COMPUTER NETWORK (BCA301)

DEPARTMENT OF COMPUTER SCIENCE

PROGRAMME: BCA

CENTRAL UNIVERSITY OF ORISSA
KORAPUT

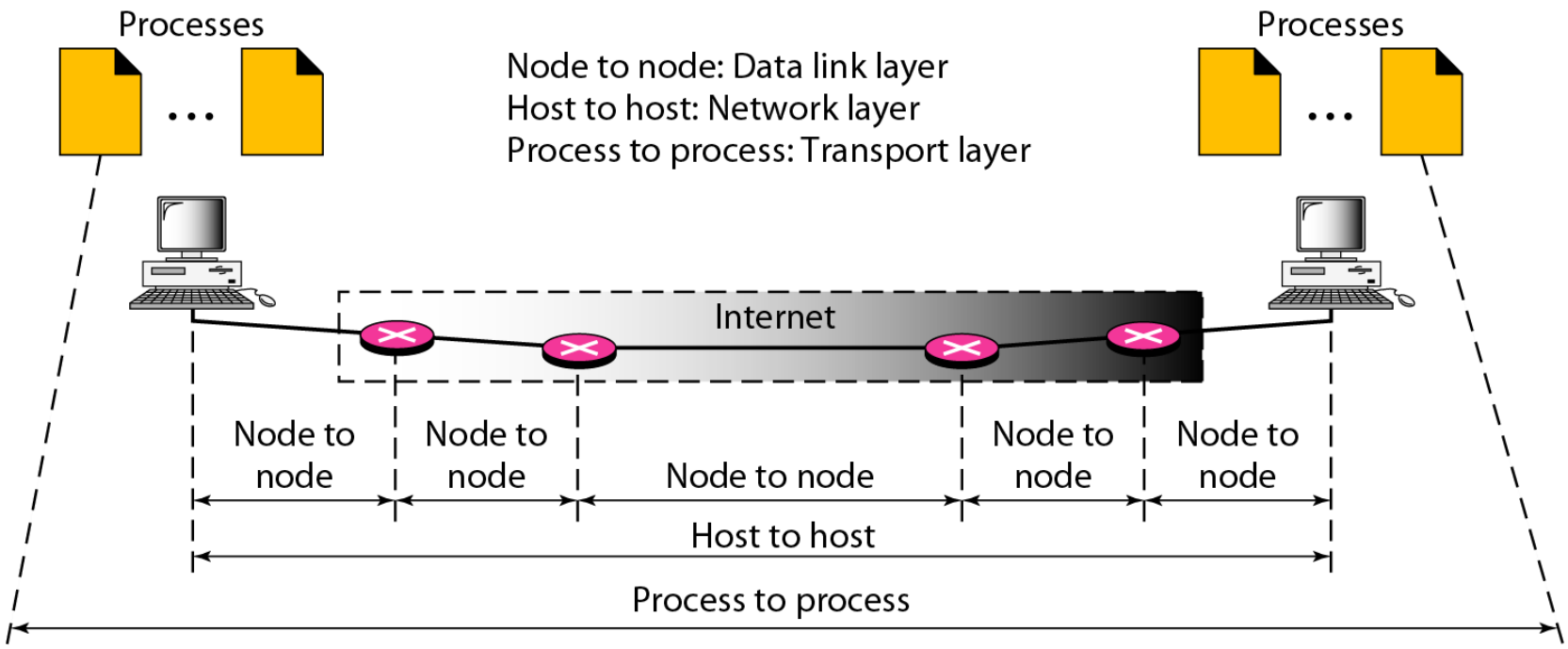
TRANSPORT LAYER

- The network layer oversees source-to-destination delivery of individual packets, it does not recognize any relationship between those packets.
- It treats each one independently, as though each piece belonged to a separate message, whether or not it does.
- The transport layer is responsible for process-to-process delivery of the entire message.
- It ensures that the whole message arrives intact and in order, overseeing both error control and flow control at the source-to-destination level.
- Source-to-destination delivery means delivery not only from one computer to the next but also from a specific process on one computer to a specific process on the other.
- The transport layer header must therefore include a type of address called a service-point address in the OSI model and port number or port addresses in the Internet and TCP/IP protocol suite.

- In the transport layer, a message is normally divided into transmittable segments.
- A connectionless protocol, such as UDP, treats each segment separately.
- A connection-oriented protocol, such as TCP and SCTP(Stream Control Transmission Protocol), creates a relationship between the segments using sequence numbers.
- Like the data link layer, the transport layer may be responsible for flow and error control.
- However, flow and error control at this layer is performed end to end rather than across a single link.

NODE-TO-NODE Vs END-TO-END Vs PROCESS -TO-PROCESS

- **Node-to-node delivery :** The data link layer is responsible for delivery of frames between two neighboring nodes over a link.
- **Host-to-host delivery :** The network layer is responsible for delivery of datagrams between two hosts.
- **Process-to-process delivery:** Communication on the Internet is not defined as the exchange of data between two nodes or between two hosts. Real communication takes place between two processes (application programs).
- At any moment, several processes may be running on the source host and several on the destination host.
- To complete the delivery, we need a mechanism to deliver data from one of these processes running on the source host to the corresponding process running on the destination host.
- Two processes communicate in a client/server relationship


Processes
Processes
Node to node: Data link layer
Host to host: Network layer
Process to process: Transport layer
Internet
Node to node
Node to node
Node to node
Node to node
Node to node
Host to host
Process to process

