Computer Networks

By

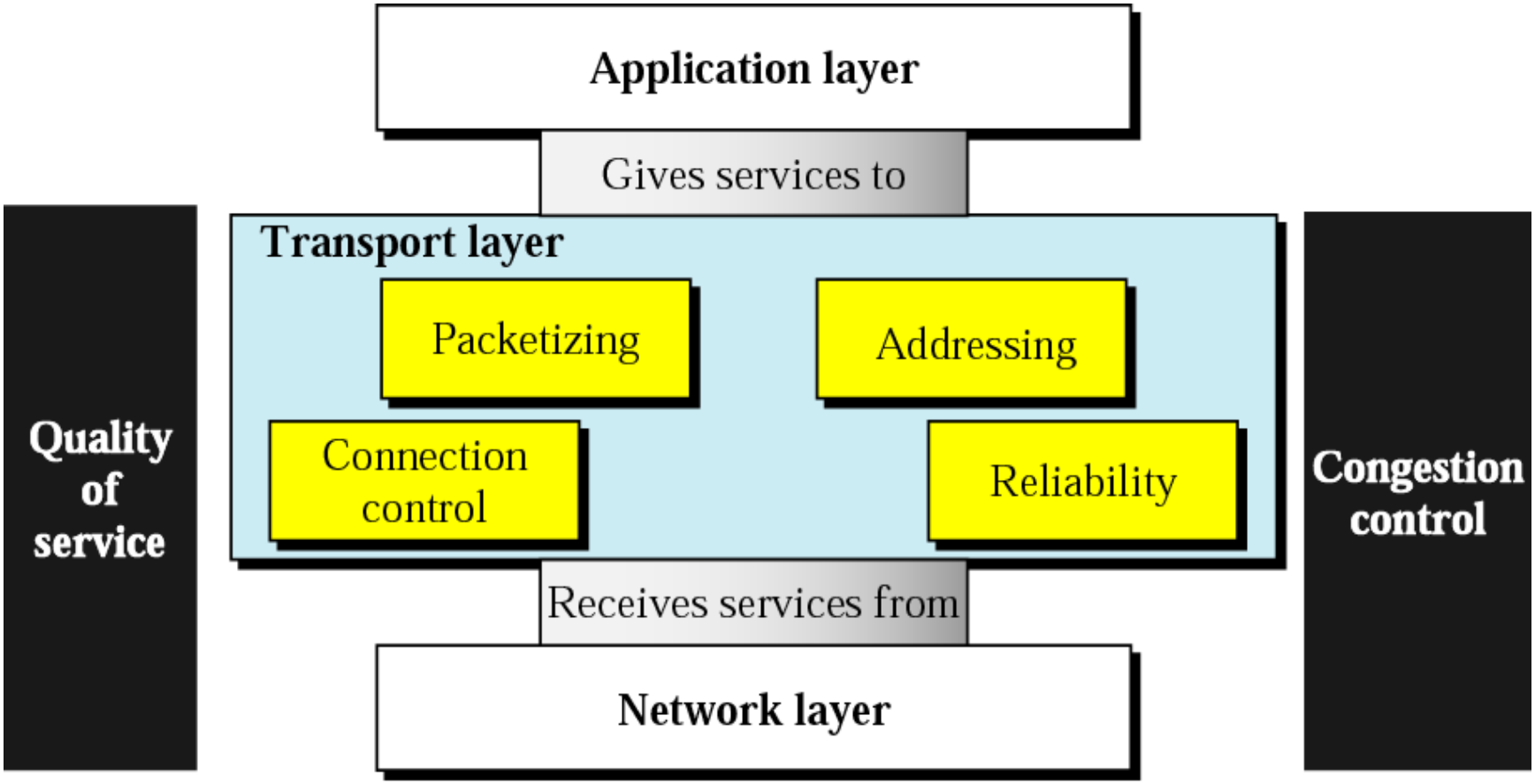
Lt Col Ishtiaq Kiani

(10 Sep 12 to 12 Jan 13)

PART 5

Transport Layer

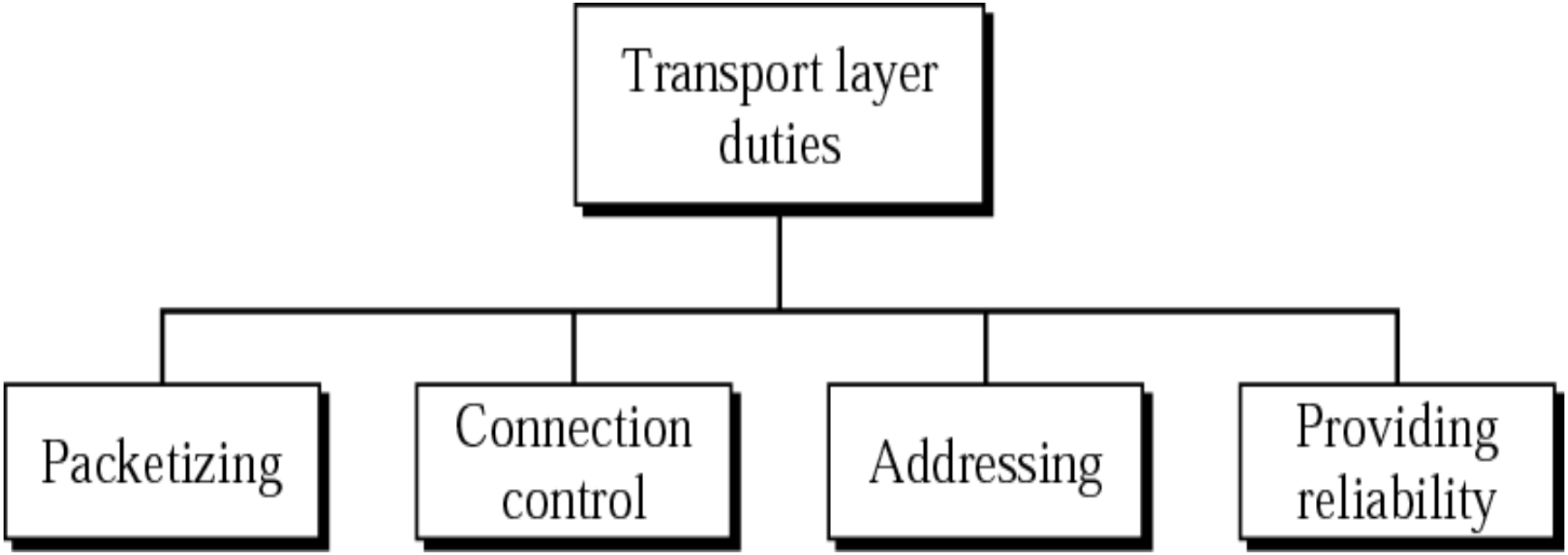
Position of the Transport Layer



Chapters

Chapter 23	Process to Process Delivery: UDP, TCP and SCTP
Chapter 24	Congestion Control and quality of Service

Duties – Transport Layer

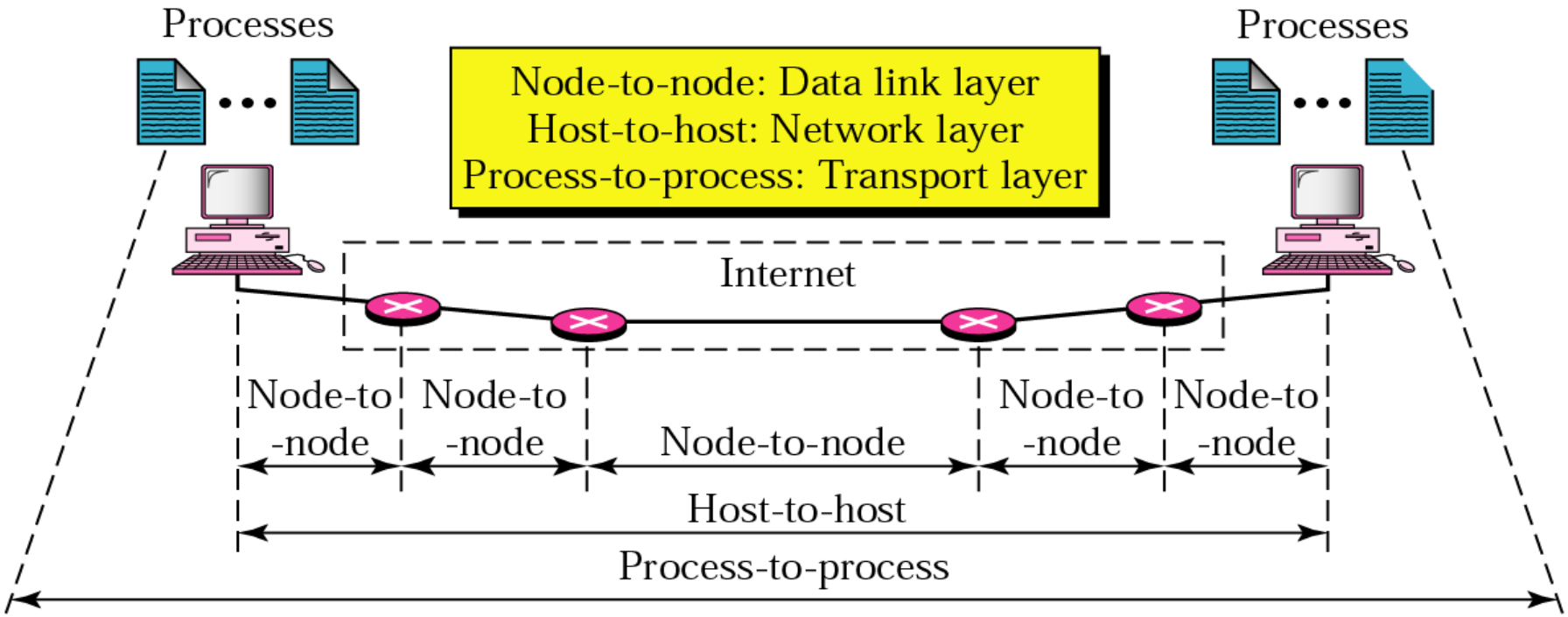


Chapter 23
Process to Process Delivery: UDP, TCP and SCTP

- **Process to Process Delivery**
- **User Datagram Protocol (UDP)**
- **TCP**
- **SCTP**

Process to Process Delivery

Transport Layer



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

Process to Process Delivery

Client Server Paradigm

Although there are several ways to achieve process-to-process communication, the most common one is through the **client/server paradigm.** A process on the local host, called a **client,** needs services from a process usually on the remote host, called a **server.**

