

# Chapter Five

## OSI Model

### Data Communication and Computer Networks

(SEng2051)

[haleluyaluya@yahoo.com](mailto:haleluyaluya@yahoo.com)

# Communication and Layer Architecture

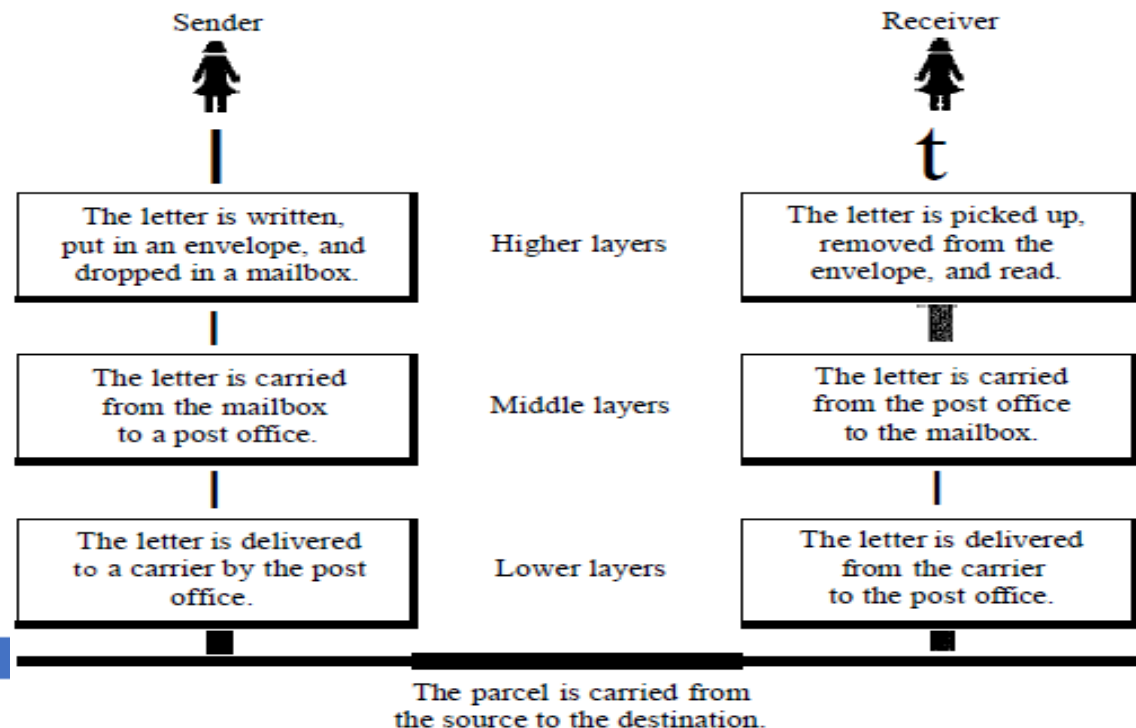
- ✚ A network is a combination of **hardware** and **software** that sends data from one location to another.
- ✚ The hardware consists of the **physical equipment** that carries signals from one point of the network to another.
- ✚ The software consists of **instruction sets** that make possible the services that we expect from a network
- ✚ For example, the task of sending an **e-mail** from one point in the world to another can be broken into several tasks, each performed by a separate software package.
- ✚ Each **software package** uses the services of another software package.
- ✚ At the lowest layer, a **signal**, or a **set of signals**, is sent from the source computer to the destination computer.

# Layered Tasks

✉ We use the concept of layers in our daily life.

✉ Eg. let us consider **two friends** who communicate through **postal mail**.

✉ The process of sending a letter to a friend would be complex if there were **no services** available from the **post office**.



## Contd.

- ✧ A **communication architecture** is a strategy for connecting host computers and other communicating equipment.
- ✧ It defines **necessary elements** for data communication between devices.
- ✧ A communication architecture, therefore, defines a **standard** for the communicating hosts.
- ✧ A **programmer** formats data in a manner defined by the communication architecture and passes it on to the **communication software**.
- ✧ Separating communication functions adds **flexibility**, for example, we do **not need to modify** the entire host software to include more communication devices.

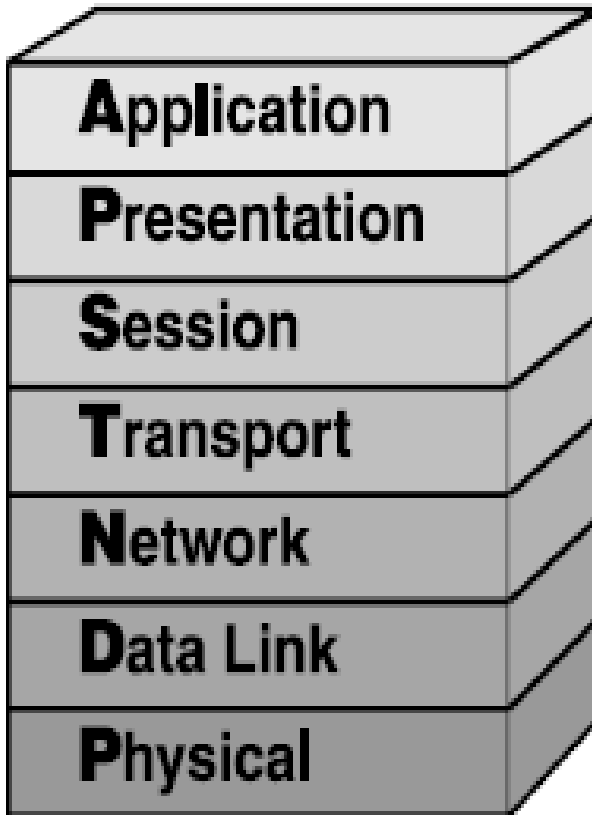
## Contd.

