Transport Layer

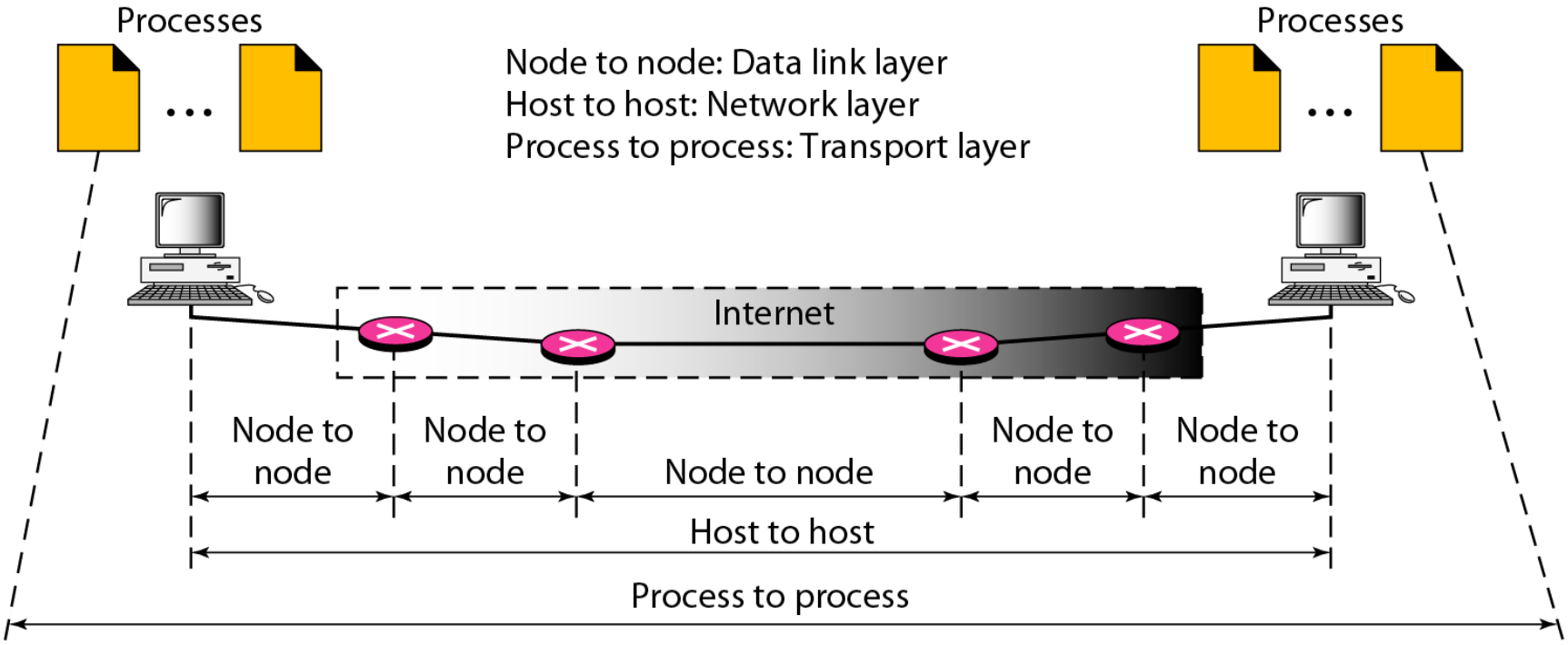
Introduction

- The transport layer is the fourth layer from the bottom in the OSI reference model.
- It is responsible for message delivery from process running in source computer to the process running in the destination computer.
- Transport layer does not perform any function in the intermediate nodes.
- It is active only in the end systems.

Introduction

- Data Link Layer is responsible for delivery of frames between two neighboring nodes over a link.
  - —This is called **node-to-node delivery**.
- Network Layer is responsible for delivery of datagrams between two hosts.
  - —This is called **host-to-host delivery**.
- Transport Layer is responsible for delivery of entire message from one process running on source to another process running on destination.
  - —This is called **process-to process delivery**.

Types of data deliveries



Transport Layer Design Issues

- The transport layer delivers the message from one process to another process running on two different hosts.
- Thus, it has to perform number of functions to ensure the accurate delivery of message.
- The various functions of transport layer are:
  - Establishing, Maintaining & Releasing Connection
  - Addressing
  - Data Transfer
  - Flow Control
  - Error Control
  - Congestion Control

Transport Layer Design Issues (2)

- **Establishing, Maintaining & Releasing Connection:**
- The transport layer establishes, maintains & releases end-to-end transport connection on the request of upper layers.
- Establishing a connection involves allocation of buffers for storing user data, synchronizing the sequence numbers of packets etc.
- A connection is released at the request of upper layer.

Transport Layer Design Issues (3)

- **Addressing**:
- In order to deliver the message from one process to another, an addressing scheme is required.
- Several process may be running on a system at a time.
- In order to identify the correct process out of the various running processes, transport layer uses an addressing scheme called port number.
- Each process has a specific port number.

Transport Layer Design Issues (4)

- **Data Transfer:**
- Transport layer breaks user data into smaller units and attaches a transport layer header to each unit forming a TPDU (TransPort Layer Data Unit).
- The TPDU is handed over to the network layer for its delivery to destination.
- The TPDU header contains port number, sequence number, acknowledgement number, checksum and other fields.

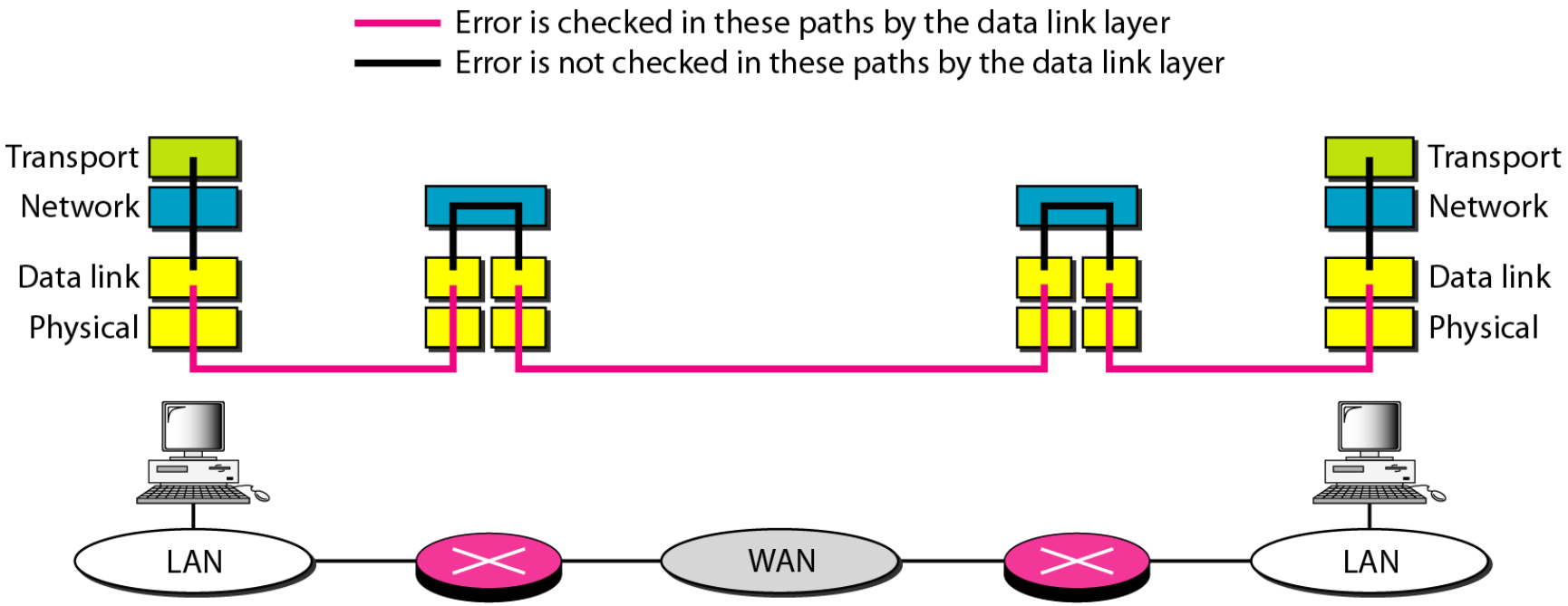
Transport Layer Design Issues (5)

- **Flow Control:**
- Like data link layer, transport layer also performs flow control.
- However, flow control at transport layer is performed end-to-end rather than node-to-node.
- Transport Layer uses a sliding window protocol to perform flow control.