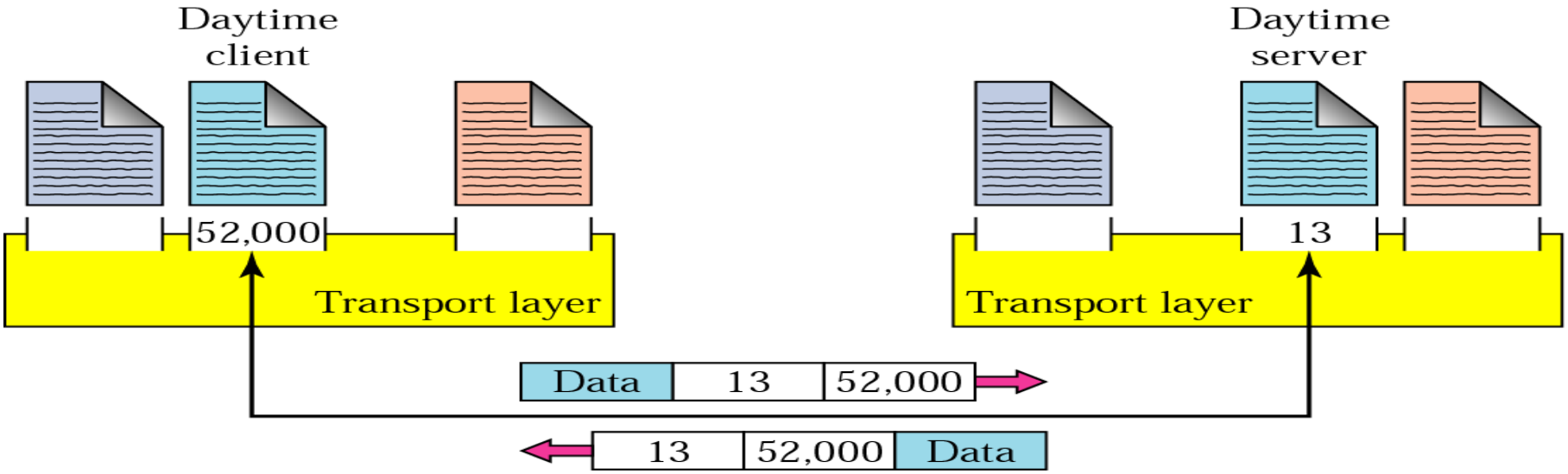
Both processes (client and server) have the same name. For example, to get the day and time from a remote machine, we need a Daytime client process running on the local host and a Daytime server process running on a remote machine.

Operating systems today support both multiuser and multiprogramming environments. A remote computer can run several server programs at the same time, just as local computers can run one or more client programs at the same time. For communication, we must define the following:

1. Local host
2. Local process
3. Remote host
4. Remote process

Process to Process Delivery

Addressing – Port Number

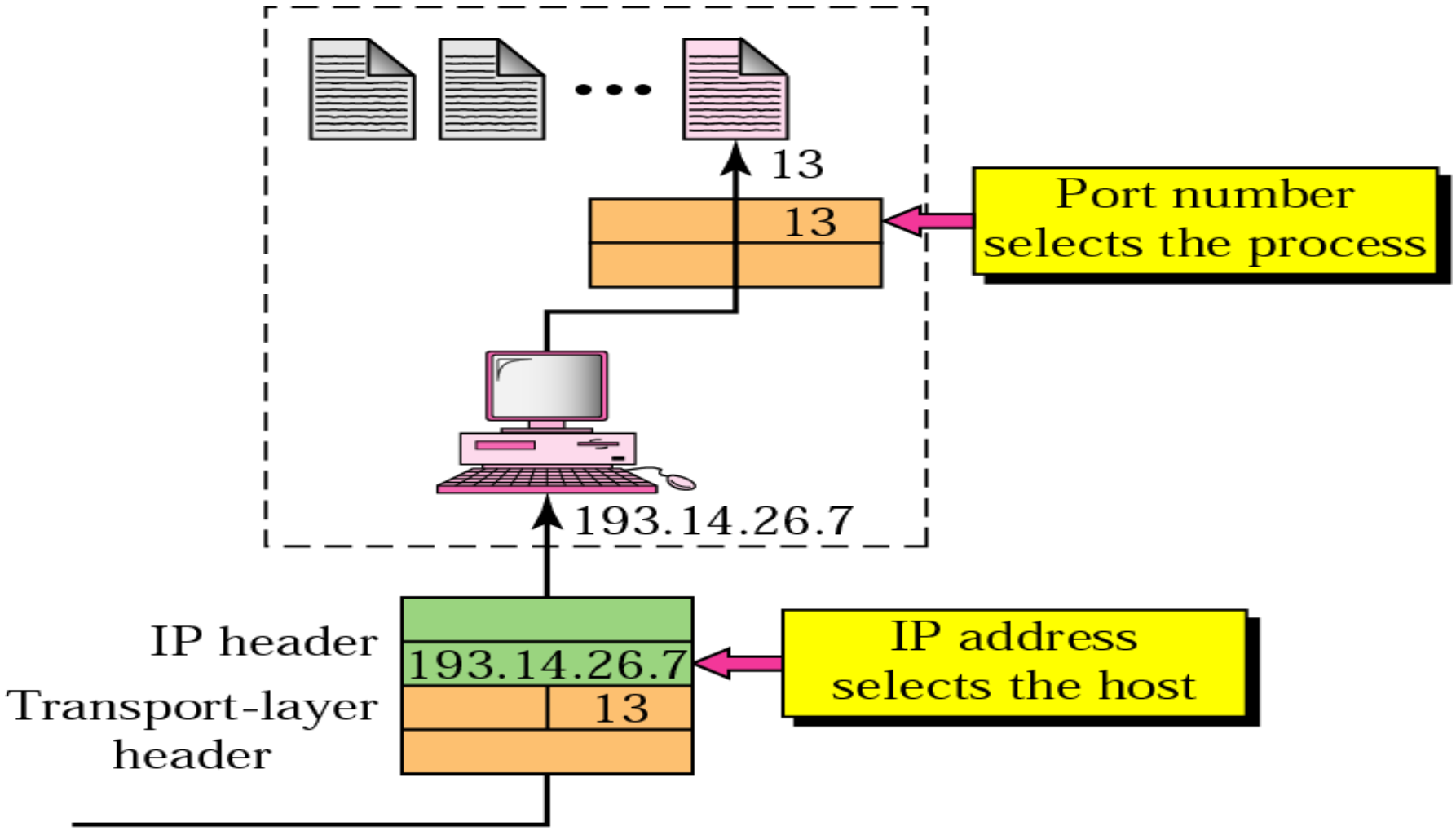


At the transport layer, we need a transport layer address, called a **port number,** to choose among multiple processes running on the destination host. The destination port number is needed for delivery; the source port number is needed for the reply.

In the Internet model, the port numbers are 16-bit integers between 0 and 65,535. The client program defines itself with a port number, chosen randomly by the transport layer software running on the client host. This is the **ephemeral port number.**

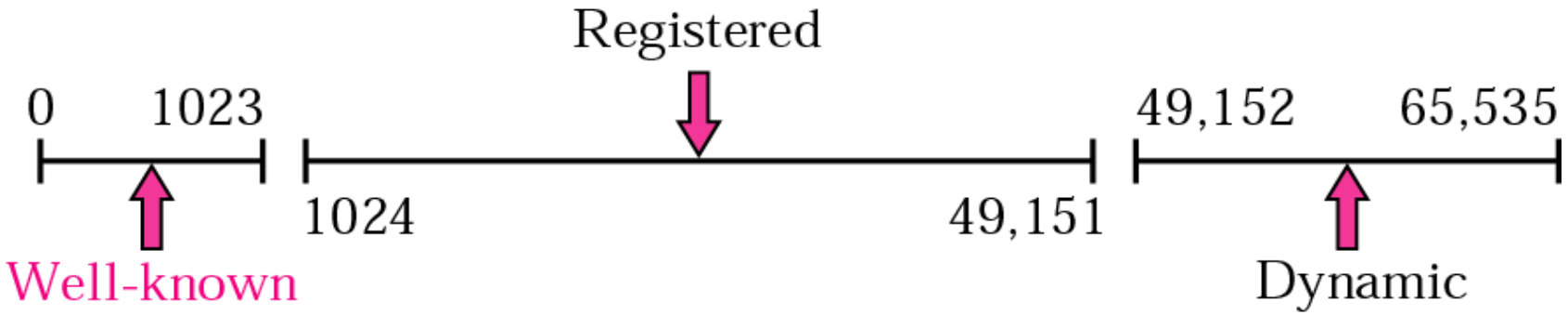
Process to Process Delivery

IP Address verses Port Address



Process to Process Delivery

IANA Ranges

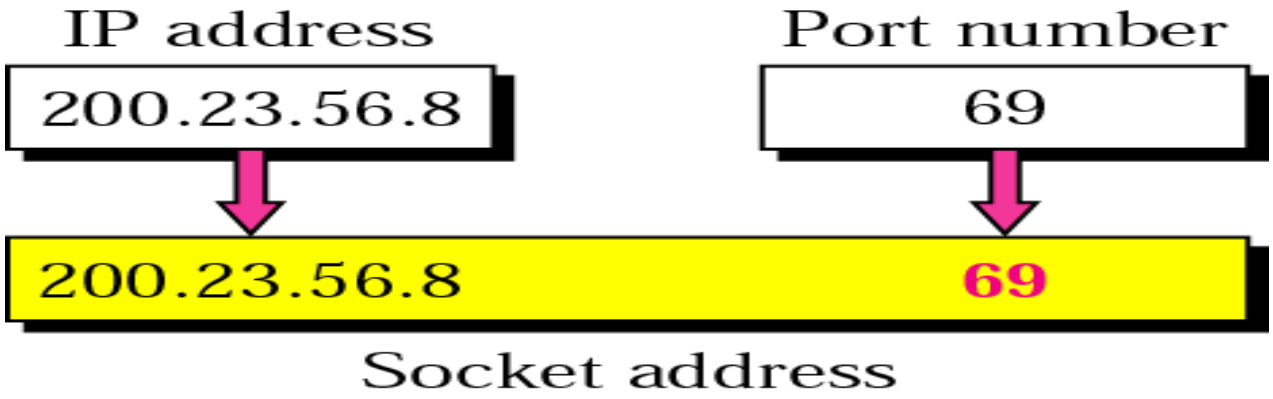


The IANA (Internet Assigned Number Authority) has divided the port numbers into three ranges: well known, registered, and dynamic (or private),

- **Well-known ports.** The ports ranging from 0 to 1023 are assigned and controlled by IANA. These are the well-known ports.
- **Registered ports.** The 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 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.