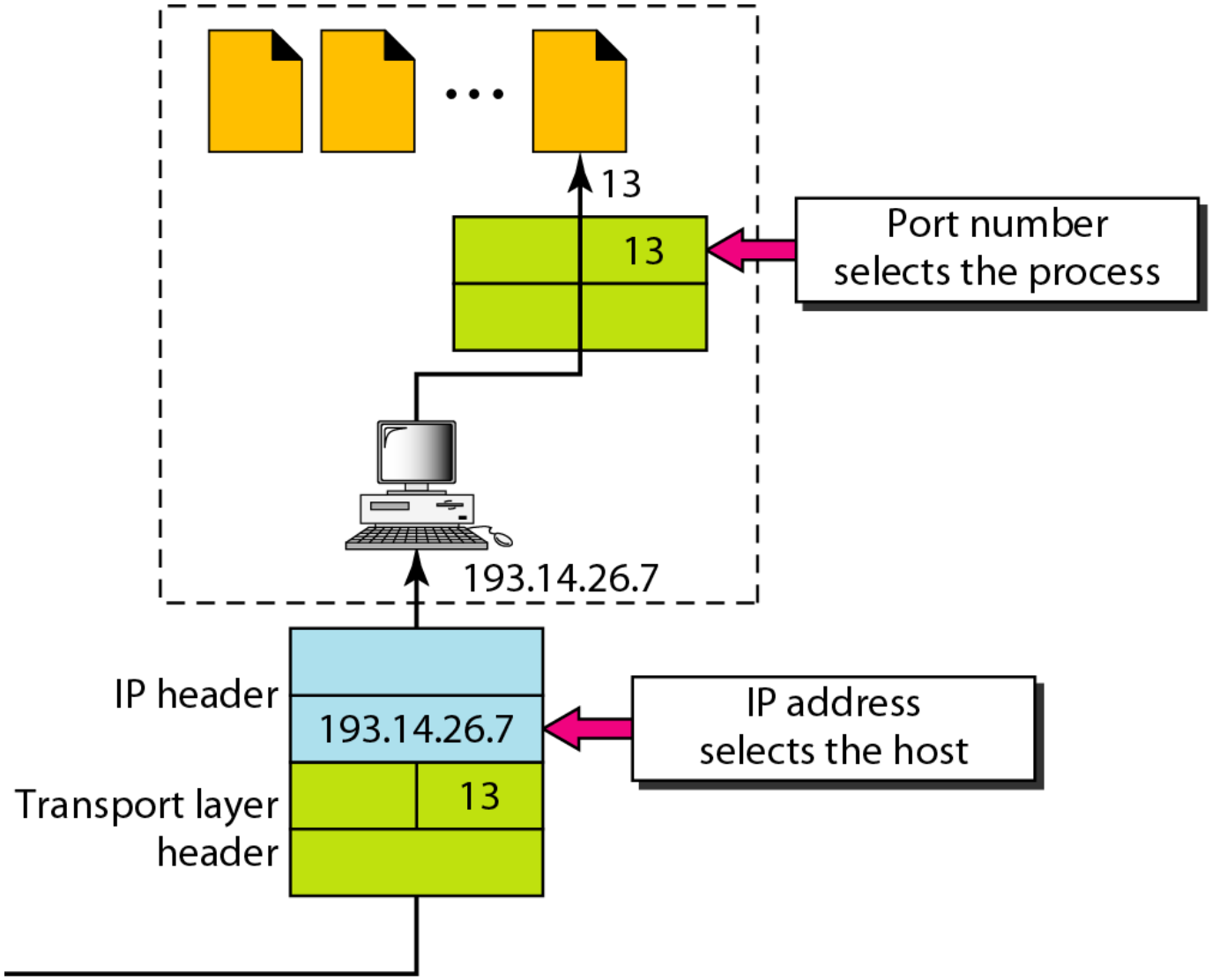
Client/Server Paradigm

- There are several ways to achieve process-to-process communication, the most common one is through the client/server paradigm.
- A process on the local host, called a client, needs services from a process usually on the remote host, called a server.
- Both processes (client and server) have the same name.
- Operating systems today support both multiuser and multiprogramming environments.
- For communication, we must define the following: Local host, Local Process, Remote host & Remote process.

Addressing

- At the data link layer, we need a MAC address to choose one node among several nodes if the connection is not point-to-point.
- A frame in the data link layer needs a destination MAC address for delivery and a source address for the next node's reply.
- At the network layer, we need an IP address to choose one host among millions.
- A datagram in the network layer needs a destination IP address for delivery and a source IP address for the destination's reply.
- At the transport layer, we need a transport layer address, called a port number, to choose among multiple processes running on the destination host.
- The destination port number is needed for delivery; the source port number is needed for the reply.