- ✚ Layer architecture simplifies the **network design**.
- ✚ It is easy to **debug network applications** in a layered architecture network.
- ✚ The **network management** is easier due to the layered architecture.
- ✚ Network layers follow a set of rules, called **protocol**.
- ✚ The protocol defines the **format of the data** being exchanged, and the **control and timing** for the **handshake** between layers.

# Open Systems Interconnection (OSI)

- ✧ International standard organization (ISO) established a committee in 1977 to develop an architecture for computer communication.
- ✧ Open Systems Interconnection (OSI) reference model is the result of this effort.
- ✧ In 1984, the Open Systems Interconnection (OSI) reference model was approved as an international standard for communications architecture.
- ✧ Term “open” denotes the ability to connect any two systems which conform to the reference model and associated standards.
- ✧ The OSI model is now considered the primary Architectural model for inter-computer communications.
- ✧ The OSI model describes how information or data makes its way from application programmes (such as spreadsheets) through a network medium (such as wire) to another application programme located on another network.
- ✧ The OSI reference model divides the problem of moving information between computers over a network medium into SEVEN smaller and more manageable problems.
- ✧ This separation into smaller more manageable functions is known as layering.

# The Seven Layers



All  
People  
Seem  
To  
Need  
Data  
Processing

Please  
Do  
Not  
Throw  
Susage  
Pizza  
Away

## Contd

Application

- File, print, message, database, and application services

Presentation

- Data encryption, compression, and translation services

Session

- Dialog control

Transport

- End-to-end connection

Network

- Routing

Data Link

- Framing

Physical

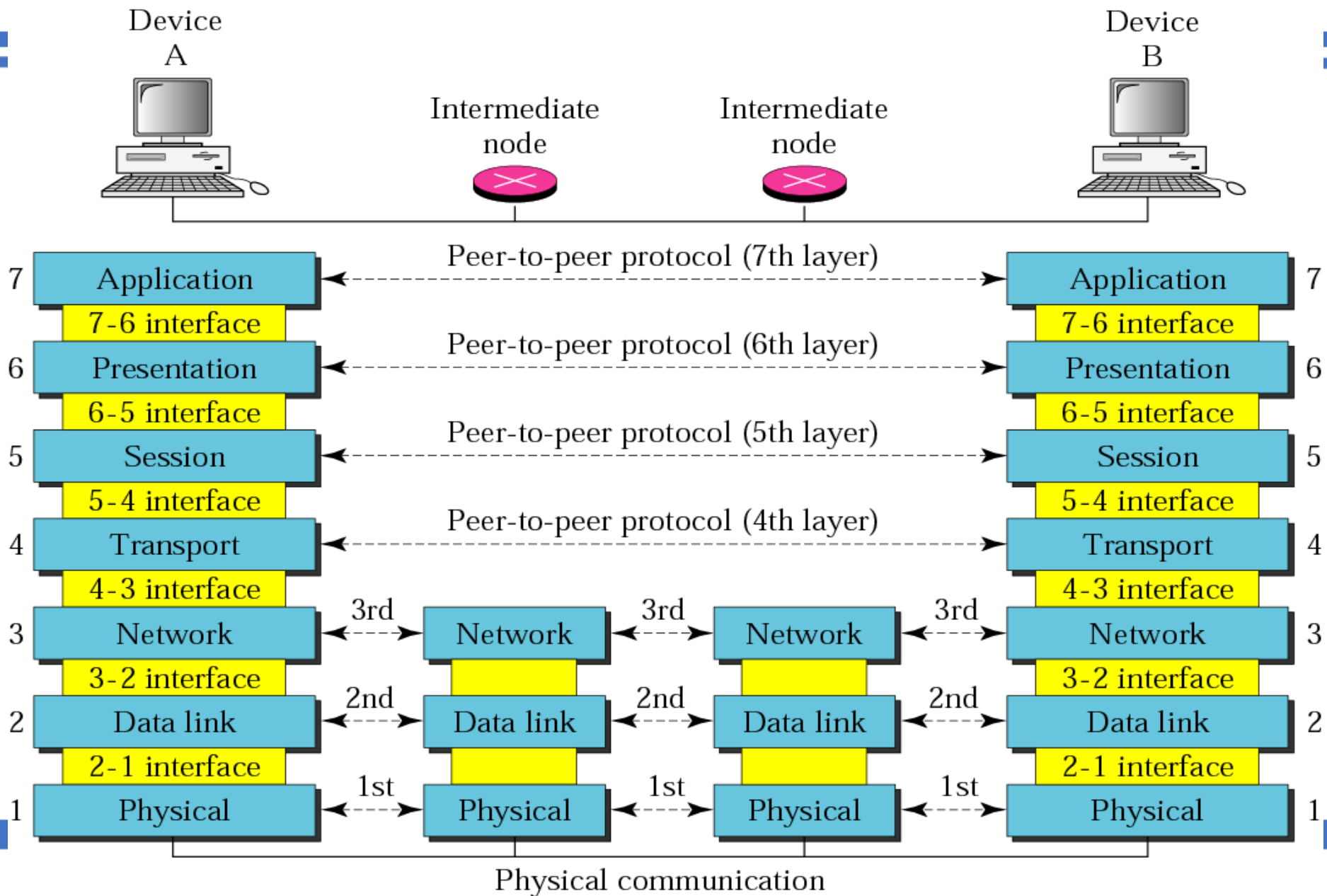
- Physical topology



# Contd.

- ✧ The OSI model is composed of seven **ordered layers**:
  - ✧ Physical (layer 1),
  - ✧ Data link (layer 2),
  - ✧ Network (layer 3),
  - ✧ Transport (layer 4),
  - ✧ Session (layer 5),
  - ✧ Presentation (layer 6), and
  - ✧ Application (layer 7).
- ✧ The figure in the next slide shows the layers involved when a message is sent from **device A to device B**. As the message travels from A to B, it may pass through many **intermediate nodes**.
- ✧ These intermediate nodes usually involve only the **first three bottom layers** of the OSI model.