Process to Process Delivery

Socket Address

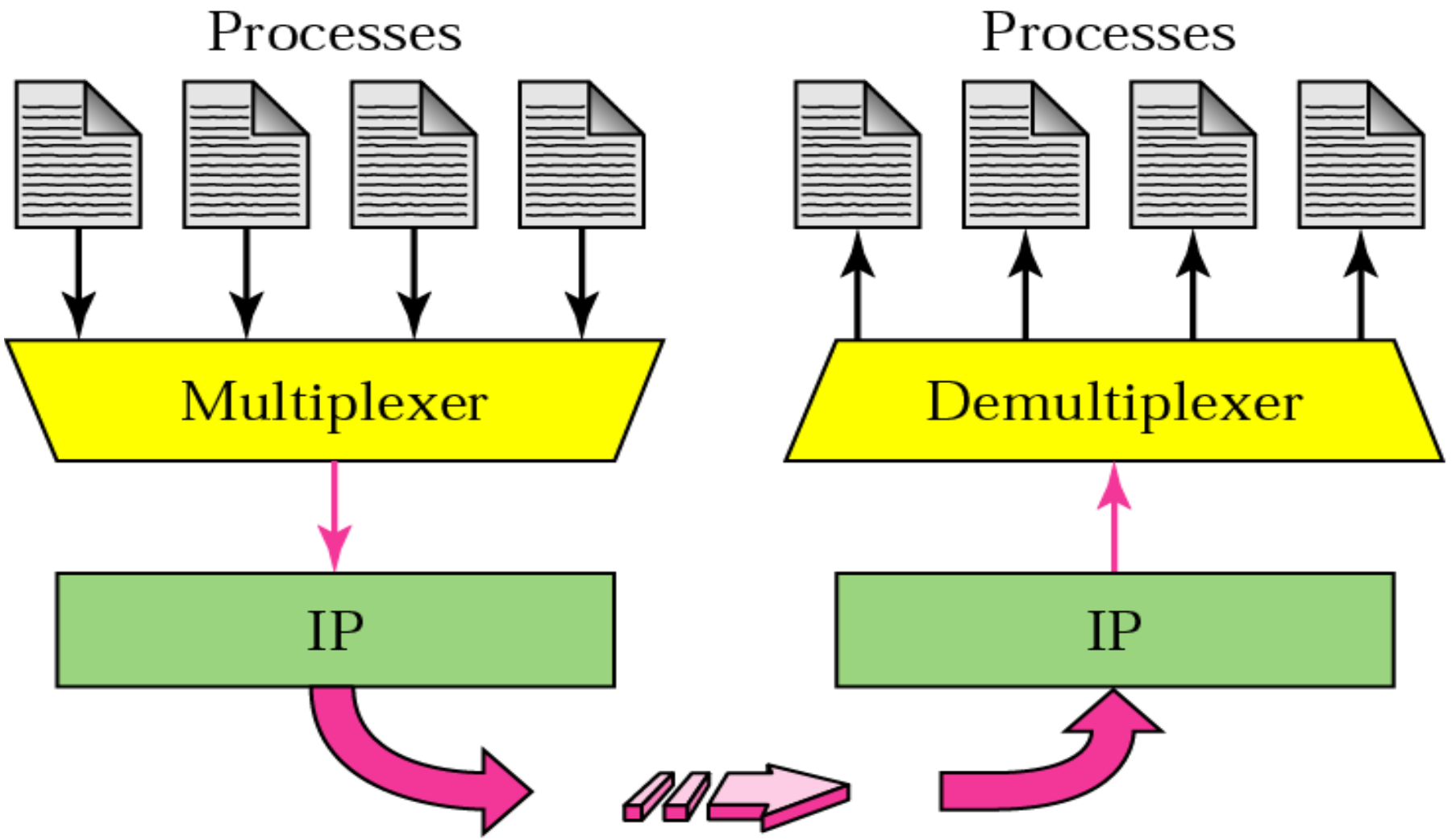


Process-to-process delivery needs two identifiers, IP address and the port number, at each end to make a connection. The combination of an IP address and a port number is called a **socket address.** The client socket address defines the client process uniquely just as the server socket address defines the server process uniquely

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.

Process to Process Delivery

Multiplexing & Demultiplexing



Process to Process Delivery

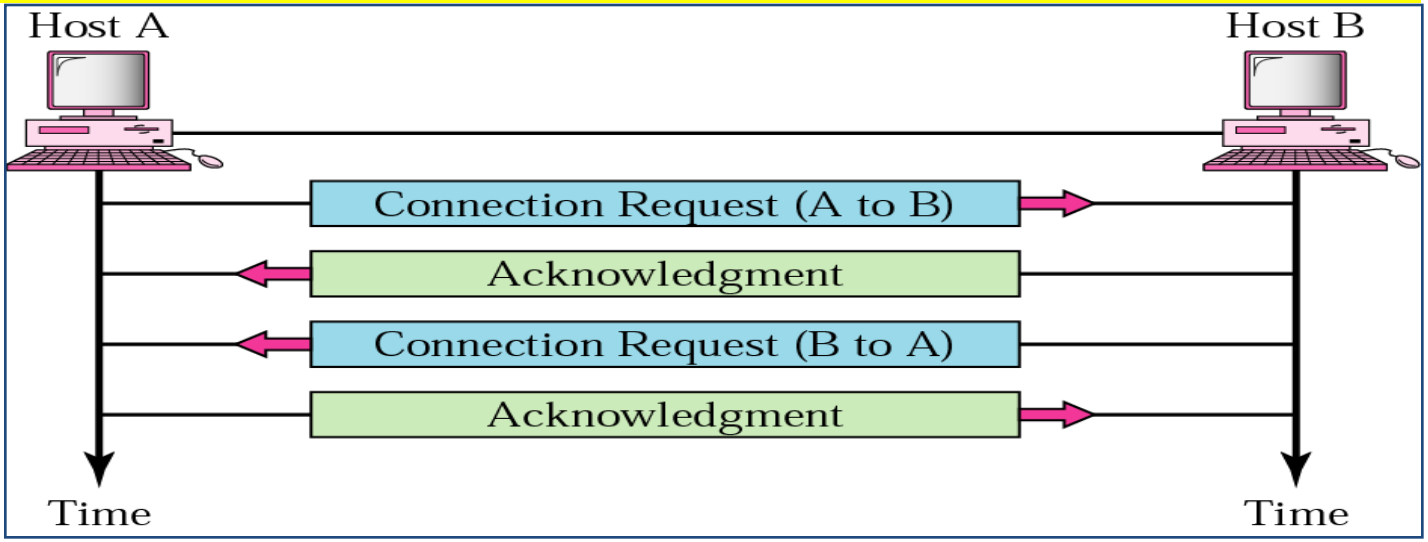
Connection Services

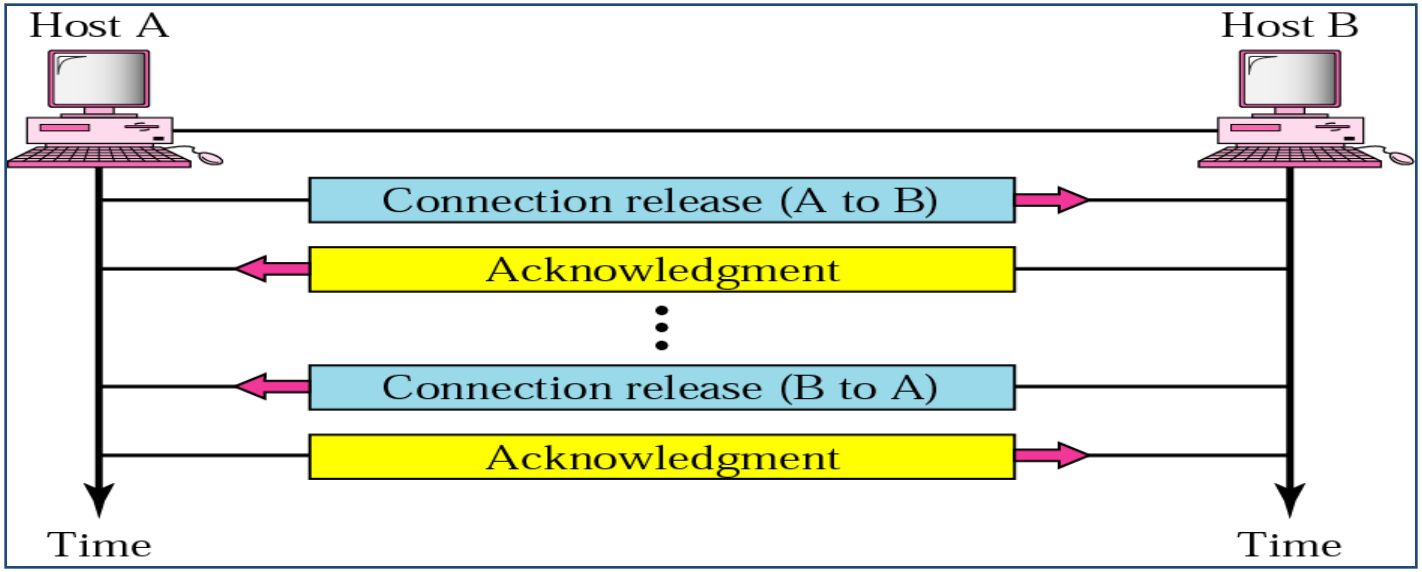
In a **connectionless service,** the packets are sent from one party to another with no need for connection establishment or connection release. The packets are not numbered; they may be delayed or lost or may arrive out of sequence. There is no acknowledgment either. We will see shortly that one of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

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. We will see shortly that TCP and SCTP are connection-oriented protocols.

Process to Process Delivery

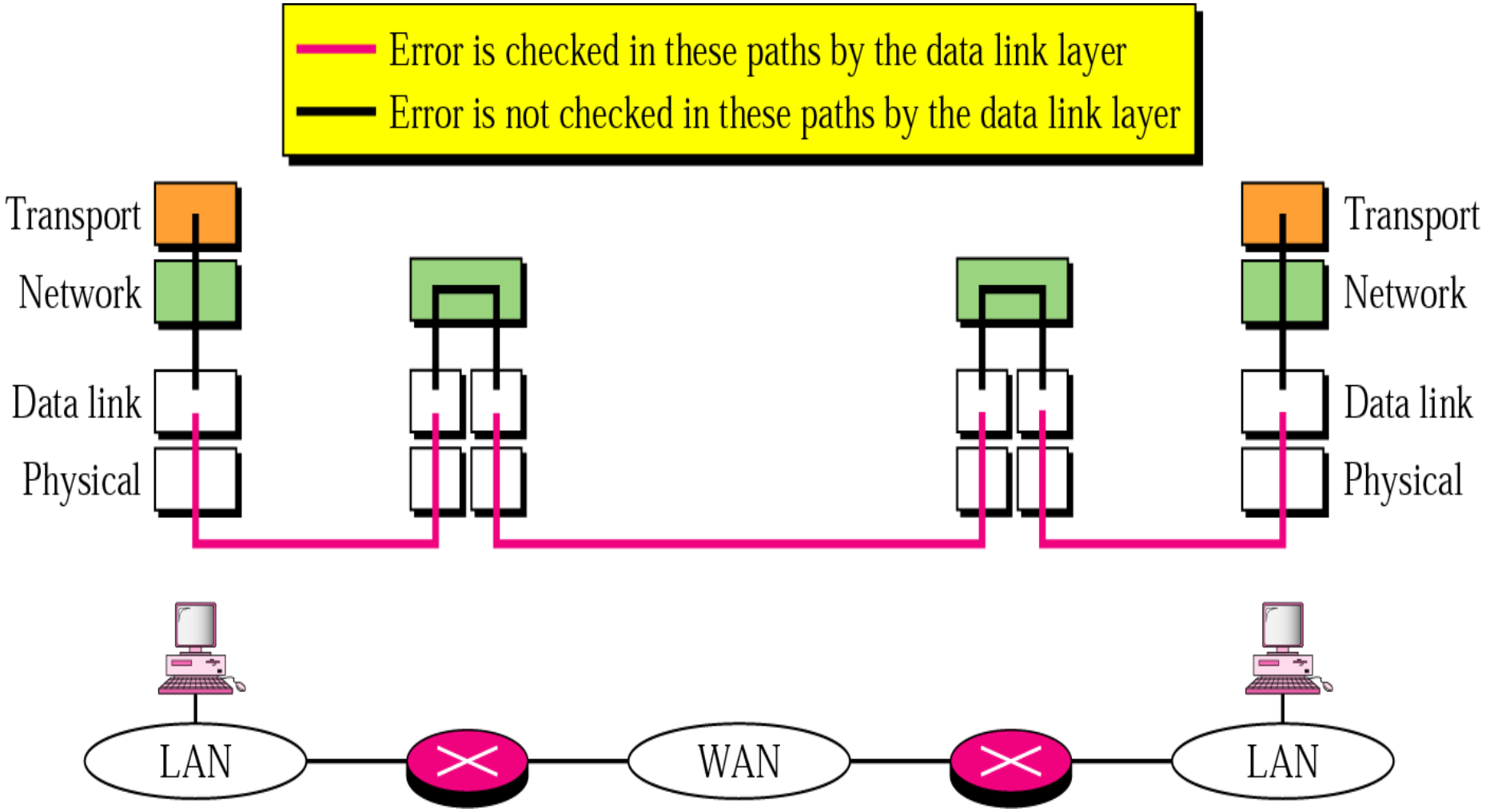
Connection Services





Process to Process Delivery

Error Control



User Datagram Protocol (UDP)

Port Numbers

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. Also, it performs very limited error checking.

