Advanced Computer Networks

Module III

Syllabus

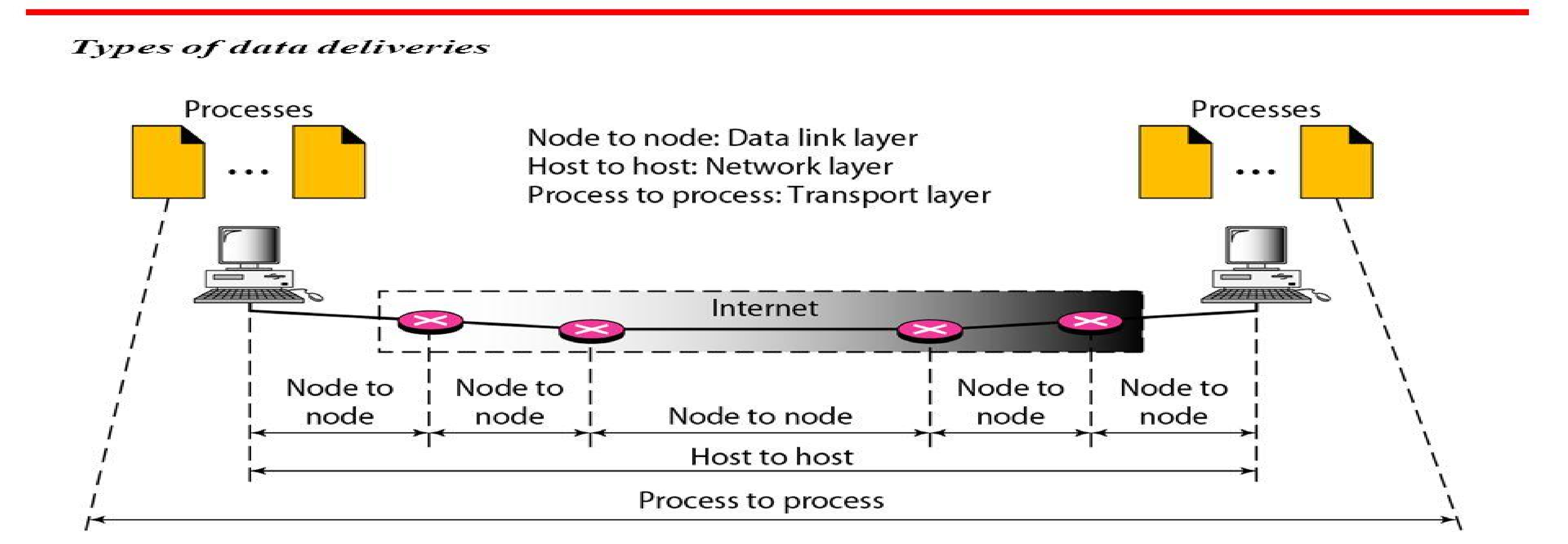
- Transport Layer and Network Layer Security: Connection oriented and connection-less services, Transmission Control Protocol and User Datagram Protocol – characteristics, reliability mechanism, performance considerations. Flow Control - Go-back N and Selective Repeat sliding window protocols and Congestion control mechanisms. IP Addressing - IPv4 Classful addressing, Subnet masking and Packet format. Unicast routing algorithms – Link- State and Distance Vector routing, Routing protocols – RIP and OSPF. SSL for secure web communication – handshake process and vulnerabilities, IPSec for secure IP communication and firewalls – types, Network Address Translation management.

Transport Layer and Network Layer Security - Introduction

- Network Layer Responsibility: Ensures delivery of datagrams between two hosts (host-to-host delivery).
- Internet communication is not just between nodes or hosts but between processes (application programs).
- So there is a need for Process-to-Process delivery.
- Real communication happens between processes, not just hosts.
- Several processes may run on both the source and destination hosts simultaneously.
- A mechanism is required to ensure data is delivered from the correct source process to the corresponding destination process.

Continues..

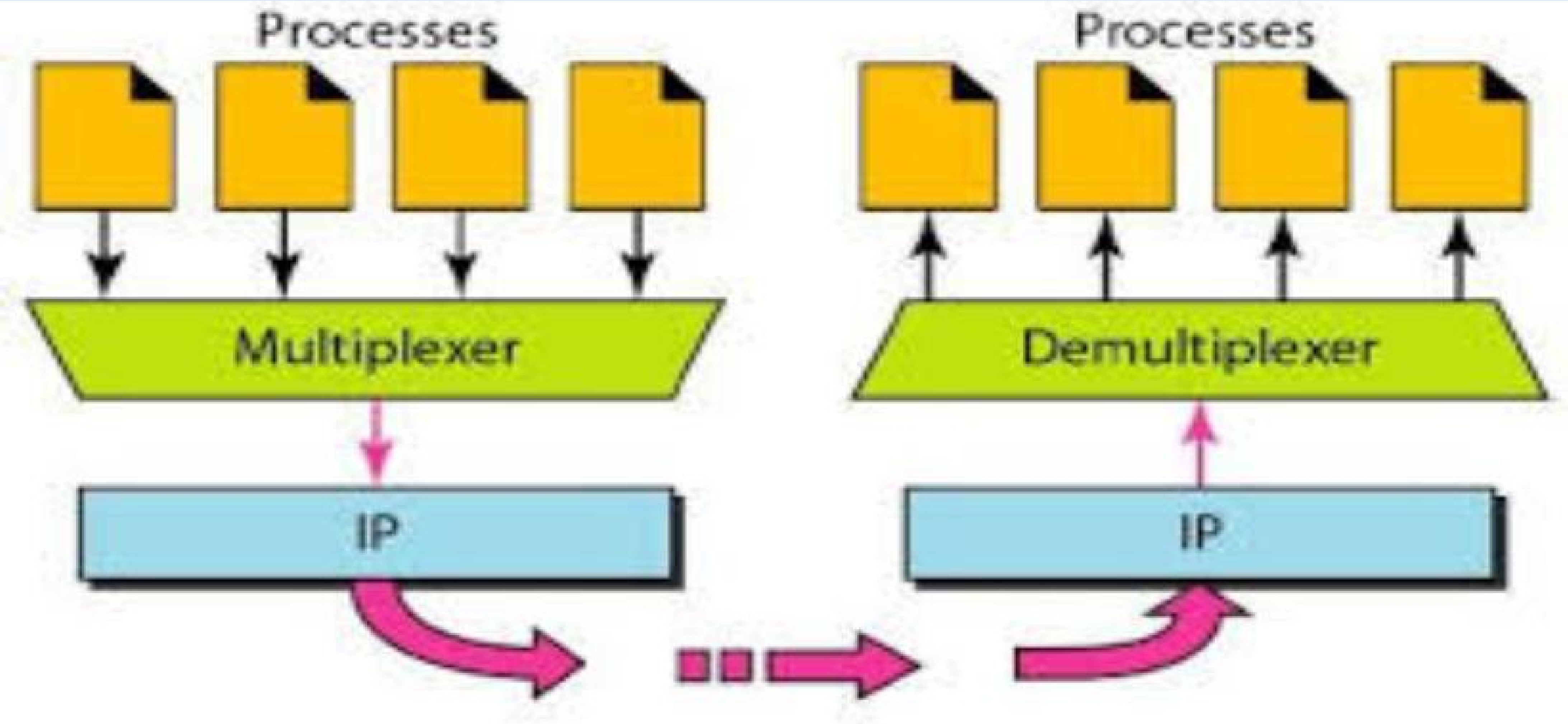
- Transport Layer ensures process-to-process delivery of packets.
- It facilitates data transfer between processes rather than just hosts.
- Processes communicate in a client-server relationship.
- There are three types of data deliveries:- Node to Node, Host to Host and Process to Process



