CN (IT-3001)

Transport Layer: Transport Protocols

Prof. Amit Jha

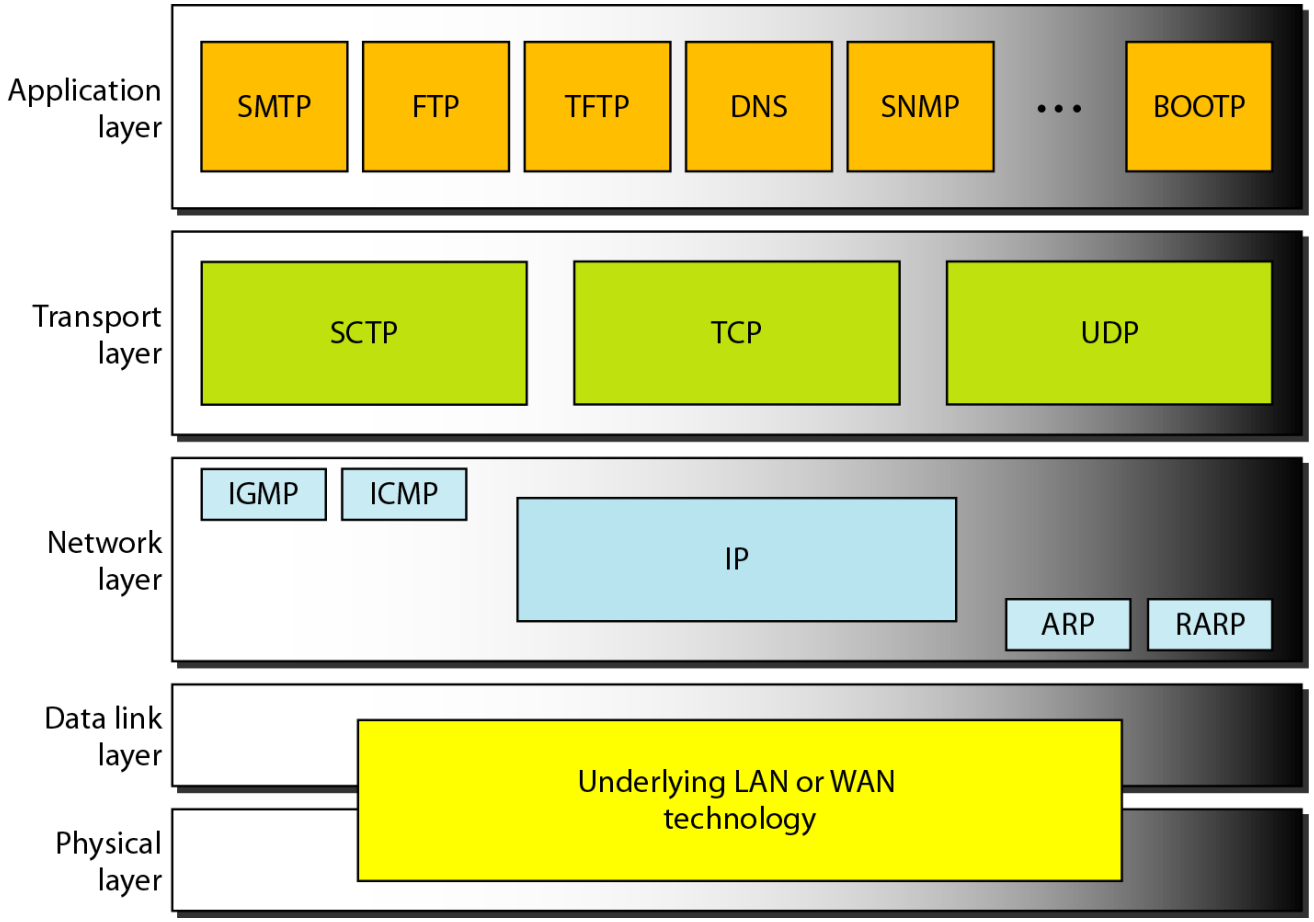
School of Electronics Engineering (SOEE)

KIIT Deemed to be University

Transport Protocols

- We discuss following protocols in the subsequent slides.
  1. User Datagram Protocol (UDP)
  2. Transmission Control Protocol (TCP)