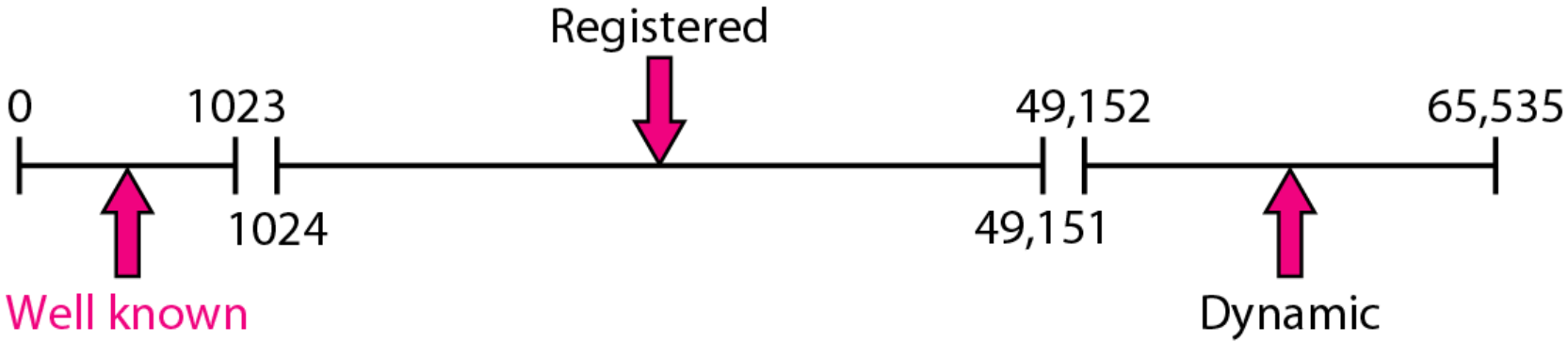
IP addresses versus port numbers



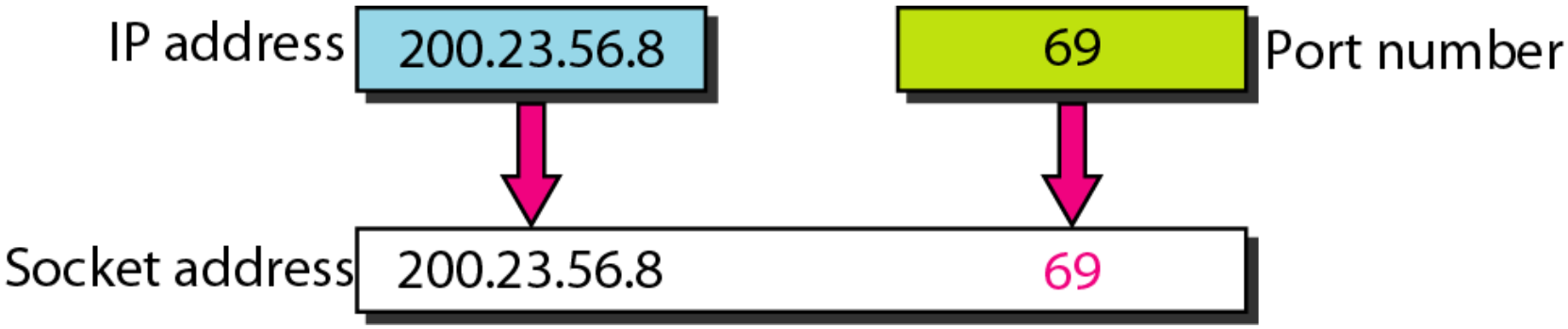
Port Number

- In the Internet model, the port numbers are **16-bit integers between 0 and 65,535.**
- The **IANA (Internet Assigned Number Authority)** has divided the port numbers into three ranges: well known, registered, and dynamic (or private)
- **Well-known ports:** The ports ranging from 0 to 1023 are assigned and controlled by IANA. These are the well-known ports. These are Universal port numbers for servers.
- **Registered ports:** The ports ranging from 1024 to 49,151 are not assigned or controlled by IANA. They can only be registered with IANA to prevent duplication.
- **Dynamic ports:** The ports ranging from 49,152 to 65,535 are neither controlled nor registered. They can be used by any process.
- The destination IP address defines the host among the different hosts in the world. After the host has been selected, the port number defines one of the processes on this particular host.

IANA ranges


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

Socket address



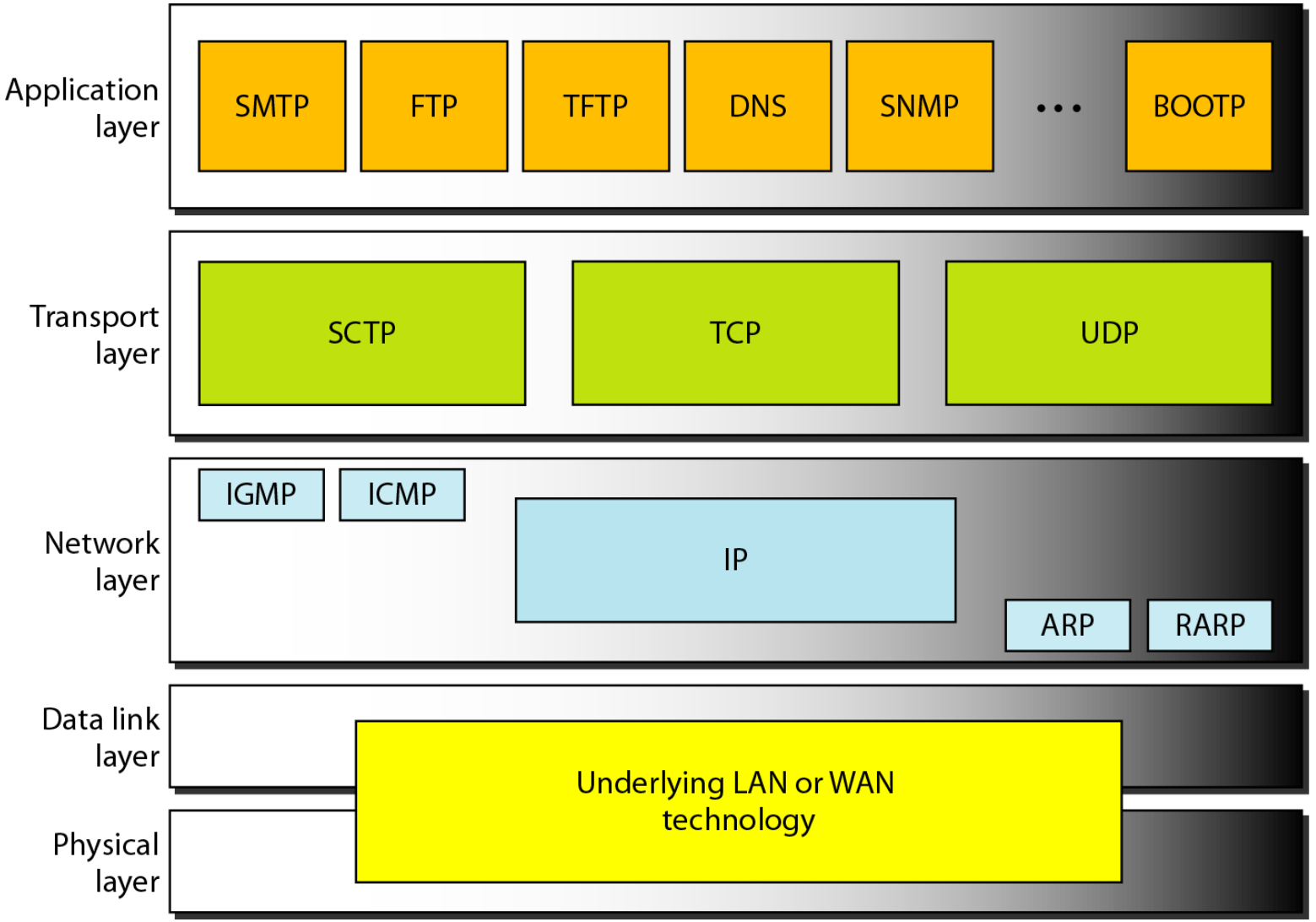
Socket Addresses

- Process-to-process delivery needs two identifiers, IP address and the port number, at each end to make a connection. The combination of an IP address and a port number is called a socket address.
- The client socket address defines the client process uniquely just as the server socket address defines the server process uniquely.
- A transport layer protocol needs a pair of socket addresses: the client socket address and the server socket address.
- These four pieces of information are part of the IP header and the transport layer protocol header. The IP header contains the IP addresses; the UDP or TCP header contains the port numbers.

Reliable Versus Unreliable

- The transport layer service can be reliable or unreliable. If the application layer program needs reliability, we use a reliable transport layer protocol by implementing flow and error control at the transport layer. This means a slower and more complex service.
- On the other hand, if the application program does not need reliability because it uses its own flow and error control mechanism or it needs fast service or the nature of the service does not demand flow and error control (real-time applications), then an unreliable protocol can be used.
- UDP is connectionless and unreliable; TCP and SCTP are connection-oriented and reliable.

