S.Y.B.Sc.IT SEM III

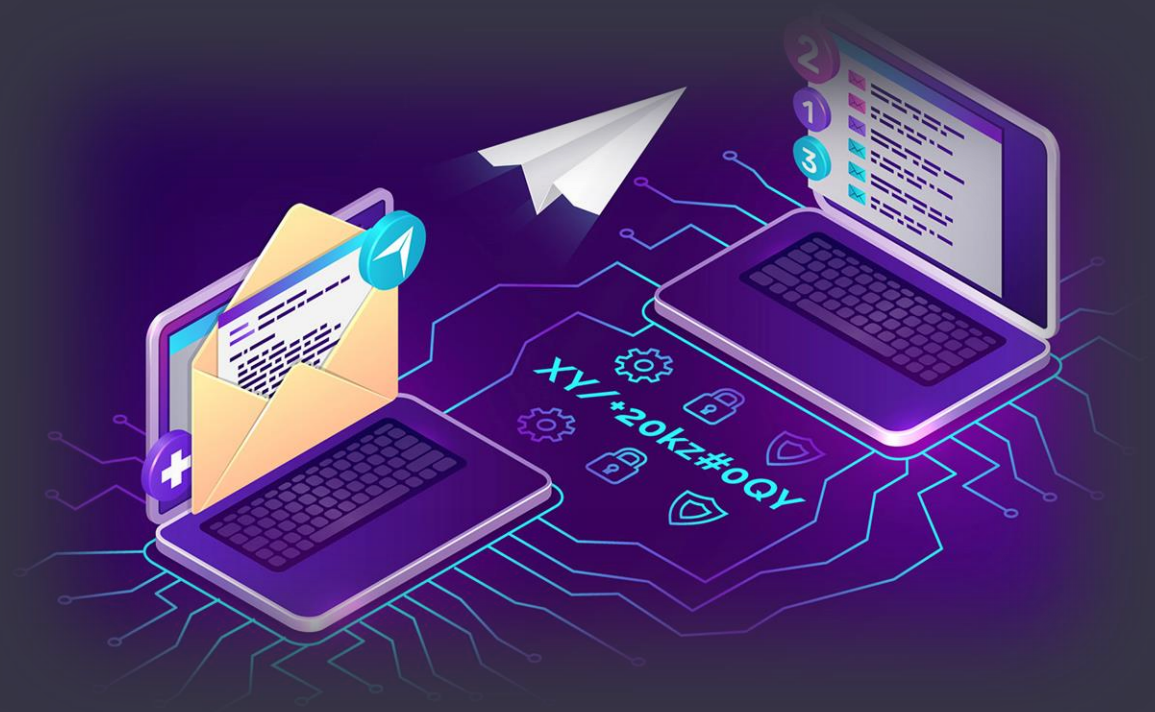
COMPUTER NETWORKS (PUSIT303)

BY,

NIKITA MADWAL

UNIT IV

7. Transport Layer



7. Transport Layer

Introduction

- Transport layer is **present between the network layer and application layer.**
- It is responsible for **providing services to the application layer; it receives services from the network layer.**
- In this chapter, we can discuss the various services that can be provided by a transport layer and different protocols present in the transport layer.

Transport Layer Services

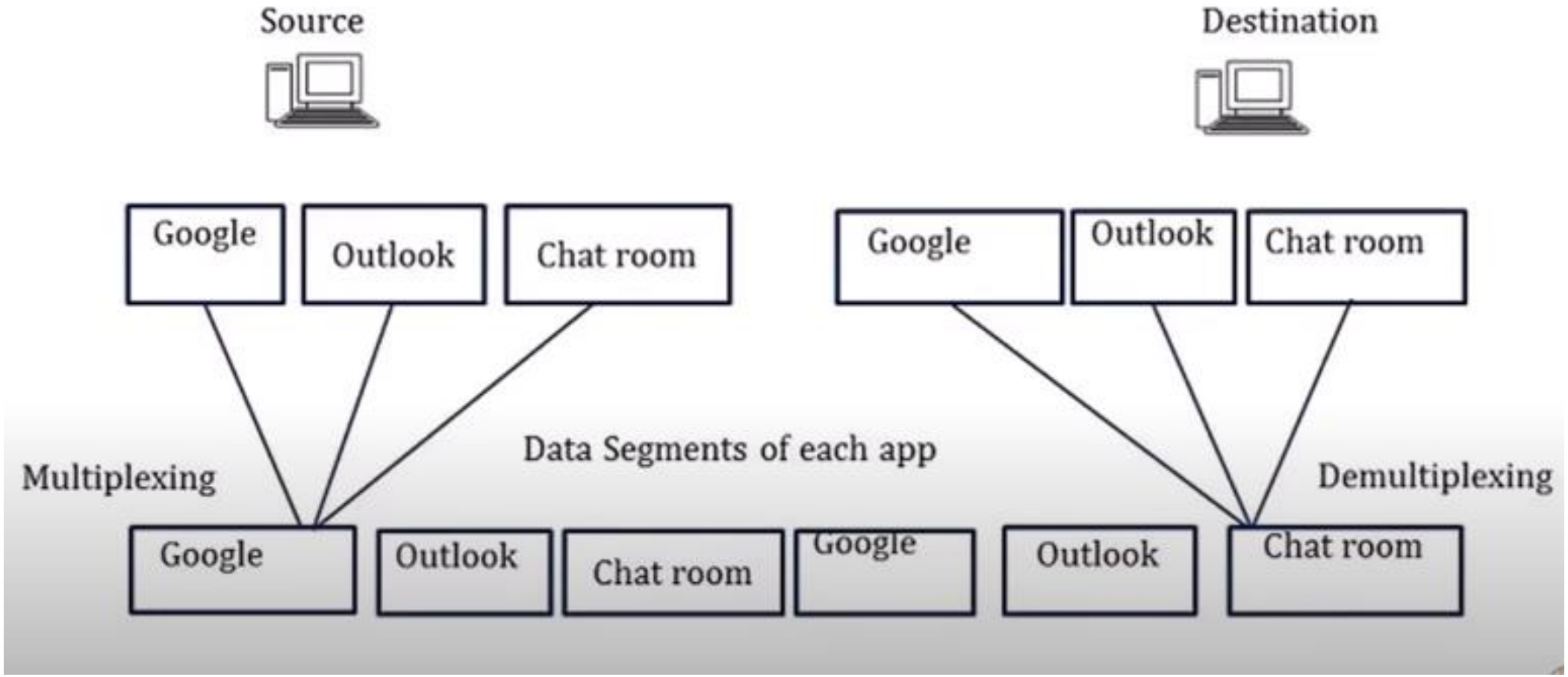
- The transport layer provides the various services such as
- **Process - to - Process Communication**
- Transport Layer is responsible for **delivery of message to appropriate application process on the host computers**
- Transport Layer **uses a port number to deliver the segmented data to the correct process amongst the multiple processes that are running on a particular host.**