# Contd.



## Contd.

- ✚ In *developing the model*, the designers distilled the process of **transmitting data to its most fundamental elements**.
- ✚ They identified which networking functions had related uses and collected those functions into discrete groups that became the layers.
- ✚ Each layer defines a family of functions distinct from those of the other layers.
- ✚ By defining and localizing functionality in this fashion, the designers created an architecture that is both **comprehensive and flexible**.
- ✚ Most importantly, the OSI model allows complete **interoperability** between otherwise incompatible systems.
- ✚ Within **a single machine**, each layer calls upon the **services of the layer just below it**.

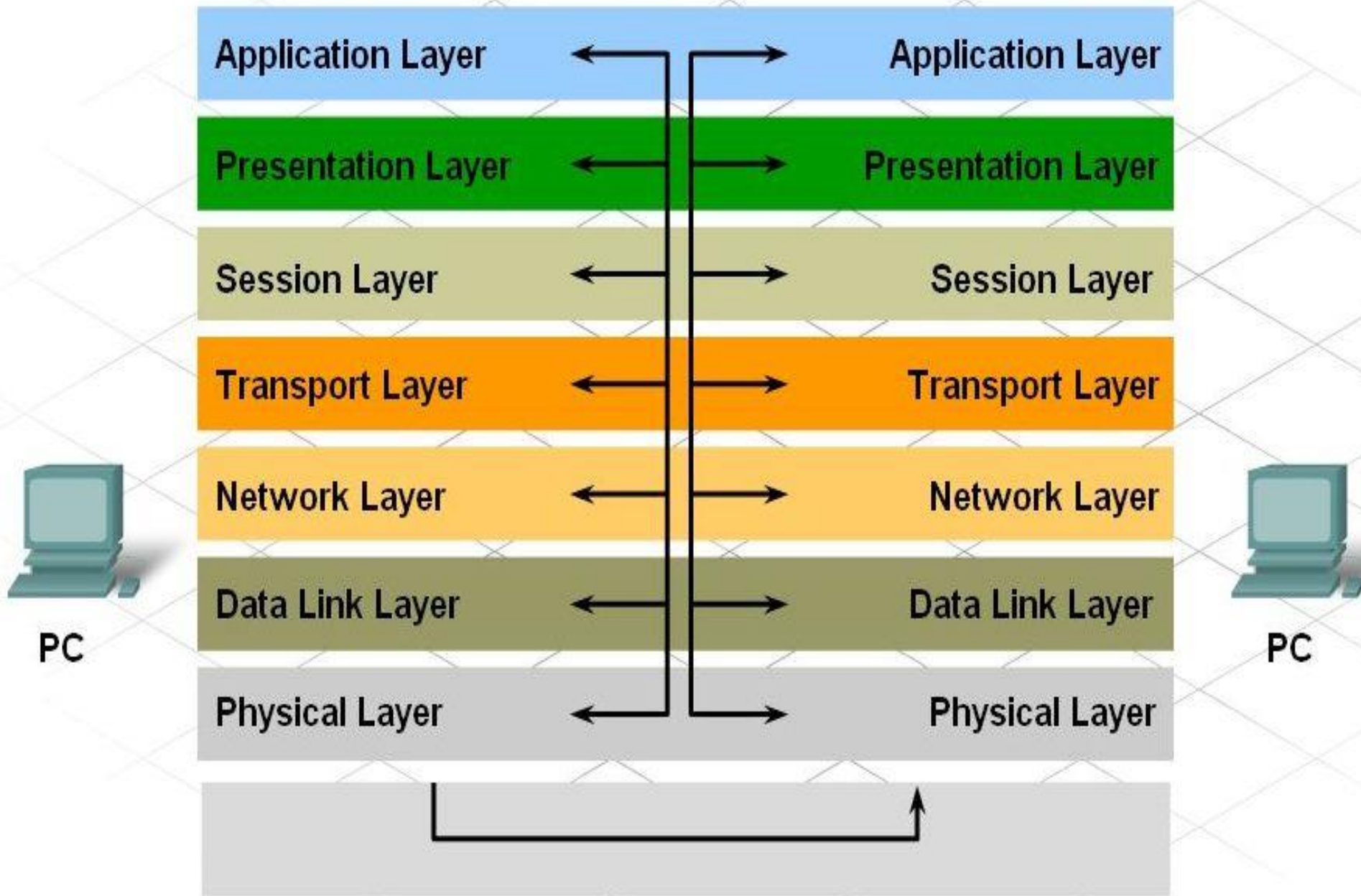
## Contd.

- Layer 3, for example, uses the services provided by layer 2 and provides services for layer 4. Between machines, layer  $x$  on one machine communicates with layer  $x$  on another machine.
- This communication is governed by an agreed-upon series of rules and conventions called protocols.
- The processes on each machine that communicate at a given layer are called peer-to-peer processes.
- Communication between machines is therefore a peer-to-peer process using the protocols appropriate to a given layer.

# Peer-to-Peer Processes

- ⚡ At the **physical layer**, communication is **direct**: At the higher layers, however, communication must move down through the layers on **device A**, over to **device B**, and then **back up** through the layers.
- ⚡ Each layer in the sending device **adds its own information** to the message it receives from the layer just above it and passes the whole package to the layer just below it.
- ⚡ At layer 1 the **entire package** is **converted to a form that can be transmitted to the receiving device**.
- ⚡ At the receiving machine, the message is **unwrapped layer by layer**, with each process **receiving and removing** the data **meant for it**.
- ⚡ For example, layer 2 removes the data meant for it, then passes the rest to layer 3. Layer 3 then removes the data meant for it and passes the rest to layer 4, and so on.