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



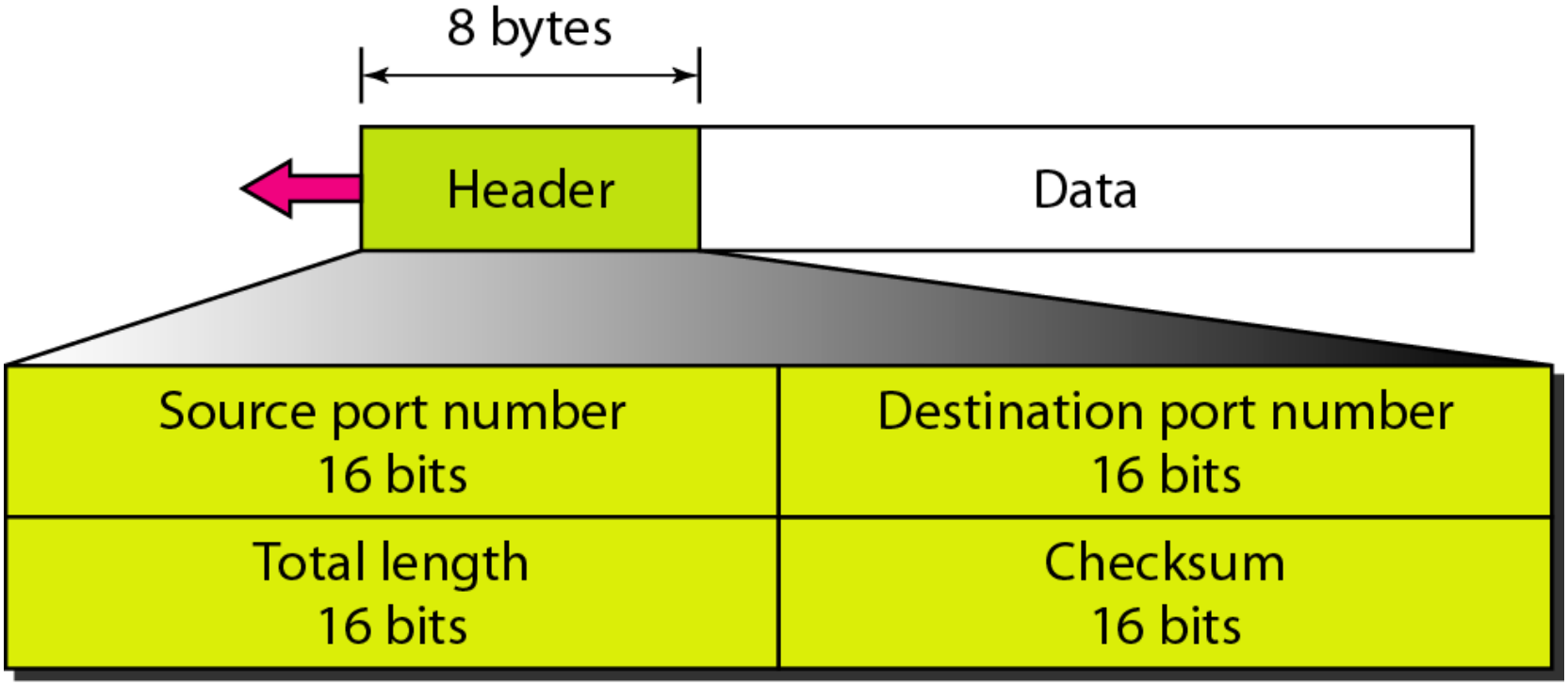
USER DATAGRAM PROTOCOL (UDP)

- The User Datagram Protocol (UDP) is called a connectionless, unreliable transport protocol.
- It does not add anything to the services of IP except to provide process-to-process communication instead of host-to-host communication. Also, it performs very limited error checking.
- Advantages: 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 or SCTP.

User Datagram

- UDP packets, called user datagrams, have a fixed-size header of 8 bytes.
- **Source port number:** This is the port number used by the process running on the source host. It is 16 bits long, which means that the port number can range from 0 to 65,535.
- **Destination port number:** This is the port number used by the process running on the destination host. It is also 16 bits long.
- **Length:** This is a 16-bit field that defines the total length of the user datagram, header plus data. The 16 bits can define a total length of 0 to 65,535 bytes.
- **Checksum:** This field is used to detect errors over the entire user datagram (header plus data).

User datagram format



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)

UDP Operation

- **Connectionless Services:** UDP provides a connectionless service. This means that 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.
- Also, there is no connection establishment and no connection termination, as is the case for TCP. This means that each user datagram can travel on a different path.
- The process that uses UDP cannot send a stream of data to UDP and expect UDP to chop them into different related user datagrams. Instead each request must be small enough to fit into one user datagram. Only those processes sending short messages should use UDP.

UDP Operation

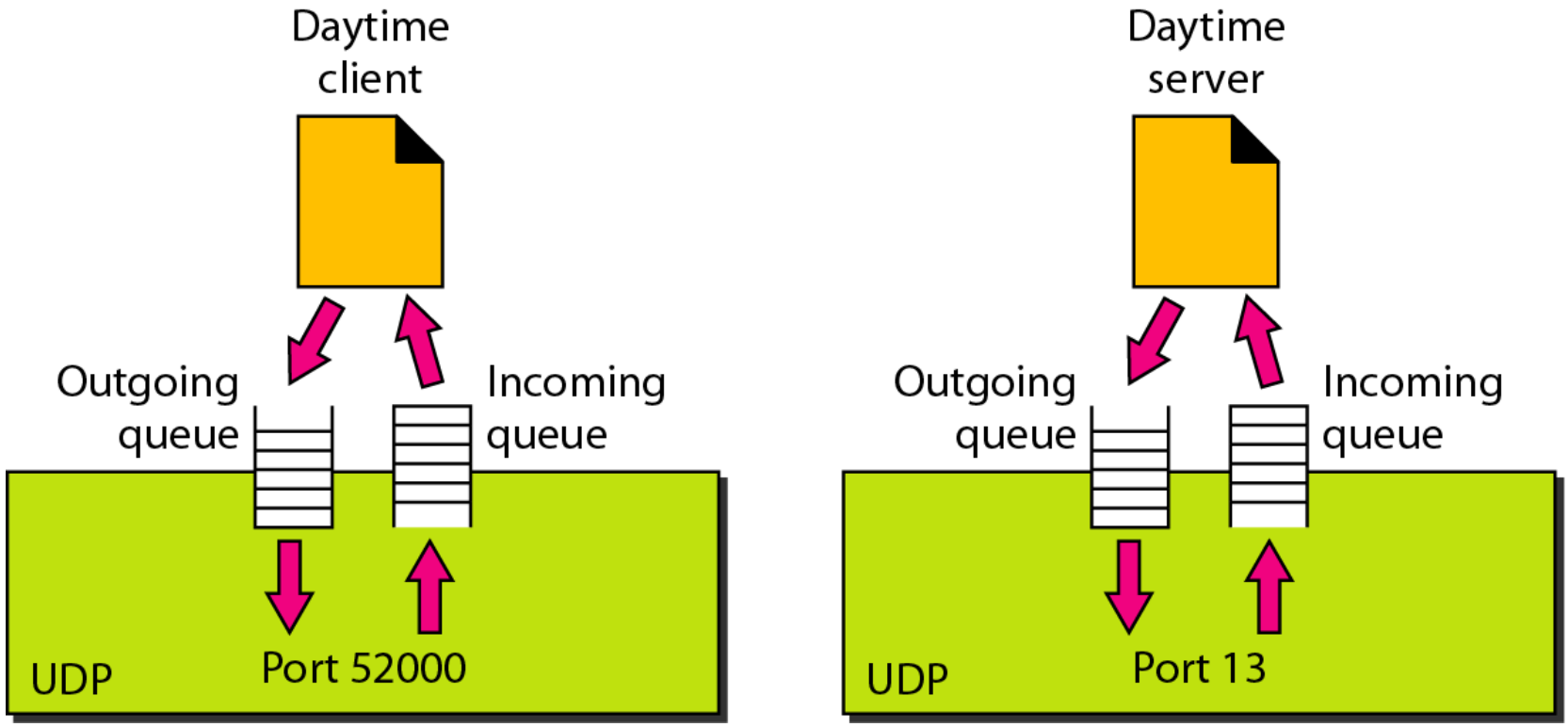
- **Flow and Error Control:** UDP is a very simple, unreliable transport protocol. There is no flow control and hence no window mechanism. The receiver may overflow with incoming messages.
- There is no error control mechanism in UDP except for the checksum. This means that the sender does not know if a message has been lost or duplicated. When the receiver detects an error through the checksum, the user datagram is silently discarded.
- The lack of flow control and error control means that the process using UDP should provide following mechanisms: Encapsulation and Decapsulation & Queuing.
- **Encapsulation and Decapsulation:** To send a message from one process to another, the UDP protocol encapsulates and decapsulates messages in an IP datagram.