Multiplexing and Demultiplexing

- Multiplexing (Sender Side)
  - Multiple processes need to send packets.
  - Only one transport layer protocol is available at a time.
  - The many-to-one relationship requires multiplexing.
  - Messages from different processes are identified by their port numbers.
  - The transport layer adds a header and forwards the packet to the network layer.
- Demultiplexing (Receiver Side)
  - The transport layer receives datagrams from the network layer.
    The one-to-many relationship requires demultiplexing.
  - Error checking is performed, and the header is removed.
  - The message is delivered to the appropriate process based on the port number.

Continues..



Connection-Oriented and Connectionless services

Difference Between Connection-Oriented and Connectionless Services

Feature	Connection-Oriented Service	Connectionless Service
Connection Setup	Required before data transfer	Not required
Packet Ordering	Ensures ordered delivery	May arrive out of order
Reliability	Reliable (uses acknowledgments)	Unreliable (no acknowledgments)
Error Handling	Error checking and retransmission	No error correction
Speed	Slower due to overhead	Faster due to minimal control
Examples	TCP, SCTP	UDP

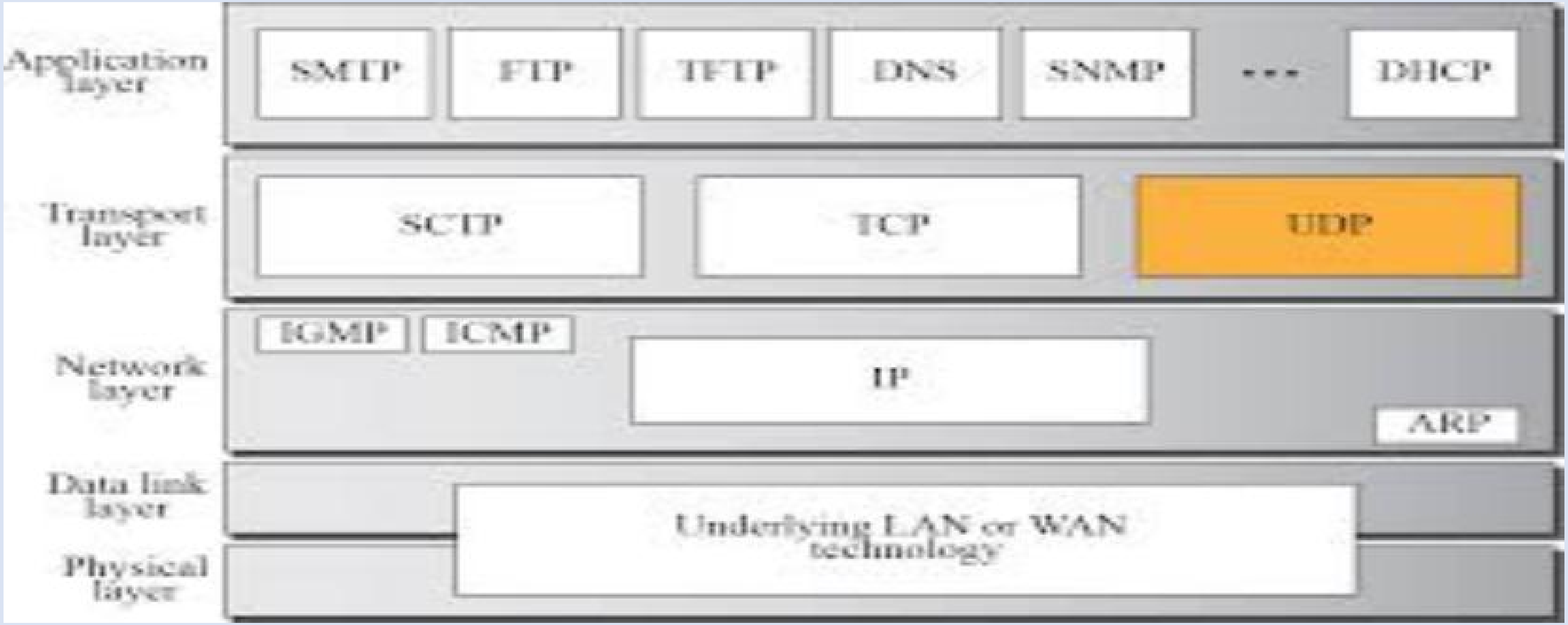
Continues..

- Reliable vs. Unreliable Transport Layer
- The transport layer can provide reliable or unreliable services based on application needs.
- Reliable Transport Service:
  - Implements flow and error control to ensure accurate data delivery.
  - More complex and slower due to additional processing.
  - Used when applications require guaranteed data integrity.
- Unreliable Transport Service:
  - No built-in flow or error control.
  - Faster but may result in packet loss or errors.
  - Suitable for real-time applications or when the application manages reliability.

Continues..

- Transport Layer Protocols in the Internet:
  - UDP: Connectionless and unreliable.
  - TCP & SCTP: Connection-oriented and reliable.
- Why Reliability at the Transport Layer?
  - The Data Link Layer ensures reliability only between two nodes.
  - The Network Layer provides best-effort delivery (unreliable).
  - Transport Layer reliability ensures end-to-end error-free communication.

Position of UDP, TCP, and SCTP in TCP IP suite



User Datagram Protocol

- Connectionless unreliable transport protocol.
- Provides process to process communication.
- Simple protocol.
- Use minimum overhead.
- Message passing with less reliability can use UDP protocol.
- Why UDP ?
  - Finer application-level control over what data is sent, and when.
  - No connection establishment.
  - No connection state.
  - Small packet header overhead

Continues..

- UDP takes messages from the application process, attaches source and destination port number fields for the multiplexing / demultiplexing service, adds two other small fields, and passes the resulting segment to the network layer.
- The network layer encapsulates the transport-layer segment into an IP datagram and then makes a best-effort attempt to deliver the segment to the receiving host.
- If the segment arrives at the receiving host, UDP uses the destination port number to deliver the segment's data to the correct application process.
- With UDP there is no handshaking between sending and receiving transport-layer entities before sending a segment.
- For this reason, UDP is said to be connectionless.