❖Encapsulation and Decapsulation

- To send a message from one process to another, the transport-layer protocol **encapsulates and decapsulates messages.**
- **Encapsulation happens at the sender site**. The transport layer **receives the data and adds the transport-layer header**.
- **Decapsulation happens at the receiver site**. When the message **arrives at the destination transport layer, the header is dropped and the transport layer delivers the message to the process running at the application layer.**

❖Multiplexing and Demultiplexing

- **Multiplexing**(many to one) is when **data is acquired from several processes from the sender and merged into one packet along with headers and sent as a single packet.**
- Multiplexing allows the simultaneous use of different processes over a network that is running on a host.
- The processes are differentiated by their port numbers.
- Similarly, **Demultiplexing**(one to many) **is required at the receiver side when the message is distributed into different processes.**
- Transport receives the segments of data from the network layer distributes and delivers it to the appropriate process running on the receiver's machine.


Source
Destination
Google
Outlook
Chat room
Google
Outlook
Chat room
Multiplexing
Data Segments of each app
Demultiplexing
Google
Outlook
Chat room
Google
Outlook
Chat room

❖Flow Control

- Flow control **makes sure that the data is transmitted at a rate that is acceptable for both sender and receiver by managing data flow**.

❖Congestion Control

- Congestion **is a situation in which too many sources over a network attempt to send data and the router buffers start overflowing due to which loss of packets occurs.**
- As a result, **the retransmission of packets from the sources increases the congestion further**. In this situation, the Transport layer provides Congestion Control in different ways.
- Congestion control mechanisms are divided **into two categories**,
- 1. **Open loop** - prevent the congestion before it happens.
- 2. **Closed loop** - remove the congestion after it happens.

❖Error Control

- Error control at the transport layer is responsible to
- 1. Detect and discard corrupted packets.
- 2. Keep track of lost and discarded packets and resend them.
- 3. Recognize duplicate packets and discard them.
- 4. Buffering out-of-order packets until the missing packets arrive

Transport Layer Protocol

- **Three protocols are associated with the Transport layer.**
- They are UDP –User Datagram Protocol , TCP – Transmission Control Protocol & SCTP - Stream Control Transmission Protocol
- **UDP** - UDP is an unreliable connectionless transport-layer protocol used for its simplicity and efficiency in applications where error control can be provided by the application-layer process.
- **TCP** - TCP is a reliable connection-oriented protocol that can be used in any application where reliability is important.
- **SCTP** - SCTP is a new transport-layer protocol designed to combine some features of UDP and TCP in an effort to create a better protocol for multimedia communication.

Port Numbers

❖**What is Port ?**

- A port is a **virtual point where network connections start and end**.
- **Ports are software-based and managed by a computer's operating system.**
- Each **port is associated with a specific process or service**.
- Ports **allow computers to easily differentiate between different kinds of traffic**

❖**What is a port number?**

- Ports are standardized across all network-connected devices, with each port assigned a number.
- Most ports are reserved for certain protocols — for example, all Hypertext Transfer Protocol (HTTP) messages go to port 80.
- While IP addresses **enable messages to go to and from specific devices**, port numbers allow targeting of specific services or applications within those devices.

Table 24.1 *Some well-known ports used with UDP and TCP*

Port	*Protocol*	*UDP*	*TCP*	*SCTP*	*Description*
7	Echo	√	√	√	Echoes back a received datagram
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
21	FTP-21		√	√	File Transfer Protocol
23	TELNET		√	√	Terminal Network
25	SMTP		√	√	Simple Mail Transfer Protocol
53	DNS	√	√	√	Domain Name Service
67	DHCP	√	√	√	Dynamic Host Configuration Protocol
69	TFTP	√	√	√	Trivial File Transfer Protocol
80	HTTP		√	√	HyperText Transfer Protocol
111	RPC	√	√	√	Remote Procedure Call
123	NTP	√	√	√	Network Time Protocol
161	SNMP-server	√			Simple Network Management Protocol
162	SNMP-client	√			Simple Network Management Protocol