Queuing

- At the client site, when a process starts, it requests a port number from the operating system.
- Even if a process wants to communicate with multiple processes, it obtains only one port number and eventually one outgoing and one incoming queue.
- The queues opened by the client are, in most cases, identified by ephemeral port numbers.
- The queues function as long as the process is running. When the process terminates, the queues are destroyed.
- The client process can send messages to the outgoing queue by using the source port number specified in the request. UDP removes the messages one by one and, after adding the UDP header, delivers them to IP.
- When a message arrives for a client, UDP checks to see if an incoming queue has been created for the port number specified in the destination port number field of the user datagram. If there is such a queue, UDP sends the received user datagram to the end of the queue. If there is no such queue, UDP discards the user datagram and asks the ICMP protocol to send a port unreachable message to the server. All the incoming messages for one particular client program, whether coming from the same or a different server, are sent to the same queue.

- At the server site, a server asks for incoming and outgoing queues, using its well-known port, when it starts running. The queues remain open as long as the server is running.
- When a message arrives for a server, UDP checks to see if an incoming queue has been created for the port number specified in the destination port number field of the user datagram.
- If there is such a queue, UDP sends the received user datagram to the end of the queue. If there is no such queue, UDP discards the user datagram and asks the ICMP protocol to send a port unreachable message to the client.
- All the incoming messages for one particular server, whether coming from the same or a different client, are sent to the same queue.
- When a server wants to respond to a client, it sends messages to the outgoing queue, using the source port number specified in the request. UDP removes the messages one by one and, after adding the UDP header, delivers them to IP. An outgoing queue can overflow.

Queues in UDP



Use of UDP

- UDP is a suitable transport protocol for multicasting.
- UDP is suitable for a process with internal flow and error control mechanisms.
- UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control.
- UDP is used for management processes such as SNMP (Simple Network Management Protocol) & for some route updating protocols such as Routing Information Protocol (RIP).

TCP

- 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.
- TCP is called a connection-oriented, reliable transport protocol because it adds connection-oriented and reliability features to the services of IP.

TCP Services

- **Full-Duplex Communication:** TCP offers full-duplex service, in which data can flow in both directions at the same time. Each TCP then has a sending and receiving buffer, and segments move in both directions.
- **Connection-Oriented Service:** The two TCPs establish a connection between them, Data are exchanged in both directions and 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.

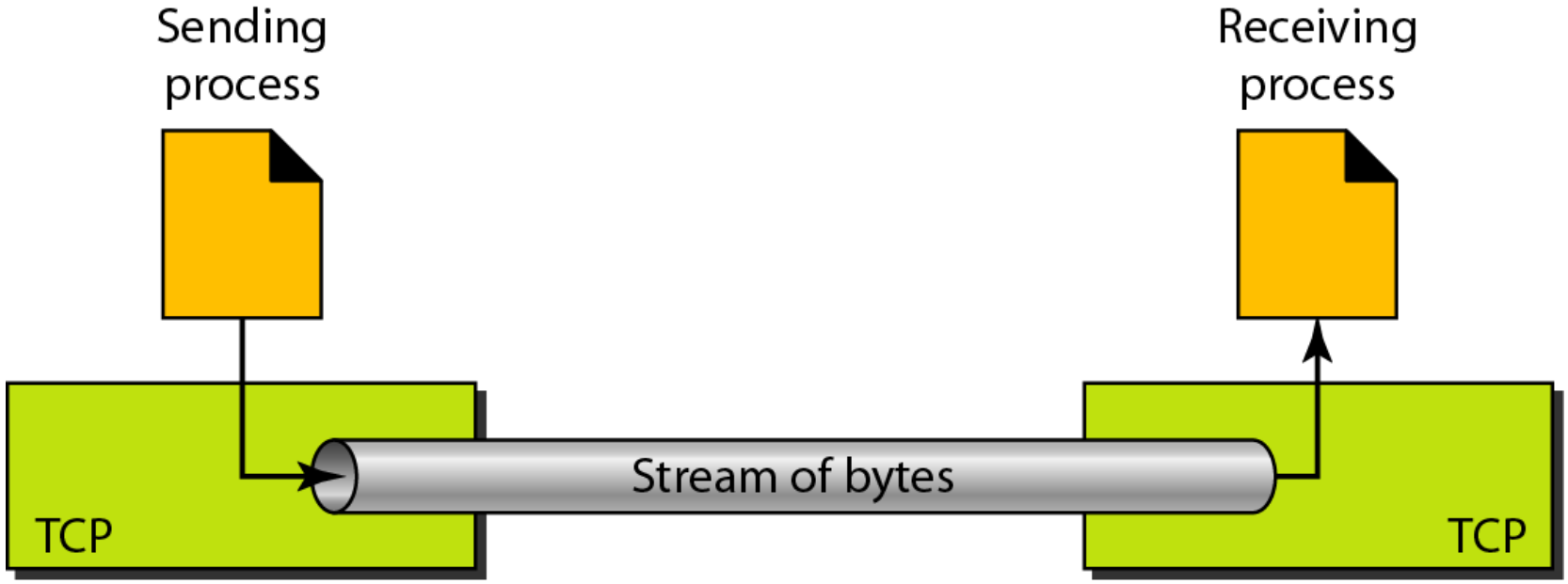
TCP Services

- **Process-to-Process Communication :** Like UDP, TCP provides process-to-process communication using port numbers.
- **Stream Delivery Service:** It is a stream-oriented protocol.
- It allows the sending process to deliver data as a stream 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 data across the Internet.
- The sending process produces (writes to) the stream of bytes, and the receiving process consumes (reads from) them.

