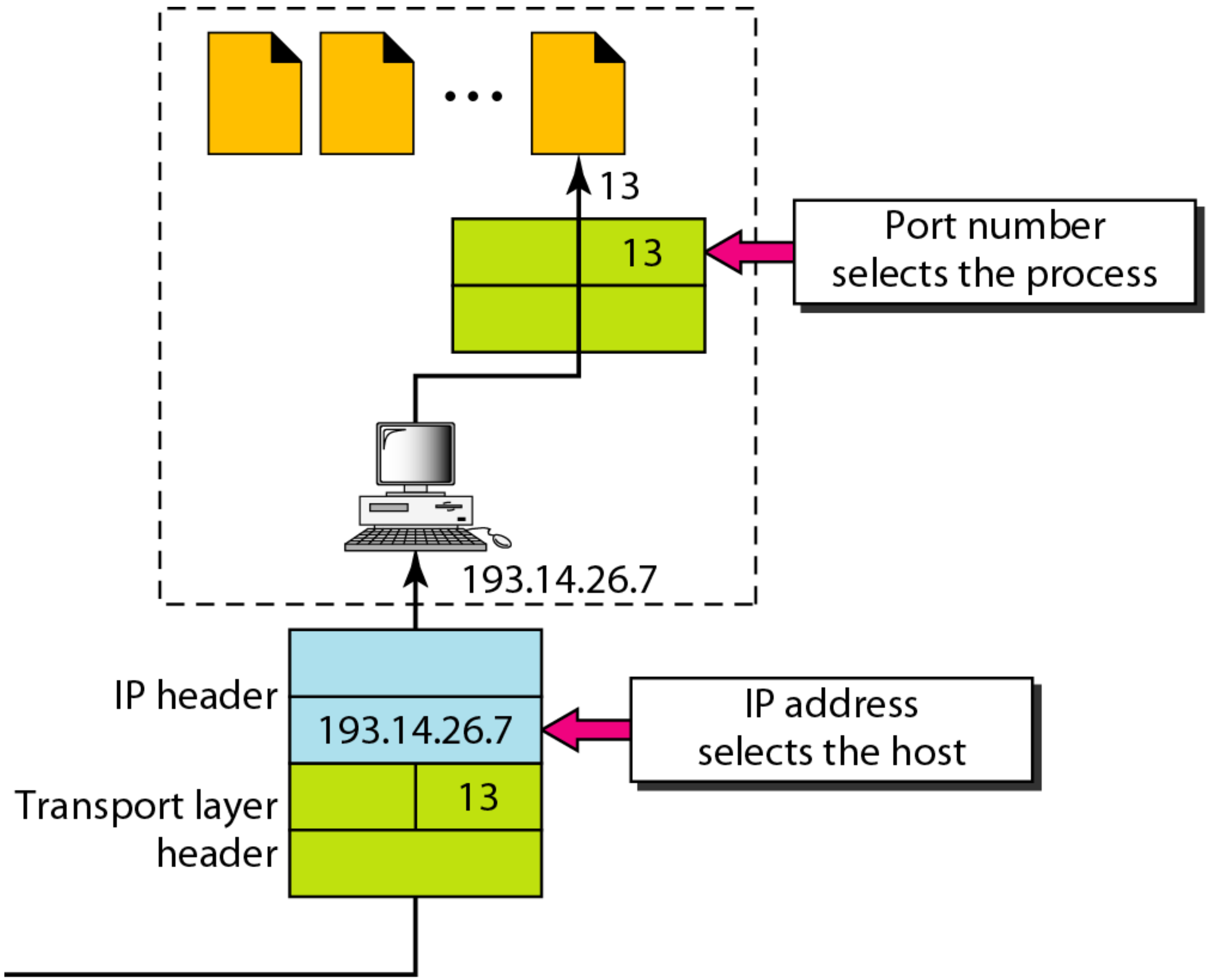
Every client process knows the well-known port number of the corresponding server process

- For example, while the Daytime client process, can use an ephemeral (temporary) port number 52,000 to identify itself, the Daytime server process must use the well-known (permanent) port number 13.

Port numbers



IP addresses versus port numbers



IANA Ranges

The IANA (Internet Assigned Number Authority) has divided the port numbers into three ranges:

- **Well- known ports**:

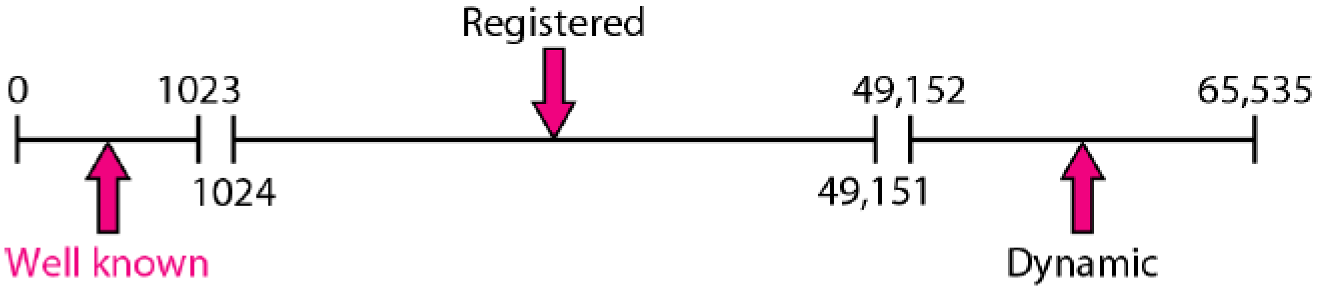
  The ports ranging from 0 to 1023 are assigned and controlled by IANA.

- **Registered ports :**

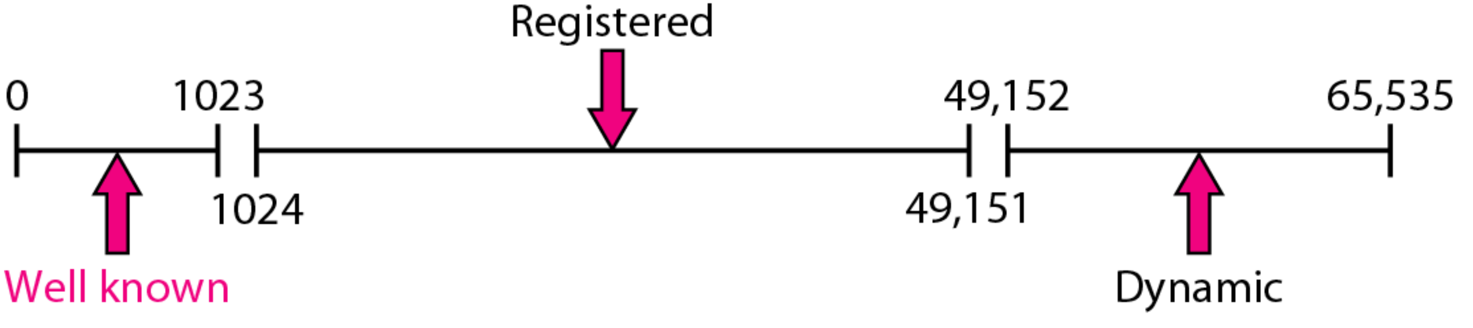
  ranging from 1024 to 49,151 are not assigned or controlled by IANA. They can only be registered with IANA to prevent duplication.

- **Dynamic (or private):**

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



IANA ranges



Registered
0
1023
49,152
65,535
1024
49,151
Well known
Dynamic

Socket Addresses

- Process-to-process delivery needs two identifiers, IP address and the port number, at each end to make a connection.
- The combination of an IP address and a port number is called a socket address.
- A transport layer protocol needs a pair of socket addresses: the client socket address and the server socket address.
- These four pieces of information are part of the IP header and the transport layer protocol header.
- The IP header contains the IP addresses; the UDP or TCP header contains the port numbers.