Well-known ports used with UDP

- Some port numbers can be used by both UDP and TCP.
- Well-known ports are predefined port numbers (0-1023) assigned by IANA (Internet Assigned Numbers Authority).
- They are used by common network services and protocols like HTTP (port 80), DNS (port 53), and DHCP (port 67/68).
- These ports help in identifying and standardizing communication between applications across networks.
- Why well-known ports?
  - Standardized communication across networks.
  - Service identification for specific applications.
  - Interoperability between systems and devices.
  - Efficient routing and traffic management.
  - Security enforcement through firewalls and access control.
  - Simplifies application development by avoiding conflicts.
  - Quick troubleshooting for network administrators.

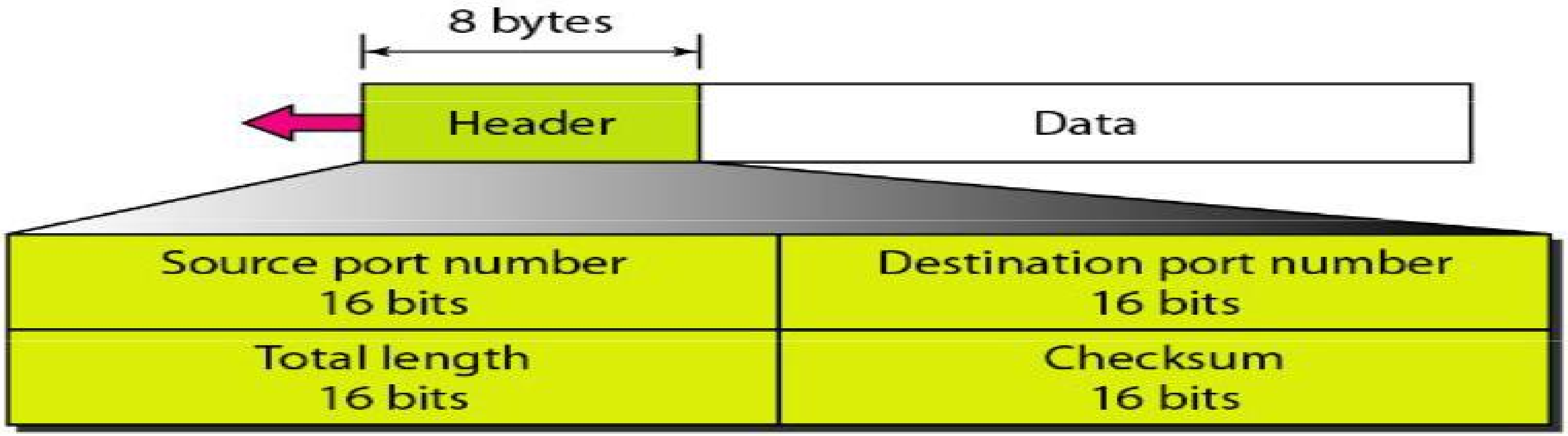
Well-known ports used with UDP

Table 4.1 Well-known port numbers used by UDP

Port	Protocol	Description
1	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	Name server	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)

Continues..

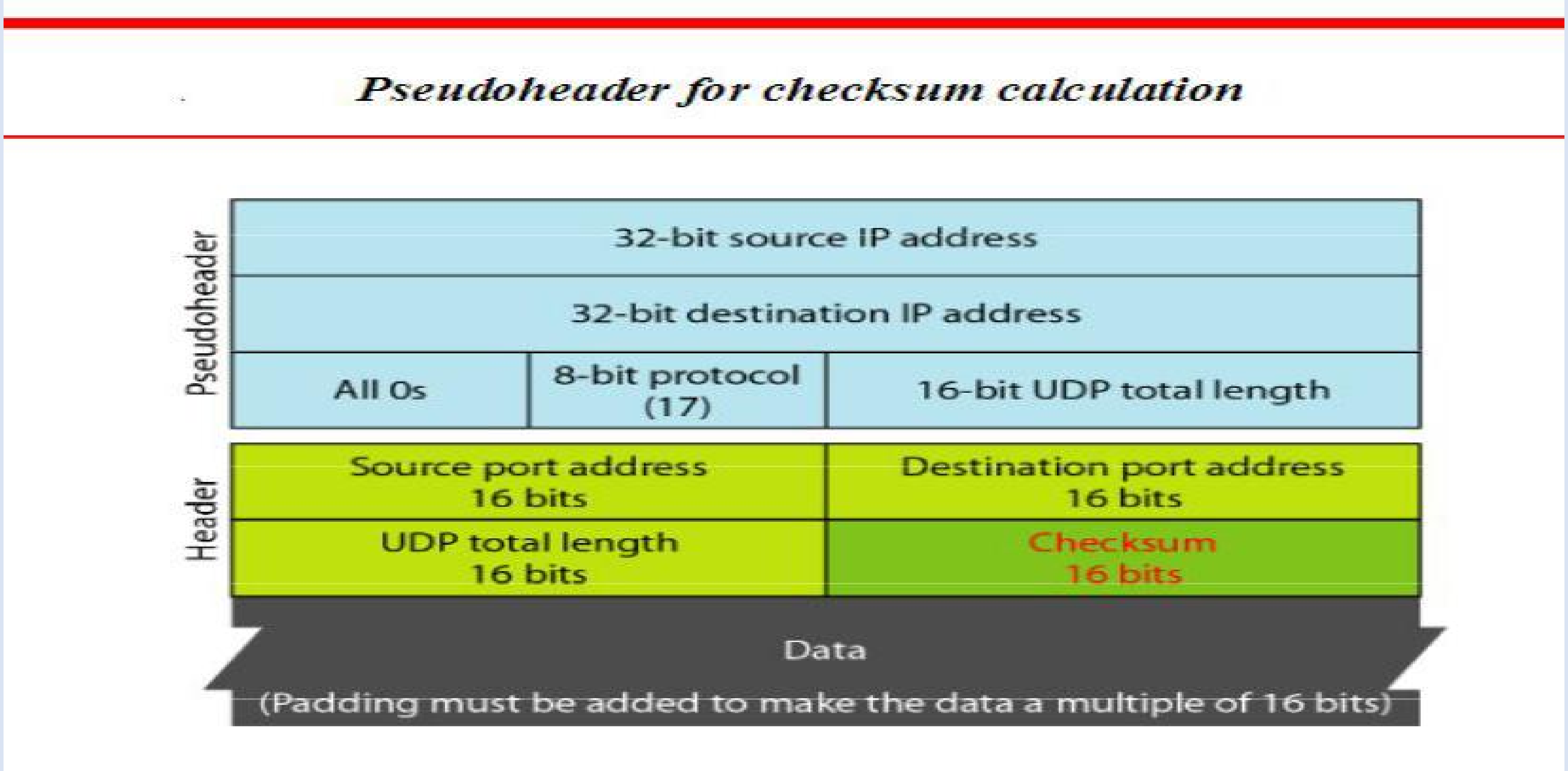
User datagram format



Continues..

- User Datagram
  - UDP packets are called user datagram.
  - Fixed size header(8 bytes)
  - Four fields (2 bytes)
  - First two fields : source and destination port address
  - Third field : total length of the user datagram, header plus data. The 16 bits can define a total length of 0 to 65535 bytes.
  - The last bit may carry optional checksum.