UDP is a connectionless, unreliable protocol that has no flow and error control. It uses port numbers to multiplex data from the application layer.

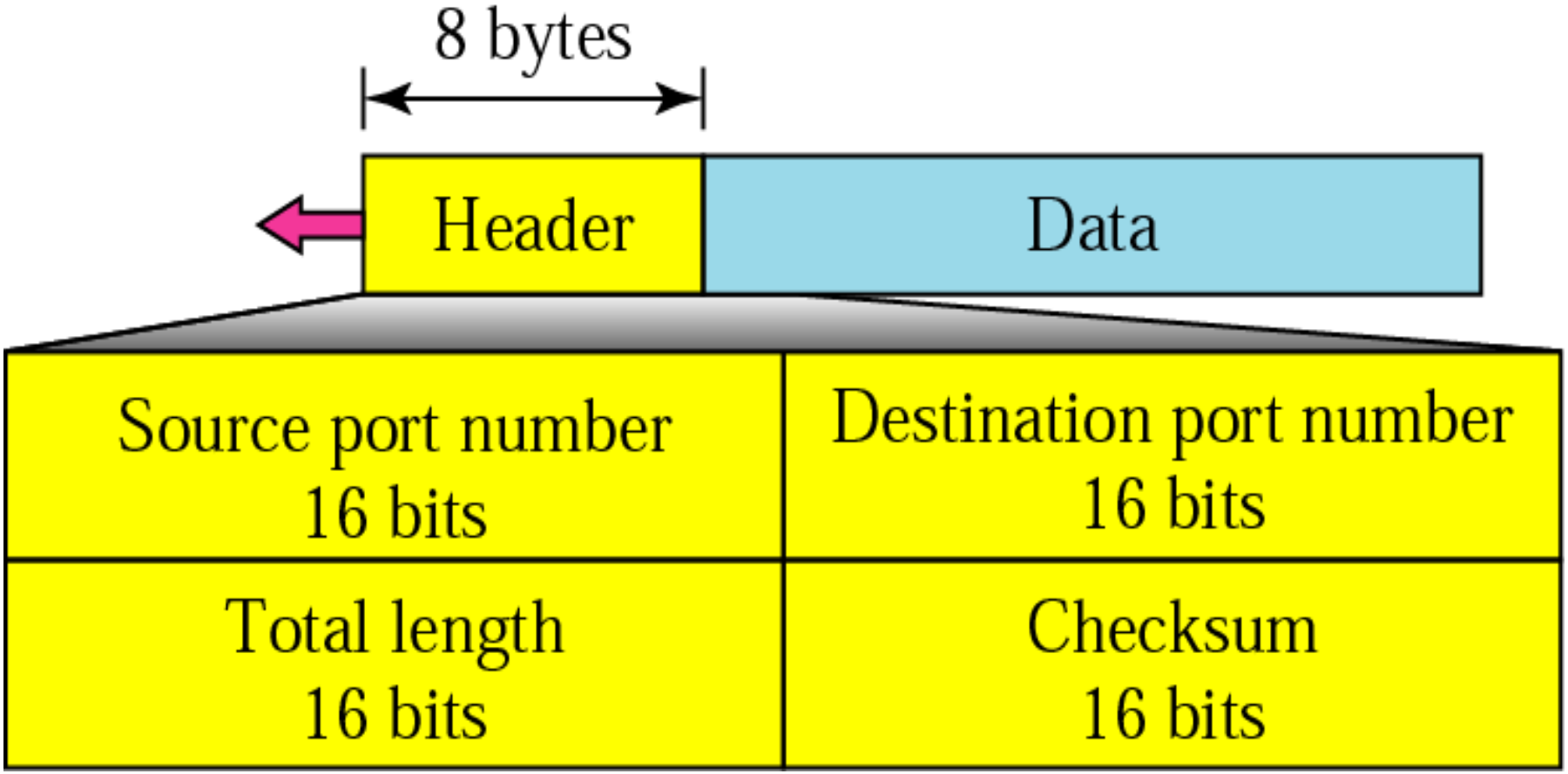
User Datagram Protocol (UDP)

Port Numbers

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)

User Datagram Protocol (UDP)

User Datagram Format



User Datagram Protocol (UDP)

User Datagram Format

Source port number. This is the port number used by the process running on the source host. It is 16 bits long, which means that the port number can range from 0 to 65,535. If the source host is the client (a client sending a request), the port number, in most cases, is an ephemeral port number requested by the process and chosen by the UDP software running on the source host. If the source host is the server (a server sending a response), the port number, in most cases, is a well-known port number.

Destination port number. This is the port number used by the process running on the destination host. It is also 16 bits long. If the destination host is the server (a client sending a request), the port number, in most cases, is a well-known port number. If the destination host is the client (a server sending a response), the port number, in most cases, is an ephemeral port number. In this case, the server copies the ephemeral port number it has received in the request packet.

User Datagram Protocol (UDP)

User Datagram Format

Length. This is a 16-bit field that defines the total length of the user datagram, header plus data. The 16 bits can define a total length of 0 to 65,535 bytes. However, the total length needs to be much less because a UDP user datagram is stored in an IP datagram with a total length of 65,535 bytes.

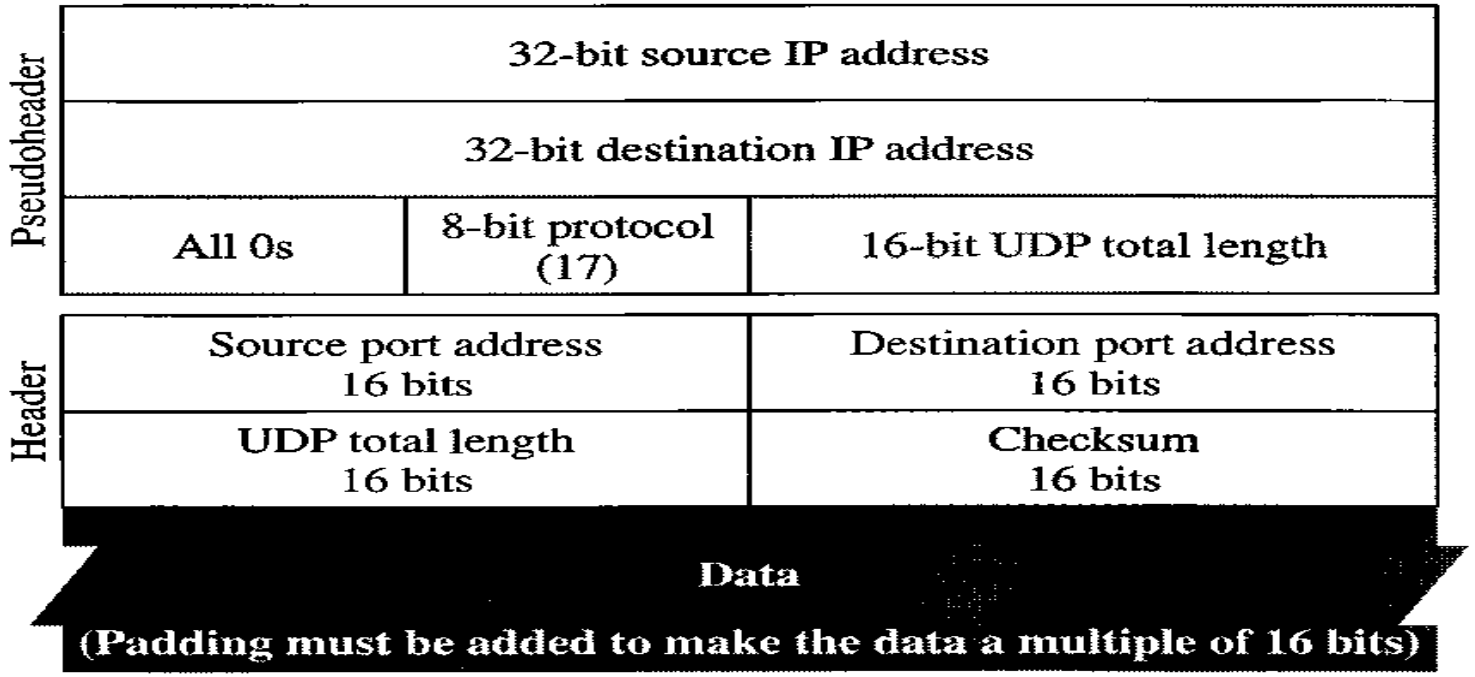
The UDP checksum calculation is different from the one for IP and ICMP. Here the checksum includes three sections: a pseudoheader, the UDP header, and the data coming from the application layer.

There is no error control mechanism in UDP except for the checksum. This means that 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 silently discarded.

The lack of **flow control** and **error control** means that the process using UDP should provide these mechanisms.

User Datagram Protocol (UDP)

User Datagram Format - Checksum



The calculation of the checksum and its inclusion in a user datagram are optional. If the checksum is not calculated, the field is filled with 1s. Note that a calculated checksum can never be all 1s because this implies that the sum is all 0s, which is impossible because it requires that the value of fields to be 0s.

User Datagram Protocol (UDP)

User Datagram Format - Checksum

153.18.8.105		
171.2.14.10		
All 0s	17	15

1087	13
15	All 0s

T	E	S	T
I	N	G	All 0s

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

User Datagram Protocol (UDP)

User 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 such as FTP that needs to send bulk data (see Chapter 26).

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. It can easily use UDP.

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 (see Chapter 28).

UDP is used for some route updating protocols such as Routing Information Protocol (RIP) (see Chapter 22).

Transmission Control Protocol (TCP)

Background

The second transport layer protocol we discuss in this chapter is called **Transmission Control Protocol (TCP).** TCP, like UDP, is a process-to-process (program-to-program) protocol. TCP, therefore, like UDP, uses port numbers. Unlike UDP, 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.

In brief, TCP is called a *connection-oriented, reliable* transport protocol. It adds connection-oriented and reliability features to the services of IP.