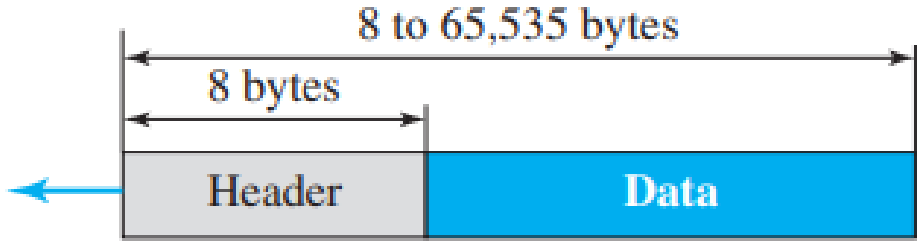
UDP (User Datagram Protocol)

- The User Datagram Protocol (UDP) is called a **connectionless, unreliable transport protocol.**
- Also, it performs **very limited error checking**.
- 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.

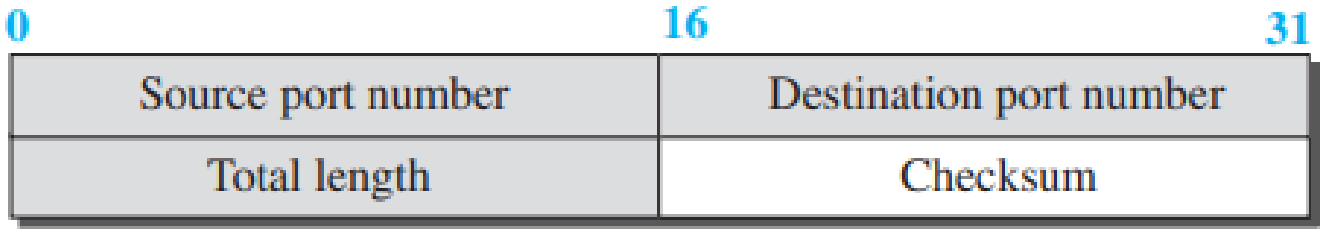
❖UDP Packet Format

- UDP packets, called **user datagrams**, have a **fixed-size header of 8 bytes.**

Figure 24.2 *User datagram packet format*



a. UDP user datagram



b. Header format

- These user datagrams, have a fixed-size header of 8 bytes made of four fields, each of 2 bytes (16 bits).

Source Port Number

- It 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

- It is the port number used by the **process running on the destination host**.
- It is also 16 bits long

Length

- This field denotes the **total length of the UDP Packet (Header + data)**
- The total length of any UDP datagram can be from 0 to 65,535 bytes.

Checksum

- It is used to detect errors over the **entire user datagram (header plus data)**

UDP Services

Process-to-Process Communication

- UDP provides process-to-process communication using socket addresses, a ***combination of IP addresses and port numbers***.

Connectionless Services

- UDP ***provides a connectionless service***.
- There is ***no connection establishment and no connection termination*** .
- Each ***user datagram sent by UDP is an independent datagram***.
- There is ***no relationship between the different user datagrams even if they are coming from the same source process and going to the same destination program.***
- The ***user datagrams are not numbered***.
- Each ***user datagram can travel on a different path***.
- Only those processes ***sending short messages should use UDP***.

Flow Control

- UDP is unreliable transport protocol.
- There is ***no flow control, and hence no window mechanism***.
- The ***receiver may overflow with incoming messages***.
- The ***lack of flow control means that the process using UDP should provide for this service, if needed.***

Error Control

- 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.***

Checksum

- UDP checksum calculation includes three sections: ***a pseudoheader***, ***the UDP header, and the data*** coming from the application layer.
- The pseudoheader is the part of the header in which the user datagram is to be encapsulated with some fields filled with 0s.

Congestion Control

- Since UDP is a ***connectionless protocol, it does not provide congestion control.***
- ***UDP assumes*** that the packets sent are small and sporadic(occasionally or at irregular intervals) and ***cannot create congestion in the network***.
- This assumption may or may not be true, when ***UDP is used for interactive real-time transfer of audio and video***.

Encapsulation and Decapsulation

- To send a message from one process to another, the ***UDP protocol encapsulates and decapsulates messages.***

Queuing

- In UDP, ***queues are associated with ports***.
- At the ***client site, when a process starts***, ***it requests a port number*** from the operating system.
- Some implementations ***create both an incoming and an outgoing queue associated with each process.***
- Other ***implementations create only an incoming queue*** associated with each process.

- **Multiplexing and Demultiplexing**
- In a host running a transport protocol suite, there is only one UDP but possibly ***several processes that may want to use the services of UDP***.
- To handle this situation, UDP multiplexes and demultiplexes.

❖Applications of UDP

- The straightforward request/response communication of relatively small amounts of data, eliminating concerns regarding controlling errors or the flow of the packets
- Routing update protocols such as Routing Information Protocol (RIP)
- Real-time applications in which the information needs to be delivered quickly and smoothly
- 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 (Simple Network Management Protocol) (i.e. communication between routers, switches, servers and wireless devices)

TCP (Transmission Control Protocol)