Continues..


Pseudoheader for checksum calculation
Pseudoheader
32-bit source IP address
32-bit destination IP address
All 0s
8-bit protocol (17)
16-bit UDP total length
Header
Source port address 16 bits
Destination port address 16 bits
UDP total length 16 bits
Checksum 16 bits
Data
(Padding must be added to make the data a multiple of 16 bits)

Continues..

- Process to process communication.
- Using socket address: a combination of IP address and port address.
- Connectionless Service.
- Each datagram send by UDP is an independent datagram.
- Packets from same source process to same destination process are also independent to each other.
- The user datagram are not numbered.
- No connection establishment and termination.
- User datagram can take different path.
- Disadvantages of UDP protocol.
- Short messages less than 65,507 bytes can use UDP.

Continues..

- Flow Control
  - Simple protocol.
  - No flow control.
  - The receiver may overflow with incoming messages.
  - The process using this UDP should provide this service.
- Error Control
  - No error control mechanism except for checksum.
  - The sender does not know if a message is lost or is duplicated.
  - When an error is detected through the checksum, the user datagram is silently discarded.
  - The process using this UDP service should provide this.

Continues..

- Checksum
  - The UDP checksum provides for error detection.
  - The checksum is used to determine whether bits within the UDP segment have been altered as it moved from source to destination.
  - UDP at the sender side performs the 1s complement of the sum of all the 16-bit words in the segment, with any overflow encountered during the sum being wrapped around. This result is put in the checksum field of the UDP segment.

Continues..

- Uses of UDP
  - Ideal for simple request-response communication with minimal error control.
  - Suitable for applications with internal flow and error control, like TFTP.
  - Supports multicasting, which TCP does not.
  - Used in network management (e.g., SNMP).Helps in routing updates, such as RIP.

Transmission Control Protocol

- TCP (Transmission Control Protocol) is a core transport layer protocol in the Internet model.
- Process-to-process communication.
- It ensures reliable, error-free, and in-order data delivery.
- Unlike UDP, TCP establishes a connection before transmitting data.
- Example: When you load a webpage, your browser establishes a TCP connection with the web server to ensure all elements (HTML, images, scripts) are received correctly.

Continues..

- Why TCP is Important?
  - Ensures Reliable Communication: Every packet is acknowledged to prevent data loss.
  - Manages Network Congestion: Dynamically adjusts speed to avoid overloading the network.
  - Guarantees Ordered Delivery: Packets arrive in sequence, avoiding jumbled data.
  - Example: When downloading a file, TCP ensures each chunk arrives in order and is complete.

Continues..

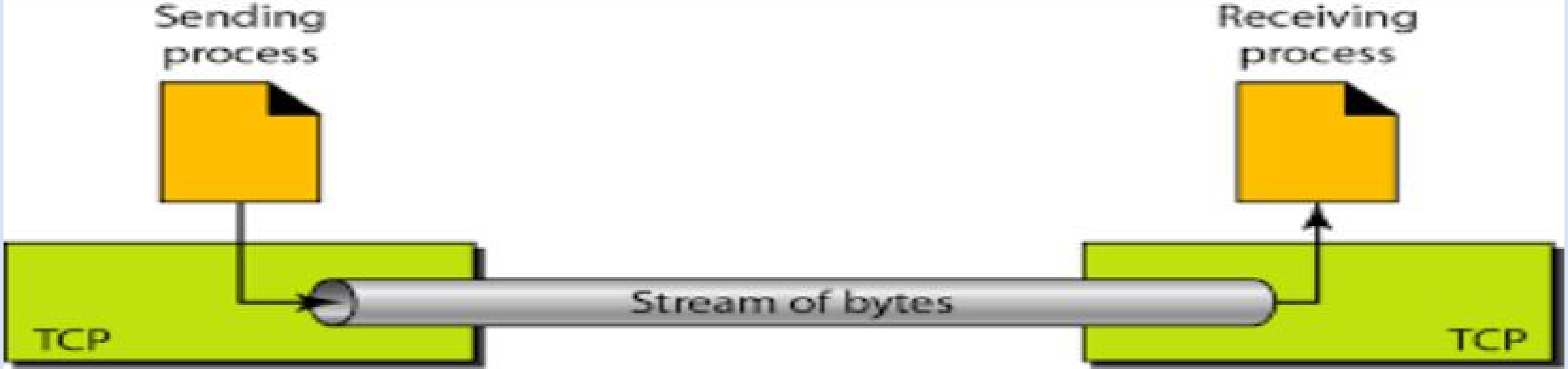
- TCP Services
  - Reliable Delivery: Resends lost packets, ensuring complete data transmission.
  - Connection-Oriented Communication: Establishes a session before sending data.
  - Flow Control: Regulates data flow to match the receiver’s processing speed.
  - Error Control: Uses checksums and acknowledgments (ACKs) to detect and correct errors.
  - Example: Sending an email via SMTP (which runs on TCP) ensures every part of the message is delivered accurately.

TCP Well Known Ports

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	FIP, Data	File Transfer Protocol (data connection)
21	FIP, Control	File Transfer Protocol (control connection)
23	TELNET	Tenninal 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

Continues..

- TCP treats data as a continuous stream of bytes, unlike UDP’s discrete messages.
- TCP provides a virtual connection between sender and receiver, simulating a "tube" for seamless data flow.
- The sending process writes data to the stream, while the receiving process reads from it continuously.