## Contd.



# Interfaces Between Layers

- ✉ The passing of the data and network information down through the layers of the sending device and back up through the layers of the receiving device is made possible by an **interface** between each pair of **adjacent layers**.
- ✉ Each interface **defines the information** and **services a layer must provide** for the layer above it.
- ✉ Well-defined **interfaces and layer functions** provide **modularity** to a network.
- ✉ As long as a layer provides the expected services to the layer above it, the specific implementation of its functions can be modified or replaced without requiring changes to the surrounding layers.

# Organization of the Layers

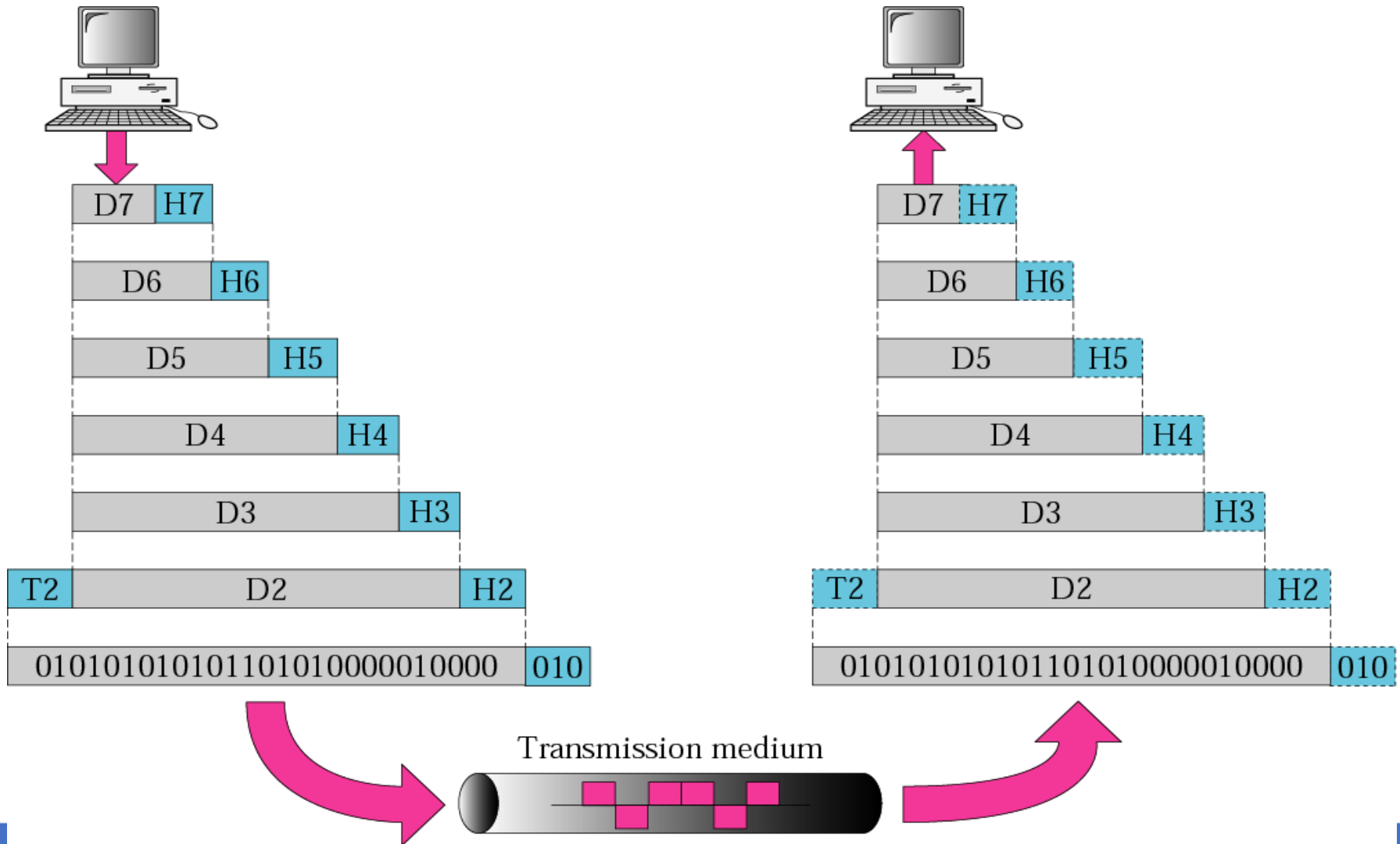
- ✚ The seven layers can be thought of as belonging **to three subgroups**.
- ✚ Layers 1, 2, and 3 - **physical, data link, and network** - are the **network support layers**; they deal with the physical aspects of moving data from one device to another (such as **electrical specifications, physical connections, physical addressing, and transport timing and reliability**).
- ✚ Layers 5, 6, and 7- **session, presentation, and application** - can be thought of as the user support layers; they allow interoperability among unrelated software systems.
- ✚ Layer 4, the **transport layer**, **links** the two subgroups and **ensures that what the lower layers have transmitted is in a form that the upper layers can use**.
- ✚ The upper OSI layers are almost always implemented in **software**; lower layers are a combination of **hardware and software**, except for the physical layer, which is mostly **hardware**.



# Contd.

- ✎ The figure shown in the next slide, gives an overall view of the OSI layers, D7 means the data unit at layer 7, D6 means the data unit at layer 6, and so on. The process starts at layer 7 (the application layer), then moves from layer to layer in descending, sequential order.
- ✎ At each layer, a header, or possibly a trailer, can be added to the data unit.
- ✎ Commonly, the trailer is added only at layer 2.
- ✎ When the formatted data unit passes through the physical layer (layer 1), it is changed into an electromagnetic signal and transported along a physical link.