Transport Layer Design Issues (6)

- **Error Control:**
- Transport layer also provides end-to-end error control facility.
- Transport layer deals with several different types of errors:
  - Error due to damaged bits.
  - Error due to non delivery of TPDUs.
  - Error due to duplicate delivery of TPDUs.
  - Error due to delivery of TPDU to a wrong destination.

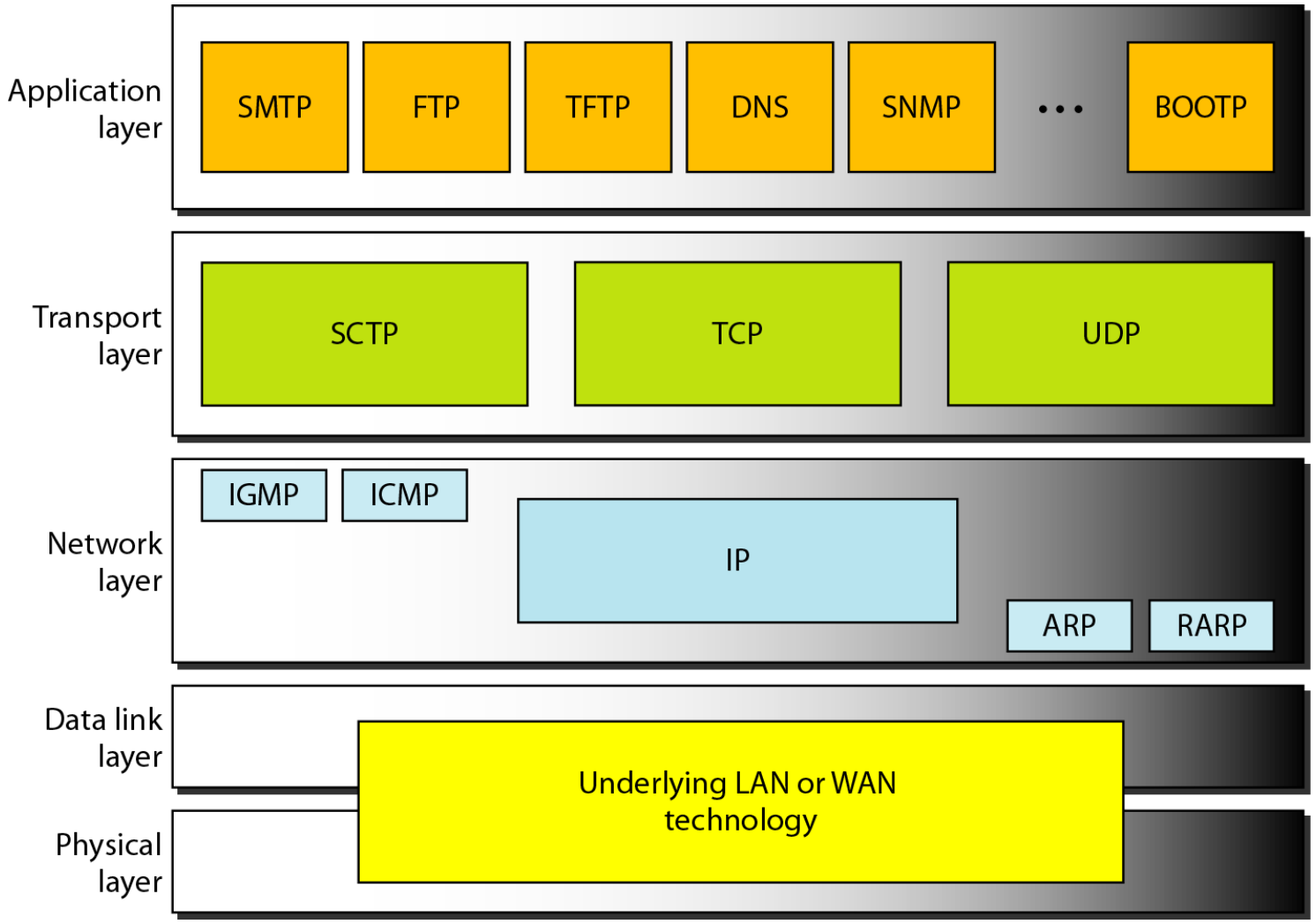
Error control



Transport Layer Design Issues (7)

- **Congestion Control:**
- Transport layer also handles congestion in the networks.
- Several different congestion control algorithms are used to avoid congestion.

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



Transport Layer Services

- Transport layer protocols can provide two types of services:
  - **Connection Oriented Service**
  - **Connectionless Service**

Transport Layer Services (2)

- **Connection Oriented Service**:
  - —In connection oriented service, a connection is first established between sender and the receiver.
  - —Then, transfer of user data takes place.
  - —At the end, connection is released.
  - —The connection oriented service is generally reliable.
  - —Transport layer protocols that provide connection oriented service are **TCP** and **SCTP**(Stream Control Transmission Protocol).

Transport Layer Services (3)

- **Connectionless Service**:
  - In the service, the packets are sent from sender to receiver without the establishment of connection.
  - In such service, packets are not numbered.
  - The packets may be lost, corrupted, delayed or disordered.
  - Connectionless service is unreliable.
  - Transport layer protocol that provides this service is **UDP.**

Elements of Transport Protocols

- **Addressing**:
  - In order to deliver data from one process to another, address is required.
  - In order to deliver data from one node to another, MAC address is required.
  - Such an address is implemented at Data Link Layer and is called Physical Addressing.

- **Addressing (Cont.):**
  - In order to deliver data from one network to another, IP address is required.
  - Such an address is implemented at Network Layer and is called Logical Addressing.
  - Similarly, in order to deliver data from a process running on source to process running on destination, transport layer defines the Service Point Addressor Port Numbers.

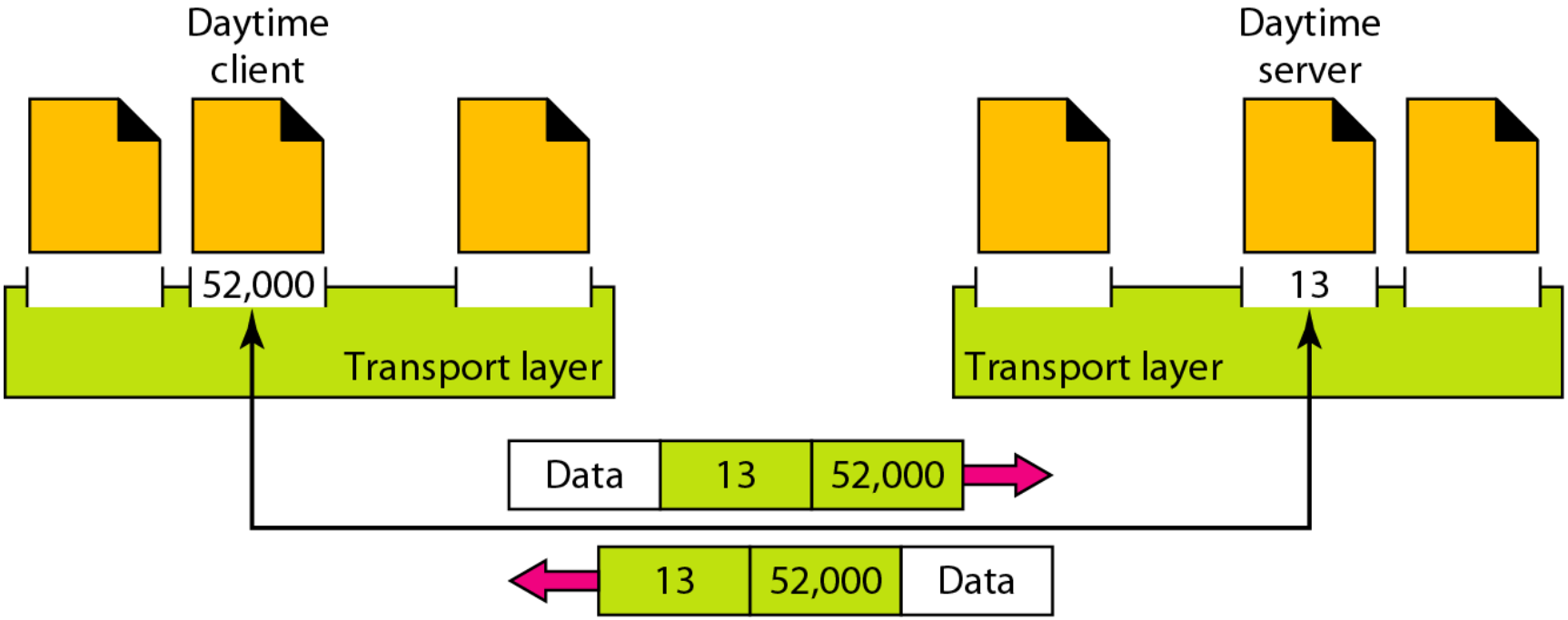
Elements of Transport Protocols (3)