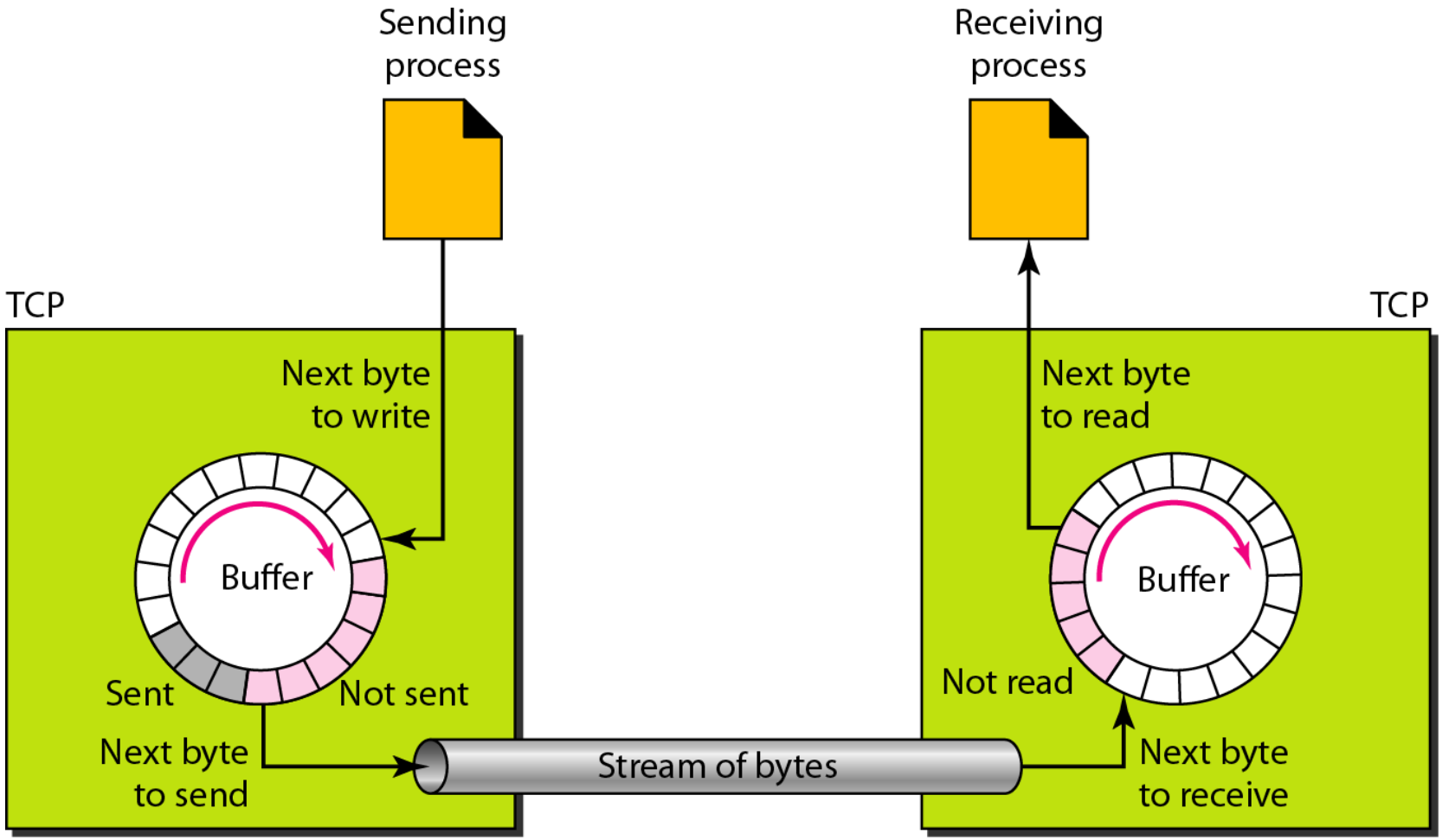
Sending and Receiving Buffers

- Because 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.
- **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.
- **Buffer at the receiver site :** 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.
- When a byte is read by the receiving process, the chamber is recycled and added to the pool of empty chambers.