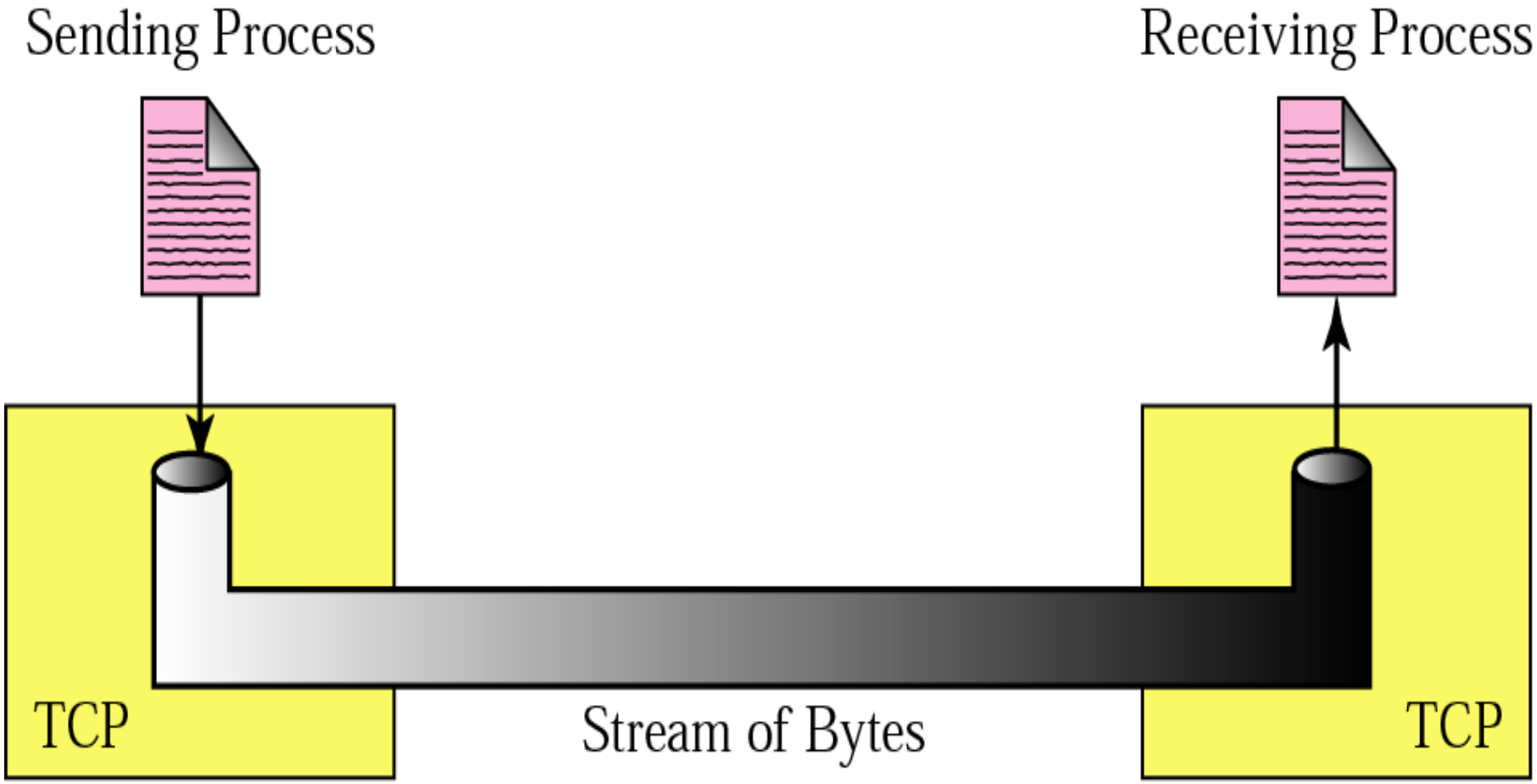
Transmission Control Protocol (TCP)

Port Number

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

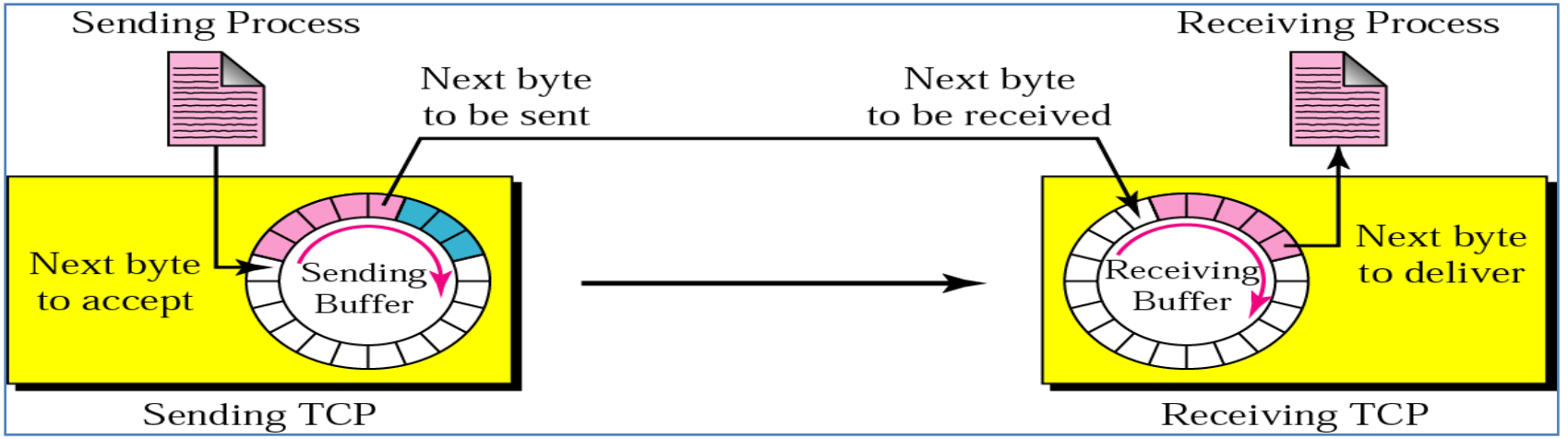
Transmission Control Protocol (TCP)

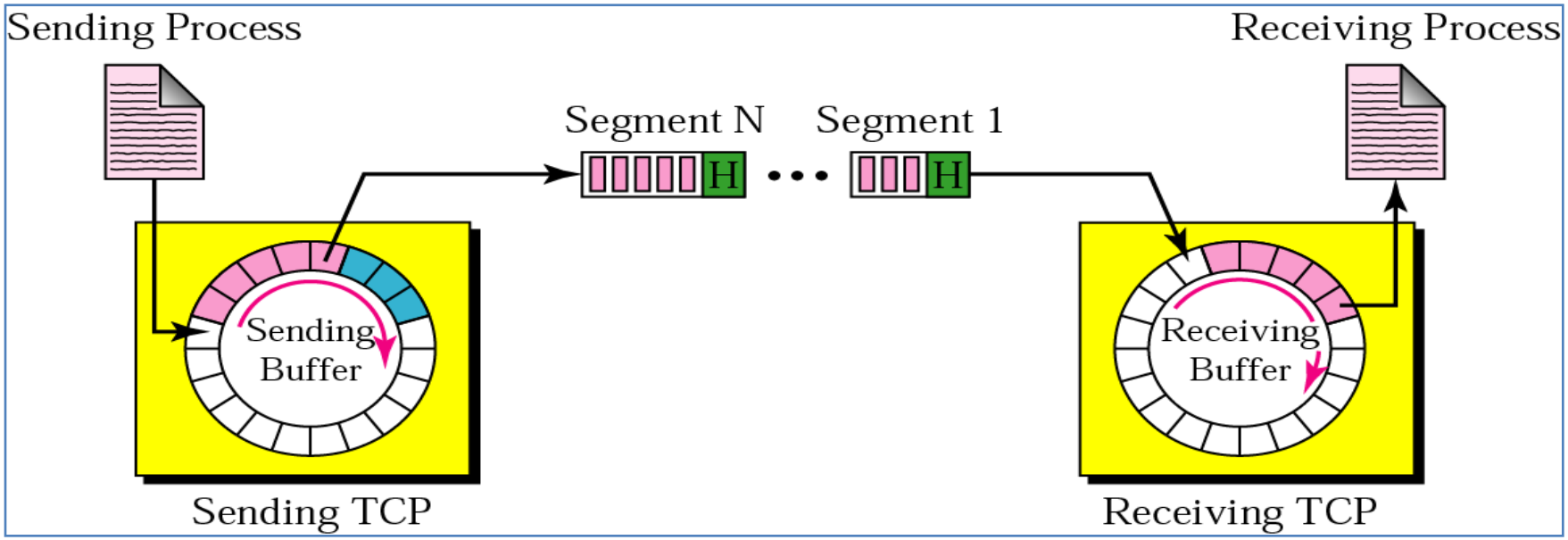
Stream Delivery



Transmission Control Protocol (TCP)

Sender/ Rx Buffers & TCP Segment





Transmission Control Protocol (TCP)

Communication and Connection Services

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

TCP, unlike UDP, is a connection-oriented protocol. When a process at site A wants to send and receive data from another process at site B, the following occurs:

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

Transmission Control Protocol (TCP)

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. The numbering does not necessarily start from 0. Instead, TCP generates a random number between 0 and $2^{32} - 1$ for the number of the first byte. For example, if the random number happens to be 1057 and the total data to be sent are 6000 bytes, the bytes are numbered from 1057 to 7056. We will see that byte numbering is used for flow and error control.

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

Transmission Control Protocol (TCP)

TCP Features

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.

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

Ack Number the acknowledgment number defines the number of the next byte that the party expects to receive. In addition, the acknowledgment number is cumulative, which means that the party takes the number of the last byte that it has received, safe and sound, adds 1 to it, and announces this sum as the acknowledgment number. The term *cumulative* here means that if a party uses 5643 as an acknowledgment number, it has received all bytes from the beginning up to 5642. Note that this does not mean that the party has received 5642 bytes because the first byte number does not have to start from 0.

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.

Transmission Control Protocol (TCP)

Example

Imagine a TCP connection is transferring a file of 6000 bytes. The first byte is numbered 10010. What are the sequence numbers for each segment if data are sent in five segments with the first four segments carrying 1000 bytes and the last segment carrying 2000 bytes?