# An exchange using the OSI model



## Contd.

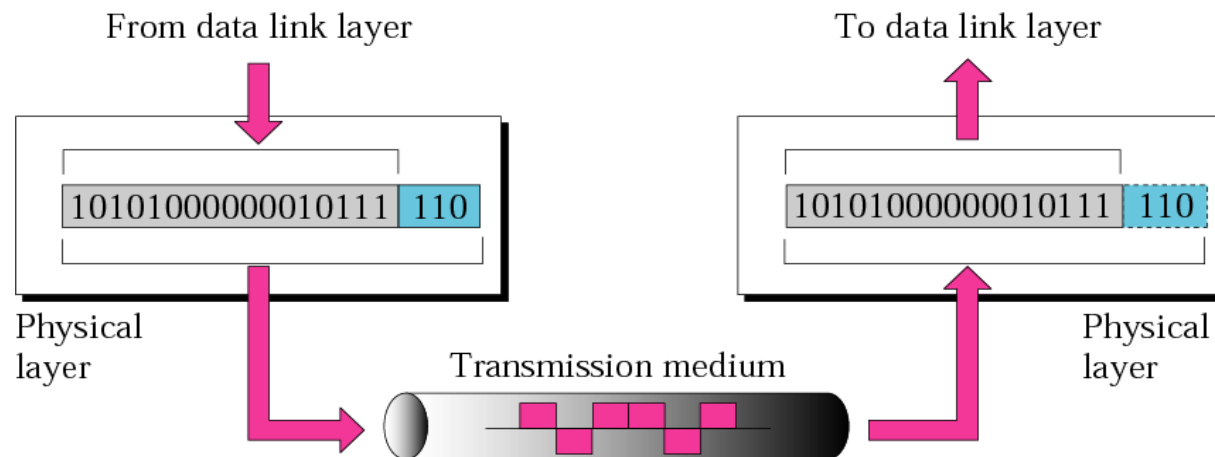
- ✈ Upon reaching its destination, the **signal passes into layer 1** and is transformed back into **digital form**. The data units then move back up through the OSI layers.
- ✈ As each block of data reaches the **next higher layer**, the **headers and trailers attached** to it at the corresponding sending layer are **removed**, and actions appropriate to that layer are taken.
- ✈ By the time it reaches **layer 7**, the message is again in a form appropriate to the application and is made available to the **recipient**.

# Encapsulation

- ✎ A packet (header and data) at level 7 is encapsulated in a packet at level 6. The whole packet at level 6 is encapsulated in a packet at level 5, and so on.
- ✎ In other words, the data portion of a packet at level  $N - 1$  carries the whole packet (data and header and may be trailer) from level  $N$ . The concept is called *encapsulation*; level  $N - 1$  is not aware of which part of the encapsulated packet is data and which part is the header or trailer.
- ✎ For level  $N - 1$ , the whole packet coming from level  $N$  is treated as one integral unit.

# Physical Layer

- ✧ The physical layer coordinates the functions required to carry a **bit stream** over a **physical medium**.
- ✧ It deals with the **mechanical and electrical specifications** of the interface and transmission medium.
- ✧ It also **defines the procedures and functions** that physical devices and interfaces have to perform for transmission to occur.



**The physical layer is responsible for movements of individual bits from one hop (node) to the next.**

## The physical layer is also concerned with the following:

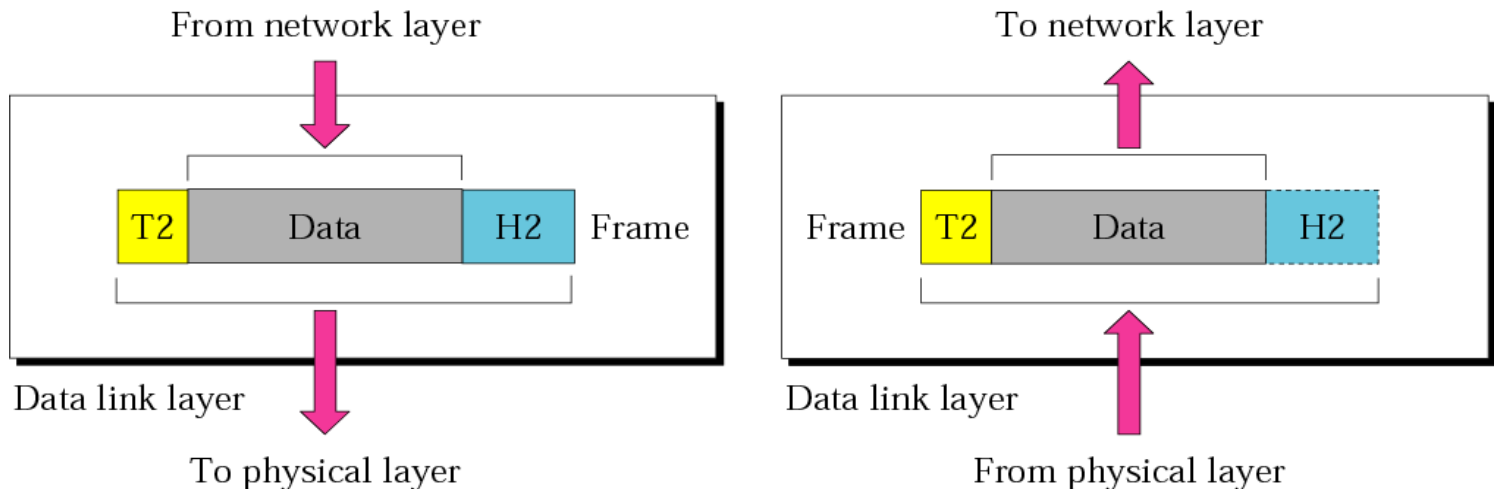
1. **Physical characteristics of interfaces and medium.** The physical layer defines the characteristics of the interface between the devices and the transmission medium. It also defines the type of transmission medium.
2. **Representation of bits.** The physical layer data consists of a stream of bits (sequence of 0s or 1s) with no **interpretation**. To be transmitted, bits must be **encoded into signals** - **electrical or optical**. The physical layer defines the type of encoding (how 0s and 1s are changed to signals) (**Refer chapter one**).
3. **Data rate.** The transmission rate - the number of bits sent each second - is also defined by the physical layer. In other words, the physical layer defines the duration of a bit, which is how long it lasts.
4. **Synchronization of bits.** The sender and receiver not only must use the same bit rate but also must be synchronized at the bit level. In other words, the sender and the receiver clocks must be synchronized.