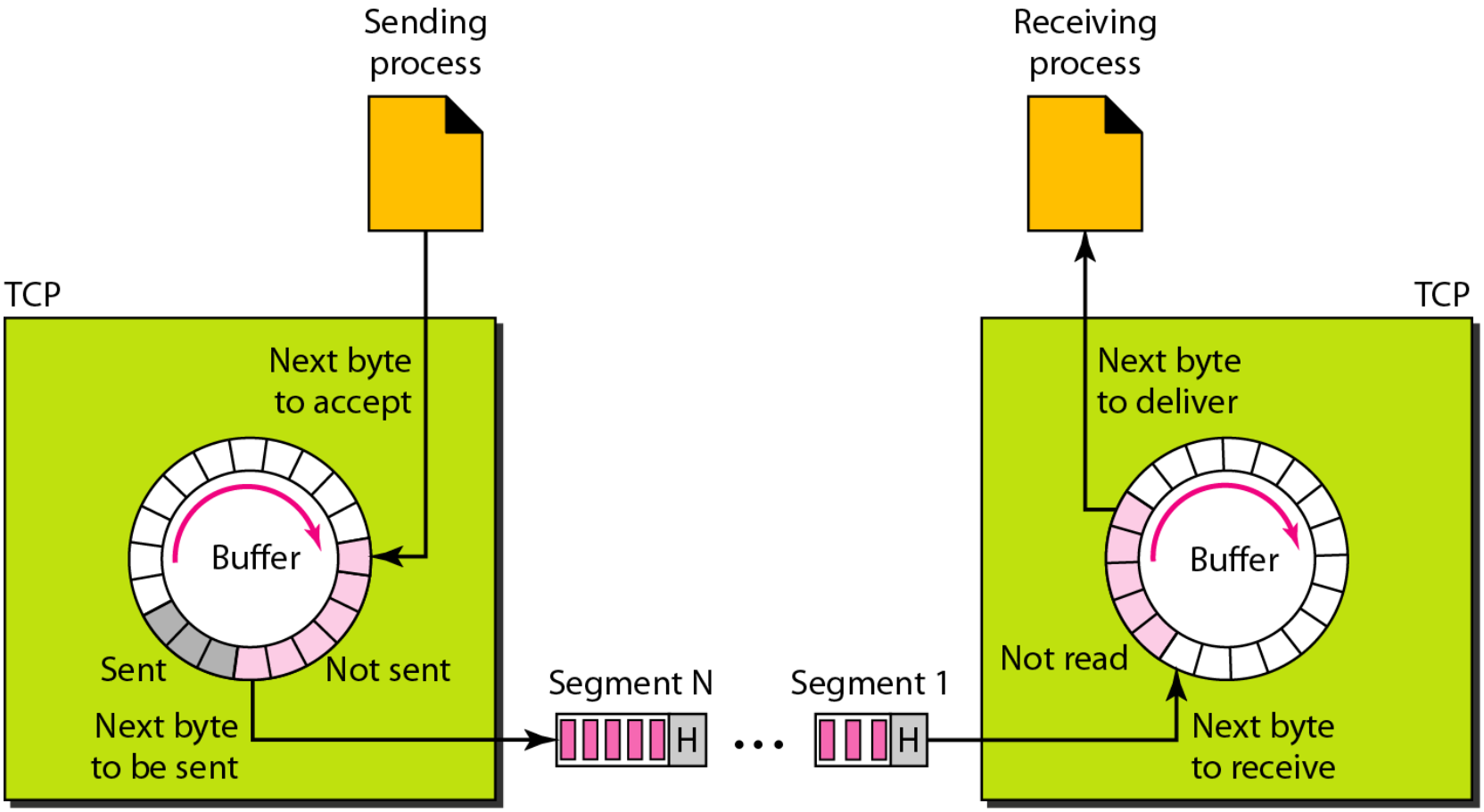
Stream delivery



Sending and receiving buffers



TCP segments



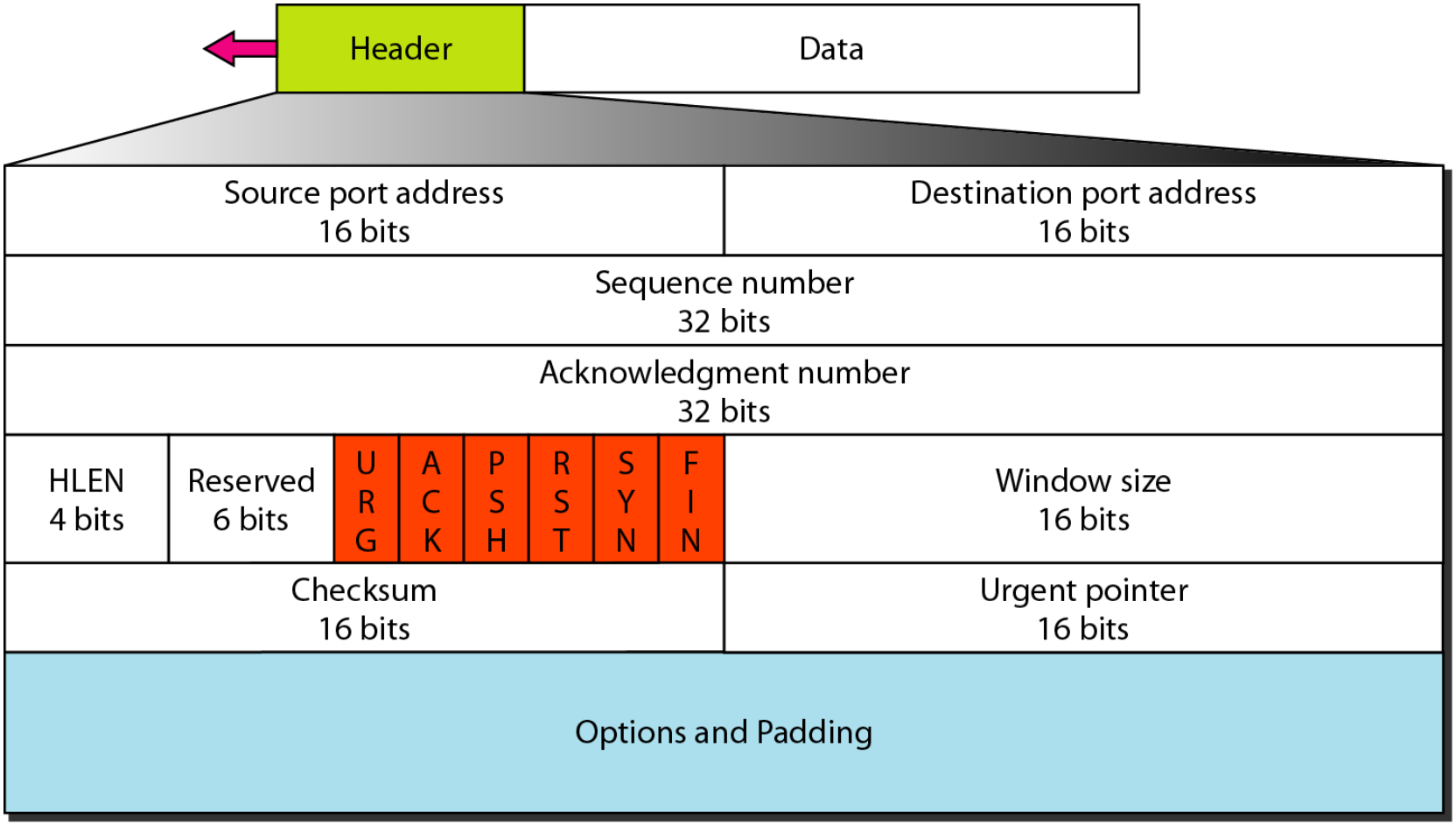
TCP Services

- **SEGMENTS:** At the transport layer, TCP groups a number of bytes together into a packet called a segment.
- TCP adds a header to each segment (for control purposes) and delivers the segment to the IP layer for transmission.
- The segments are encapsulated in IP datagrams and transmitted.
- The segments are not necessarily the same size.
- The segment consists of a 20- to 60-byte header, followed by data from the application program. The header is 20 bytes if there are no options and up to 60 bytes if it contains options.
- **Source port address:** This is a 16-bit field that defines the port number of the application program in the host that is sending the segment.
- **Destination port address:** This is a 16-bit field that defines the port number of the application program in the host that is receiving the segment.
-

SEGMENTS

- **Sequence number:** This 32-bit field defines the number assigned to the first byte of data contained in this segment. 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.
- **Header length:** This 4-bit field indicates the number of 4-byte words in the TCP header.
- **Reserved:** This is a 6-bit field reserved for future use.
- **Control:** This field defines 6 different control bits or flags.
- **Window size:** This field defines the size of the window, in bytes, that the other party must maintain.
- **Checksum:** This 16-bit field contains the checksum.
- **Urgent pointer:** This l6-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data.
- **Options:** There can be up to 40 bytes of optional information in the TCP header.