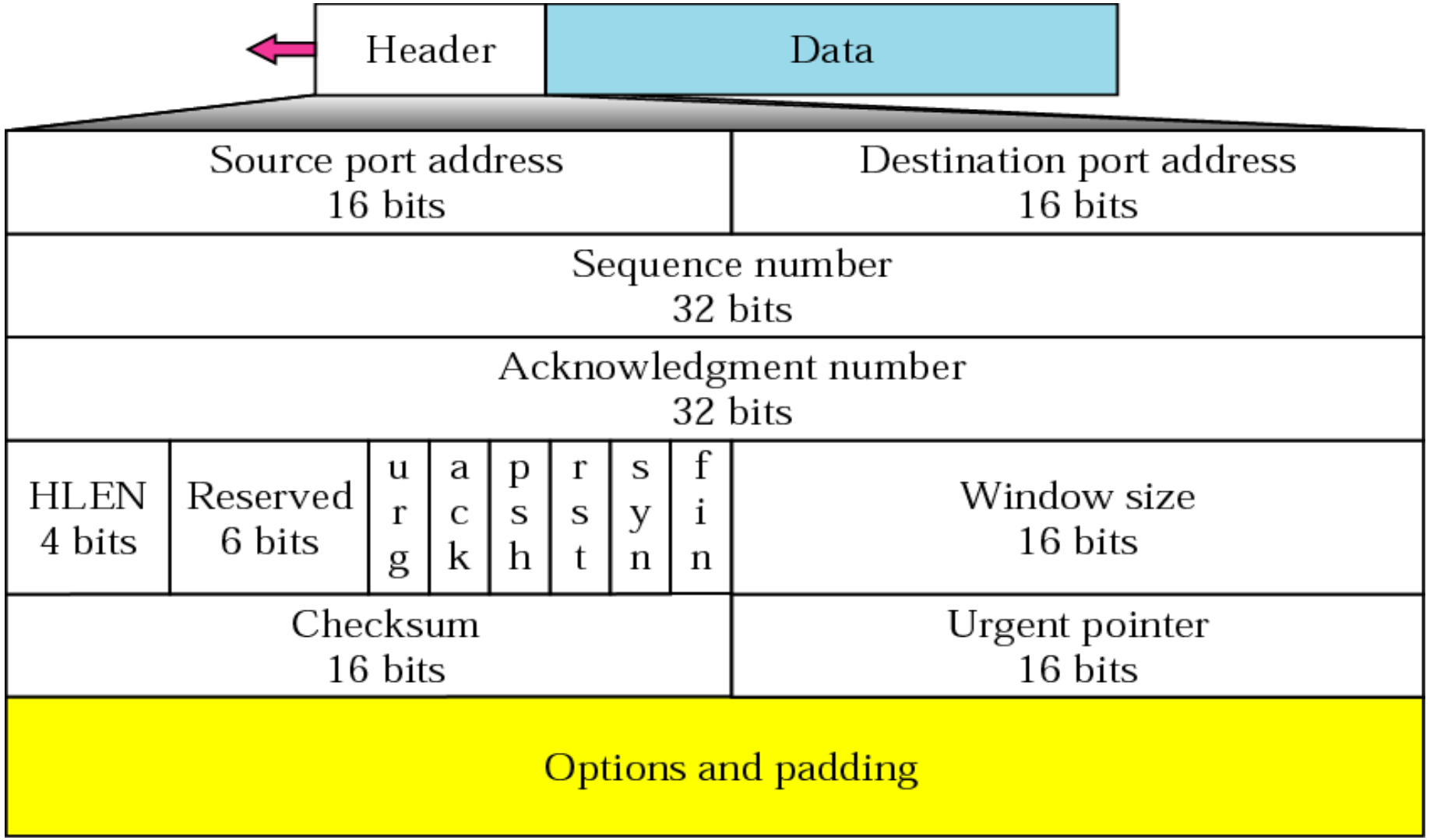
Solution

The following shows the sequence number for each segment:

Segment 1 ==> sequence number: 10,010 (range: 10,010 to 11,009)
Segment 2 ==> sequence number: 11,010 (range: 11,010 to 12,009)
Segment 3 ==> sequence number: 12,010 (range: 12,010 to 13,009)
Segment 4 ==> sequence number: 13,010 (range: 13,010 to 14,009)
Segment 5 ==> sequence number: 14,010 (range: 14,010 to 16,009)

Transmission Control Protocol (TCP)

TCP Segment Format



Transmission Control Protocol (TCP)

TCP Segment Format

Source port address. This is a 16-bit field that defines the port number of the application program in the host that is sending the segment. This serves the same purpose as the source port address in the UDP header.

Destination port address. This is a 16-bit field that defines the port number of the application program in the host that is receiving the segment. This serves the same purpose as the destination port address in the UDP header.

Sequence number. This 32-bit field defines the number assigned to the first byte of data contained in this segment. As we said before, TCP is a stream transport protocol. To ensure connectivity, each byte to be transmitted is numbered. The sequence number tells the destination which byte in this sequence comprises the first byte in the segment. During connection establishment, each party uses a random number generator to create an **initial sequence number (ISN),** which is usually different in each direction.

Acknowledgnient number. This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party. If the receiver

Transmission Control Protocol (TCP)

TCP Segment Format

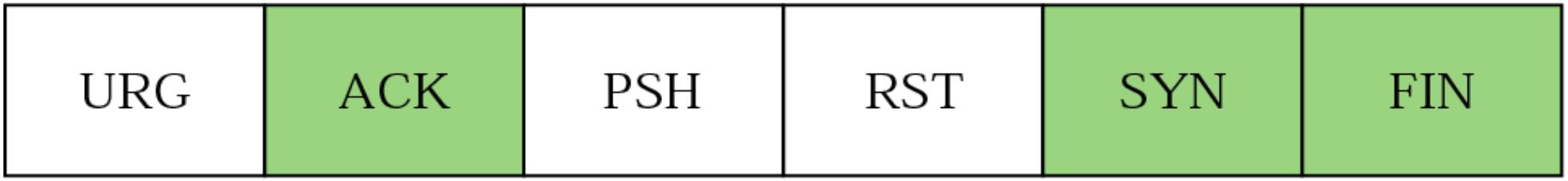
of the segment has successfully received byte number x from the other party, it defines $x + 1$ as the acknowledgment number. Acknowledgment and data can be piggybacked together.

Header length. This 4-bit field indicates the number of 4-byte words in the TCP header. The length of the header can be between 20 and 60 bytes. Therefore, the value of this field can be between 5 ($5 \times 4 = 20$) and 15 ($15 \times 4 = 60$).

Reserved. This is a 6-bit field reserved for future use.

Control. This field defines 6 different control bits or flags as shown in Figure 23.17. One or more of these bits can be set at a time.

URG: Urgent pointer is valid	RST: Reset the connection
ACK: Acknowledgment is valid	SYN: Synchronize sequence numbers
PSH: Request for push	FIN: Terminate the connection



Transmission Control Protocol (TCP)

TCP Segment Format

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	The connection must be reset.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.

Transmission Control Protocol (TCP)

TCP Segment Format

Window size. This field defines the size of the window, in bytes, that the other party must maintain. Note that the length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes. This value is normally referred to as the receiving window (rwnd) and is determined by the receiver. The sender must obey the dictation of the receiver in this case.

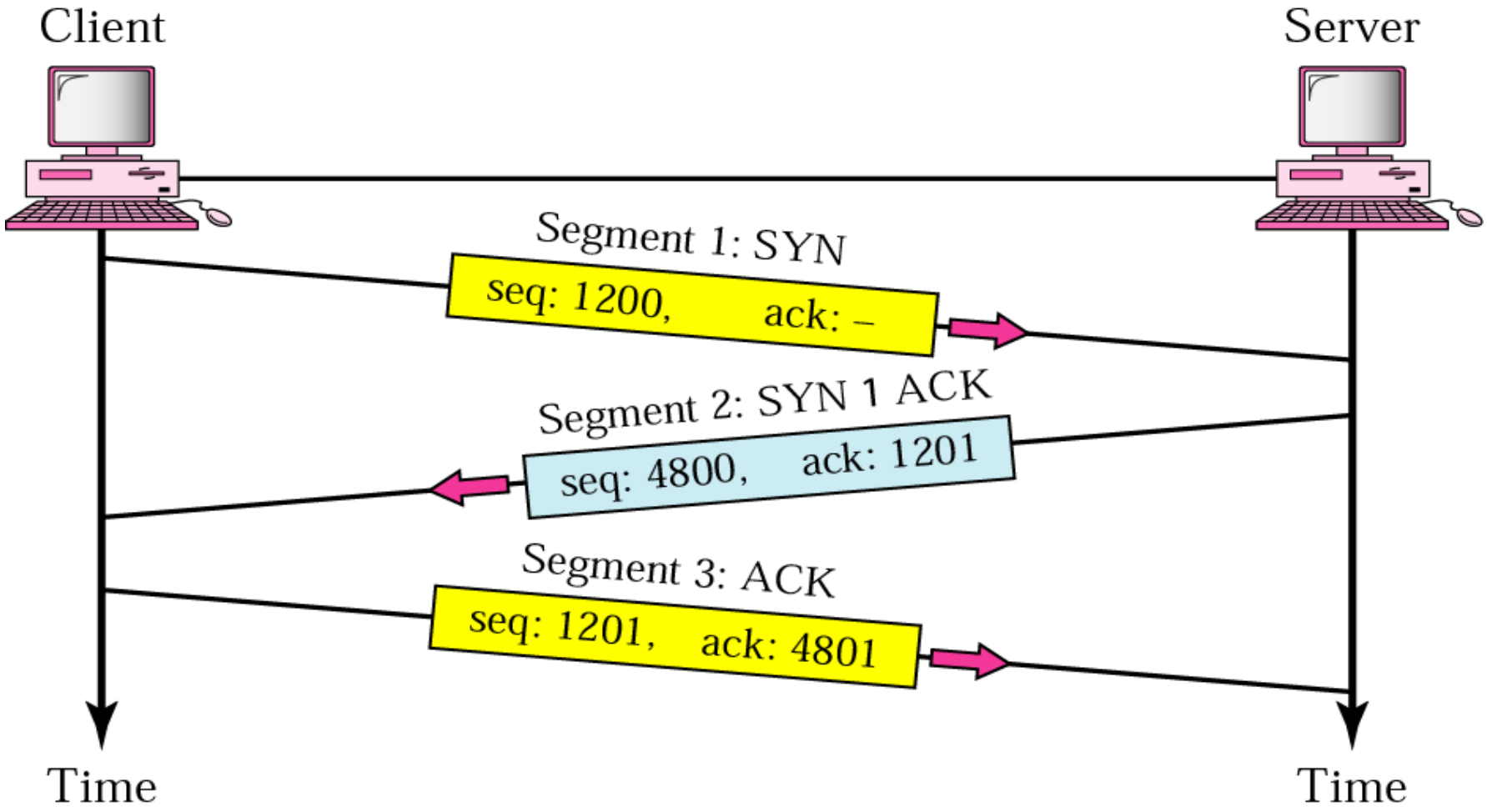
Checksum. This 16-bit field contains the checksum. The calculation of the checksum for TCP follows the same procedure as the one described for UDP. However, the inclusion of the checksum in the UDP datagram is optional, whereas the inclusion of the checksum for TCP is mandatory. The same pseudoheader, serving the same purpose, is added to the segment. For the TCP pseudoheader, the value for the protocol field is 6.

Urgent pointer. This 16-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data. It defines the number that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment. This will be discussed later in this chapter.

Options. There can be up to 40 bytes of optional information in the TCP header. We will not discuss these options here; please refer to the reference list for more information.

Transmission Control Protocol (TCP)

TCP Connection Establishment



Transmission Control Protocol (TCP)

TCP Connection Establishment

1. The client sends the first segment, a SYN segment, in which only the SYN flag is set. This segment is for synchronization of sequence numbers. It consumes one sequence number. When the data transfer starts, the sequence number is incremented by 1. We can say that the SYN segment carries no real data, but we can think of it as containing 1 imaginary byte.