- **Port Numbers**:
  - —Each communicating process is assigned a specific port number.
  - —In order to select among multiple processes running on a destination host, a port number is required.
  - —The port numbers are 16-bit integers between 0 and 65,535.

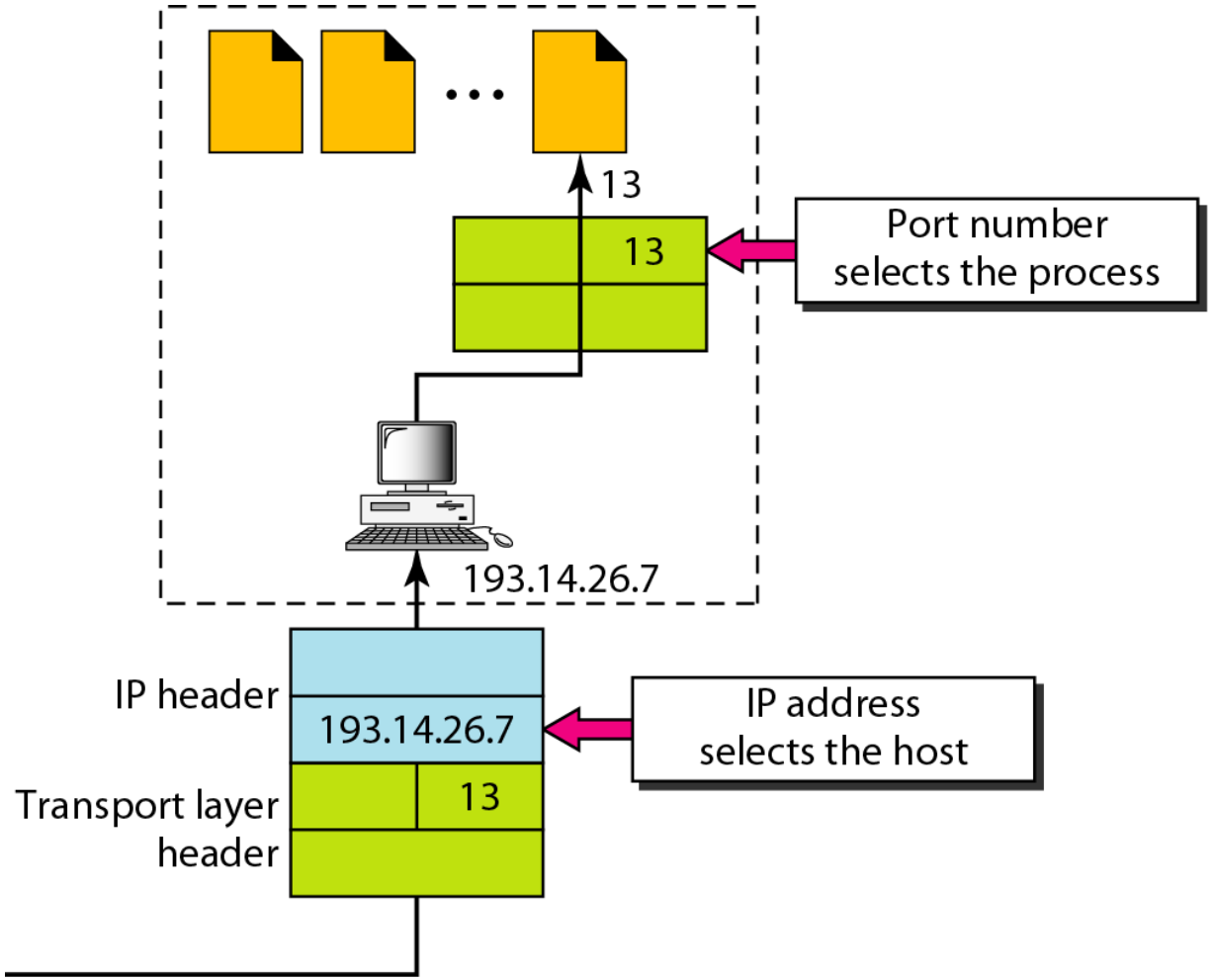
Elements of Transport Protocols (4)

- **Port Numbers (Cont.):**
  - Port numbers are assigned by Internet Assigned Number Authority (IANA).
  - IANA has divided the port numbers in three categories:
    - Well Known Ports: The ports ranging from 0 to 1023. For e.g.: HTTP: 80, SMTP: 25, FTP: 21.
    - Registered Ports: The ports ranging from 1024 to 49,151. These are not controlled by IANA.
    - Dynamic Ports: The ports ranging from 49,152 to 65,535. These can be used by any process.

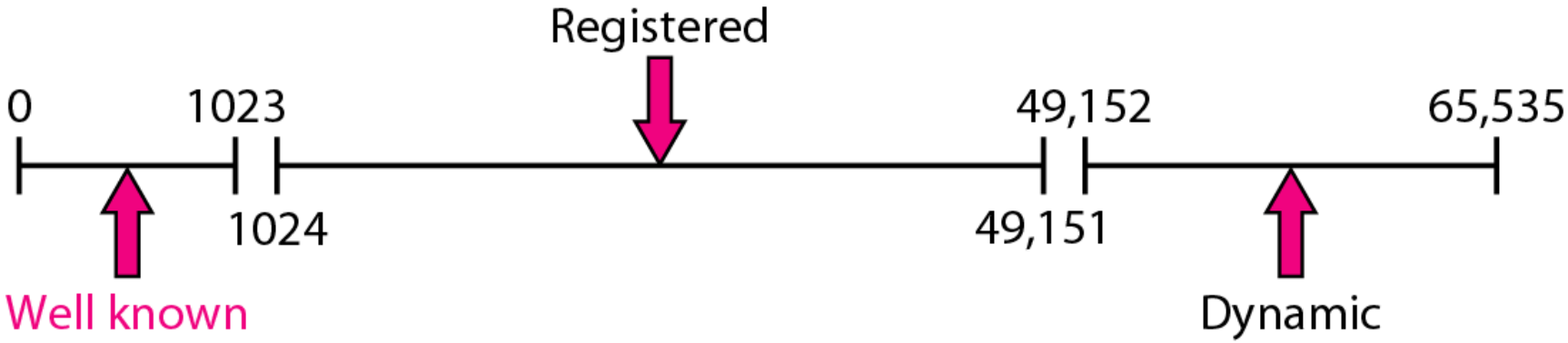
Port numbers



IP addresses versus port numbers



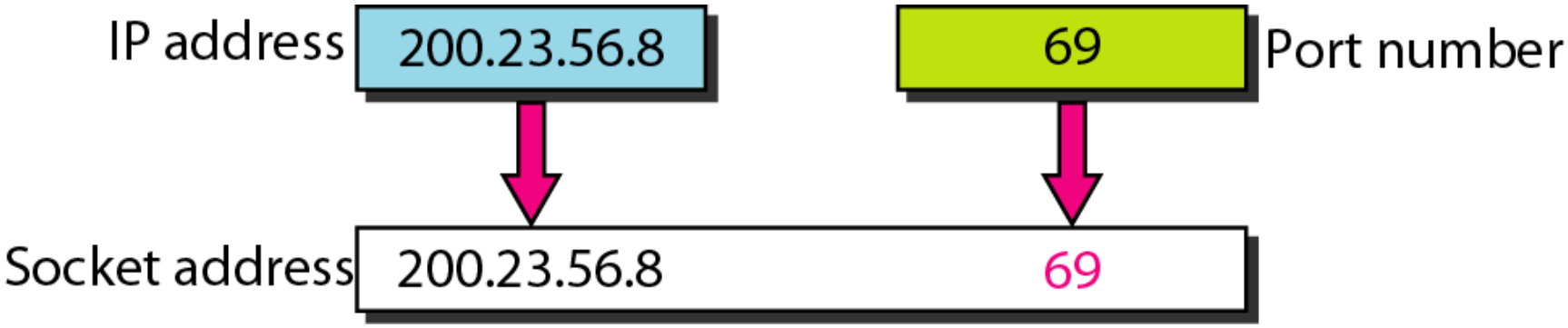
IANA ranges


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

Elements of Transport Protocols (5)

- **Socket Address**:
  - Socket address is a combination of IP address and port number.
  - In order to provide communication between two different processes on different networks, both IP address and port number, i.e. socket address is required.