## Contd.

5. **Line configuration.** The physical layer is concerned with the connection of devices to the media (point – to – point or multipoint).
6. **Physical topology.** The physical topology defines how devices are connected to make a network.
6. **Transmission mode.** The physical layer also defines the direction of transmission between two devices: simplex, half-duplex, or full-duplex.

# Data Link Layer

- The data link layer transforms the physical layer, a raw transmission facility, to a **reliable link**.
- It makes the physical layer appear **error-free** to the upper layer (**network layer**). The following figure shows the relationship of the data link layer to the network and physical layers.




The data link layer is responsible for moving frames from one hop (node) to the next.



# Layer 2 frame structure

Start Frame (Flag)	Header		Data	Trailer	Stop Frame (Flag)
	Address	Type/Length		FCS	

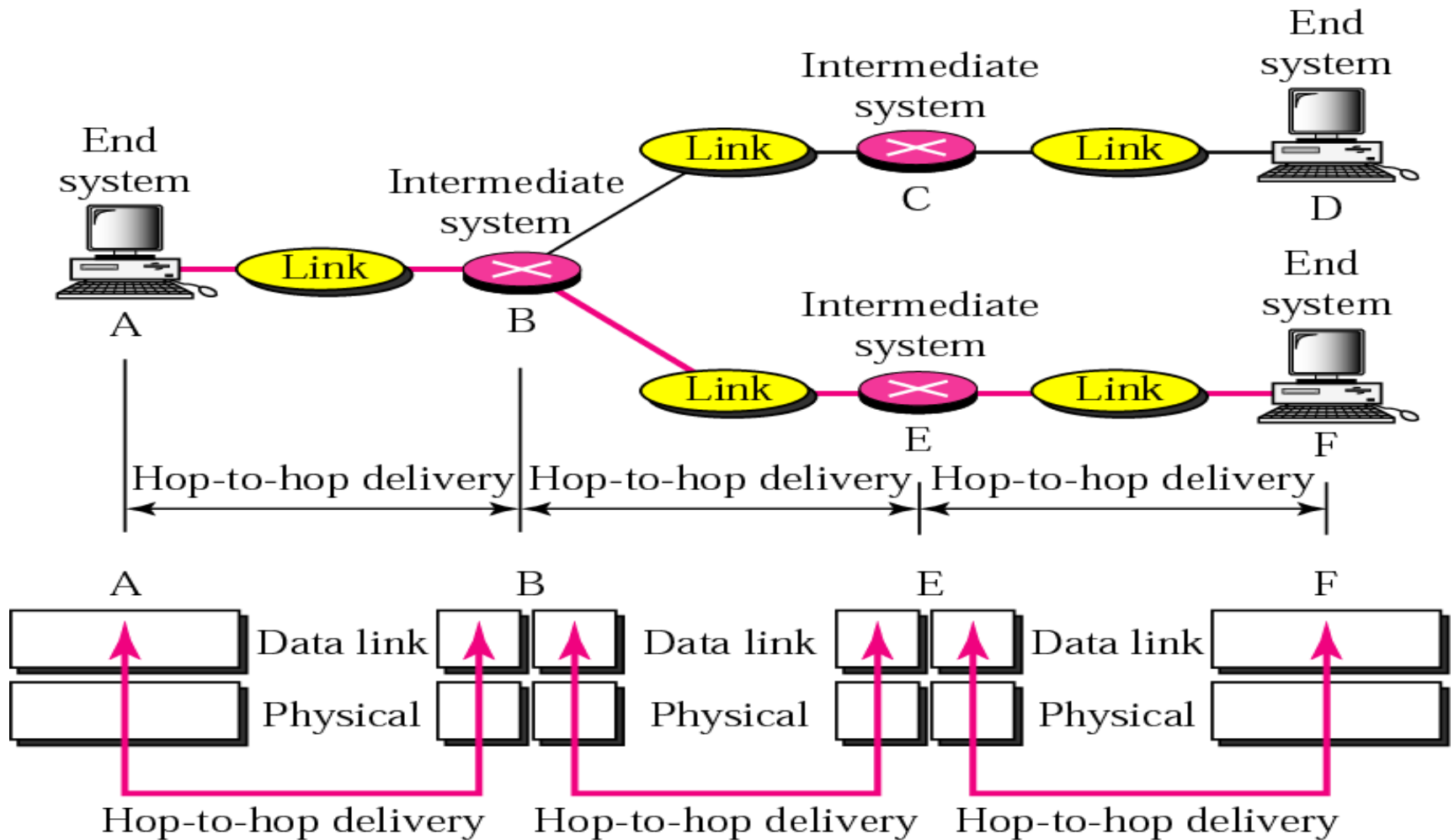
## Data Link Layer Responsibilities

1. **Framing.** The data link layer divides the stream of bits received from the network layer into manageable data units called frames.
2. **Physical addressing.** If frames are to be distributed to different systems on the network, the data link layer adds a header to the frame to define the sender and/or receiver of the frame.
  -  If the frame is intended for a system outside the sender's network, the receiver address is the address of the device that connects the network to the next one.