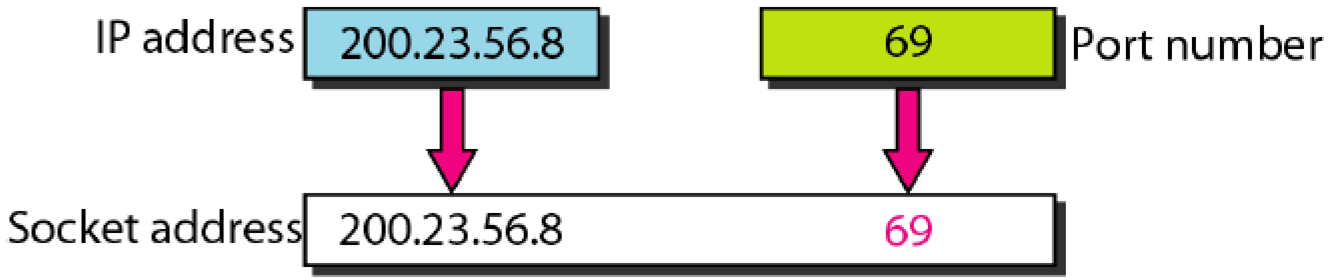


Figure 23.5 *Socket address*

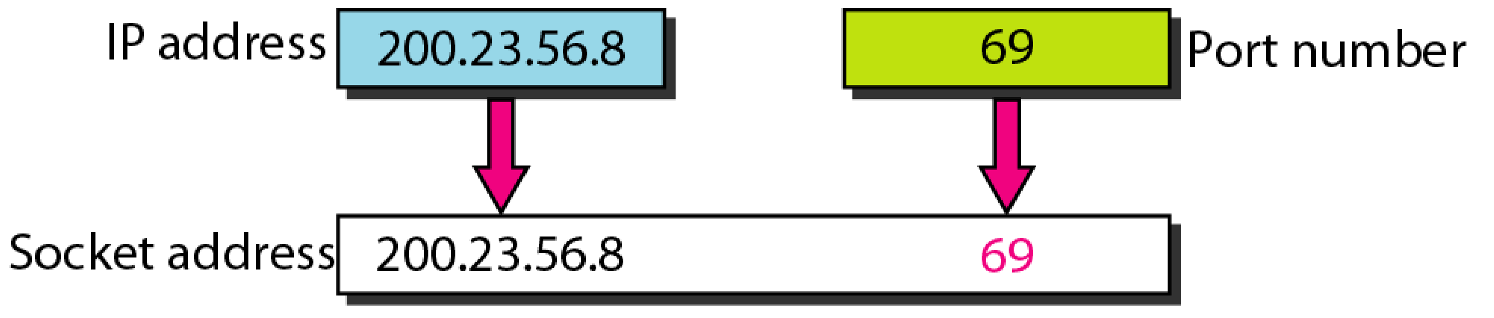
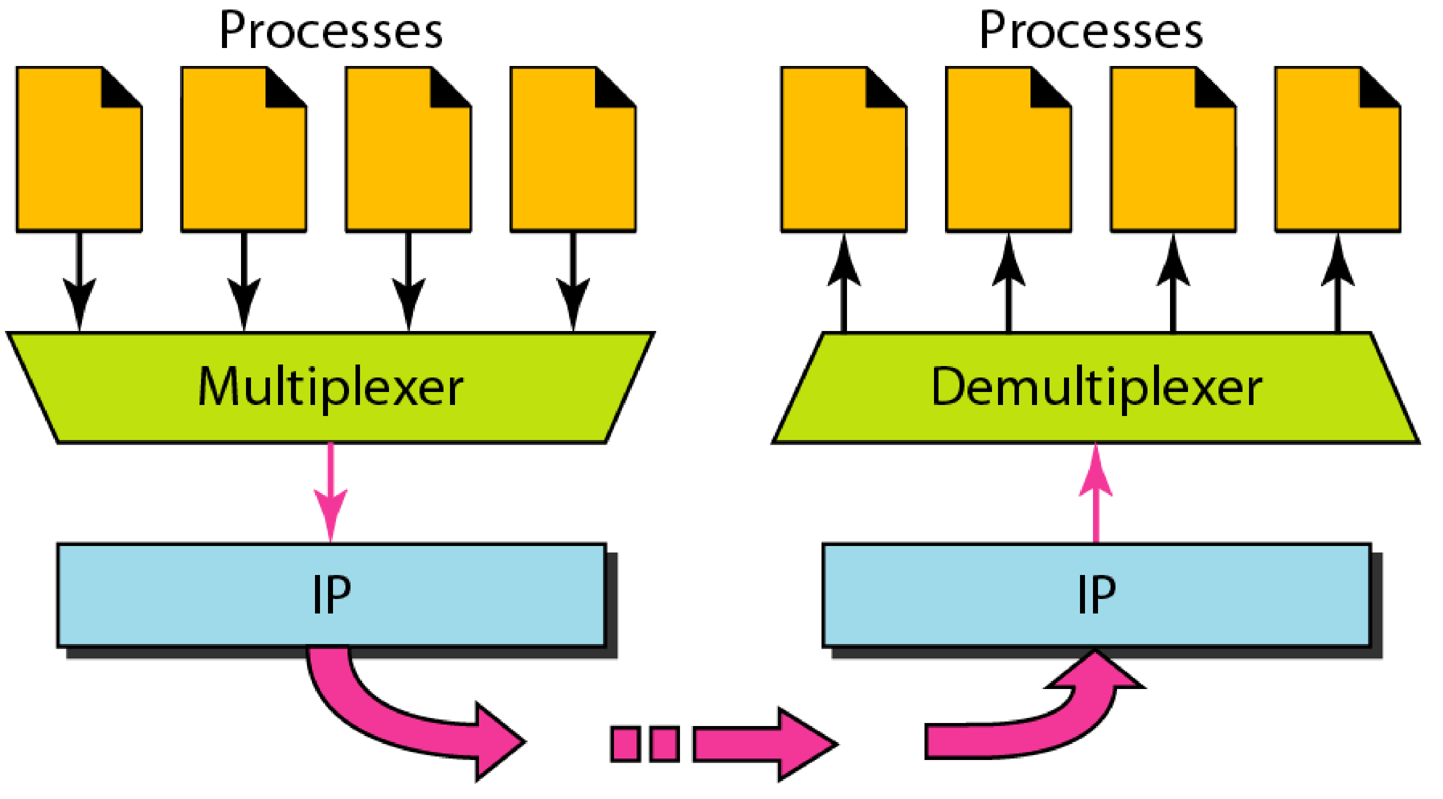


Figure 23.6 *Multiplexing and demultiplexing*



Connectionless Versus Connection-Oriented Service

- A transport layer protocol can either be connectionless or connection-oriented.
- 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 .
- **UDP** is a **connectionless** transport layer protocols.

Connectionless Versus Connection-Oriented Service

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 then an unreliable protocol can be used.
- UDP is connectionless and unreliable;
- TCP and SCTP are connection oriented and reliable.
- These three protocols can respond to the demands of the application layer programs.