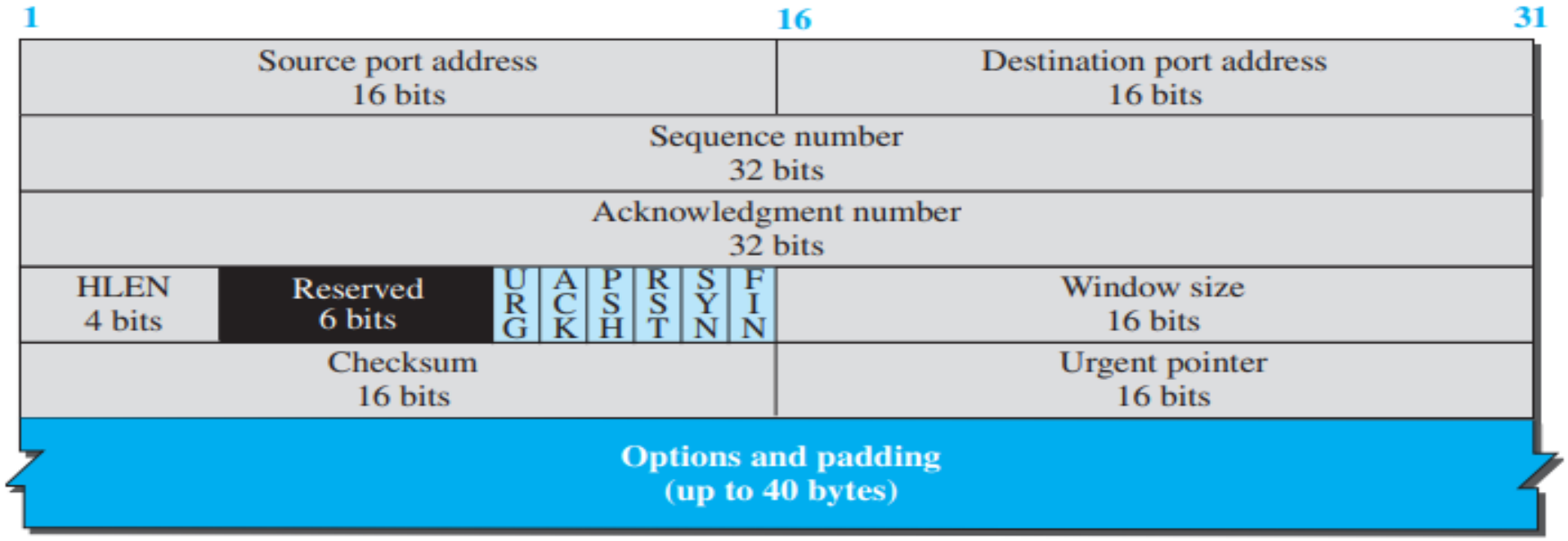
- TCP, like UDP, is a **process-to-process** (program-to-program) protocol.
- TCP, therefore, like UDP, **uses port numbers**.
- Unlike UDP, TCP is a **connection oriented protocol; it creates a virtual connection between two TCPs to send data**.
- In addition, TCP **uses flow and error control mechanisms at the transport level.**
- In brief, TCP is called a connection-oriented, **reliable transport protocol.**

❖TCP Packet Format

- A packet in TCP is called a **segment.**

Figure 24.7 *TCP segment format*

a. Segment



b. Header

- The segment consists of a header of 20 to 60 bytes, followed by data from the application program.

➢ **Source port address**

- It is a 16-bit field that ***defines the port number of the application program in the host that is sending*** the segment.

➢ **Destination port address**

- It is a 16-bit field that ***defines the port number of the application program in the host that is receiving*** the segment

➢ **Sequence number**

- This 32-bit field defines the ***number assigned to the first byte of data contained in this segment. 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.***

Acknowledgment 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 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 ***TCP header length***.
- The length of the header can be between 20 and 60 bytes.

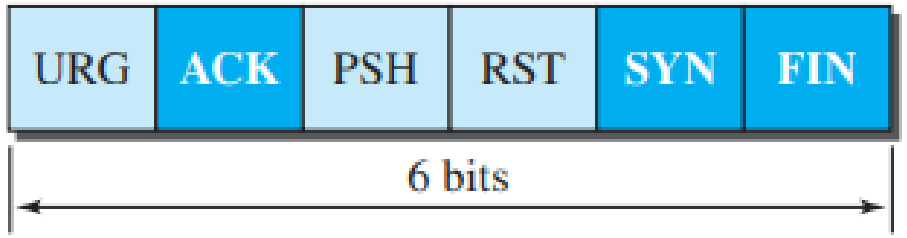
Reserved

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

Control

- This field defines ***6 different control bits or flags***, as shown in Figure 24.8. One or more of these bits can be set at a time.
- These ***bits enable flow control, connection establishment and termination, connection abortion, and the mode of data transfer in TCP***

Figure 24.8 *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.

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 and ***is determined by the receiver. The sender must obey the dictation of the receiver in this case.***
- ***It defines receiver's window size and acts as flow control***

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.***
- The same pseudoheader, serving the same purpose, is added to the segment.

Urgent pointer

- This l6-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.***

Options

- There can be up to 40 bytes of optional information in the TCP header.
- Other function that are not covered by header

❖TCP Services

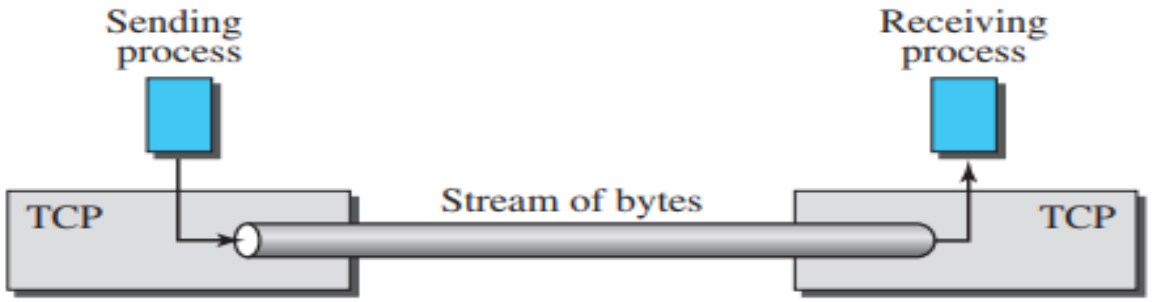
➢Process-to-process communication

- TCP also provides process-to-process communication like UDP by using port numbers.