## Contd.

3. **Flow control.** If the rate at which the data are absorbed by the receiver is less than the rate at which data are produced in the sender, the data link layer imposes a flow control mechanism to avoid overwhelming the receiver.
4. **Error control.** The data link layer adds reliability to the physical layer by adding mechanisms to detect and retransmit damaged or lost frames.
  - ✎ It also uses a mechanism to recognize duplicate frames.
  - ✎ Error control is normally achieved through a **trailer** added to the end of the frame.
5. **Access control.** When two or more devices are connected to the same link, data link layer protocols are necessary to determine which device has control over the link at any given time.

# Hop-to-hop (node-to-node) delivery by data link layer



# Contd.

✎ As the figure above shows, communication at the **data link layer** occurs between two **adjacent nodes**.

- To send data from A to F, three partial deliveries are made.
- First, the data link layer at A sends a frame to the data link layer at B (a router).
- Second, the data link layer at B sends a new frame to the data link layer at E.
- Finally, the data link layer at E sends a new frame to the data link layer at F.

✎ Note that the frames that are exchanged between the three nodes have **different values** in the **headers**. The frame from **A to B** has **B** as the **destination address** and **A** as the **source address**. The frame from **B to E** has **E** as the **destination address** and **B** as the **source address**. The frame from **E to F** has **F** as the **destination address** and **E** as the **source address**. The values of the **trailers** can also be different if error checking includes the header of the frame.

# Framing

- ✉ The data link layer, needs to pack bits into frames, so that each frame is distinguishable from another. Our postal system practices a type of framing.
- ✉ The simple act of inserting a letter into an envelope separates one piece of information from another; the envelope serves as the delimiter.
- ✉ In addition, each envelope defines the sender and receiver addresses since the postal system is a many-to-many carrier facility.
- ✉ Framing in the data link layer separates a **message** from one source to a destination, or from other messages to other destinations, by adding a sender address and a destination address.
- ✉ The destination address defines where the packet is to go; the sender address helps the recipient acknowledge the receipt.

# Contd.

- ✚ NB: Addressing here is about the next node in the LAN
- ✚ Although the whole message could be packed in one frame, that is not normally done.
- ✚ One reason is that a **frame can be very large**, making **flow** and **error control** very inefficient.
- ✚ When a message is carried in one very large frame, even a single-bit error would require the retransmission of the whole message.
- ✚ When a message is divided into smaller frames, a single-bit error affects only that small frame.

# Frames can be of fixed or variable size

1. **Fixed-Size Framing**:- In **fixed-size framing**, there is no need for defining the boundaries of the frames; the size itself can be used as a **delimiter**. An example of this type of framing is the ATM wide-area network, which uses frames of fixed size called cells.
2. **Variable-Size Framing**:- variable-size framing is prevalent in **local area networks**. In variable-size framing, we need a way to define the **end of the frame** and the **beginning of the next**. Historically, two approaches were used for this purpose: a **character-oriented** approach and a **bit-oriented** approach.

Structure of the Ethernet Frame

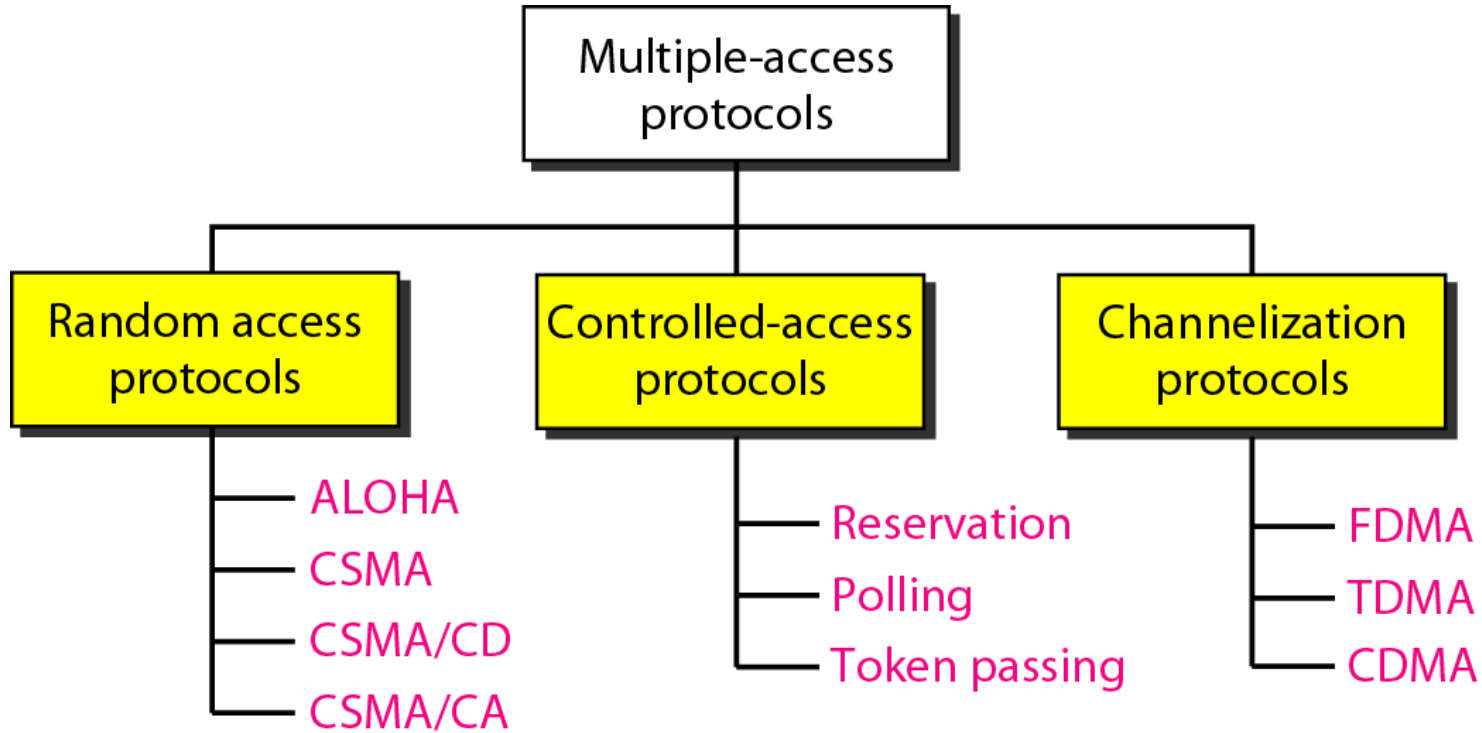
Preamble	SFD	Destination MAC Address	Source MAC Address	Length / Type	Encapsulated Data	FCS
7	1	6	6	2	46 to 1500	4