Position of UDP, TCP and SCTP in TCP/IP Suite



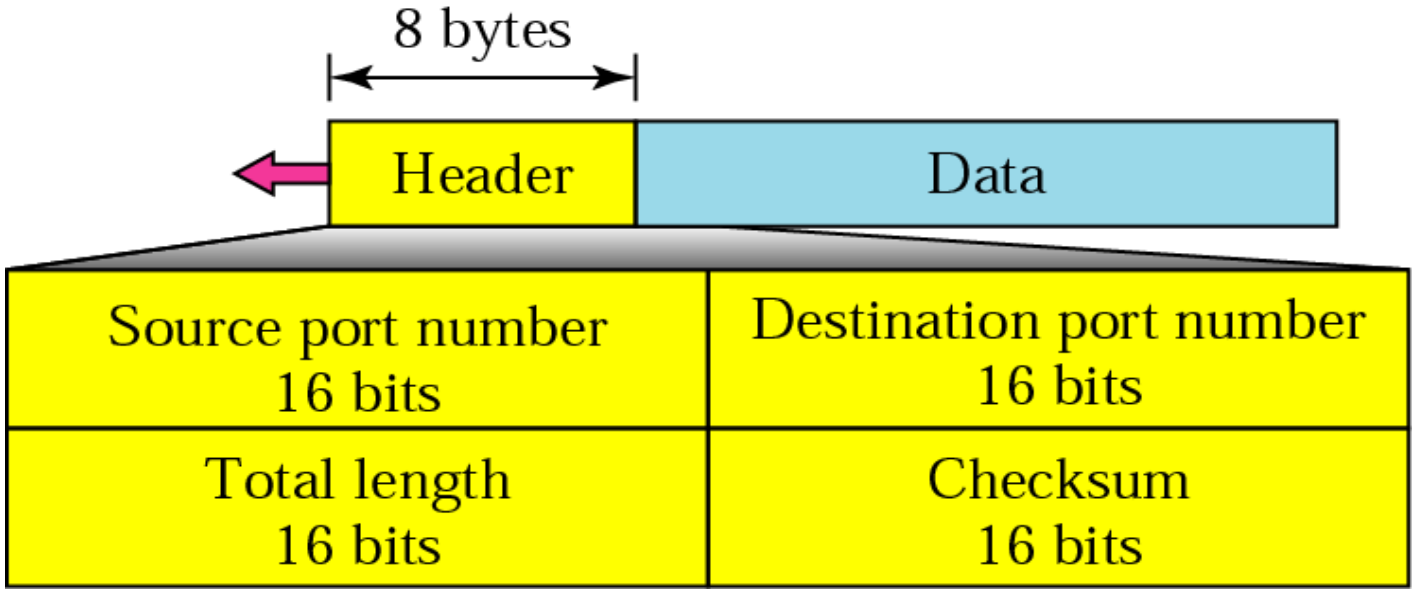
User Datagram Protocol (UDP)

- It is a connectionless and thus, unreliable transport protocol.
- It does not add anything to the service of IP except to provide process-to-process communication instead of host-to-host.
- It does not support flow control thus, no window mechanism.
- There is no error control mechanism except for the checksum.
- To handle multiple processes there is provision of a *queue*.

Q) If UDP is so powerless, why would a process want to use it?

Ans: UDP is 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.

UPD header format



Application of UDP:

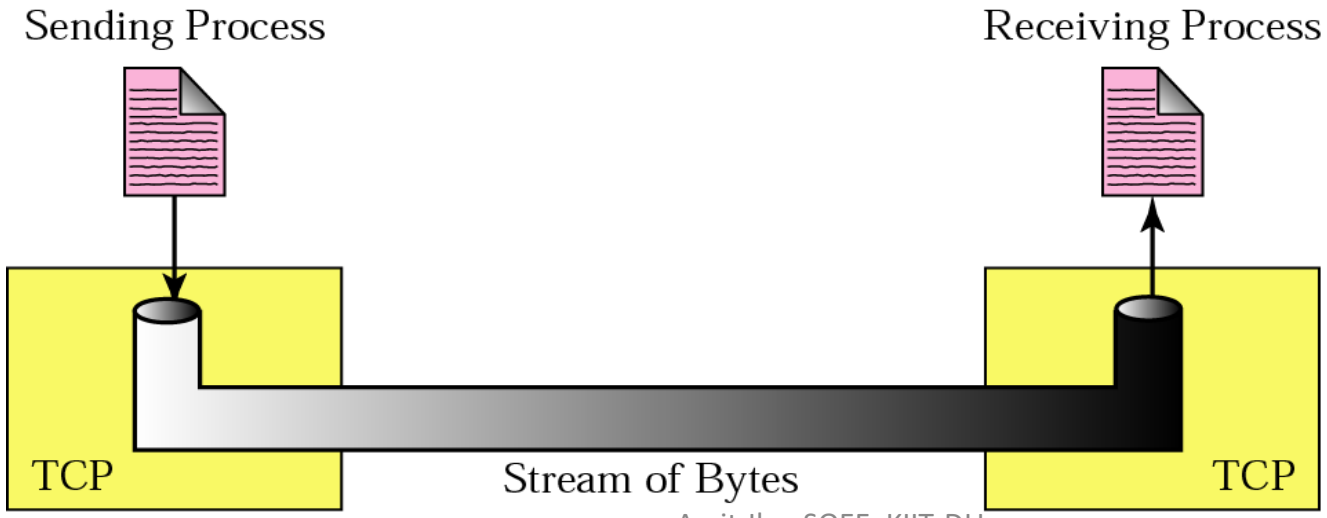
- UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control. It is not usually used for a process such as **FTP that needs to send bulk data**.
- UDP is suitable for a process with internal flow and error control mechanisms. For example, 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, DNS**.
- UDP is used for some route updating protocols such as **Routing Information Protocol(RIP)**.

Transmission Control Protocol (TCP)