➢Stream Delivery Service

- TCP is a ***stream-oriented protocol***.
- TCP ***allows the sending process to deliver data as a stream*** (***continuous flow of data***) 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 bytes*** across the Internet.

Figure 24.4 *Stream delivery*



Full-Duplex Communication

- TCP offers full-duplex service, ***where data can flow in both directions at the same time.***
- Each TCP endpoint then has its ***own sending and receiving buffer, and segments move in both directions***

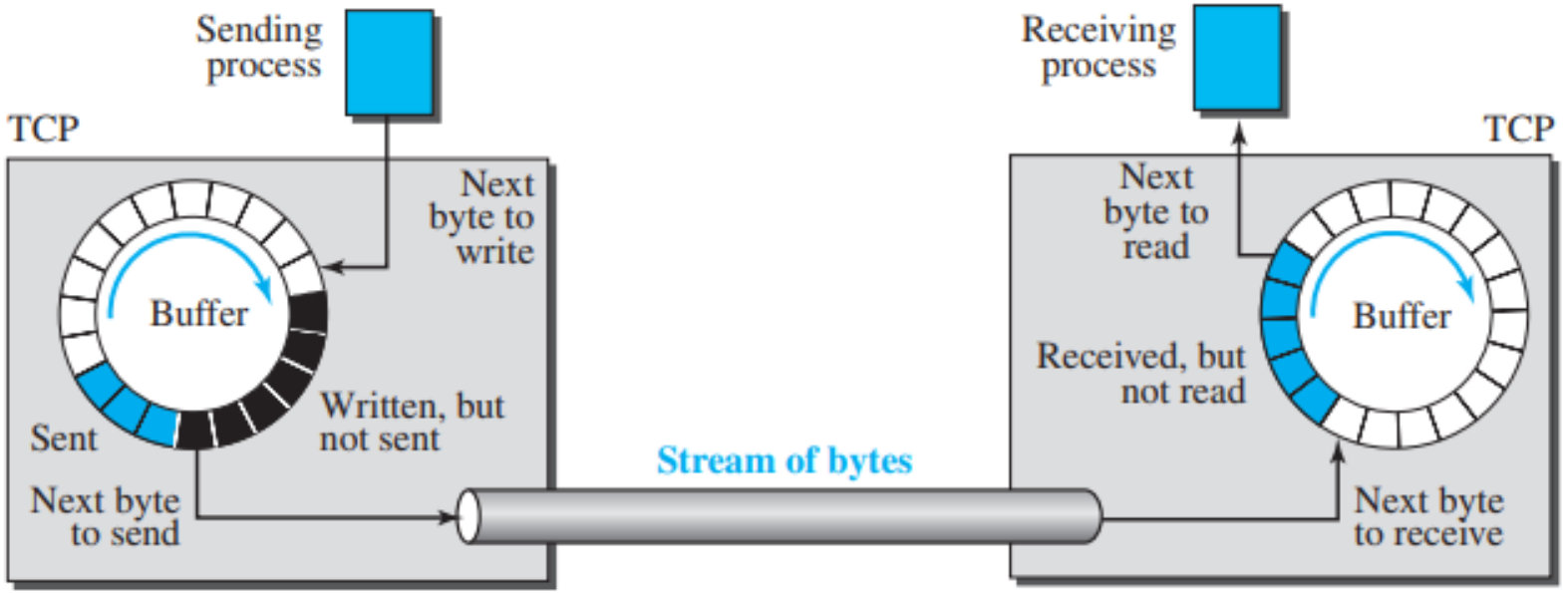
Multiplexing and Demultiplexing

- TCP performs ***multiplexing at the sender and demultiplexing at the receiver.***

Sending and Receiving Buffers

- The ***sending and the receiving processes may not 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.

Figure 24.5 *Sending and receiving buffers*



- At the **sending site**, the buffer has **three types of chambers**.
- The **white section** contains **empty chambers that can be filled by the sending process** (producer).
- The **gray area** holds bytes **that have been sent but not yet acknowledged.**
- TCP keeps these bytes in the buffer until it receives an acknowledgment.
- The **colored area** contains **bytes to be sent by the sending TCP**.
- The operation of the buffer at the ***receiver site*** is simpler. The circular buffer is ***divided into two areas*** (shown as white and colored).
- The ***white area*** contains ***empty chambers to be filled by bytes received from the network***.
- The ***colored*** sections contain ***received bytes that can be read by the receiving process.***

Connection-Oriented Service

- TCP is a ***connection-oriented protocol***.
- A connection needs to be established for each pair of processes.
- When a process at site A wants to send to and receive data from another process at site B, the following three phases occur:
- 1. The ***two TCP's establish a logical 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.***

Flow Control

- TCP provides flow control.
- The ***sending TCP controls how much data can be accepted*** from the sending process; the ***receiving TCP controls how much data can to be sent*** by the sending TCP.
- 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***.