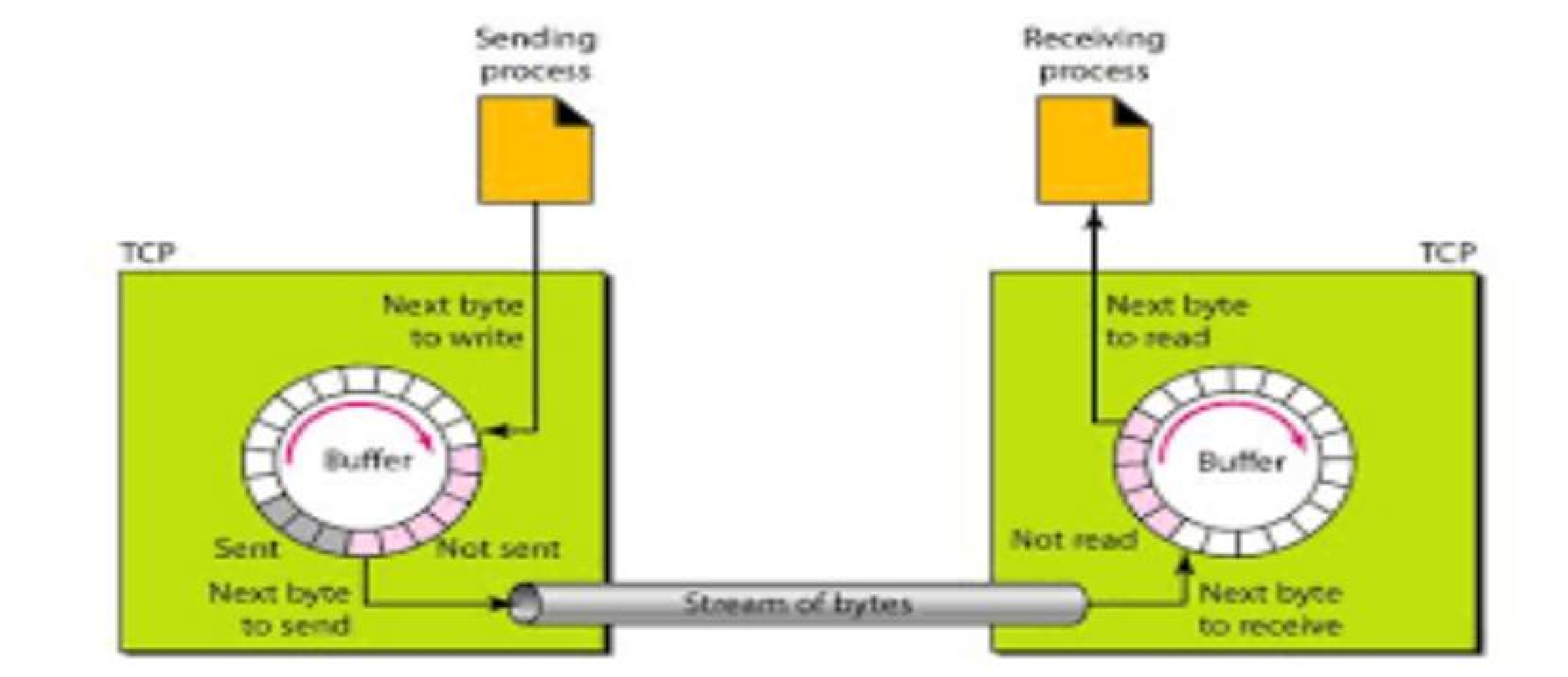
Continues..

- TCP uses buffers to handle differences in the speed of data writing (sending) and reading (receiving).
- Each direction has its own buffer—one for sending and one for receiving.
- Buffers help with flow and error control in TCP communication.
- A circular array is one way to implement these buffers, storing data in a continuous loop.
- Buffer sizes vary—they can be hundreds or thousands of bytes and are not always equal in size.

Continues..

- Sending buffer has three sections:
  - Empty chambers (ready for new data).
  - Sent but unacknowledged bytes (held until acknowledged).
  - Bytes ready to be sent (may be sent partially due to network conditions).
- Acknowledged bytes are recycled, making space for new data in the circular buffer.
- Receiving buffer has two sections:
  - Empty chambers (for incoming data).
  - Received bytes (ready to be read by the receiving process).
- TCP segments data before sending, grouping bytes into packets called segments.
- Segments are encapsulated in IP datagrams for transmission, and TCP handles issues like reordering, loss, or corruption transparently.

Continues..



Sending process
Receiving process
TCP
TCP
Next byte to write
Next byte to read
Buffer
Buffer
Sent
Not sent
Not read
Next byte to send
Stream of bytes
Next byte to receive

Full-Duplex Communication

- TCP supports full-duplex communication, allowing data to flow simultaneously in both directions.
- Each TCP connection has both a sending and receiving buffer, enabling bidirectional data transfer.
- Segments move in both directions independently, ensuring continuous and efficient communication.

Continues..

- Connection-oriented service
  - TCP establishes a connection before data exchange and terminates it afterward.
  - The connection is virtual, not physical, as data travels in IP datagrams that may take different paths.
  - TCP ensures ordered and reliable delivery, even if packets are lost, out of order, or corrupted.
  - TCP creates a stream-oriented environment, similar to a bridge connecting two sites for seamless communication.
- Reliable service
  - TCP ensures reliable data transmission.
  - Acknowledgment Mechanism: It verifies the successful arrival of data.
  - Error Control

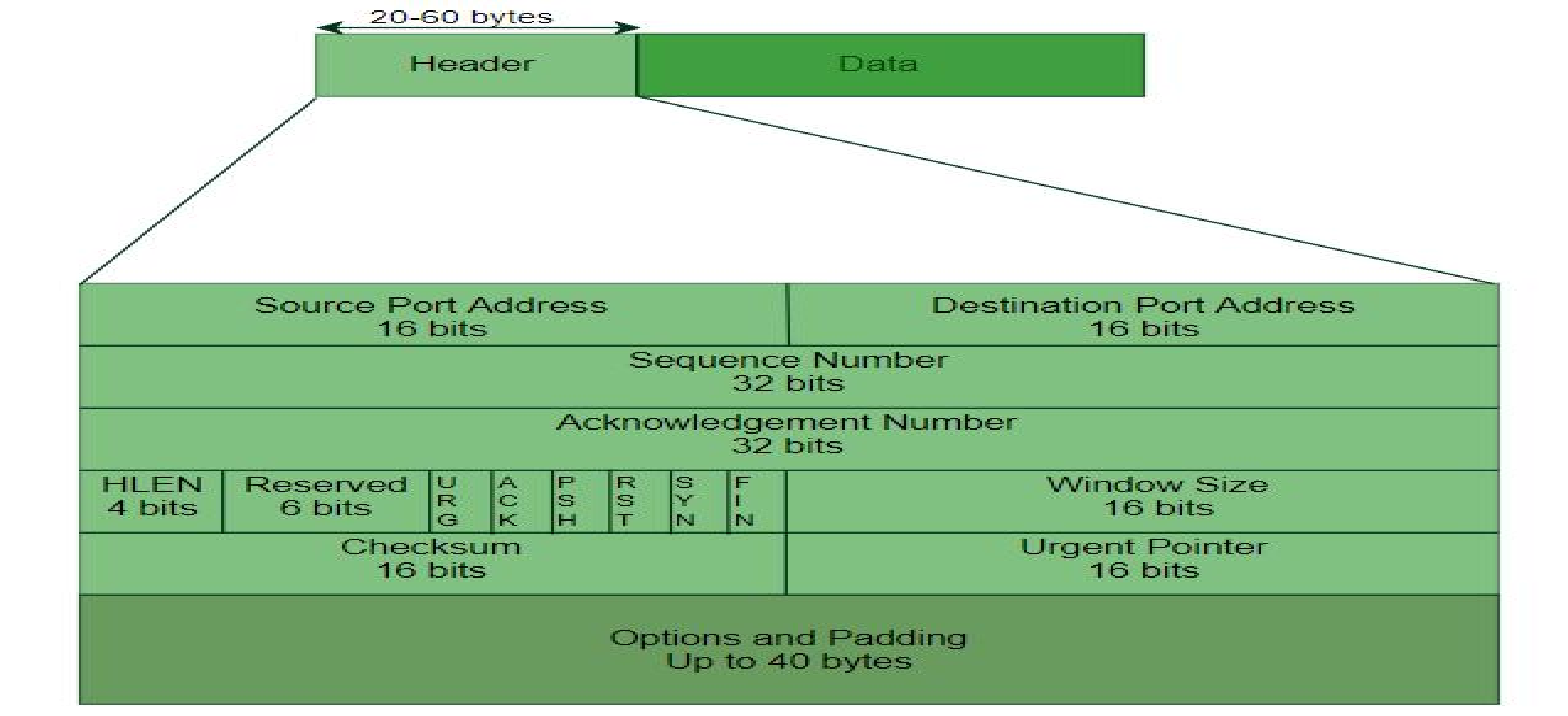
TCP Features

- No Segment Number Field: TCP tracks transmitted/received segments but does not use a segment number field in the header.
- Sequence & Acknowledgment Numbers: These fields represent byte numbers, not segment numbers.
- Byte Numbering: TCP numbers all data bytes independently in each direction.
- Random Starting Number: The first byte number is randomly chosen between 0 and 232−12 32 −1.
- Flow & Error Control: Byte numbering is crucial for managing flow and error control.
- Sequence Number: Each segment's sequence number is the number of its first byte.
- Example
  - Random Start: The first byte number is randomly chosen (e.g., 1057).
  - Byte Range: If 6000 bytes are sent, they are numbered from 1057 to 7056.