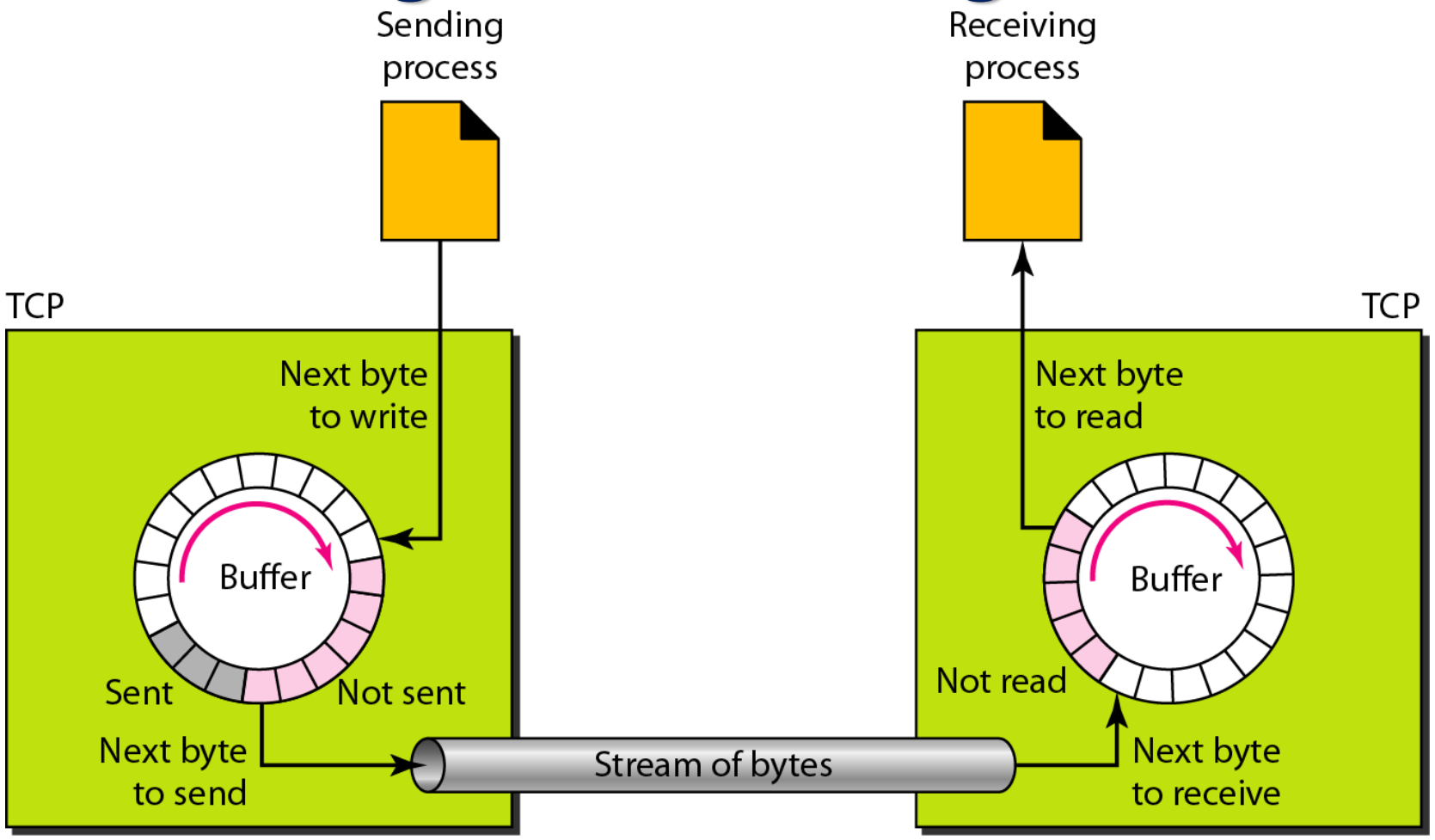
- It is a connection oriented protocol.
- It provides *process-to-process* communication.
- It is reliable and provides flow and error control mechanism.
- It is a stream oriented protocol.
- Provides full duplex communication.
- It supports multiplexing and demultiplexing.
- It also implements the congestion control mechanism.

Transmission Control Protocol (TCP)

- Like UDP, it provides process-to-process communication using port number.
- It is a *connection-oriented*, thus *reliable* transport protocol.
- Unlike UDP, it is **a stream oriented protocol**.
- Supports full-duplex service.