Figure 23.7 *Error control*

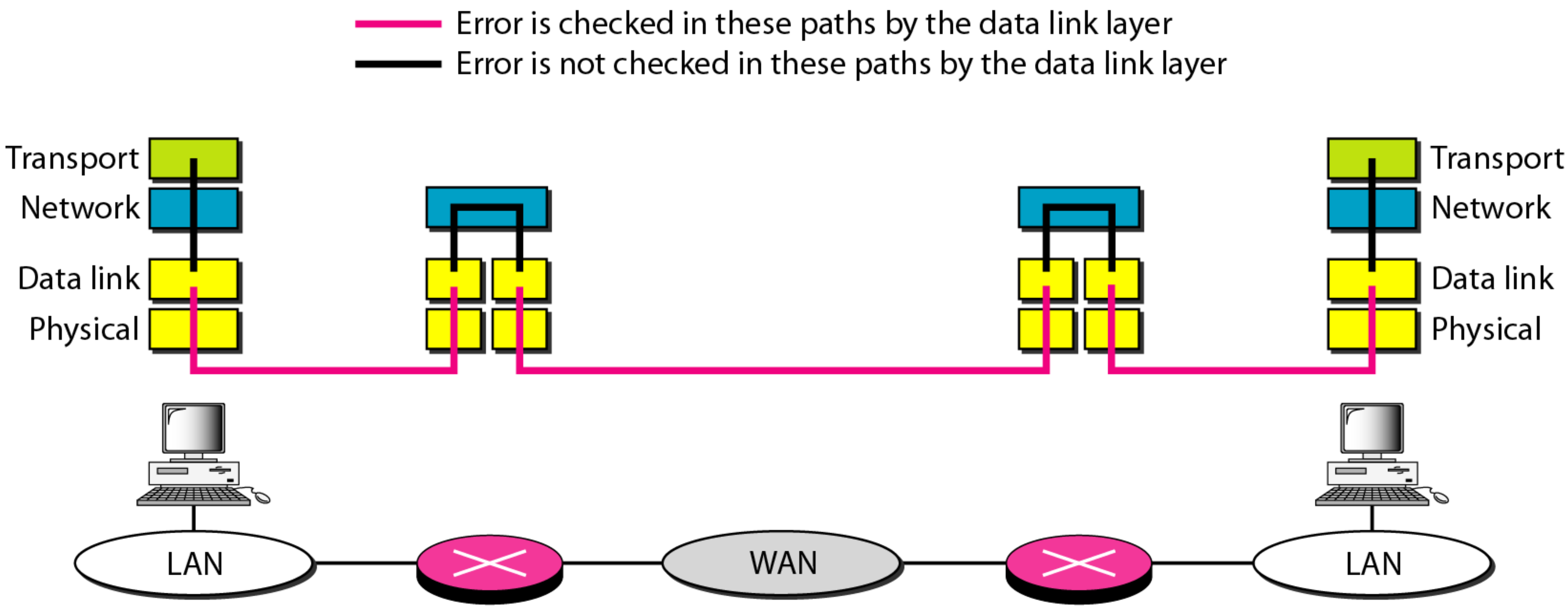
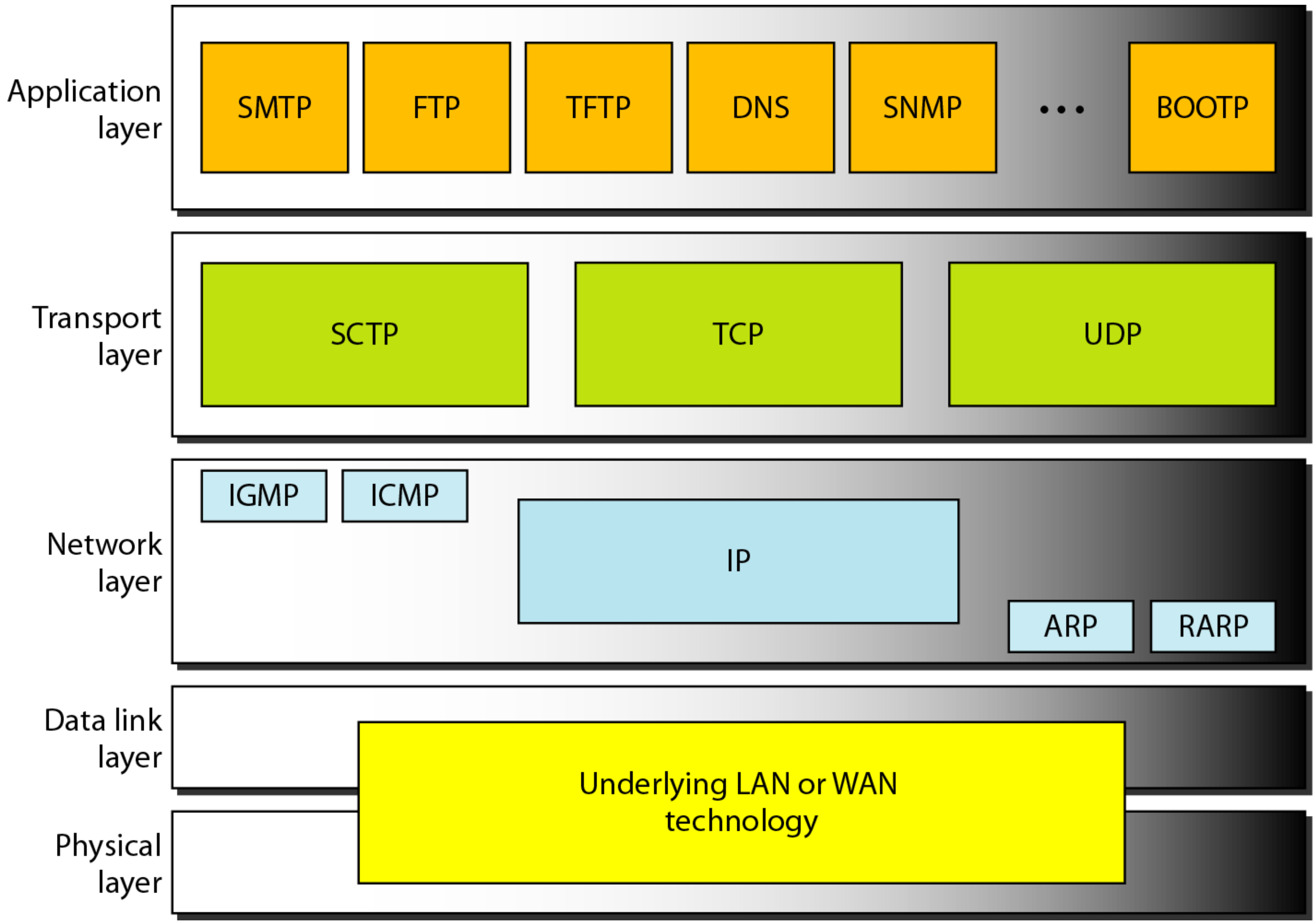


Figure 23.8 *Position of UDP, TCP, and SCTP in TCP/IP suite*



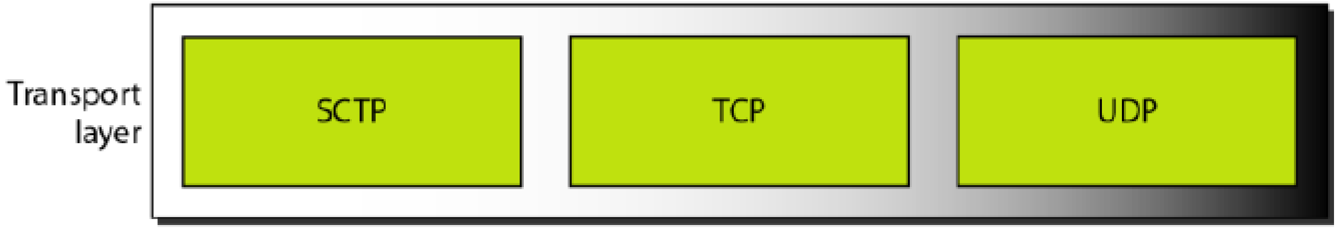
23-2 USER DATAGRAM PROTOCOL (UDP)

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:

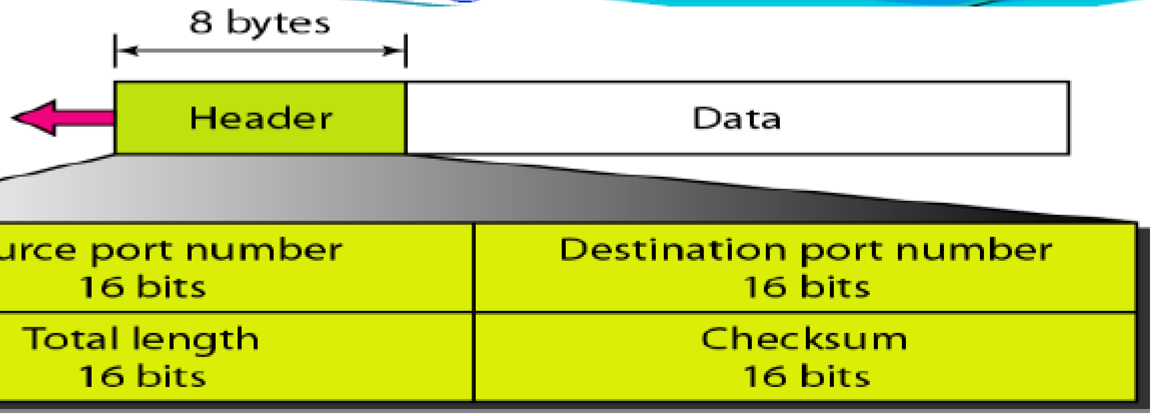
Well-Known Ports for UDP
User Datagram
Checksum
UDP Operation
Use of UDP

User Datagram Protocol (UDP



- UDP is 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.
- UDP is a very simple protocol using a minimum of overhead.
  - If a process wants to send a small message and does not care much about reliability, it can use UDP.
  - Sending a small message by using UDP takes much less interaction between the sender and receiver than using TCP or SCTP.

User datagram format



- UDP packets, called user datagrams, have a fixed size header of 8 bytes.
- **Source port number**: This is the port number used by the process running on the source host.
- **Destination port number**: This is the port number used by the process running on the destination host.
- **Length**: This is a 16-bit field that defines the total length of the user datagram.
- **Checksum**: This field is used to detect errors over the entire user datagram (header plus data). The inclusion of the checksum in the UDP datagram is optional

UDP Operation

Connectionless Services

UDP provides a connectionless service

- no relationship between the different user datagram even if they are coming from the same source process and going to the same destination program.
- Also, there is no connection establishment and no connection termination.
- Each user datagram can travel on a different path.
- The user datagrams are **not numbered.**
- Each UDP user datagram request must be small enough to fit into one user datagram. Only those processes sending short messages should use UDP.