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

---

2. The server sends the second segment, a SYN + ACK segment, with 2 flag bits set: SYN and ACK. This segment has a dual purpose. It is a SYN segment for communication in the other direction and serves as the acknowledgment for the SYN segment. It consumes one sequence number.

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

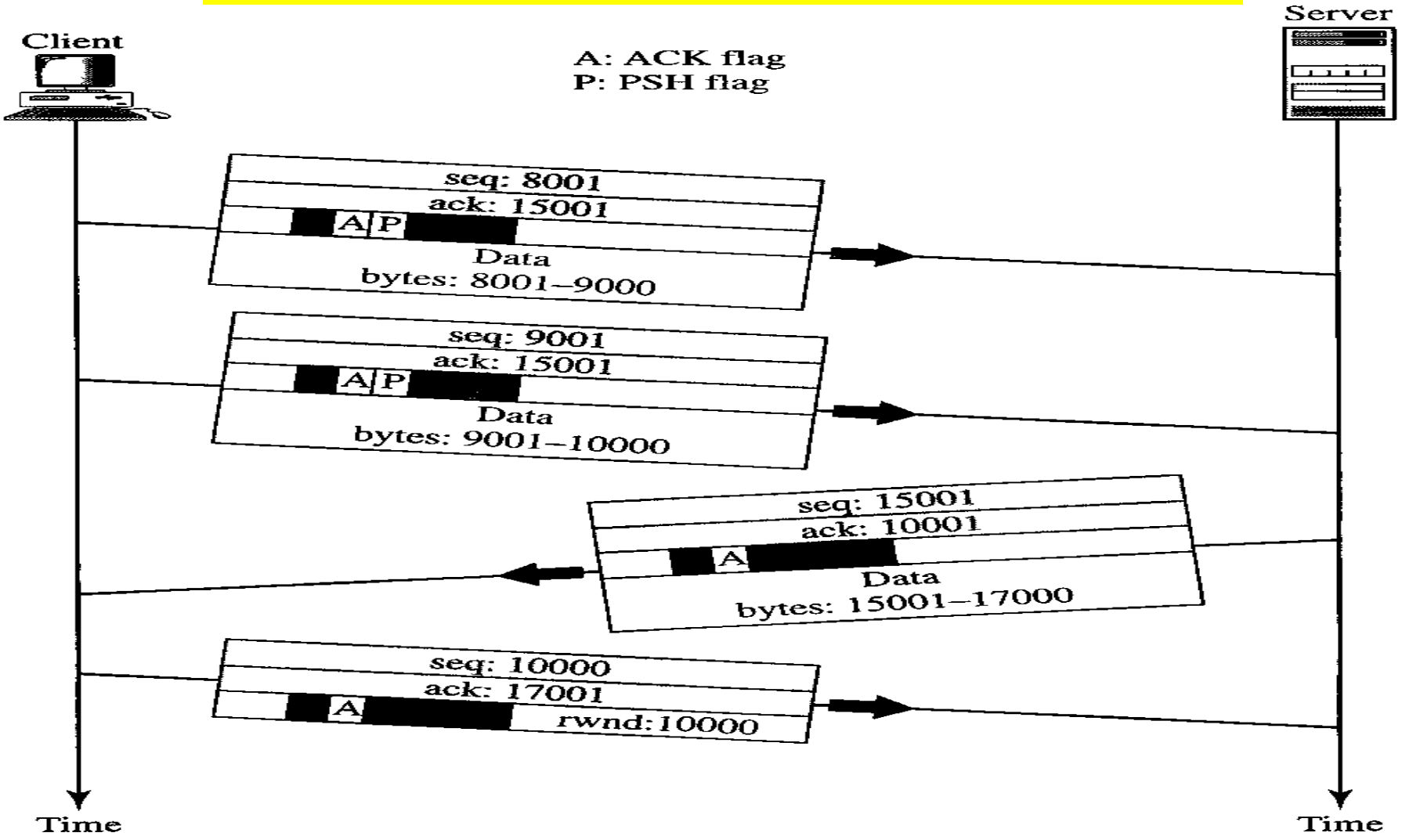
---

3. The client sends the third segment. This is just an ACK segment. It acknowledges the receipt of the second segment with the ACK flag and acknowledgment number field. Note that the sequence number in this segment is the same as the one in the SYN segment; the ACK segment does not consume any sequence numbers.

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

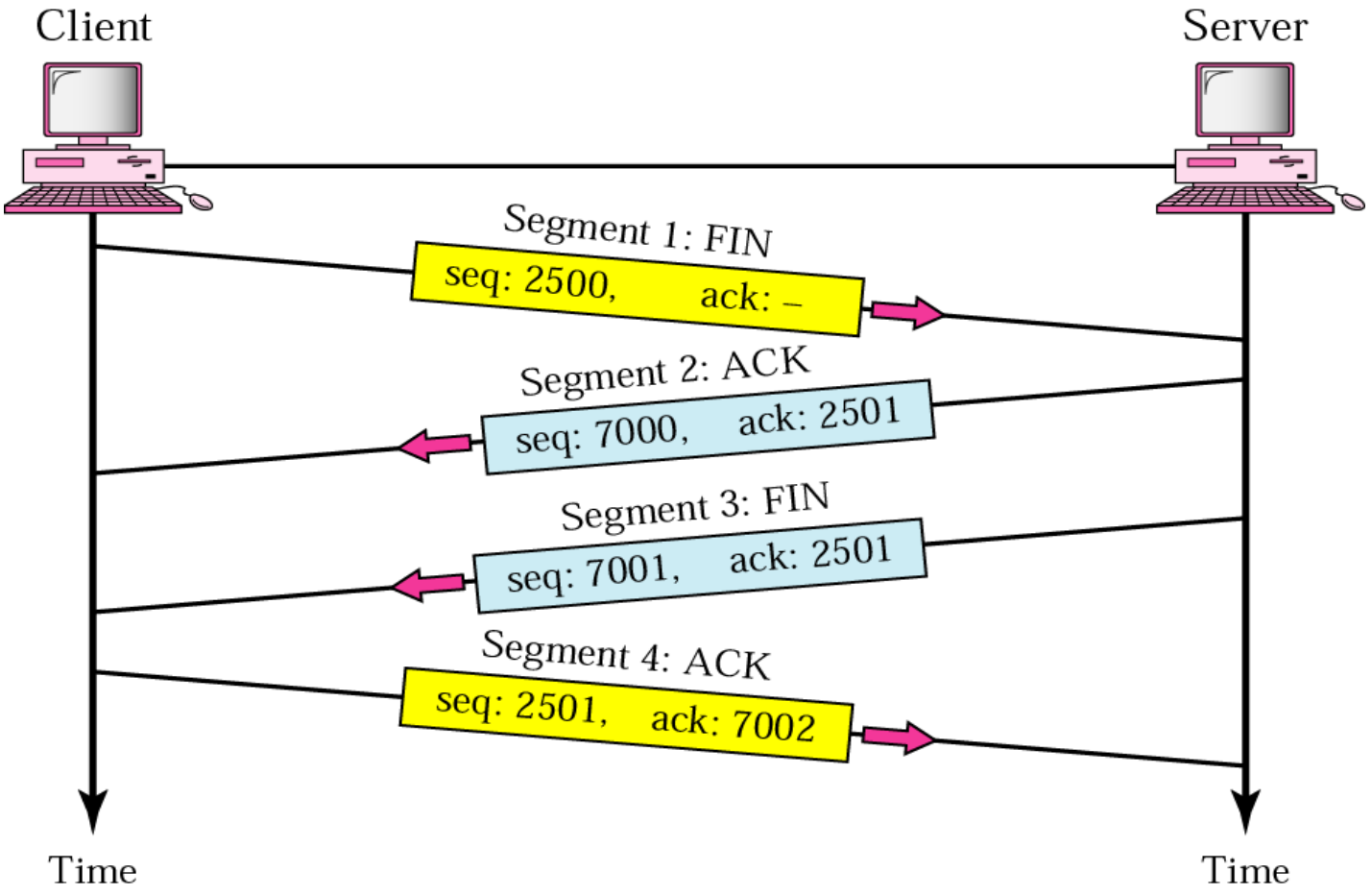
Transmission Control Protocol (TCP)

TCP Data Transfer



Transmission Control Protocol (TCP)

TCP Connection Termination



Transmission Control Protocol (TCP)

TCP Connection Termination

Three-Way Handshaking Most implementations today allow *three-way handshaking* for connection termination as shown in Figure 23.20.

1. In a normal situation, the client TCP, after receiving a close command from the client process, sends the first segment, a FIN segment in which the FIN flag is set. Note that a FIN segment can include the last chunk of data sent by the client, or it can be just a control segment as shown in Figure 23.20. If it is only a control segment, it consumes only one sequence number.

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

---

2. The server TCP, after receiving the FIN segment, informs its process of the situation and sends the second segment, a FIN + ACK segment, to confirm the receipt of the FIN segment from the client and at the same time to announce the closing of the connection in the other direction. This segment can also contain the last chunk of data from the server. If it does not carry data, it consumes only one sequence number.

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

---

3. The client TCP sends the last segment, an ACK segment, to confirm the receipt of the FIN segment from the TCP server. This segment contains the acknowledgment number, which is 1 plus the sequence number received in the FIN segment from the server. This segment cannot carry data and consumes no sequence numbers.

Transmission Control Protocol (TCP)