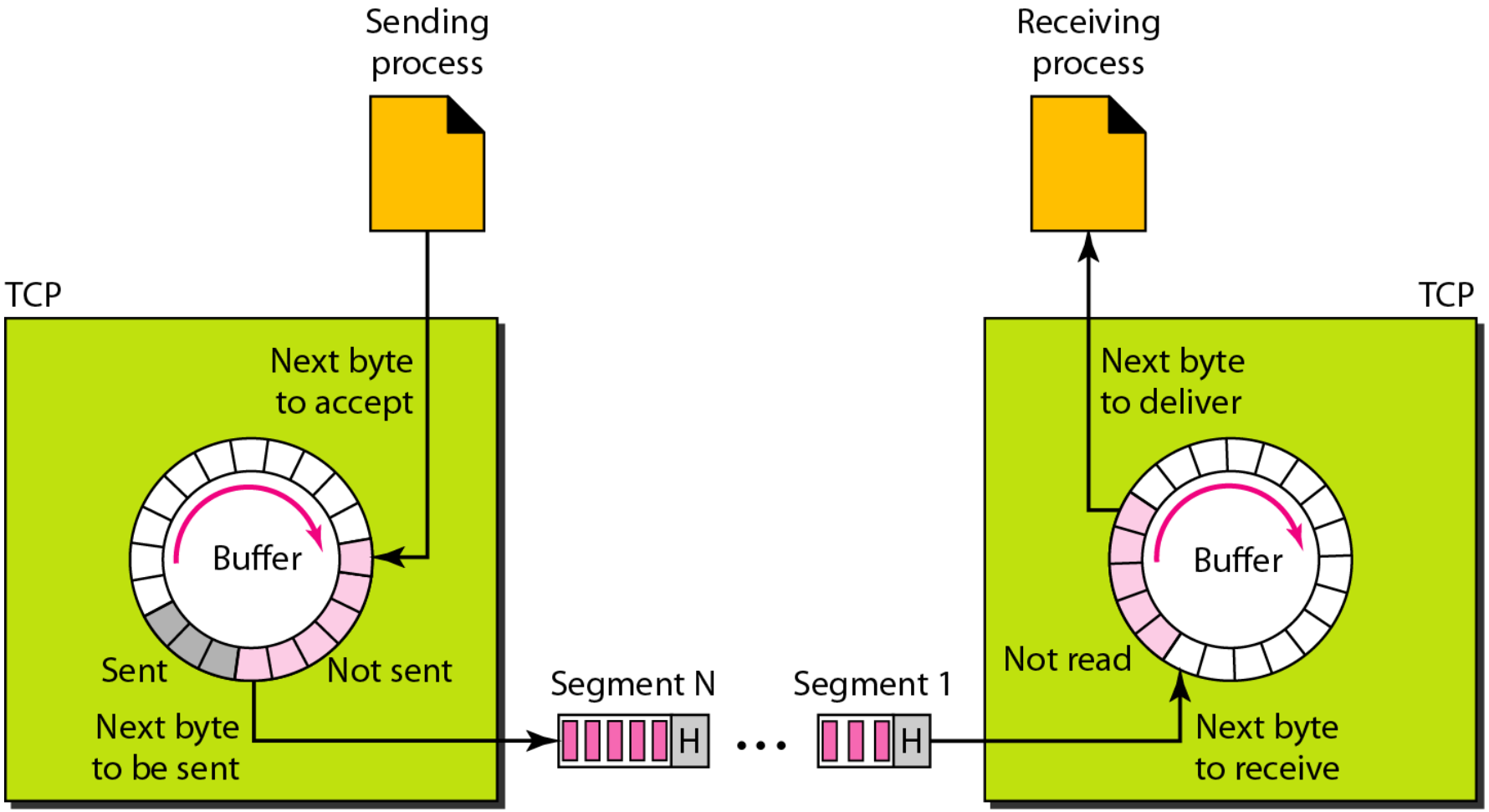


TCP: Sending and Receiving buffer



Note: It is needed when sender and receiver do not operate at the same speed.

TCP Segments



The IP layer, as a service provider for TCP, needs to send data in packets, not as a stream of bytes. At the transport layer, TCP groups a number of bytes together into a packet called a segment. TCP adds header to each segment.

TCP Features

Several features of TCP to provide various services as discussed before, are described below:

- **Numbering System:** The bytes of data being transferred in each connection are numbered by TCP. The numbering starts with a randomly generated number between 0 to $2^{32} - 1$.