UDP Operation

Flow and Error Control

- UDP is a very simple, unreliable transport protocol.
- There is no flow control: The receiver may overflow with incoming messages.
- There is no error control mechanism in UDP except for the checksum.
- The sender does not know if a message has been lost or duplicated.
- When the receiver detects an error through the checksum, the user datagram is **discarded.**

Use of UDP

- UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control.
- UDP is suitable for a process with internal flow and error control mechanisms. For example, the Trivial File Transfer Protocol (TFTP) process includes flow and error control..
- UDP is a suitable transport protocol for multicasting. Multicasting capability is embedded in the UDP software but not in the TCP software.

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.

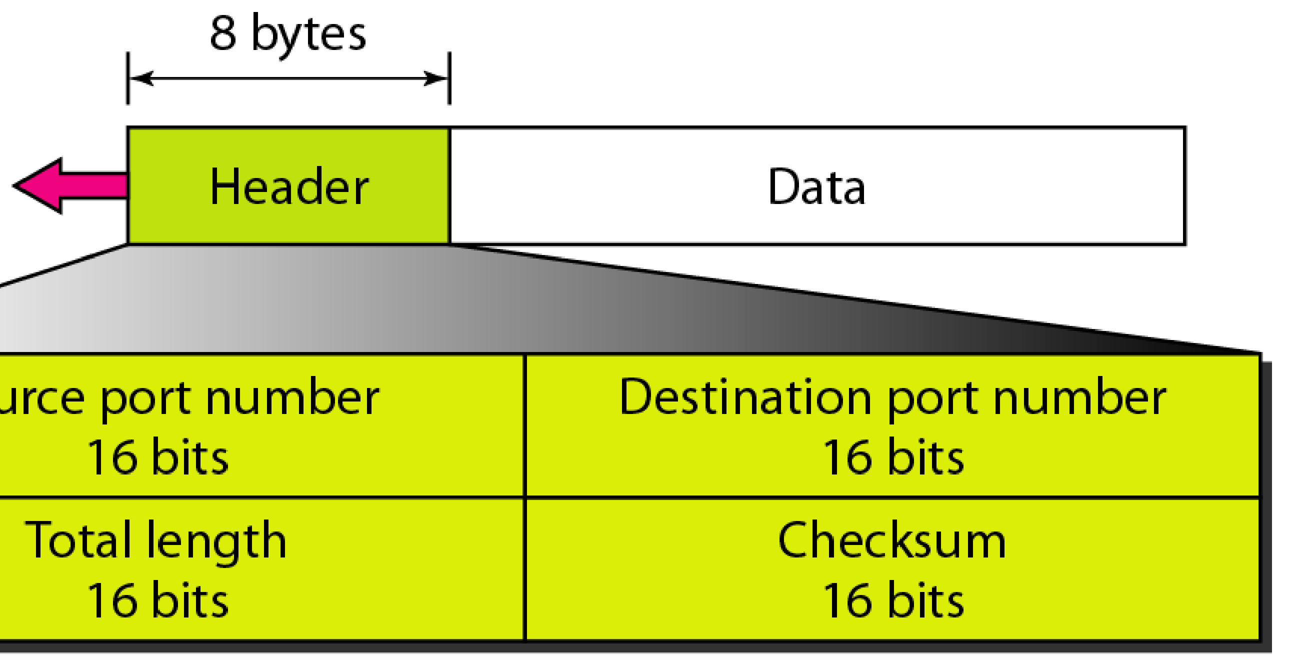
```
$ grep     ftp    /etc/services
ftp              21/tcp
ftp              21/udp
```

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*

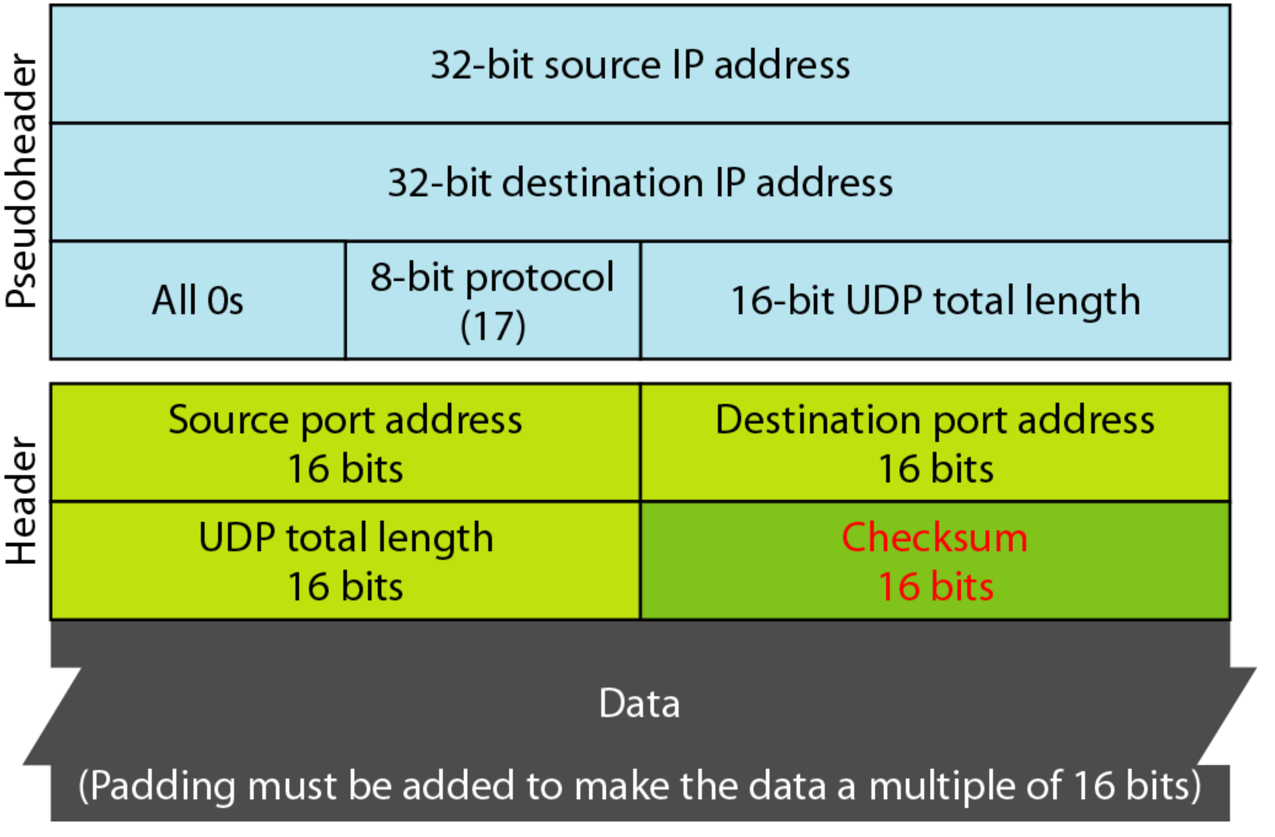


Note

UDP length

= IP length – IP header's length

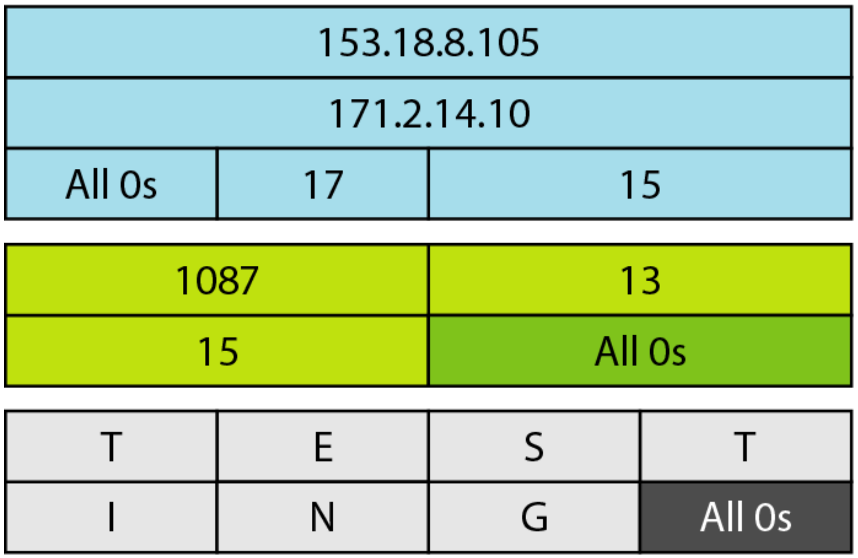
Figure 23.10 *Pseudoheader for checksum calculation*



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

Example 23.2.1

Show the entries for the header of a UDP user datagram that carries a message from a TFTP client to a TFTP server. Fill the checksum with 0s. Choose an appropriate ephemeral port number and the correct well-known port number. The length of data is 40 bytes. Show the UDP packet format.