Socket address



Socket

- Sockets are a service provided by transport layer.
- A socket is one endpoint of a two-way communication link between two programs running on the network.

Berkeley Socket

- Berkeley sockets is an application programming interface (API) for Internet sockets and UNIX domain sockets.
- It is used for inter-process communication (IPC).
- It is commonly implemented as a library of linkable modules.
- It originated with the 4.2BSD UNIX released in 1983.

Primitive used in Berkeley Socket

Primitives	Meaning
SOCKET	Create a New Communication Endpoint.
BIND	Attach a Local Address to a SOCKET.
LISTEN	Shows the Willingness to Accept Connections.
ACCEPT	Block the Caller until a Connection Attempts Arrives.
CONNECT	Actively Attempt to Establish a Connection.
SEND	Send Some Data over Connection.
RECEIVE	Receive Some Data from the Connection.
CLOSE	Release the Connection.

Socket Programming

- **Server side:**
  - —Server startup executes SOCKET, BIND & LISTEN primitives.
  - —LISTEN primitive allocate queue for multiple simultaneous clients.
  - —Then it use ACCEPT to suspend server until request.
  - —When client request arrives: ACCEPT returns.
  - —Start new socket (thread or process) with same properties as original, this handles the request, server goes on waiting on original socket.
  - —If new request arrives while spawning thread for this one, it is queued.
  - —If queue full it is refused.

Socket Programming (2)

- **Client side**:
  - —It uses SOCKET primitives to create.
  - —Then use CONNECT to initiate connection process.
  - —When this returns the socket is open.
  - —Both sides can now SEND, RECEIVE.
  - —Connection not released until both sides do CLOSE.
  - —Typically client does it, server acknowledge

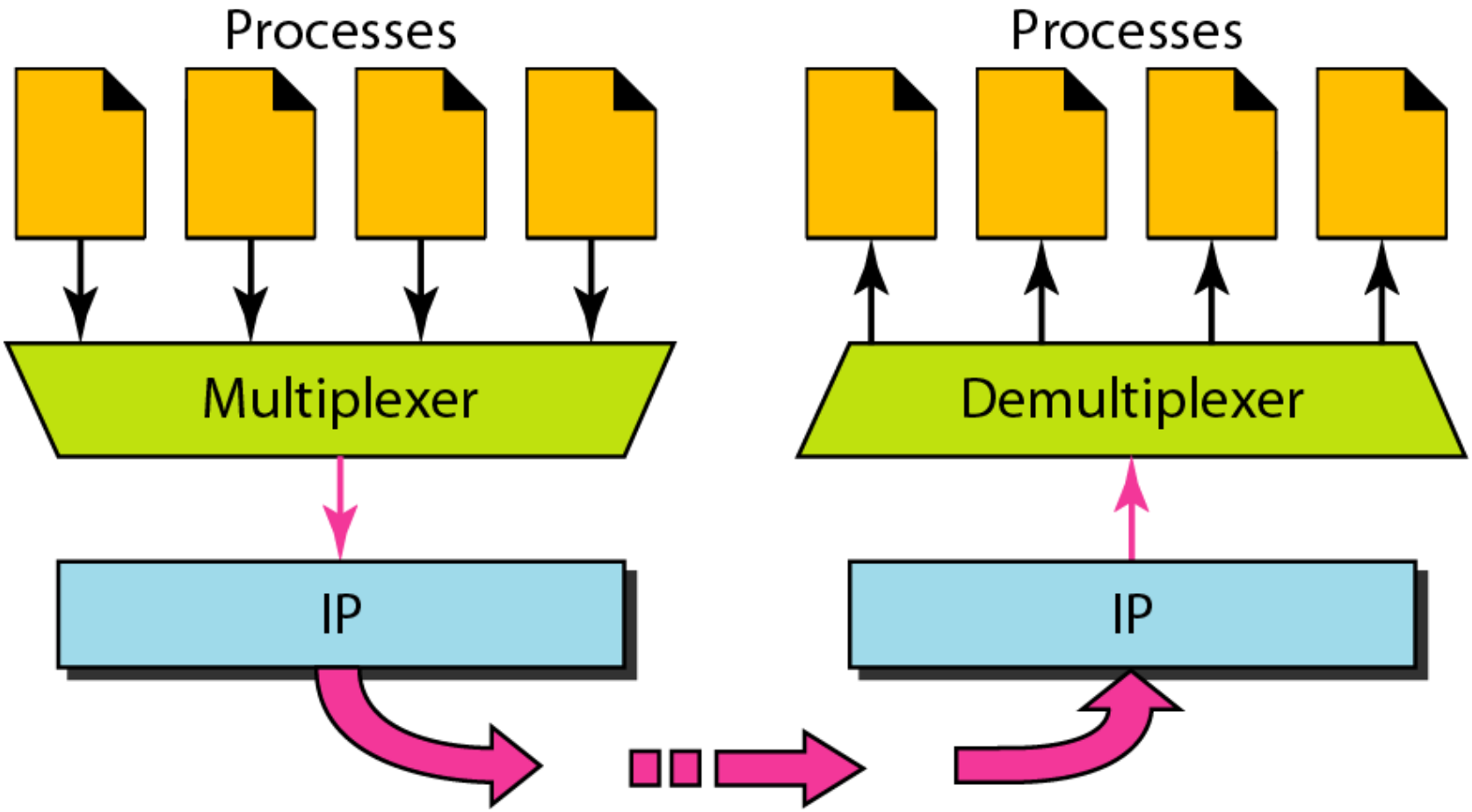
Elements of Transport Protocols (6)

- **Multiplexing & Demultiplexing**:
  - A network connection can be shared by various applications running on a system.
  - There may be several running processes that want to send data and only one transport layer connection available, then transport layer protocols may perform multiplexing.
  - The protocol accepts the messages from different processes having their respective port numbers, and add headers to them.

Elements of Transport Protocols (7)

- **Multiplexing & Demultiplexing (Cont.):**
- The transport layer at the receiver end performs demultiplexing to separate the messages for different processes.
- After checking for errors, the headers of messages are dropped and each message is handed over to the respective processes based on their port numbers.

Multiplexing and demultiplexing



Elements of Transport Protocols (8)

- **Connection Establishment**:
- Before communicating, the source device must first determine the availability of the other to exchange data.
- Path must be found through the network by which the data can be sent.
- This is called Connection Establishment

Elements of Transport Protocols (9)

- **Connection Establishment (Cont.)**:
  - Connection establishment involves **Three-Way Handshaking** mechanism:
  - The source sends a connection request packet to the destination.
  - The destination returns a confirmation packet back to the source.
  - The source returns a packet acknowledging the confirmation.

Elements of Transport Protocols (10)

- **Connection Release:**
  - Once all of the data has been transferred, the connection must be released.
  - It also requires a **Three-Way Handshaking mechanism**:
  - The source sends a disconnect request packet to the destination.
  - The destination returns a confirmation packet back to the source.
  - The source returns a packet acknowledging the confirmation.