TCP segment format



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

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.

TCP Features

- **Numbering System:** It keeps track of the segments being transmitted or received, there is no field for a segment number value in the segment header. Instead, there are two fields called the sequence number and the acknowledgment number. These two fields refer to the byte number and not the segment number.
- **Byte Number :** The bytes of data being transferred in each connection are numbered by TCP. The numbering starts with a randomly generated number.
- **Sequence Number:** After the bytes have been numbered, TCP assigns a sequence number to each segment that is being sent. The sequence number for each segment is the number of the first byte carried in that segment. The value in the sequence number field of a segment defines the number of the first data byte contained in that segment.
- **Acknowledgment Number:** The value of the acknowledgment field in a segment defines the number of the next byte a party expects to receive. The acknowledgment number is cumulative

TCP Features

- **Flow Control :** TCP, unlike UDP, provides flow control. The receiver of the data controls the amount of data that are to be sent by the sender.
- **Error Control:** To provide reliable service, TCP implements an error control mechanism.
- **Congestion Control:** 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 in the network.

TCP Connection

- TCP is connection-oriented. A connection-oriented transport protocol establishes a virtual path between the source and destination.
- All the segments belonging to a message are then sent over this virtual path.
- Using a single virtual pathway for the entire message facilitates the acknowledgment process as well as retransmission of damaged or lost frames.
- In TCP, connection-oriented transmission requires three phases: connection establishment, data transfer, and connection termination.

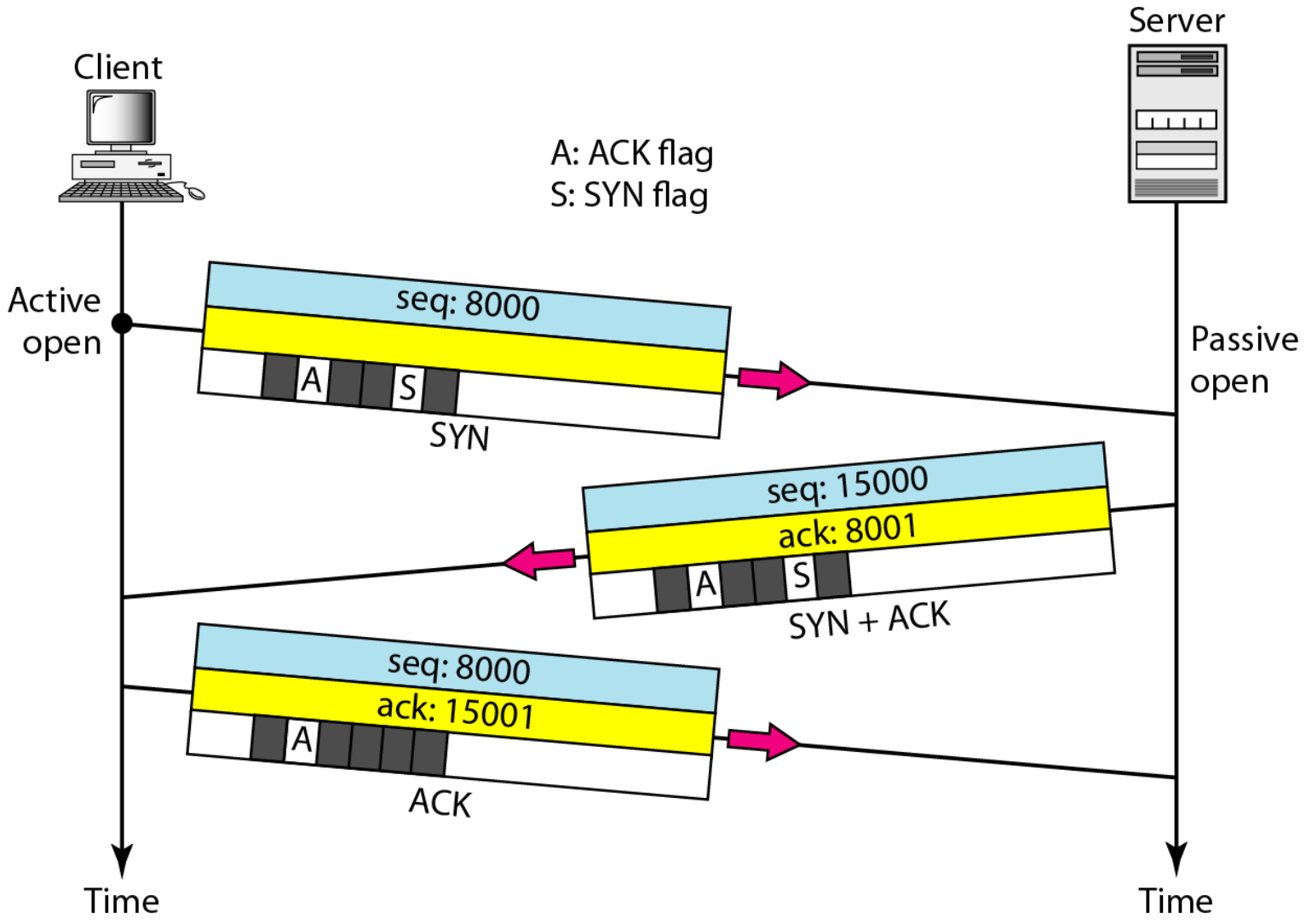
Connection Establishment

- 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.

Connection establishment using three-way handshaking

- The three steps in this phase are as follows.
- 1. The client sends the first segment, a SYN segment, in which only the SYN flag is set. This segment is for synchronization of sequence numbers. It consumes one sequence number. When the data transfer starts, the sequence number is incremented by 1.
- 2. The server sends the second segment, a SYN + ACK segment, with 2 flag bits set: SYN and ACK. This segment has a dual purpose. It is a SYN segment for communication in the other direction and serves as the acknowledgment for the SYN segment.
- 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.

Connection establishment using three-way handshaking



REFRENCES

- “ DATA COMMUNICATIONS AND NETWORKING ”, Behrouz A. Forouzan And Sophia Chung Fegan , Fourth Edition , McGraw-Hill