Example 23.2.2

A client has a packet of 68000 bytes, can this packet be transferred by a single UDP datagram?

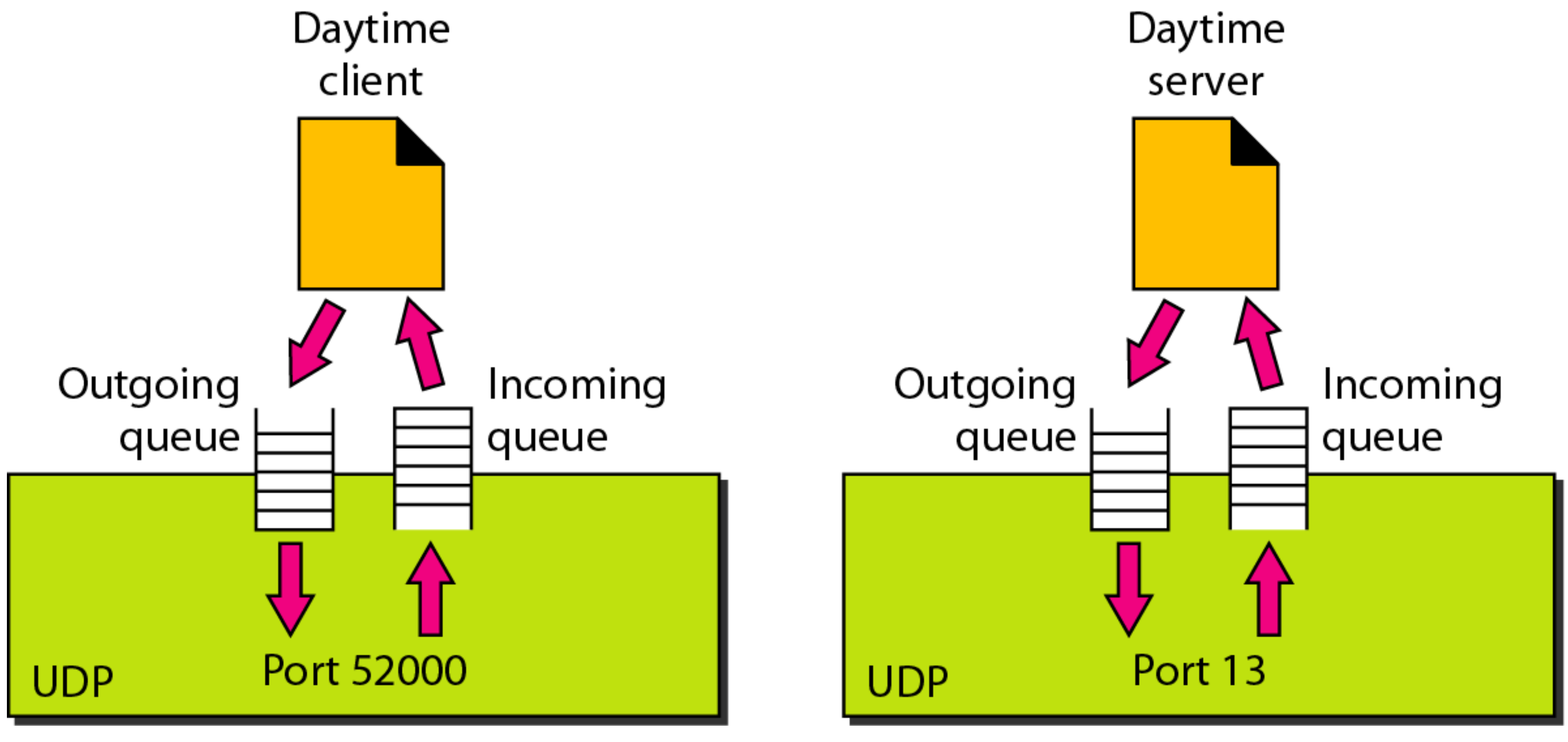
Example 23.2.3

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?
Is packet directed from a client to server or vice versa?
What is the client process?

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

TCP Services

Process-to-Process Communication

Like UDP, TCP provides process-to-process communication using port numbers. Next Table 23.2 lists some well-known port numbers used by TCP.

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

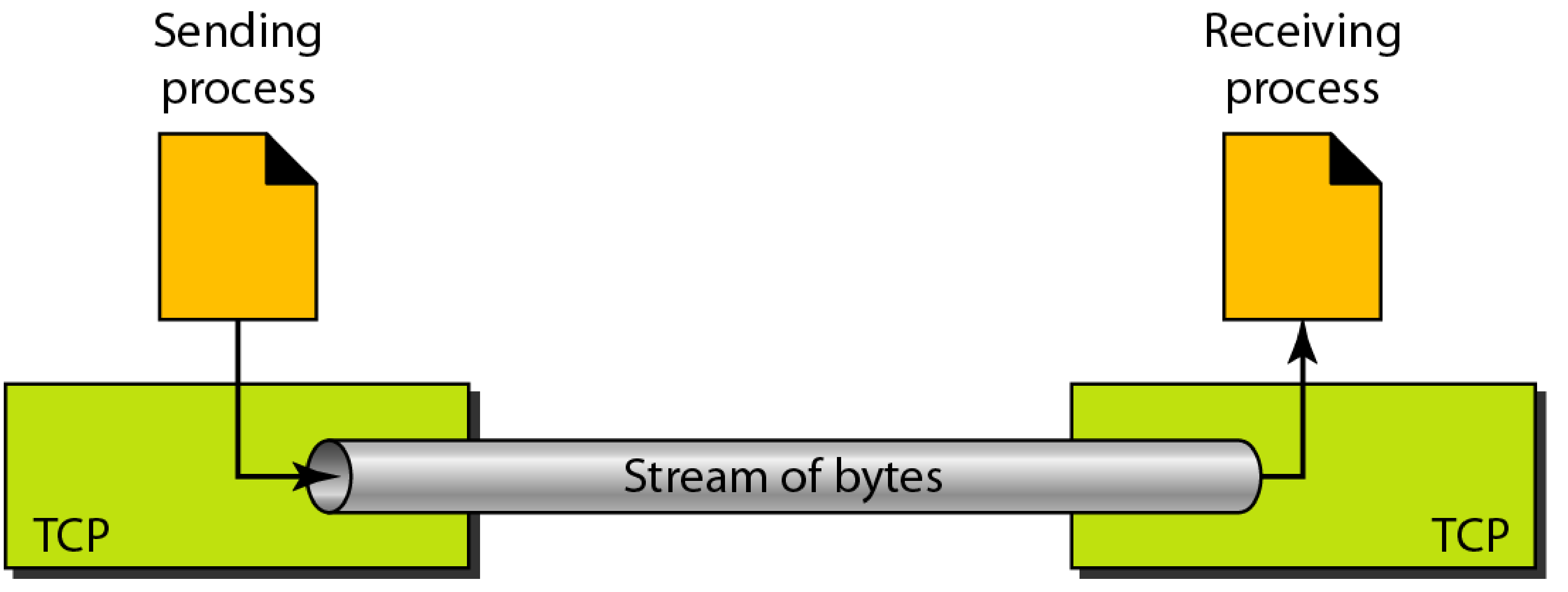
Stream delivery

TCP, unlike UDP, is a stream-oriented protocol. In UDP, a process (an application program) sends messages, with predefined boundaries, to UDP for delivery.

UDP adds its own header to each of these messages and delivers them to IP for transmission. Each message from the process is called a user datagram and becomes, eventually, one IP datagram. Neither IP nor UDP recognizes any relationship between the datagrams.

TCP, on the other hand, allows the sending process to deliver data as a stream of bytes and allows the receiving process to obtain data as a stream of bytes. TCP creates
an environment in which the two processes seem to be connected by an imaginary "tube” that carries their data across the Internet.

Figure 23.13 ***Stream delivery***



Sending and Receiving Buffers

Because the sending and the receiving processes maynot write or read data at the same speed, TCP needs buffers for storage.

There are two buffers, the sending buffer and the receiving buffer, one for each direction

One way to implement a buffer is to use a circular array of I-byte locations as shown in Figure 23.14.

For simplicity, we have shown two buffers of 20 bytes each; normally the buffers are hundreds or thousands of bytes, depending on the implementation.