Transport Layer Protocols

- Transport layer provides two types of services:
  - Connection Oriented Service
  - Connectionless Service
- For this, transport layer defines two different protocols:
  - **Transmission Control Protocol (TCP)**
  - **User Datagram Protocol (UDP)**

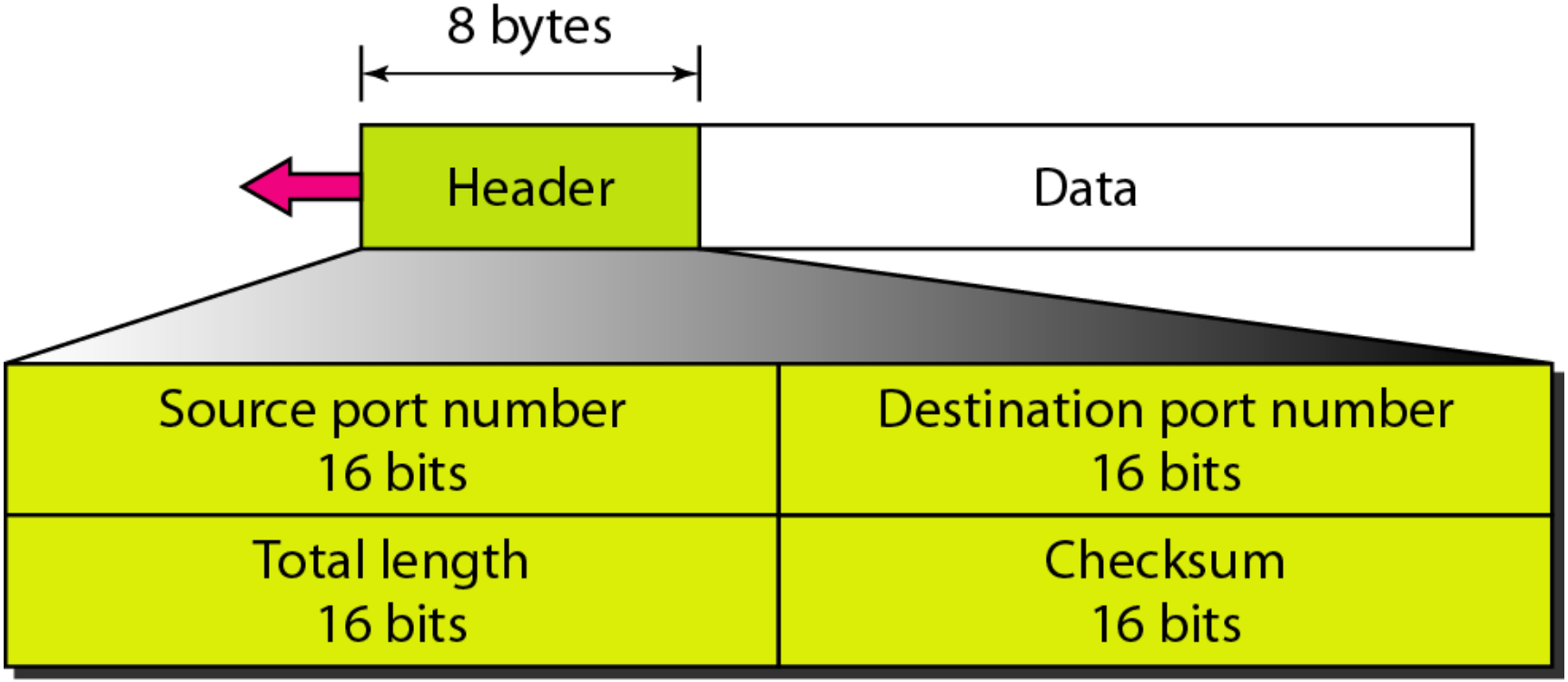
User Datagram Protocol (UDP)

- UDP is a simple protocol and it provides nonsequenced transport functionality.
- UDP is a connectionless protocol.
- This type of protocol is used when reliability and security are less important than speed and size.
- UDP is an end-to-end transport level protocol that adds transport-level addresses, checksum error control, and length information to the data from the upper layer.
- The packet produced by the UDP protocol is known as a **user datagram**.

Well-known ports used with UDP

Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
53	Nameserver	Domain Name Service
67	BOOTPs	Server port to download bootstrap information
68	BOOTPc	Client port to download bootstrap information
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)

User Datagram Format

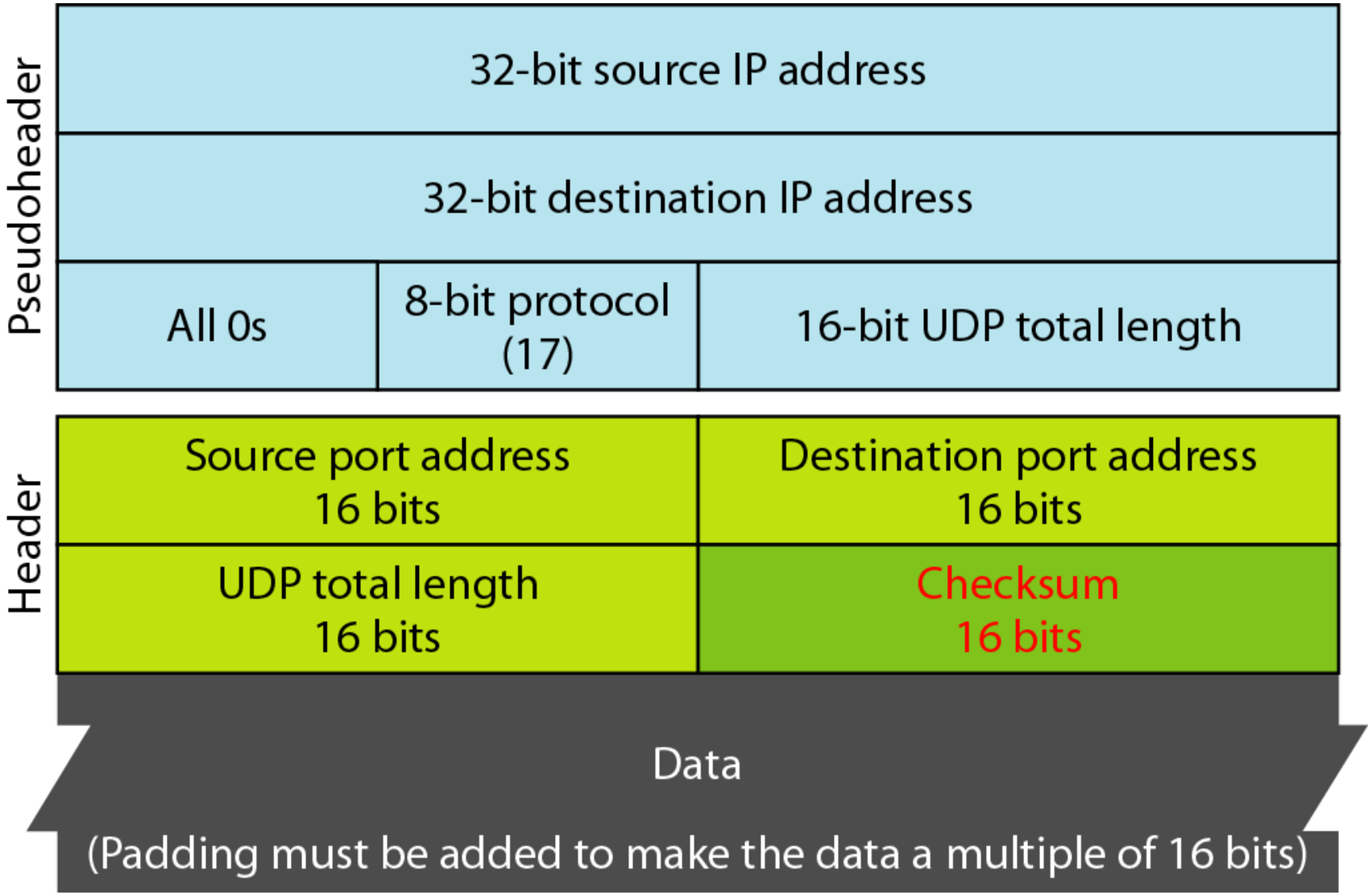


UDP length
= IP length – IP header's length

User Datagram Format (2)

- UDP header is **8-bytes** fixed and simple header.
- **Source Port** : Source Port is 2 Byte long field used to identify port number of source.
- **Destination Port** : It is 2 Byte long field, used to identify the port of destined packet.
- **Length** : Length is the length of UDP including header and the data. It is 16-bits field.
- **Checksum** : Checksum is 2 Bytes long field. It is the 16-bit one's complement of the one's complement sum of the UDP header, pseudo header of information from the IP header and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

Pseudoheader for checksum calculation



Applications of UDP

- Used for simple request response communication when size of data is less and hence there is lesser concern about flow and error control.
- It is suitable protocol for multicasting as UDP supports packet switching.
- UDP is used for some routing update protocols like RIP(Routing Information Protocol).
- Normally used for real time applications which can not tolerate uneven delays between sections of a received message.