- **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.
- **Acknowledgement Number:** The acknowledgment number defines the number of the next byte that the party expects to receive. It is cumulative in nature.

Example 3.9: Imagine a TCP connection is transferring a file of 6000 bytes. The first byte is numbered 10,001. 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:

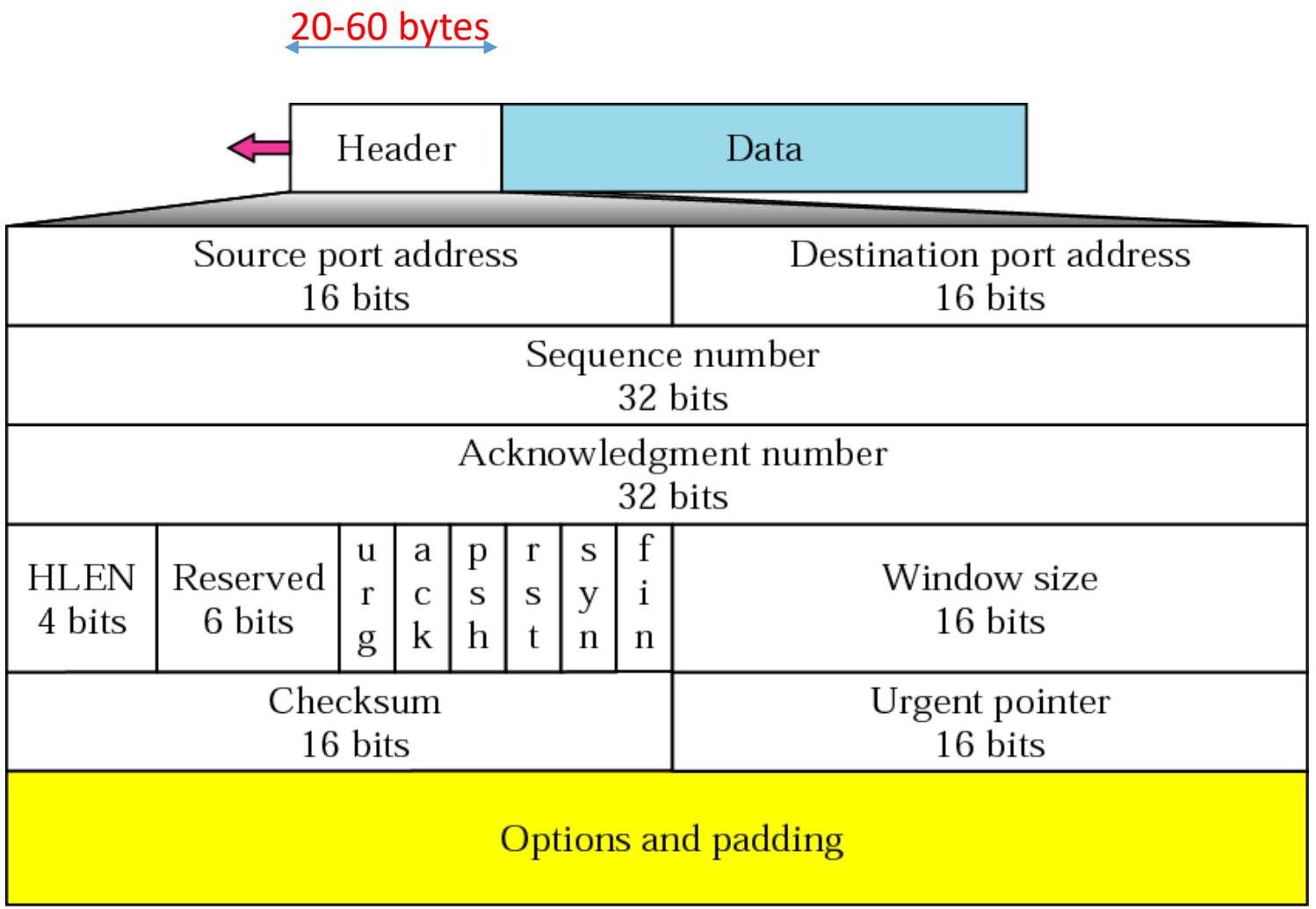
Segment 1 ==> sequence number: 10,001 (range: 10,001 to 11,000)
Segment 2 ==> sequence number: 11,001 (range: 11,001 to 12,000)
Segment 3 ==> sequence number: 12,001 (range: 12,001 to 13,000)
Segment 4 ==> sequence number: 13,001 (range: 13,001 to 14,000)
Segment 5 ==> sequence number: 14,001 (range: 14,001 to 16,000)