Figure 23.14 *Sending and receiving buffers*

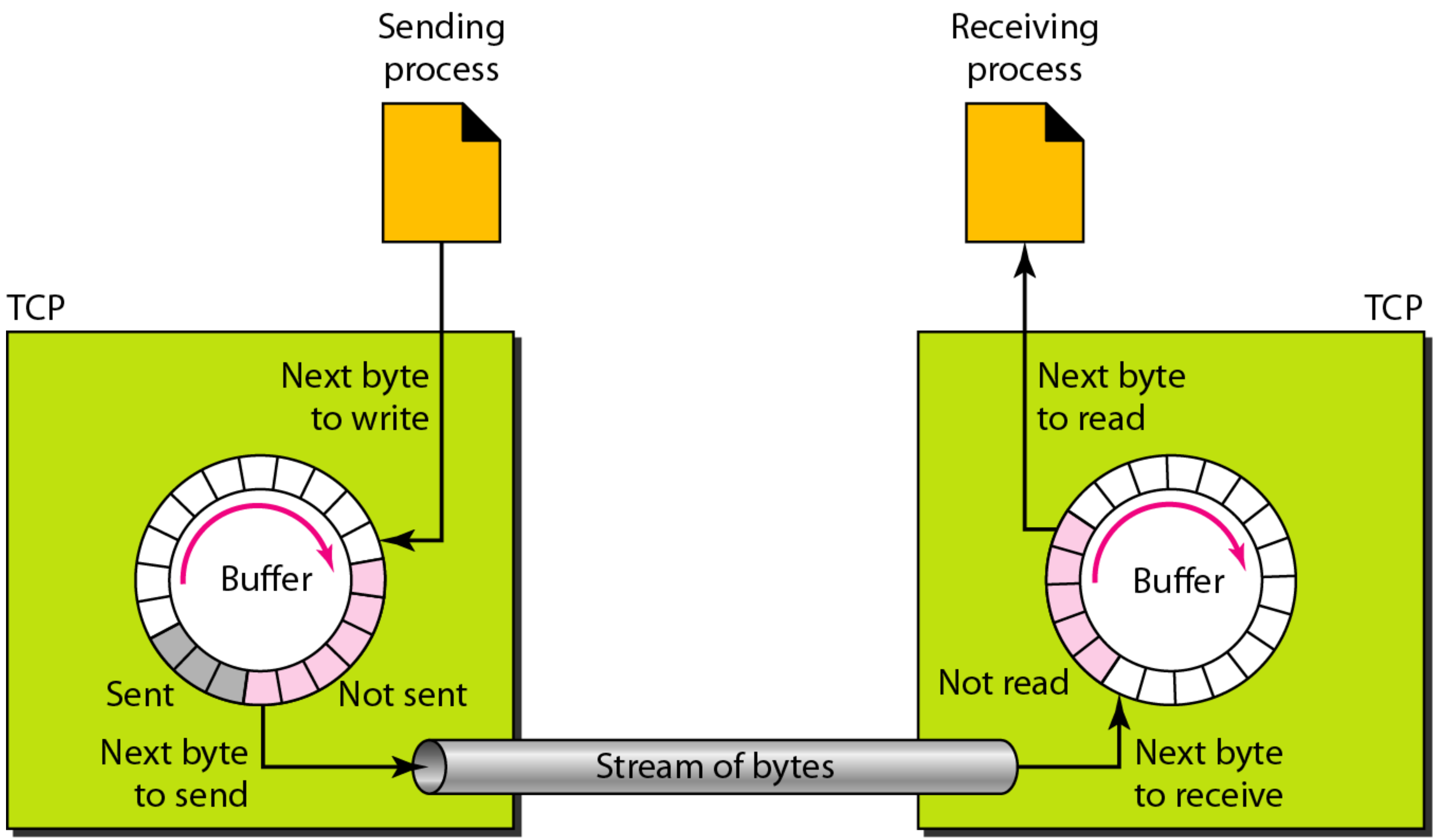
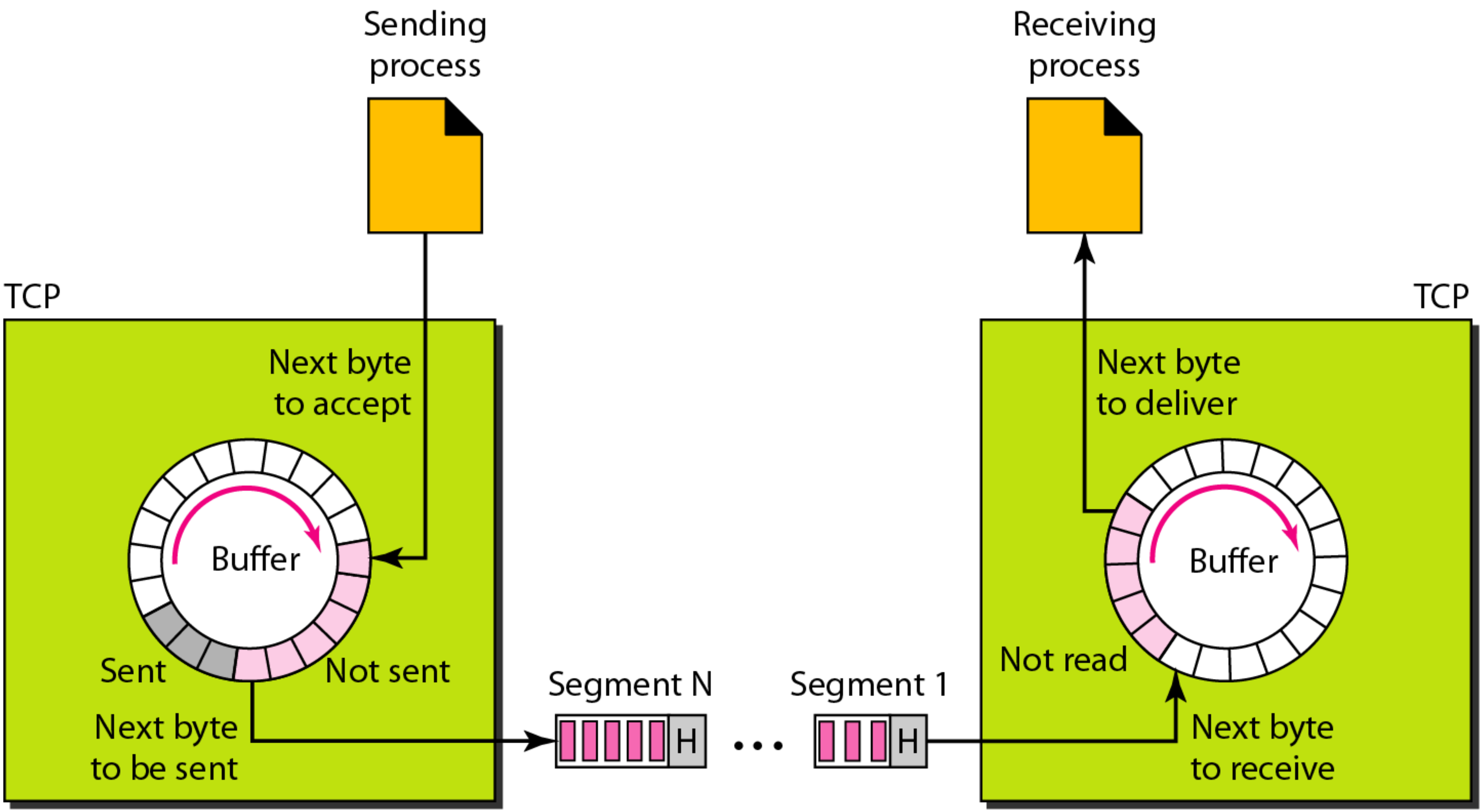


Figure 23.15 *TCP segments*



Full-Duplex Communication

TCP offers full-duplex service, in which data can flow in both directions at the same time. Each TCP then has a sending and receiving buffer, and segments move in both directions.

Connection-Oriented Service

1.The two TCPs establish a connection between them.
2. Data are exchanged in both directions.
3. The connection is terminated.

Reliable Service

TCP is a reliable transport protocol. It uses an acknowledgment mechanism to check the safe and sound arrival of data. We will discuss this feature further in the section on error control.

TCP Features

Numbering System

Although the TCP software keeps track of the segments being transmitted or received, there is no field for a segment number value in the segment header. Instead, there are two fields called the sequence number and the acknowledgment number. These two fields refer to the byte number and not the segment number.

Byte Number
TCP numbers all data bytes that are transmitted in a connection. Numbering is independent in each direction. When TCP receives bytes of data from a process, it stores them in the sending buffer and numbers them.

Sequence Number **After the bytes have been numbered, TCP assigns a sequence number to each segment that is being sent. The sequence number for each segment is the number of the first byte carried in that segment.**

Note

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

Suppose a TCP connection is transferring a file of 5000 bytes. The first byte is numbered 1O,00l.
What are the sequence numbers for each segment if data are sent in five segments, each carrying 1000 bytes?