Congestion Control

- TCP 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 determined by the level of congestion, if any, in the network.***

❖TCP Connection

- TCP is connection-oriented.
- A connection-oriented transport protocol **establishes a logical path between the source and destination.**
- All of the **segments belonging to a message are then sent over this logical path.**
- In TCP, connection-oriented transmission requires three phases: **Connection Establishment, Data Transfer and Connection Termination**.

Connection Establishment

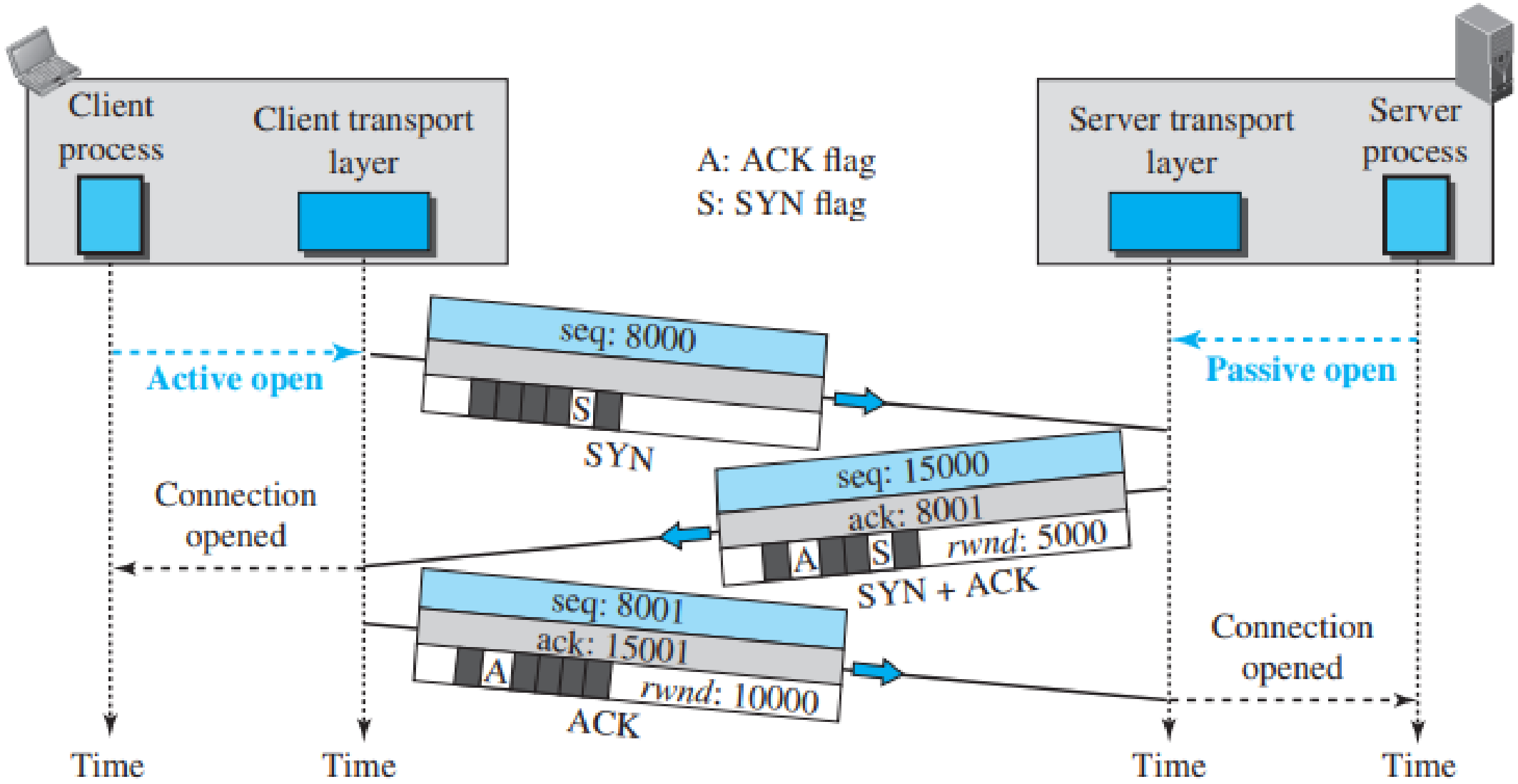
- TCP **transmits data in full-duplex mode**.
- When two TCPs in **two machines are connected, they are able to send segments to each other simultaneously.**
- This implies that each party **must initialize communication and get approval from the other party before any data are transferred**.
- **Connection establishment in TCP is a three-way handshaking.**

Three Way Handshaking

- The **connection establishment in TCP is called three way handshaking.**
- In our example, **an application program, called the client, wants to make a connection with another application program, called the server**, using TCP as the transport layer protocol.
- The **process starts with the server**.
- The **server program** tells its TCP that it is ready to accept a connection. This is **called a request for a passive open**.
- Although the server TCP is ready to accept any connection from any machine in the world, ***it cannot make the connection itself***.
- The **client program issues a request for an active open**.
- A **client that wishes to connect to an open server tells its TCP** that it needs to be connected to that particular server.