TCP Features….contd

- **Flow Control:** Unlike UDP, TCP provides flow control. The numbering system allows TCP to use a byte-oriented flow control.
- **Error Control:** To provide reliable service, TCP implements a byte-oriented error control mechanism.
- **Congestion Control:** TCP, unlike UDP, takes into account congestion in the network.

Note: A packet in TCP is called segment.

TCP Header Format



TCP Header...contd

- **Header length (HLEN):** This 4-bit field indicates the length of the header in the units of 4-byte words.
- **Reserved:** This is a 6-bit field reserved for future use.
- **Control:** This field defines 6 different control bits or flags. 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

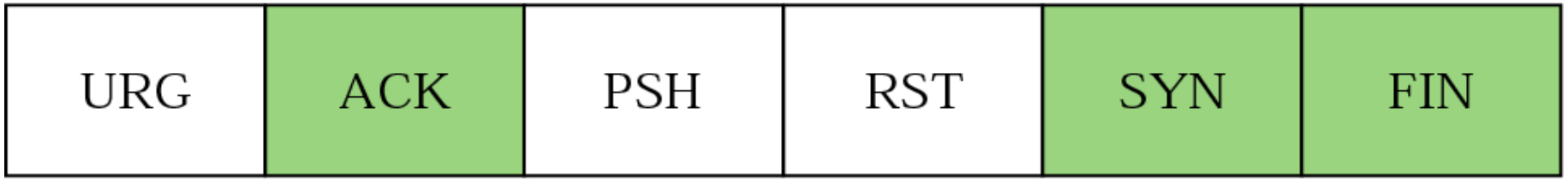


Fig: Control Field

TCP Header...contd

- **Window size:** This field defines the size of the window, in bytes, that the other party must maintain. As the length of this field is 16 bits, the maximum size of the window is 65,535 bytes.
- **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, where as the inclusion of the checksum for TCP is mandatory.
- **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.

TCP Connection

- TCP 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 events occur:
  1. ***Connection Establishment*:** The two TCPs establish a connection between them.
  2. ***Data Transfer:*** Data are exchanged in both directions.
  3. ***Connection Termination:*** The connection is terminated.

- **Note:** This is a virtual connection, not a physical connection.

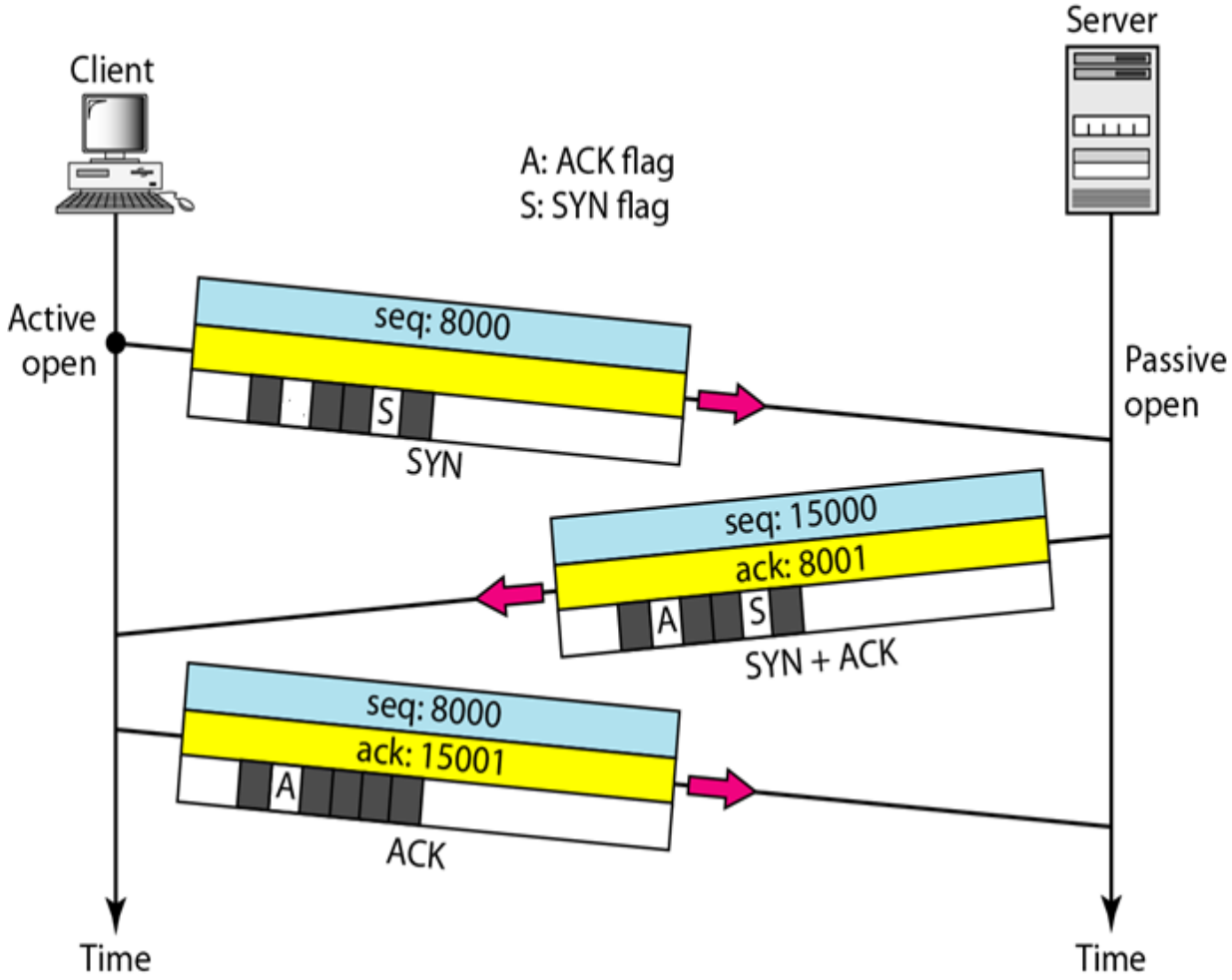
Connection Establishment: A 3-Way Handshake