Applications of UDP (2)

- Application layer can do some of the tasks through UDP-
  - Trace Route
  - Record Route
  - Time stamp
- UDP takes datagram from Network Layer, attach its header and send it to the user. So, it works fast.
- Actually UDP is null protocol if you remove checksum field.
- Unlike TCP, **Checksum calculation is not mandatory in UDP**. No Error control or flow control is provided by UDP. Hence UDP depends on IP and ICMP for error reporting.

Transmission Control Protocol (TCP)

- It provides full transport layer services to applications.
- It is a connection-oriented protocol means the connection established between both the ends of the transmission.
- For creating the connection, TCP generates a virtual circuit between sender and receiver for the duration of a transmission.

Features Of TCP protocol

- **Stream data transfer**: TCP protocol transfers the data in the form of contiguous stream of bytes.
- **Reliability**: TCP assigns a sequence number to each byte transmitted and expects a positive acknowledgement from the receiving TCP.
- **Flow Control**: When receiving TCP sends an acknowledgement back to the sender indicating the number the bytes it can receive without overflowing its internal buffer.
- **Full Duplex**: TCP provides Full Duplex service, i.e., the data flow in both the directions at the same time. To achieve Full Duplex service, each TCP should have sending and receiving buffers so that the segments can flow in both the directions.

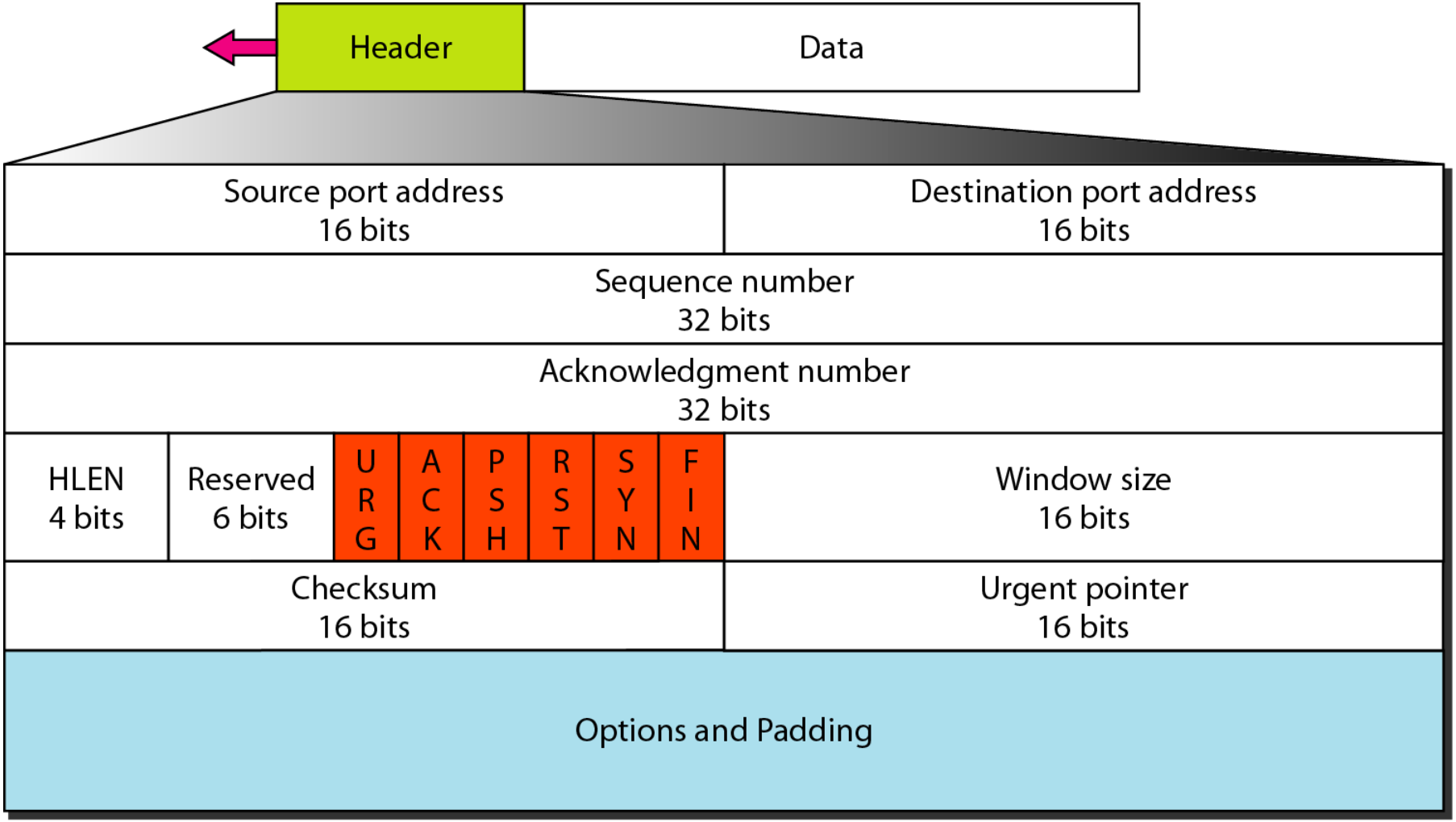
Features Of TCP protocol (2)

- **Multiplexing**:
  - —Multiplexing is a process of accepting the data from different applications and forwarding to the different applications on different computers.
  - —At the receiving end, the data is forwarded to the correct application. This process is known as demultiplexing.
- **Logical Connections**: The combination of sockets, sequence numbers, and window sizes, is called a logical connection. Each connection is identified by the pair of sockets used by sending and receiving processes.

Well-known ports used by TCP

Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FTP, Data	File Transfer Protocol (data connection)
21	FTP, Control	File Transfer Protocol (control connection)
23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

TCP segment format



TCP segment format (2)

- **Source port address**: It is used to define the address of the application program in a source computer. It is a 16-bit field.
- **Destination port address**: It is used to define the address of the application program in a destination computer. It is a 16-bit field.
- **Sequence number**: A stream of data is divided into two or more TCP segments. The 32-bit sequence number field represents the position of the data in an original data stream.

TCP segment format (3)

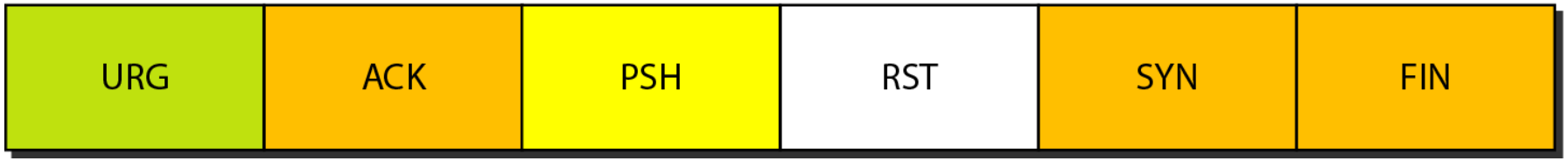
- **Acknowledgement number**: A 32-field acknowledgement number acknowledge the data from other communicating devices. If ACK field is set to 1, then it specifies the sequence number that the receiver is expecting to receive.
- **Header Length (HLEN):** It specifies the size of the TCP header in 32-bit words. The minimum size of the header is 5 words, and the maximum size of the header is 15 words. Therefore, the maximum size of the TCP header is 60 bytes, and the minimum size of the TCP header is 20 bytes.
- **Reserved**: It is a six-bit field which is reserved for future use.