- TCP can now start the three-way handshaking process as shown in Figure 23.18.
- Each segment has value for all its header fields and perhaps for some of its option fields, too.
- We **show the sequence number, the acknowledgment number, the control flags (only those that are set), and the window size, if not empty.**

Figure 24.10 *Connection establishment using three-way handshaking*



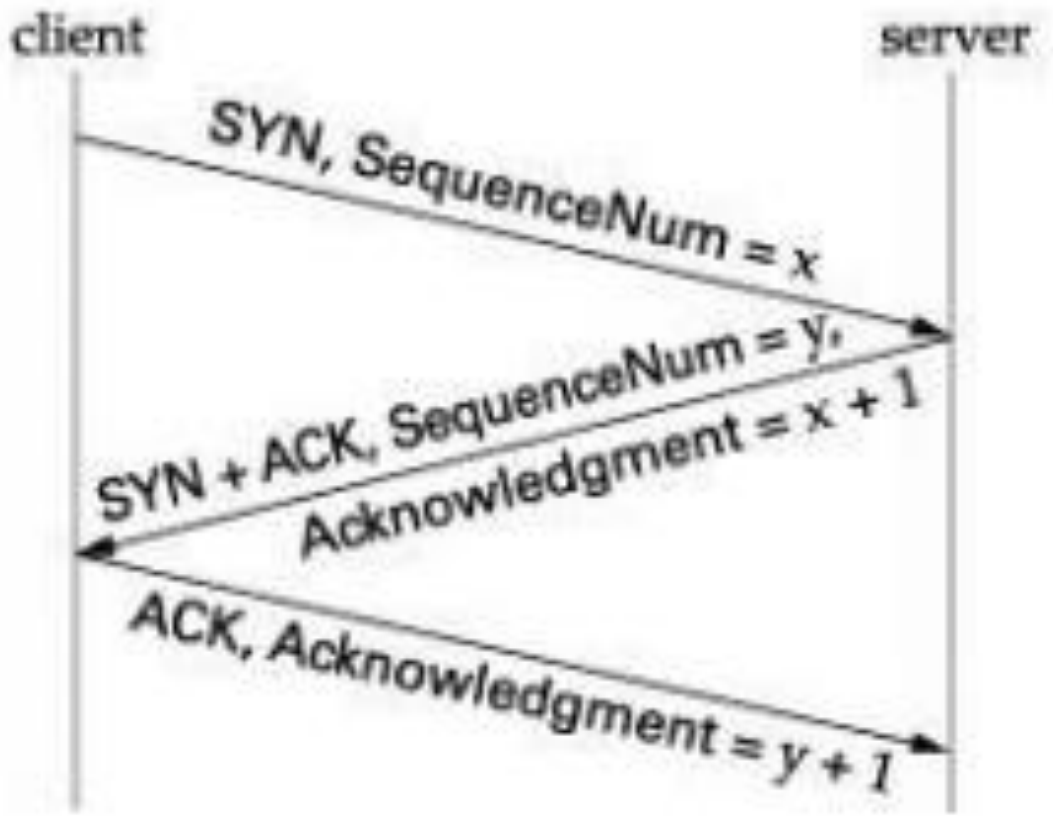
➢ 1

- 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**.
- The client in our example **chooses a random number as the first sequence number and sends this number to the server**.

➢ 2

- The **server sends the second segment**, a **SYN + ACK segment** with two flag bits set as: SYN and ACK.
- This segment has a **dual purpose**.
- First, it is a **SYN segment for communication in the other direction**. The **server uses this segment to initialize a sequence number** for numbering the bytes sent from the server to the client.
- The **server also acknowledges the receipt of the SYN** segment from the client **by setting the ACK flag and displaying the next sequence number it expects to receive from the client**.

- 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.