- The connection establishment in TCP is called three-way handshaking.
- **Explanation:** Consider 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***.
- 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.

Illustration of a 3-Way Handshaking for *Connection Establishment*

Questions

- Why does SYN carry no data?
- Why sequence no is required?
- In case of sequence no, what should be the value of sequence no?



...contd

1. **First:** 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.

- **Note: A SYN segment can not carry data, but it consumes one sequence number.**

2. **Second:** 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.

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

3. **Third:** 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.

- **Note: An ACK segment, if carrying no data, consumes no sequence number.**

Data Transfer in TCP

- After connection is established, the client and server can both send data and acknowledgments.

Explanation: After connection is established, the client sends 2000 bytes of data in two segments. The server then sends 2000 bytes in one segment. The client sends one more segment. The first three segments carry both data and acknowledgment, but the last segment carries only an acknowledgment because there are no more data to be sent.

Note: The data segments sent by the client have the PSH(push) flag set so that the server TCP knows to deliver data to the server process as soon as they are received.

Illustration of the *Data Transfer* Phase in TCP

Note: The acknowledgment is piggybacked with the data.

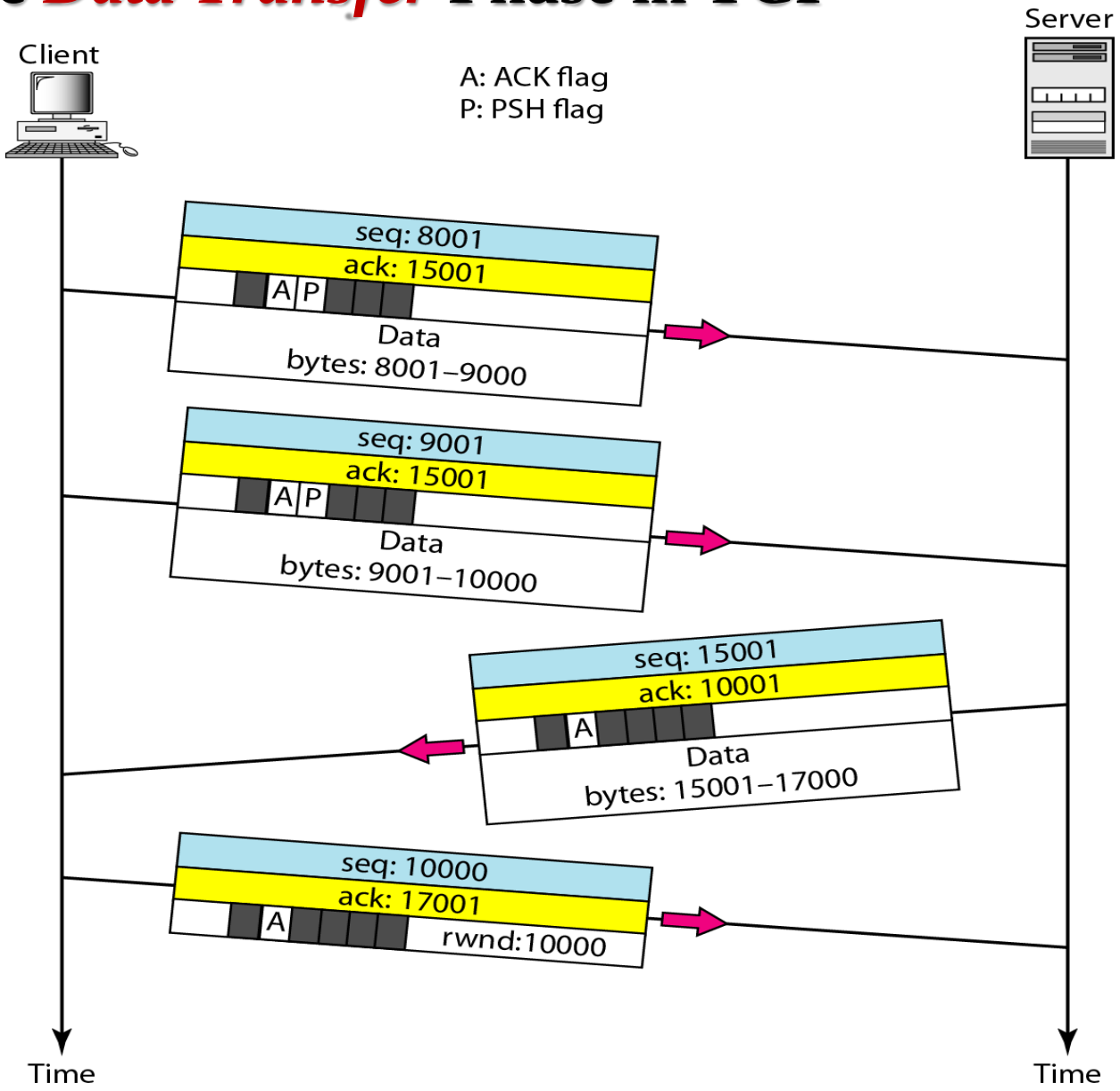
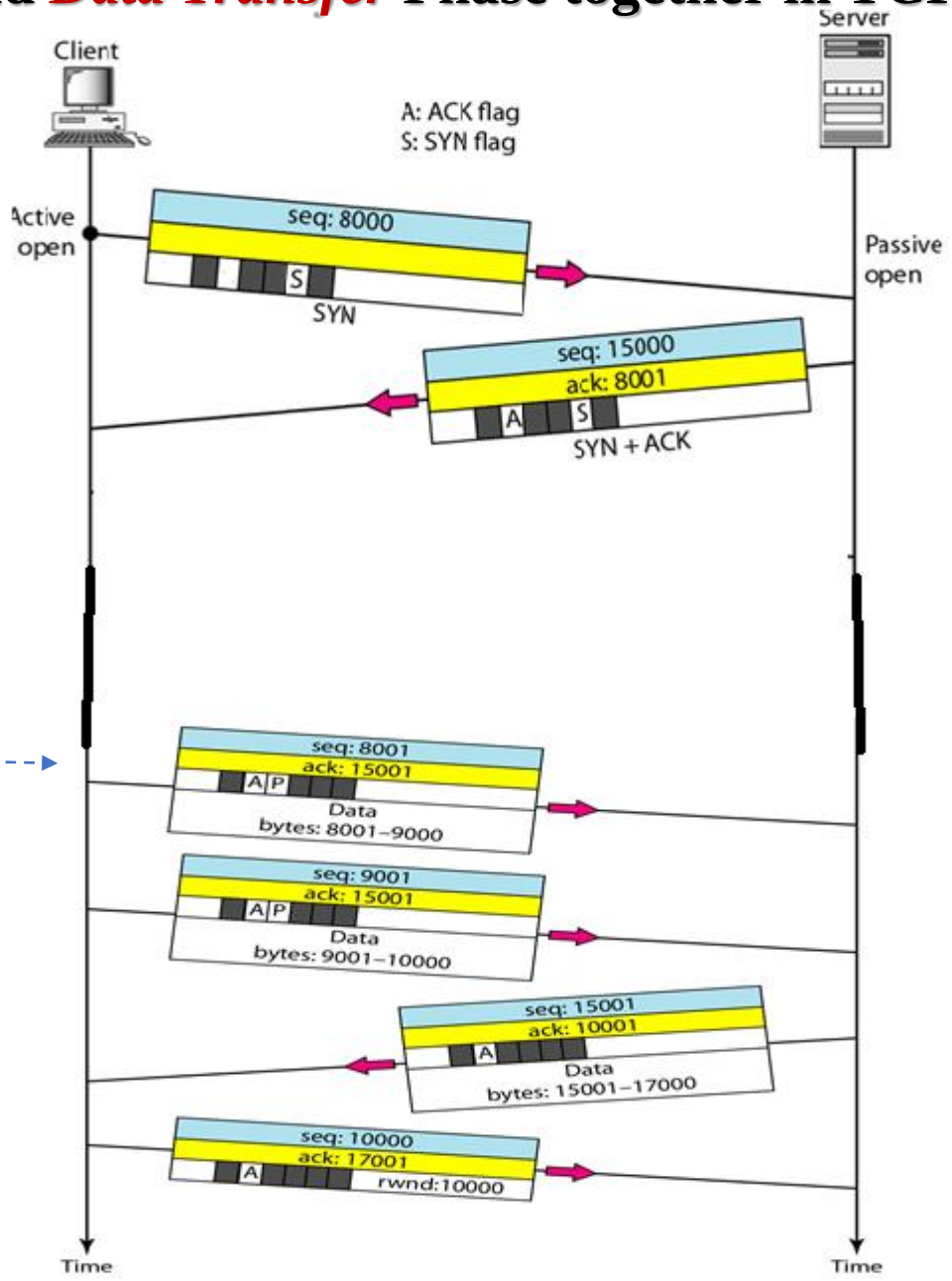


Illustration of the *connection establishment* and *Data Transfer* Phase together in TCP