Features of TCP

- Flow Control: TCP prevents the receiver from being overwhelmed by regulating the data sent.
- Byte-Oriented Control: TCP uses a numbering system for byte-based flow control.
- Error Control: Ensures reliable transmission by detecting lost or corrupted segments.
- Segment-Based Detection: Errors are detected at the segment level but managed at the byte level.
- Congestion Control: TCP adjusts data transmission based on network congestion, unlike UDP.

Segment Format



Continues..

- Segment Structure: Consists of a header (20–60 bytes) and application data.
- Header Size: 20 bytes without options, up to 60 bytes with options.
- Fields
  - Source Port Address: 16-bit field specifying the sender's application port.
  - Destination Port Address: 16-bit field specifying the receiver's application port.
  - Sequence Number:
    - 32-bit field indicating the first byte number in the segment.
    - Byte Numbering: TCP numbers each byte to ensure reliable transmission.
    - Initial Sequence Number (ISN): Randomly generated during connection setup, different for each direction.

Continues..

- Acknowledgment Number Field:
  - A 32-bit field in the TCP header.
  - Specifies the next expected byte from the sender.
  - How It Works:
  - If the receiver successfully receives byte x, it sends x + 1 as the acknowledgment number.
  - Ensures reliable communication and prevents data loss or duplication.
  - Example Scenario:
    - Sender → Receiver: Sends 1000 bytes starting from byte 5000 (bytes 5000 to 5999).
    - Receiver → Sender: Acknowledges the reception by sending ACK = 6000.
    - Sender → Receiver: Sends bytes 6000–6999.
    - Receiver → Sender: Sends ACK = 7000 to indicate the next expected byte.

Continues..

- Header Length:
  - A 4-bit field indicating the number of 4-byte words in the TCP header.
  - The header length ranges from 20 to 60 bytes.
  - Field value can be between 5 (5 x 4 = 20 bytes) and 15 (15 x 4 = 60 bytes).
- Reserved:
  - A 6-bit field reserved for future use.
  - Currently not used but may be allocated in future versions of TCP.
- Control:
  - A field that defines 6 control bits/flags.
  - These flags control various functions in TCP communication (e.g., SYN, ACK, FIN).

Continues..

Table 4.3 Description of flags in the control field

Flag	Description
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledge field is valid.
PSH	Push the data.
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection
FIN	Terminate the connection.

Continues..

- Window Size:
  - The window size in TCP is used for flow control to manage the amount of data that can be sent before receiving an acknowledgment.
  - It helps in preventing congestion and ensures that the receiver's buffer is not overwhelmed by too much data.
  - Defines the size of the window (in bytes) that the receiver maintains.
  - The field is 16 bits long, allowing a maximum window size of 65,535 bytes.
  - Referred to as the receiving window (rwnd), determined by the receiver.
  - The sender must comply with the receiver's window size.

Continues..

- Checksum:
  - A 16-bit field used for error-checking.
  - The checksum is mandatory in TCP (unlike in UDP, where it's optional).
  - The same checksum calculation method as UDP is used, including a pseudoheader with a protocol field value of 6.
- Urgent Pointer:
  - A 16-bit field, valid only if the urgent flag is set.
  - Indicates the last byte of urgent data by adding the value to the sequence number.
- Options:
  - Up to 40 bytes of optional information can be included in the TCP header.