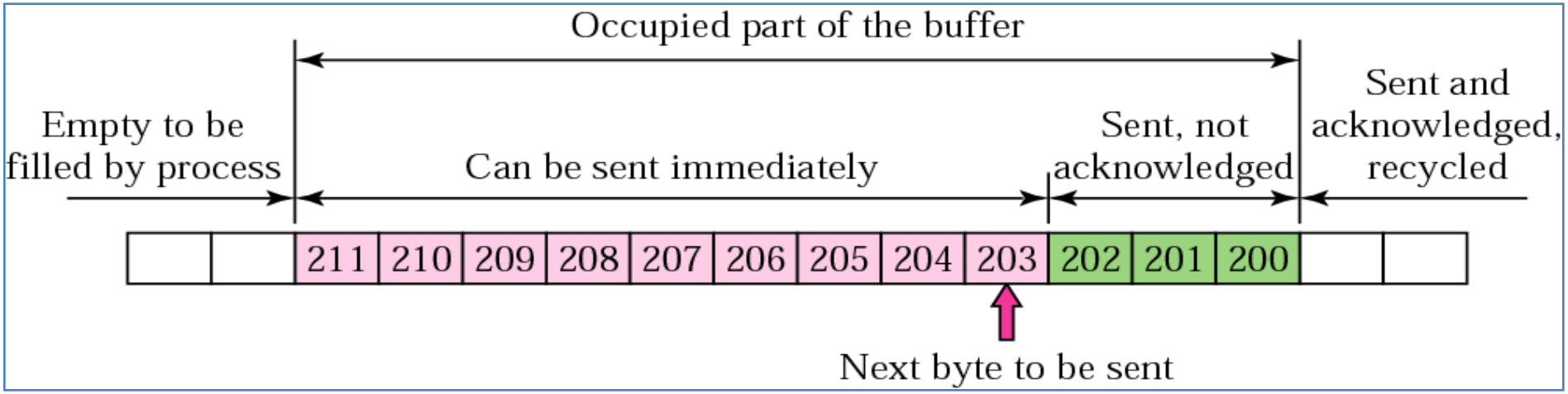
TCP Status

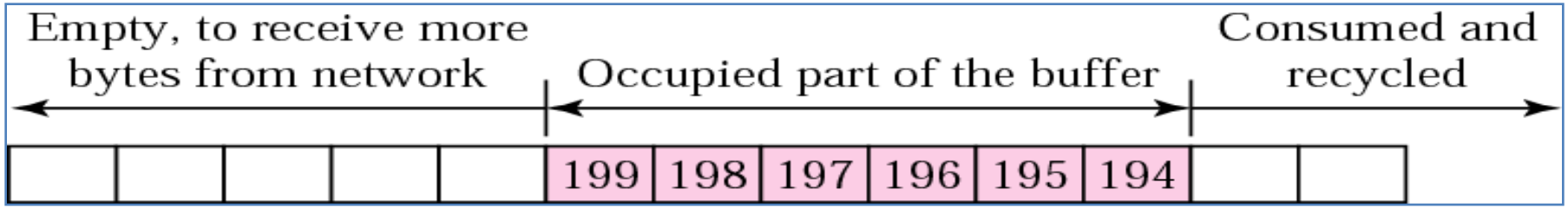
State	Description
CLOSED	There is no connection.
LISTEN	The server is waiting for calls from the client.
SYN-SENT	A connection request is sent; waiting for acknowledgment.
SYN-RCVD	A connection request is received.
ESTABLISHED	Connection is established.
FIN-WAIT-1	The application has requested the closing of the connection.
FIN-WAIT-2	The other side has accepted the closing of the connection.
TIME-WAIT	Waiting for retransmitted segments to die.
CLOSE-WAIT	The server is waiting for the application to close.
LAST-ACK	The server is waiting for the last acknowledgment.

Transmission Control Protocol (TCP)

Flow Control

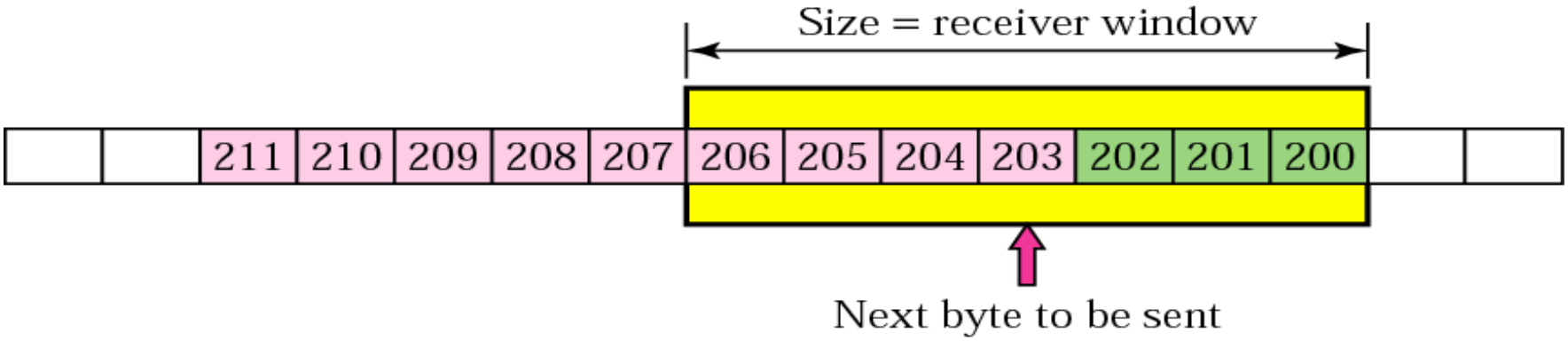
A sliding window is used to make transmission more efficient as well as to control the flow of data so that the destination does not become overwhelmed with data. TCP's sliding windows are byte-oriented.



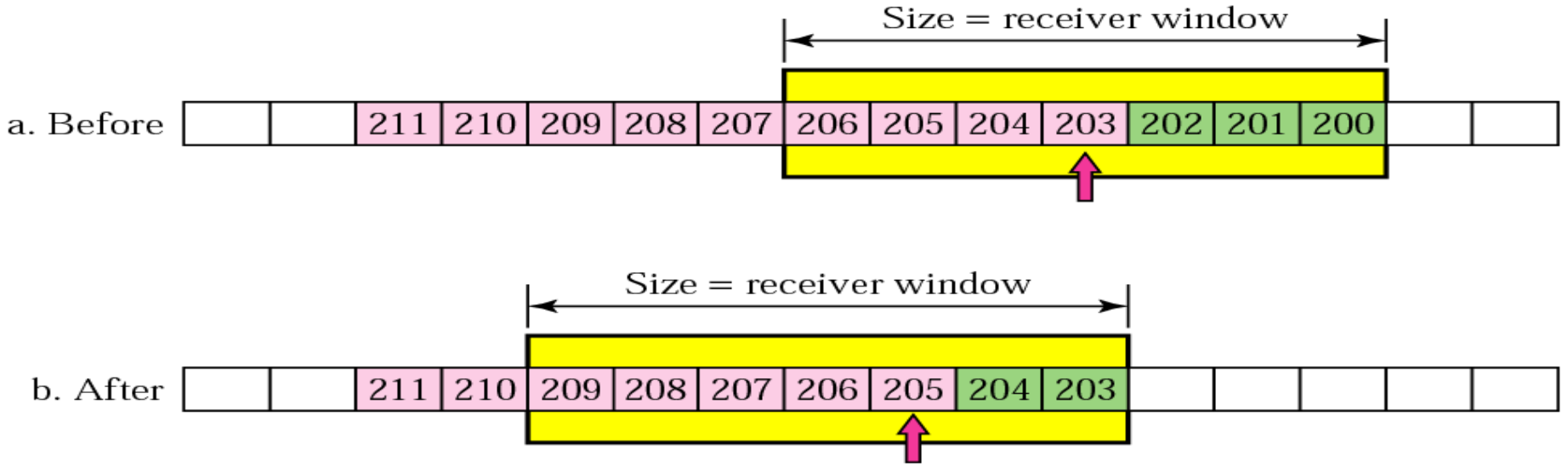


Transmission Control Protocol (TCP)

Flow Control – Sender Buffer & Window

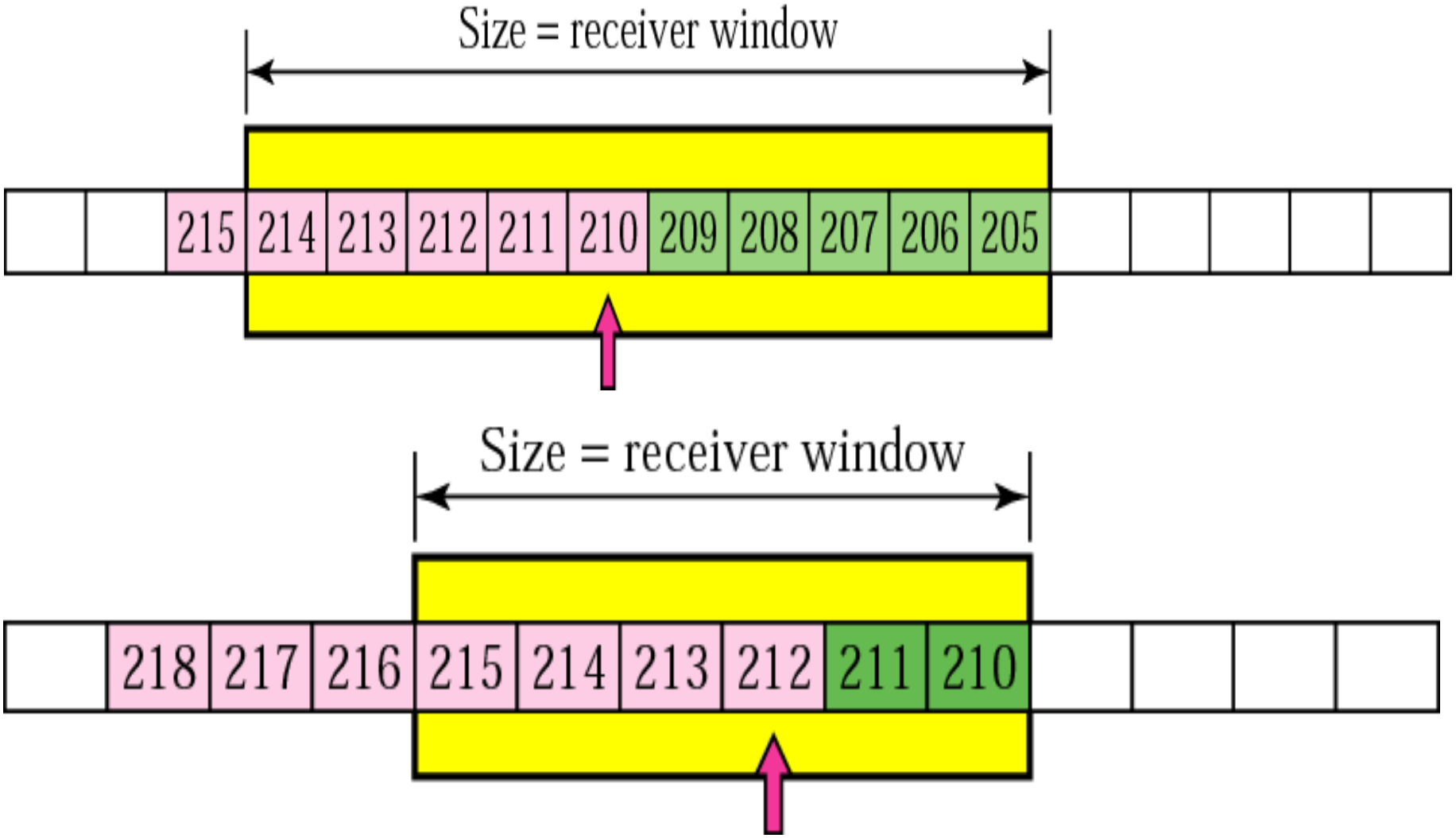


Flow Control – Sliding Rx Window



Transmission Control Protocol (TCP)

Flow Control – Shrinking & Expanding



Transmission Control Protocol (TCP)

Flow Control – Window Size

In TCP, the sender window size is totally controlled by the receiver window value (the number of empty locations in the receiver buffer). However, the actual window size can be smaller if there is congestion in the network.

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.

Stream Control Transmission Protocol

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, such as IUA (ISDN over IP), M2UA and M3UA (telephony signaling), H.248 (media gateway control), H.323 (IP telephony), and SIP (IP telephony), need a more sophisticated service than TCP can provide. SCTP provides this enhanced performance and reliability.

SCTP combines the best features of UDP and TCP. SCTP is a reliable message-oriented protocol. It preserves the message boundaries and at the same time detects lost data, duplicate data, and out-of-order data. It also has congestion control and flow control mechanisms. Later we will see that SCTP has other innovative features unavailable in UDP and TCP.