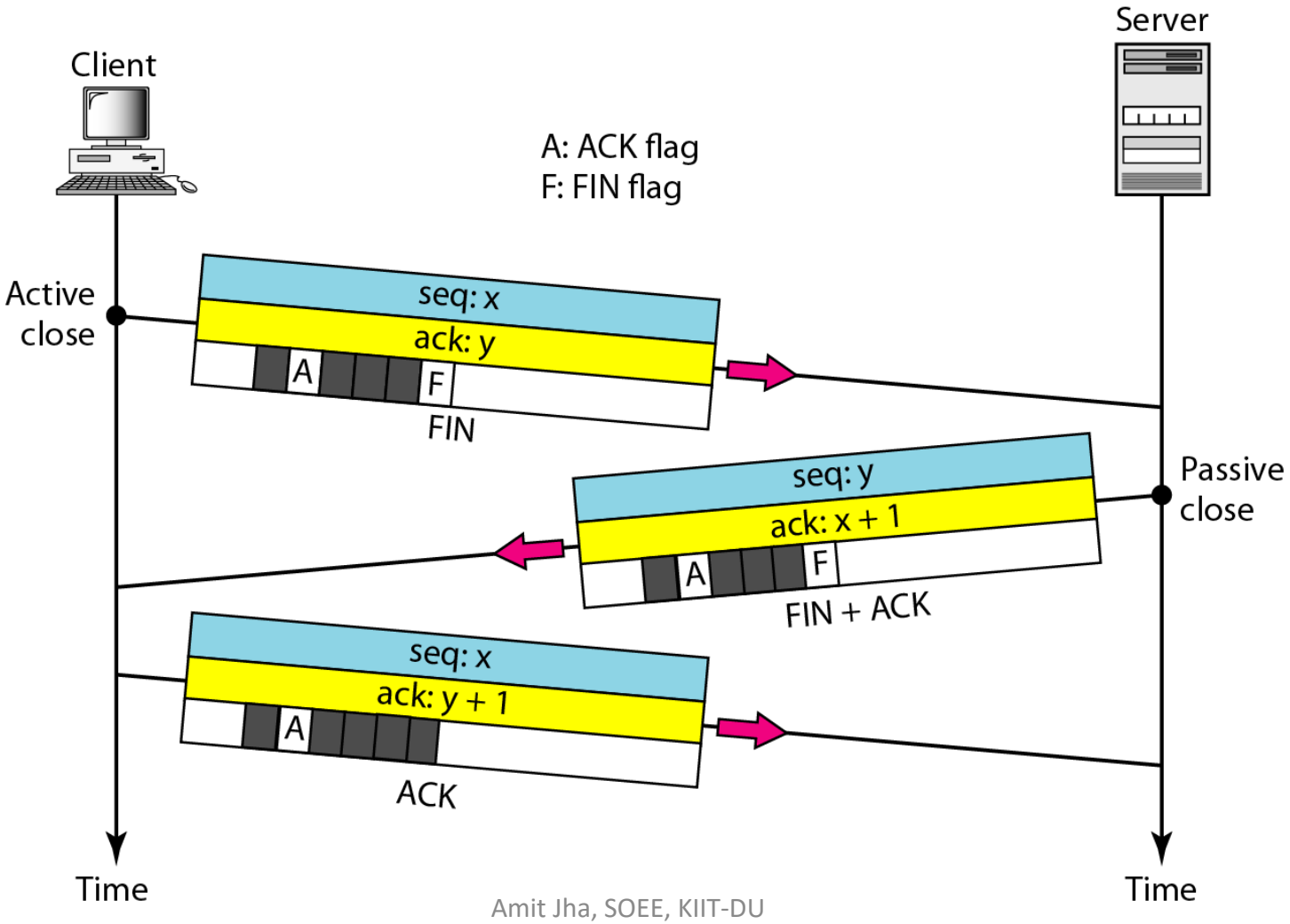
Three-way handshaking for connection establishment

Piggybacking

Data Transfer



A 3-Way Handshaking for *Connection Termination*



...contd

- Generally initiated by client.

1. **First:** 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.
   - **Note:** The **FIN** segment consumes one sequence number if it does not carry data.
2. **Second:** 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.
   - **Note:** The **FIN+ACK** segment consumes one sequence number if it does not carry data.
3. **Third:** The client TCP sends the last segment, an **ACK segment**, to confirm the receipt of the FIN segment from the TCP server.
   - **Note:** This segment can not carry data and consumes no sequence numbers.

Half Close

- Is half close possible?
- If yes, then elaborate.

- HYU:
  - With the help of **state transition diagram**, explain the process of ***connection establishment***, ***data transfer*** and ***connection termination***.

Homework

- Draw the client/server diagram for all three phases of TCP connection like; connection establishment, data transfer, and connection termination together.

Homework

1. The clients and server wants to communicate with each other using TCP. The client wants to send 2000 bytes of data in two segments of 1000 bytes each. Then server responds by sending 2000 bytes of data in one segment only. Explain the process explicitly by showing: 3-way handshaking for connection establishment, data transfer, and 3-way handshaking for connection termination. Show only the relevant fields of TCP header.
2. In relevant to TCP, explain the process of 3-way handshaking for connection establishment, data transfer and 3-way handshaking for connection termination for the following conditions:
    a. Client sends two segments namely, Seg1.1 and Seg1.2 of 1000 bytes each.
    b. Server sends two segments namely, Seg2.1 and Seg2.2 of 1000 and 1500 bytes resp.
    c. Client sends Seg1.3 of size 2000 bytes.
    d. Finally, Server replies with Seg2.3 of 500 bytes.

TCP Key- Points

- Features:
  - Connection-oriented
  - Byte-stream service
- Advantages:
  - Reliable, Full duplex
  - implements error control, flow control and congestion control
- Disadvantages:
  - More Complex Transmitter and receiver
  - more overhead as compared to UDP
  - Higher delay than UDP
- Applications: Most of the applications use TCP. To name a few:
  - HTTP, SMTP, FTP, TELNET, POP3, DHCP,…….many more