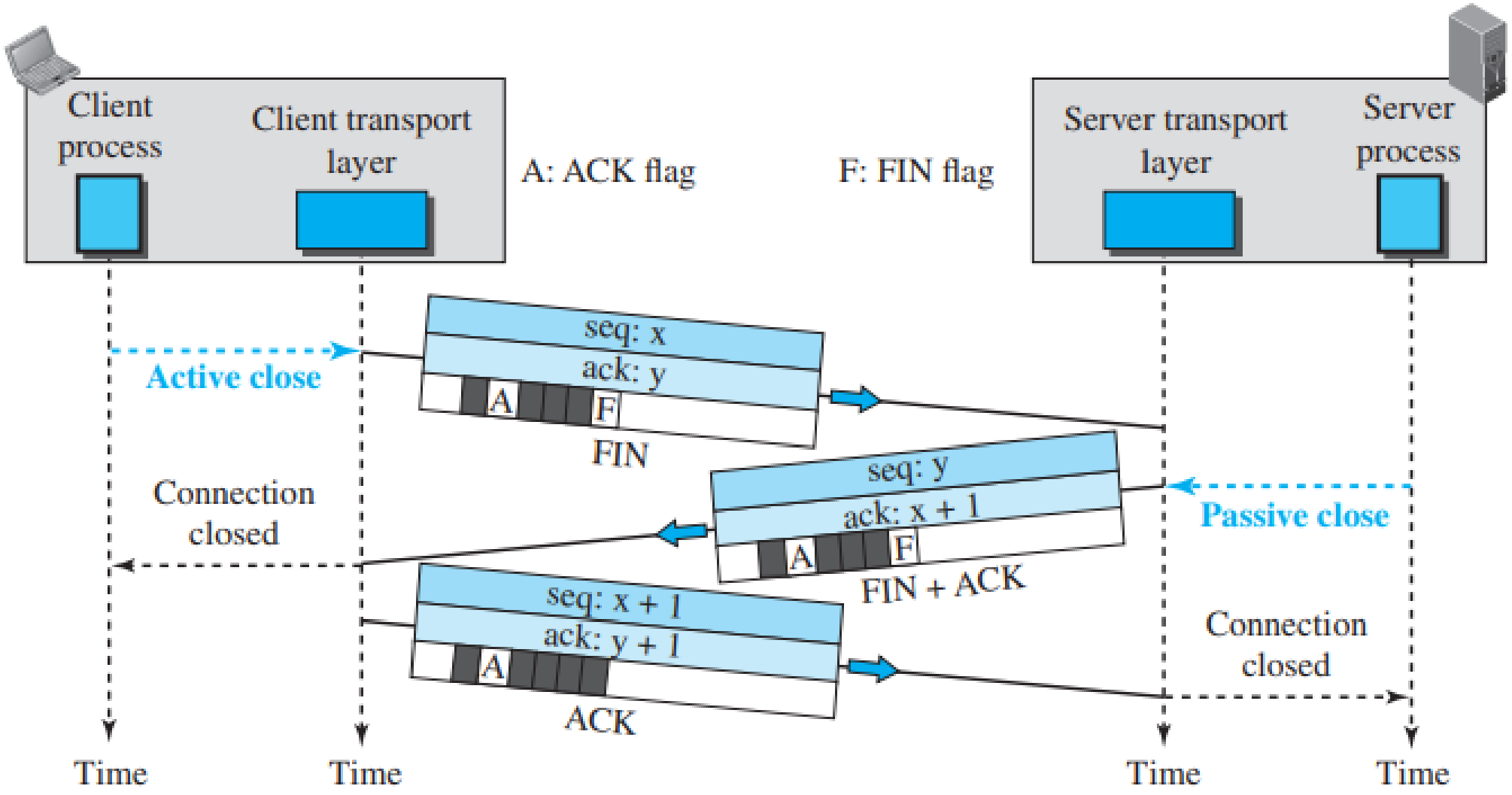
Data Transfer

- After **connection is established**, **bidirectional data transfer can take place.**
- **The client and server can send data and acknowledgments in both directions.**
- The **data traveling in the same direction as an acknowledgment are carried on the same segment**.
- The acknowledgment is piggybacked with the data.

Connection Termination

- Any of the **two parties involved in exchanging data (client or server) can close the connection**, although it is usually initiated by the client.
- Most implementations today allow **three-way handshaking for connection termination**

Figure 24.12 *Connection termination using three-way handshaking*

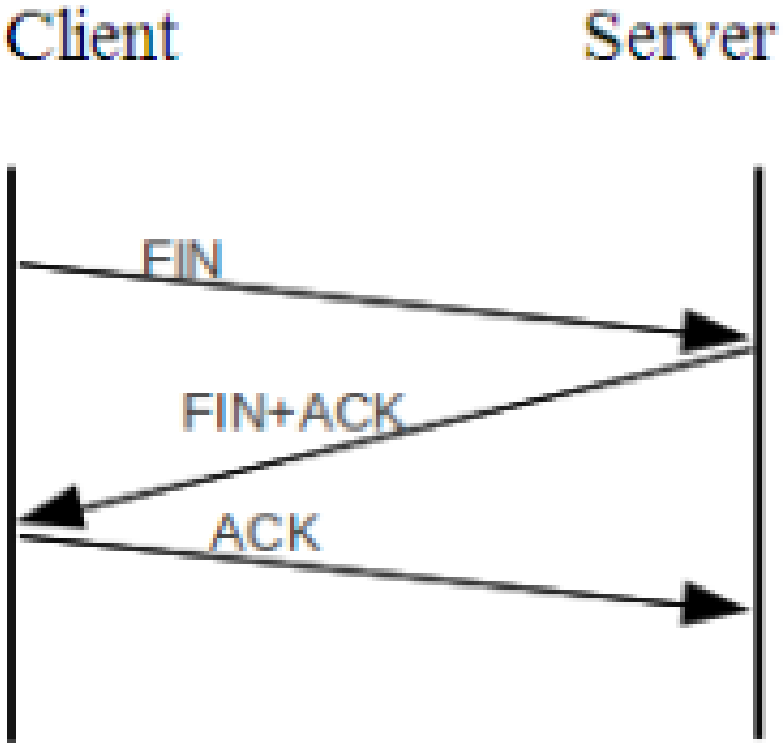


Three Way Handshaking

- 1
  - The **client usually initiates the process of termination** and sends a **FIN segment with FIN flag set to the server.**
  - This segment contains sequence number and can contain an acknowledgment number if it is the acknowledgement of the previous segment received from the server.

- 2
  - The **server on receiving the FIN segment sends a FIN + ACK segment back to the client**.
  - This **segment has 2 purposes**:
  - This act as the **acknowledgement of the FIN segment received from the client** as well as the **FIN segment of the server to client to announce the closing of the connection** in the other direction.

- 3
  - The **client at last sends the ACK segment to the server confirming the receipt of the previous segment** from server and the **connection is closed successfully from both ends**



TCP vs UDP

Key	TCP	UDP
Connection type	Requires an established connection before transmitting data	No connection is needed to start and end a data transfer
Data sequence	Can sequence data (send in a specific order)	Cannot sequence or arrange data
Data retransmission	Can retransmit data if packets fail to arrive	No data retransmitting. Lost data can't be retrieved
Delivery	Delivery is guaranteed	Delivery is not guaranteed
Check for errors	Thorough error-checking guarantees data arrives in its intended state	Minimal error-checking covers the basics but may not prevent all errors
Broadcasting	Not supported	Supported
Speed	Slow, but complete data delivery	Fast, but at risk of incomplete data delivery

Header size	TCP uses a variable-length (20-60) bytes header.	UDP has a fixed-length header of 8 bytes.
Handshake	Handshakes such as SYN, ACK, and SYN+ACK are used.	It's a connectionless protocol, which means it doesn't require a handshake.