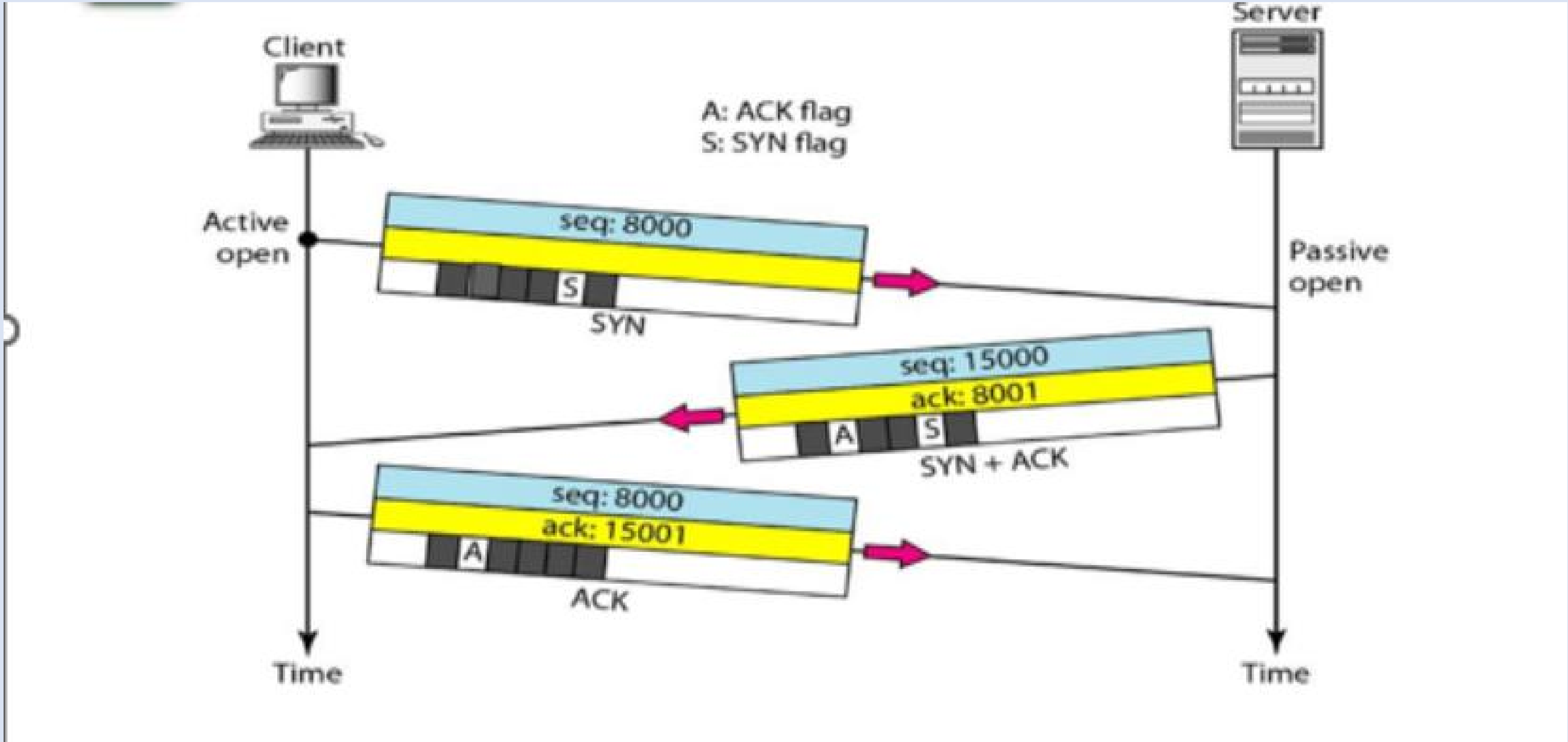
TCP Connection

- TCP as a Connection-Oriented Protocol
  - Establishes a virtual path between source and destination.
  - All message segments travel through this virtual path.
  - Ensures acknowledgment and retransmission of lost or damaged segments.
- Three Phases of a TCP Connection
  - Connection Establishment
    - TCP operates in full-duplex mode.
    - Both parties must agree before data transfer begins.
  - Data Transfer
    - Ensures reliable, ordered delivery of segments.
    - Retransmits lost or corrupted segments.
  - Connection Termination
    - Properly closes the communication channel after data transfer.

Continues..

- Three-Way Handshaking (Connection Establishment Process)
  - Step 1: Server initiates a passive open, indicating readiness to accept connections.
  - Step 2: Client requests an active open, signaling the need for a connection.
  - Step 3: TCP initiates a three-way handshake to establish the connection.

Continues..


Client
Server
A: ACK flag
S: SYN flag
Active open
seq: 8000
S
SYN
Passive open
seq: 15000
ack: 8001
A
S
SYN + ACK
seq: 8000
ack: 15001
A
ACK
Time
Time

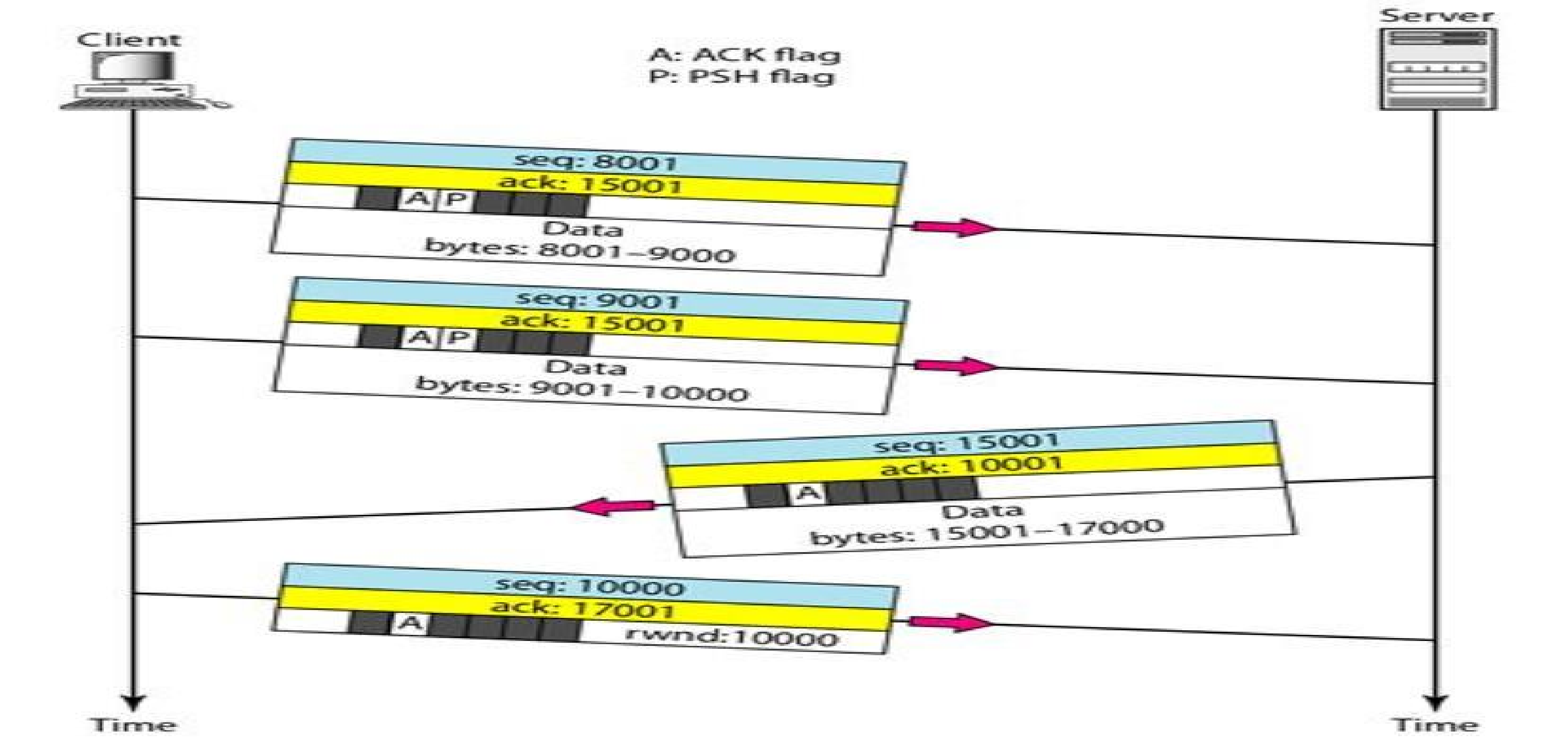
Continues..

- Step 1:
  - The client sends a SYN segment with only the SYN flag set.
  - Used to synchronize sequence numbers with the server.
  - Consumes one sequence number, incrementing it by 1 for data transfer.
  - Though it carries no real data, it is treated as having one imaginary byte.
- Step 2:
  - The server sends a SYN + ACK segment with both SYN and ACK flags set.
  - This segment acknowledges the client's SYN request.
  - It also acts as a SYN segment to initiate communication in the other direction.
  - The segment consumes one sequence number.

Continues..

- Step 3:
  - The client sends the third segment, which is an ACK segment.
  - It acknowledges the second segment using the ACK flag and acknowledgment number.
  - The sequence number remains the same as in the SYN segment.
  - The ACK segment does not consume any sequence numbers.
- **What is Simultaneous Open?**
  - Occurs when both processes issue an **active open** simultaneously.
  - Both sides send a **SYN + ACK** segment to each other.
  - A **single connection** is established between them.

Data Transfer



Continues..

- Bidirectional Data Transfer
  - Once the connection is established, both client and server can send data and acknowledgments.
  - Acknowledgments can be piggybacked with data traveling in the same direction.
  - The PSH (push) flag is used to indicate immediate data delivery.