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.

Flow Control

TCP, unlike UDP, provides *flow control. The receiver of the data controls the amount of* data that are to be sent by the sender. This is done to prevent the receiver from being overwhelmed with data. The numbering system allows TCP to use a byte-oriented flow control.

Error Control

To provide reliable service, TCP implements an error control mechanism. Although error control considers a segment as the unit of data for error detection (loss or corrupted segments), error control is byte-oriented, as we will see later.

Congestion Control

TCP, unlike UDP, takes into account congestion in the network. The amount of data sent by a sender is not only controlled by the receiver (flow control), but is also detennined by the level of congestion in the network.

Figure 23.16 *TCP segment format*

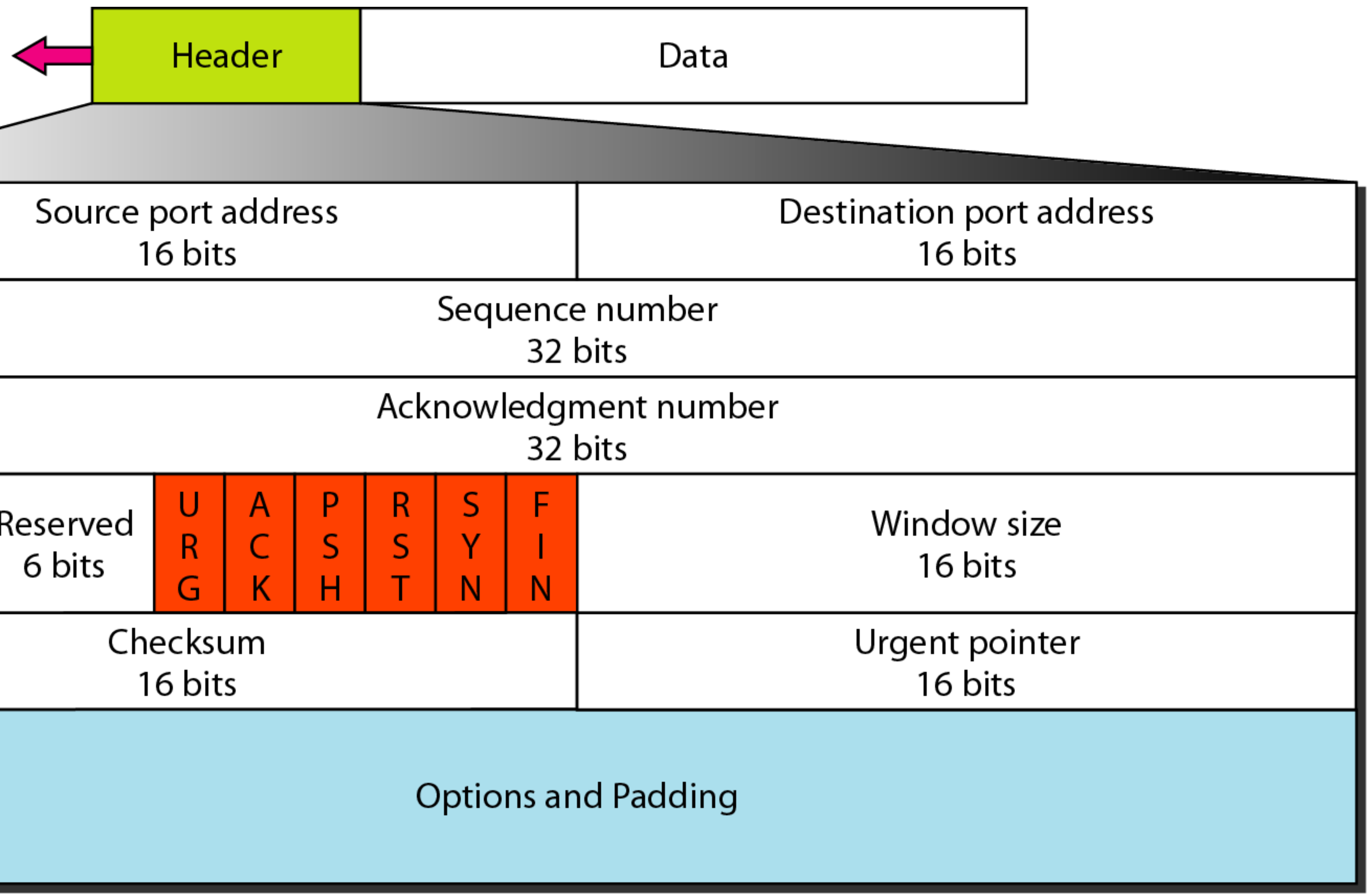


Figure 23.17 *Control field*

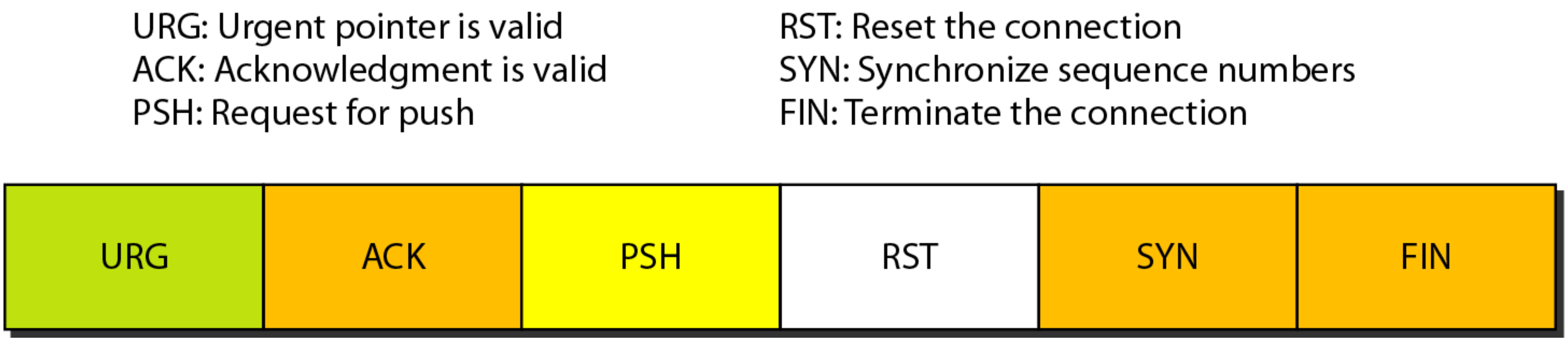
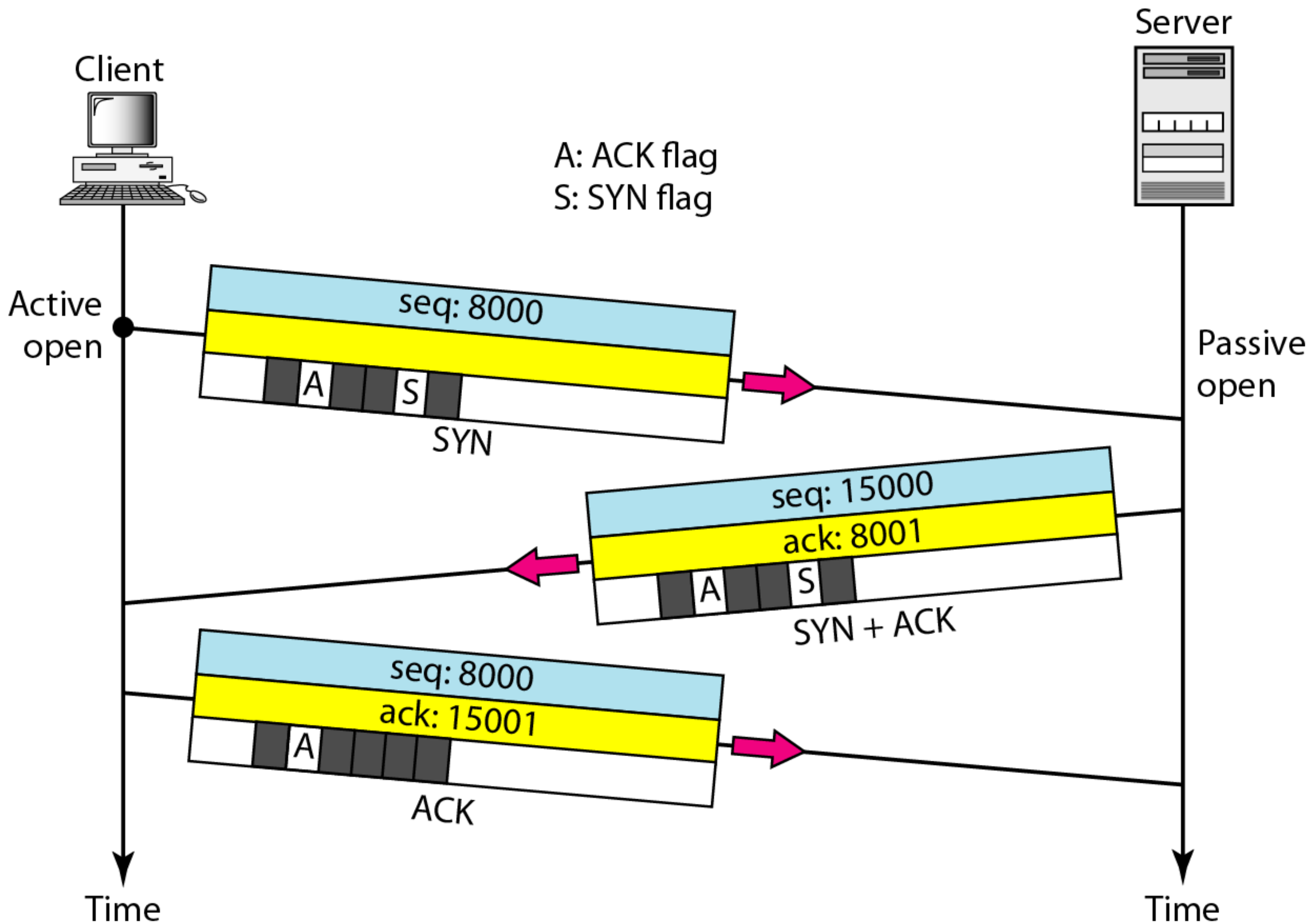