Control field

- **Control bits**: Each bit of a control field functions individually and independently. A control bit defines the use of a segment or serves as a validity check for other fields.

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



Description of flags in the control field

- **URG**: The URG field indicates that the data in a segment is urgent.
- **ACK**: When ACK field is set, then it validates the acknowledgement number.
- **PSH**: The PSH field is used to inform the sender that higher throughput is needed so if possible, data must be pushed with higher throughput.
- **RST**: The reset bit is used to reset the TCP connection when there is any confusion occurs in the sequence numbers.

Description of flags in the control field (2)

- **SYN**: The SYN field is used to synchronize the sequence numbers in three types of segments: connection request, connection confirmation (with the ACK bit set), and confirmation acknowledgement.
- **FIN**: The FIN field is used to inform the receiving TCP module that the sender has finished sending data. It is used in connection termination in three types of segments: termination request, termination confirmation, and acknowledgement of termination confirmation.

TCP segment format (4)

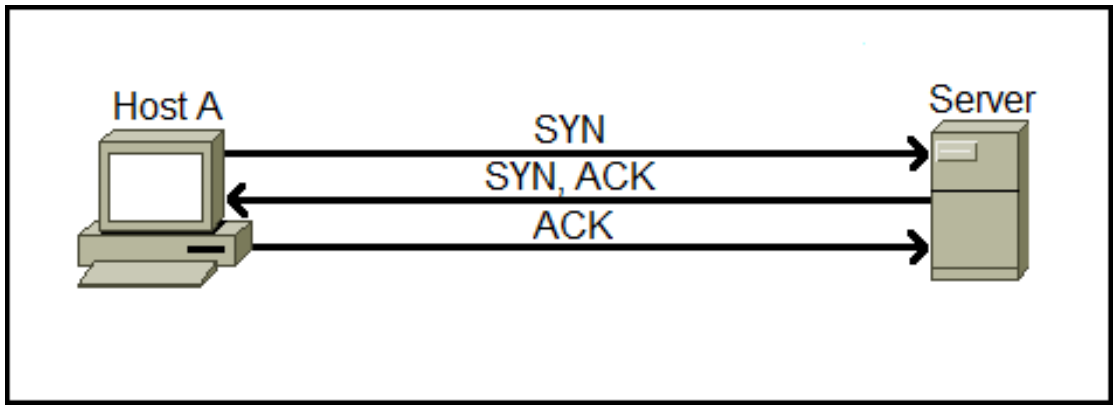
- **Window Size**: The window is a 16-bit field that defines the size of the window.
- **Checksum**: The checksum is a 16-bit field used in error detection.
- **Urgent pointer**: If URG flag is set to 1, then this 16-bit field is an offset from the sequence number indicating that it is a last urgent data byte.
- **Options and padding**: It defines the optional fields that convey the additional information to the receiver.

Differences b/w TCP & UDP

Basis for Comparison	TCP	UDP
Definition	TCP establishes a virtual circuit before transmitting the data.	UDP transmits the data directly to the destination computer without verifying whether the receiver is ready to receive or not.
Connection Type	It is a Connection-Oriented protocol	It is a Connectionless protocol
Speed	slow	high
Reliability	It is a reliable protocol.	It is an unreliable protocol.
Header size	20 bytes	8 bytes
acknowledgement	It waits for the acknowledgement of data and has the ability to resend the lost packets.	It neither takes the acknowledgement, nor it retransmits the damaged frame.

TCP three-way handshake

- Since TCP is a connection-oriented protocol, a connection needs to be established before two devices can communicate.
- TCP uses a process called three-way handshake to negotiate the sequence and acknowledgment fields and start the session. Here is a graphical representation of the process:

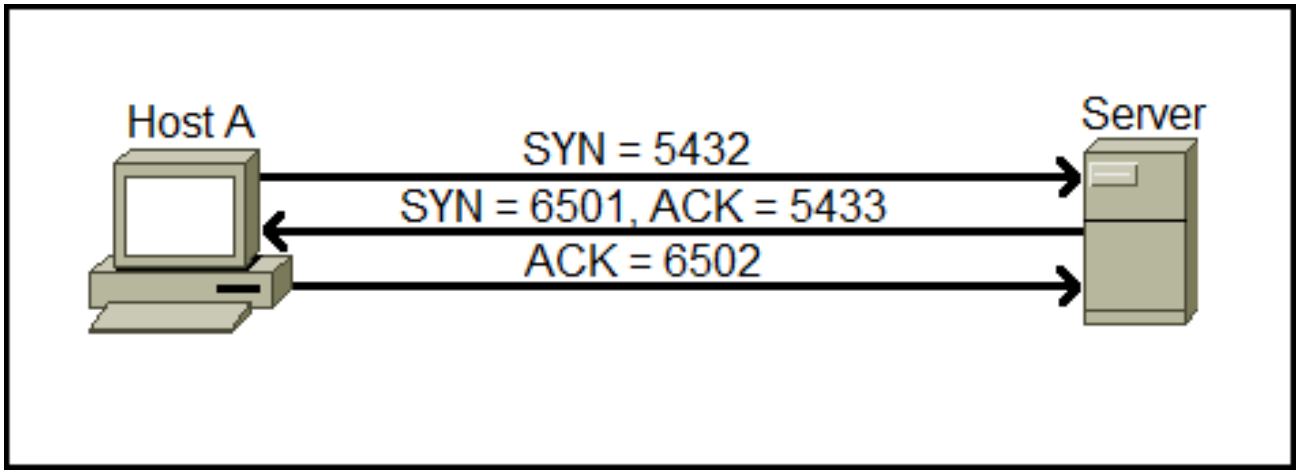


TCP three-way handshake

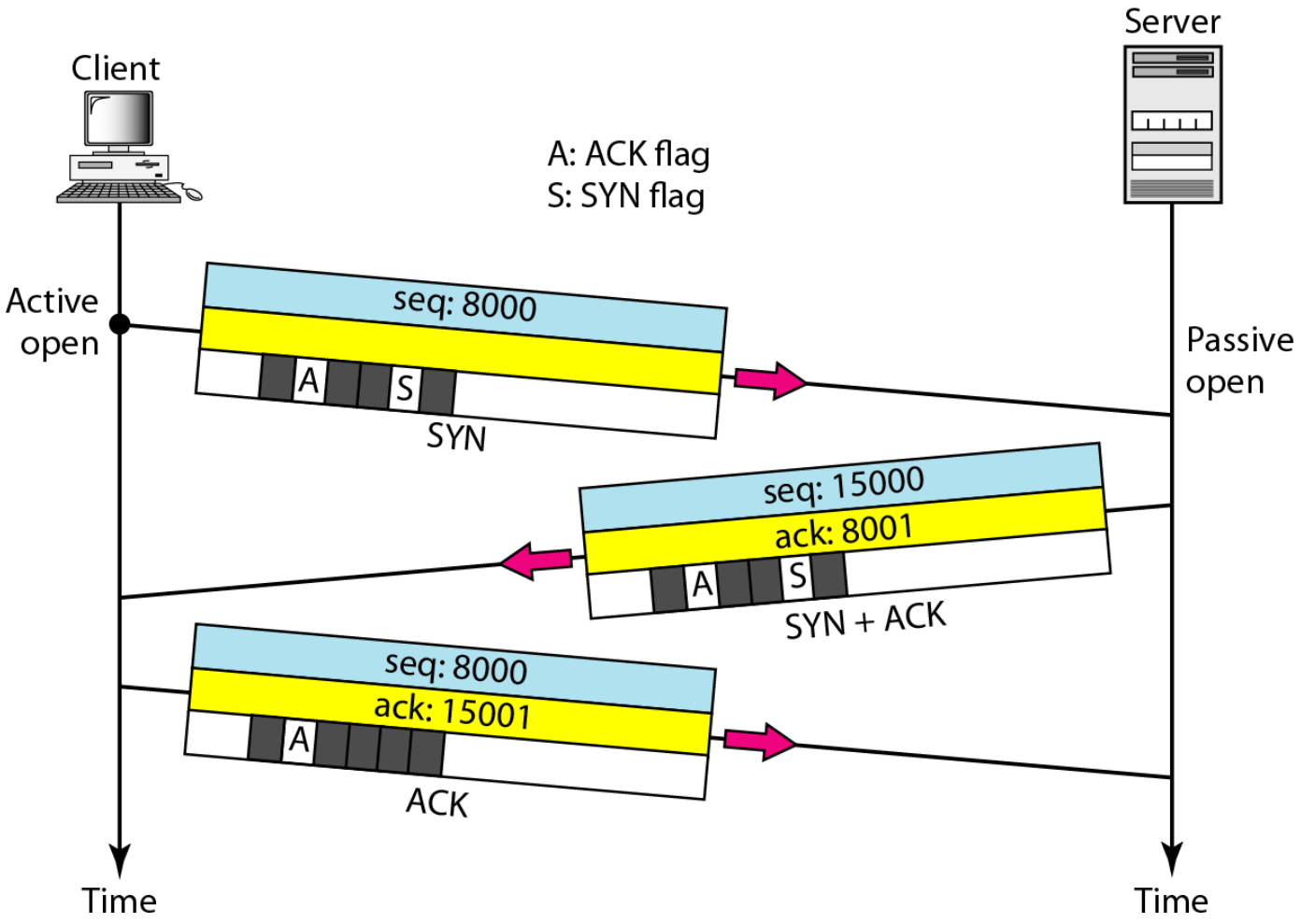
- As the name implies, the three way handshake process consists of three steps:
- Host A initiates the connection by sending the TCP SYN packet to the destination host. The packet contains the random sequence number (e.g. 5432) which marks the beginning of the sequence numbers for data that the Host A will transmit.
- The Server receives the packet and responds with its own sequence number. The response also includes the acknowledgment number, which is Host A's sequence number incremented by 1 (in our case, that would be 5433).
-