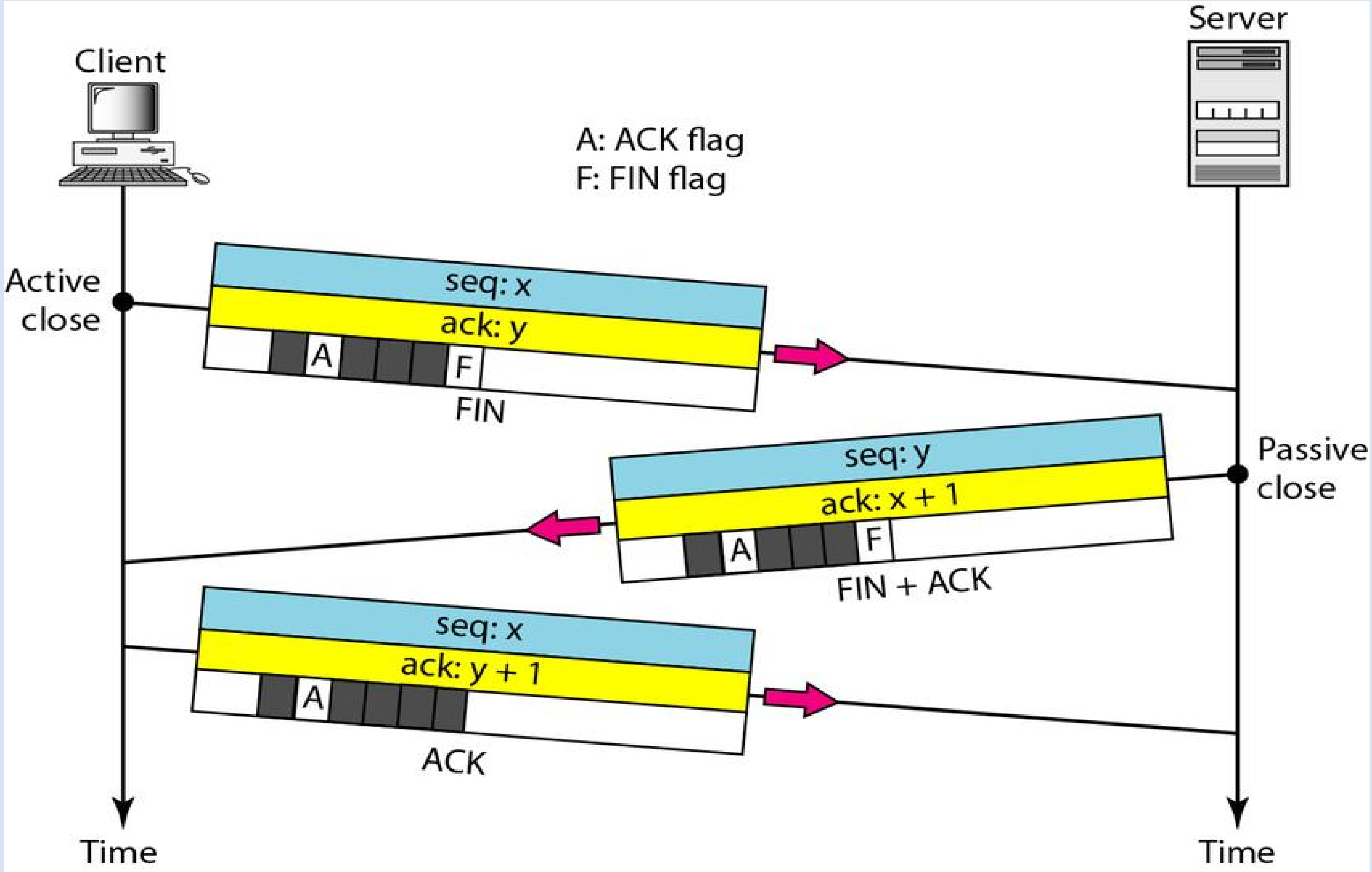
Continues..

- Example Scenario Explanation
  - Client Sends Data (2000 bytes in Two Segments)
    - The client has 2000 bytes of data to send.
    - It divides this data into two segments (e.g., 1000 bytes each).
    - Each segment contains both data and an acknowledgment for previous server messages (if any).
  - Server Responds with 2000 Bytes in One Segment
    - The server receives both segments from the client.
    - It then sends 2000 bytes of its own data in one single segment instead of splitting it.
    - This segment also includes an acknowledgment for the client's data.
  - Client Sends a Final Acknowledgment
    - After receiving the server's 2000-byte segment, the client has no more data to send.
    - It still needs to acknowledge the received data.
    - It sends a final segment that contains only an acknowledgment (ACK) with no data.

Continues..

- Urgent Data
  - TCP is stream-oriented, treating data as a continuous stream of bytes.
  - Each byte has a position in the stream.
  - Sometimes, an application needs to send urgent data that must be processed out of order.
  - Urgent data allows the receiving application to prioritize specific bytes.
  - For that send a segment with the URG bit set.

Connection Termination



Continues..

- Connection Termination can be initiated by either the client or server, but it's usually initiated by the client.

Three-Way Handshaking for Connection Termination

- Step 1
    - The **client TCP** sends a **FIN segment** after receiving a close command from the client process.
    - The **FIN segment** may include the **last chunk of data** from the client.
    - This marks the start of the termination process in a normal situation.

Continues..

- Step 2
  - After receiving the FIN segment, the server TCP informs its process.
  - The server sends a FIN + ACK segment to:
    - Confirm receipt of the FIN segment from the client.
    - Announce the closure of the connection in the other direction.
  - The FIN + ACK segment may also include the last chunk of data from the server.
  - If no data is included, the segment consumes only one sequence number.
- Step 3
  - The client TCP sends the last segment, an ACK segment.
  - This segment confirms receipt of the FIN segment from the server.
  - The acknowledgment number is 1 plus the sequence number received in the server's FIN segment.
  - The ACK segment cannot carry data and consumes no sequence numbers.

Flow Control

- What is Flow Control?
  - Flow control is a technique used to manage the rate of data transmission between two devices in a network.
  - It ensures the sender doesn't overload the receiver with too much data.
- Why is Flow Control Important?
  - Prevents buffer overflow at the receiving end.
  - Ensures efficient and reliable communication between devices.
  - Helps avoid data loss and retransmissions.
- Types of Flow Control Mechanisms:
  - Stop-and-Wait: The sender waits for an acknowledgment after sending each frame.
  - Sliding Window: Allows the sender to send multiple frames before needing an acknowledgment.
  - Credit-Based Flow Control: The receiver grants permission (credits) to the sender on how many packets it can send.

Continues..

- Flow Control in TCP:
  - Uses a sliding window to control the flow of data.
  - The window size adjusts dynamically based on the receiver’s buffer capacity.
- Key Benefits of Flow Control:
  - Prevents data congestion.
  - Enhances network reliability.
  - Optimizes bandwidth utilization.

Sliding Window Protocol