Table 23.3 ***Description of flags in the control field***

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*

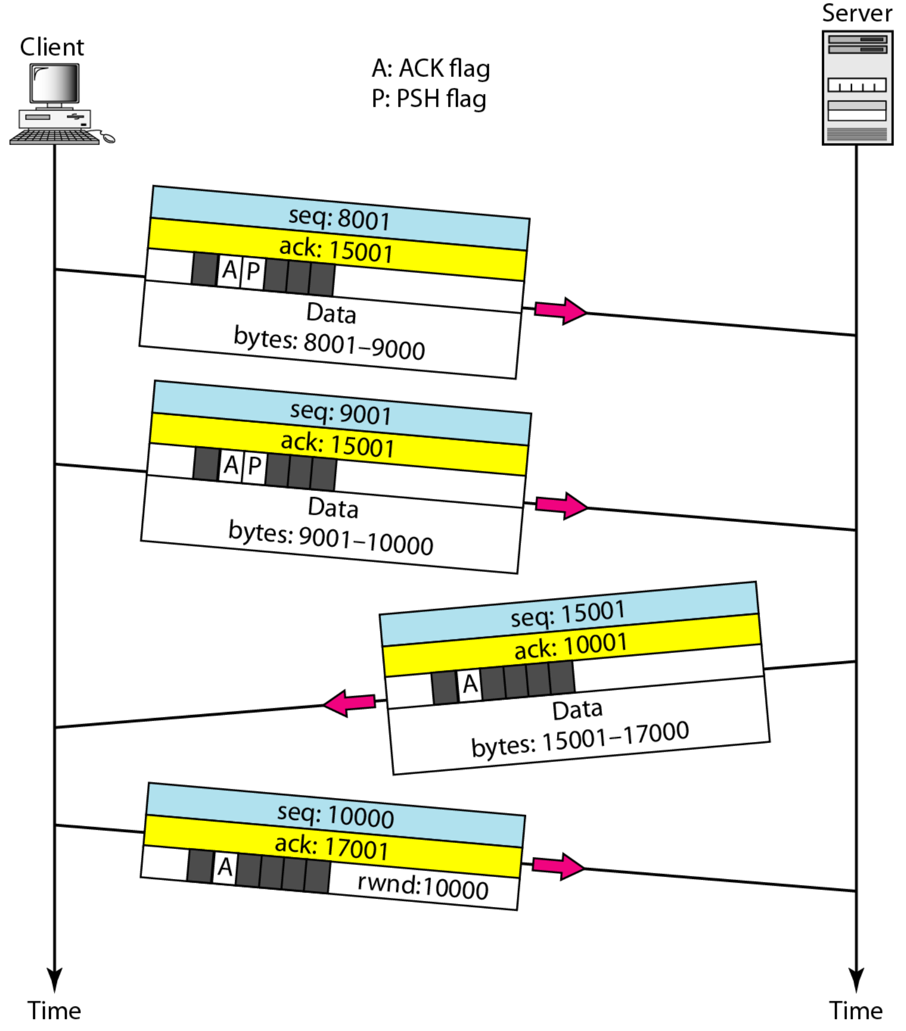
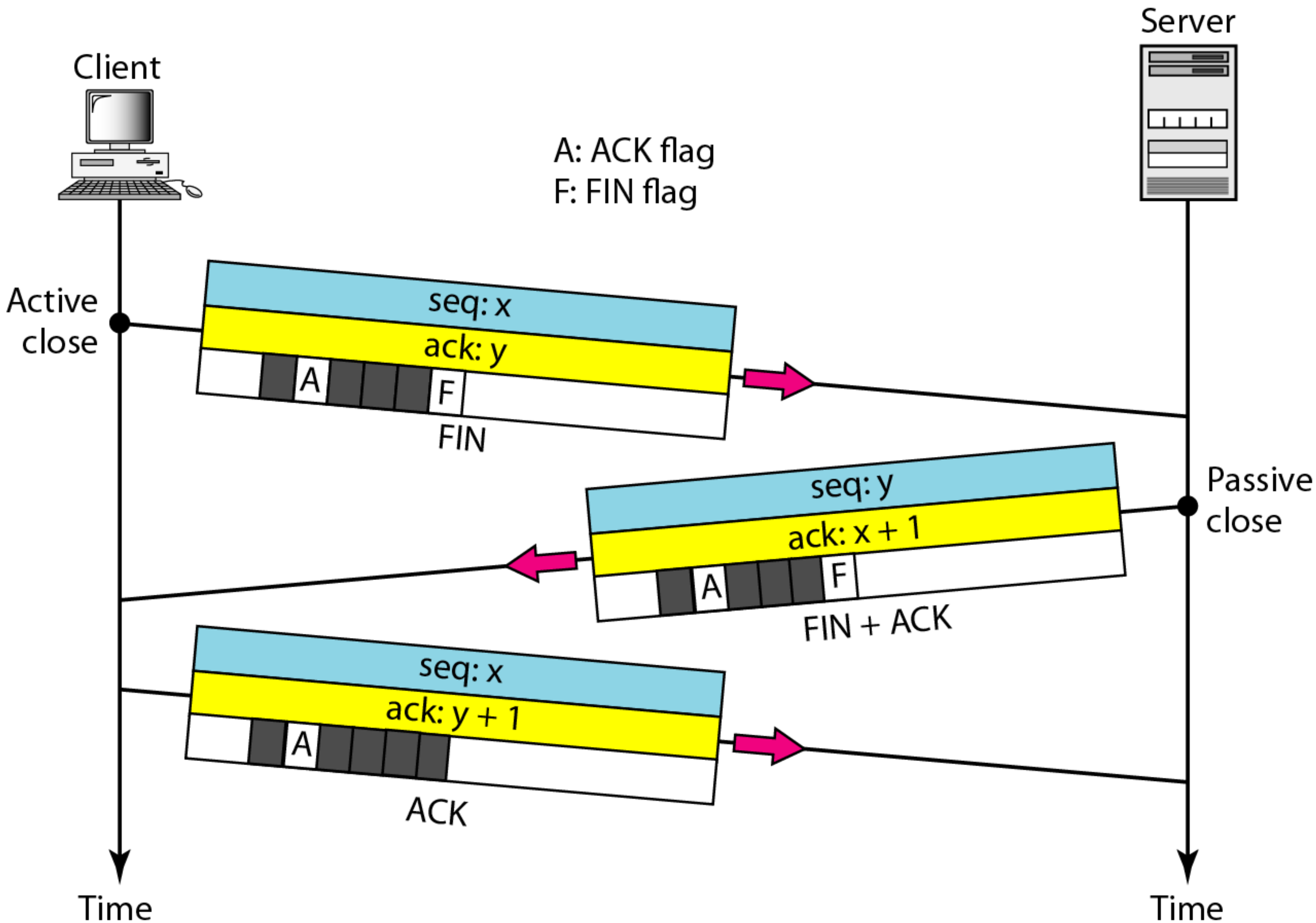


Figure 23.20 *Connection termination using three-way handshaking*



Note

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

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?