TCP three-way handshake

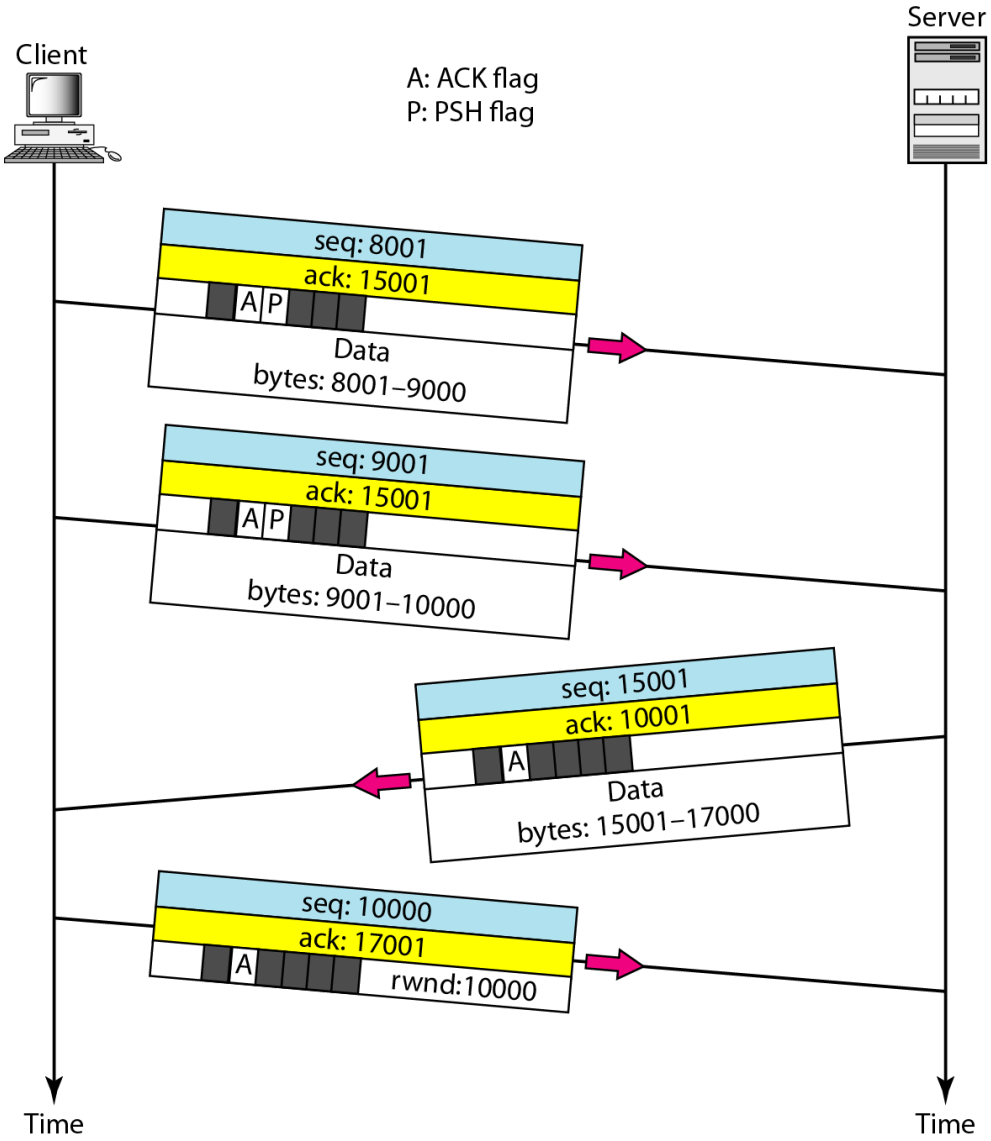
- Host A acknowledges the response of the Server by sending the acknowledgment number, which is the Server's sequence number incremented by 1.



Connection establishment using three-way handshaking

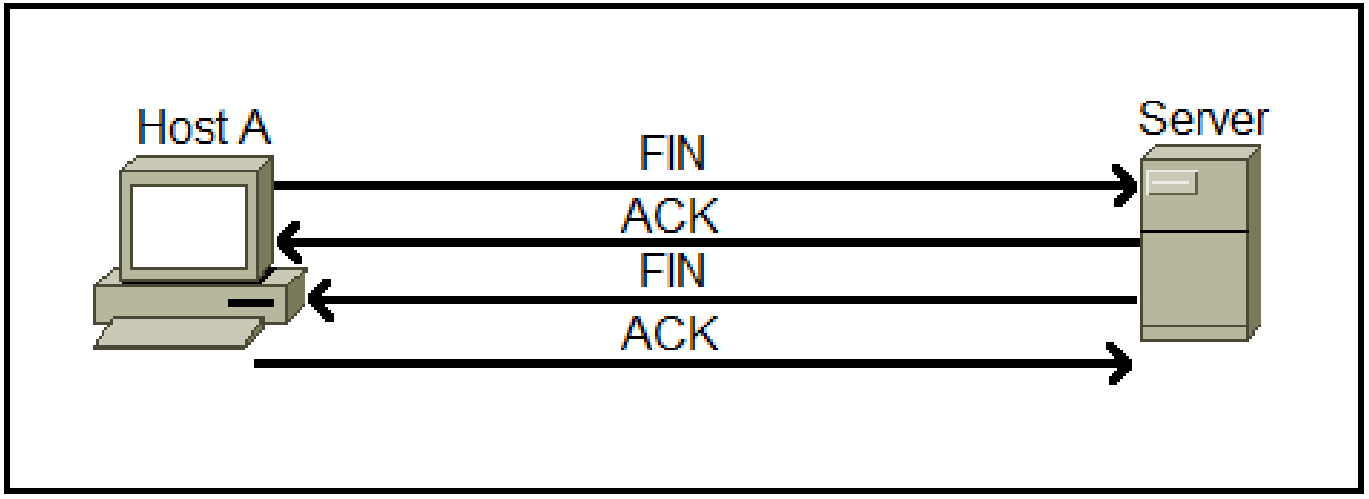


Data transfer



TCP three-way handshake

- After the data transmission process is finished, TCP will terminate the connection between two endpoints.
- This four-step process is illustrated below:



TCP three-way handshake

- The client application that wants to close the connection sends a TCP segment with the FIN (Finished) flag set to 1.
- The server receives the TCP segment and acknowledges it with the ACK segment.
- Server sends its own TCP segment with the FIN flag set to 1 to the client in order to terminate the connection.
- The client acknowledges the server's FIN segment and closes the connection.

Connection termination using three-way handshaking