# Media Access control

- The data link layer can further be divided into two layers: the **upper sub-layer** that is responsible for **flow and error control** is called the **logical link control (LLC) layer**; the lower sub-layer that is mostly responsible for **multiple access resolution** is called the **media access control (MAC) layer**
- When nodes or stations are connected and use a **common link**, called a **multipoint or broadcast link**, we need a **multiple-access protocol** to coordinate access to the link. The problem of controlling the access to the medium is similar to the rules of **speaking in an assembly**. The procedures guarantee that the right to speak is upheld and ensure that **two people do not speak at the same time**, **do not interrupt each other**, **do not monopolize the discussion**, and so on.



# Contd.



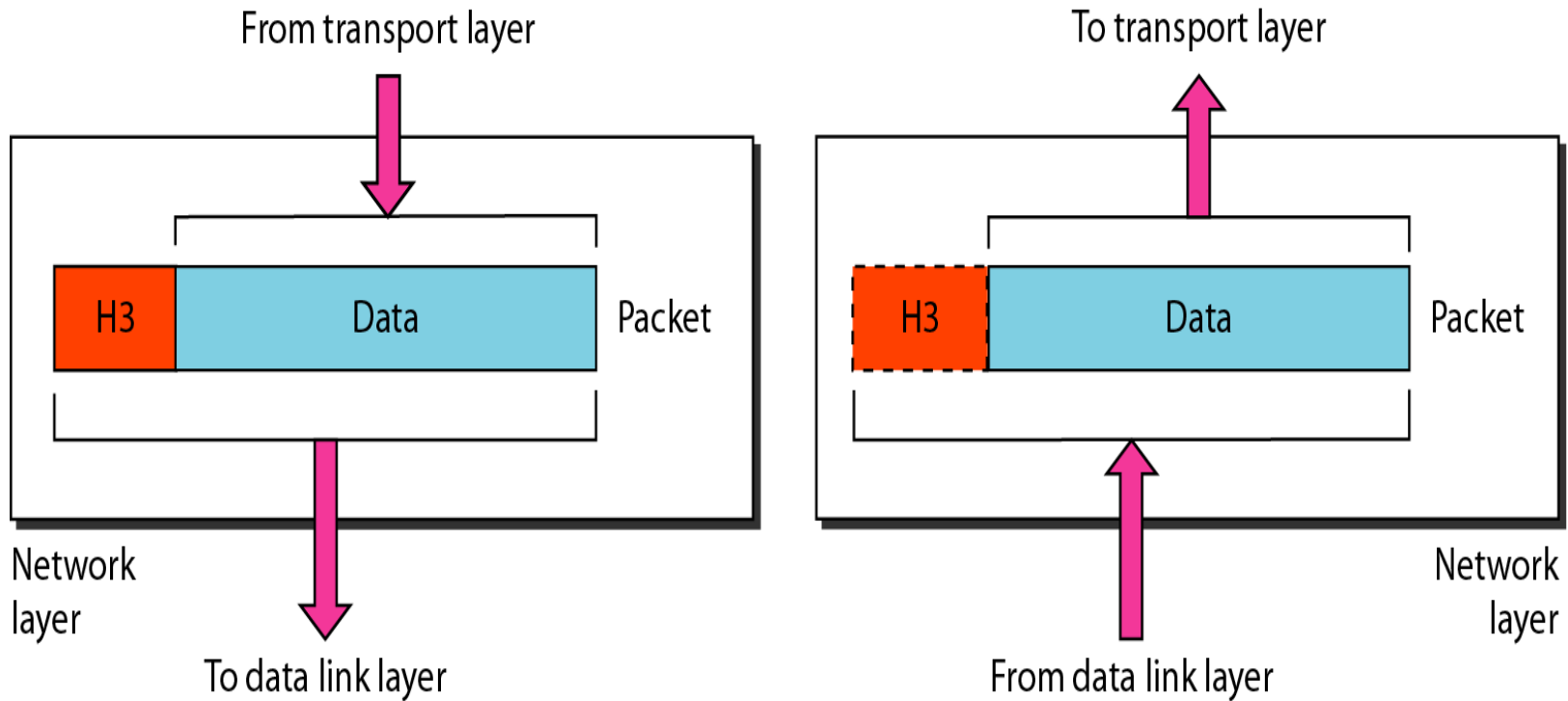
- Read about these?

## Network Layer (3<sup>rd</sup> OSI Layer)

- Concerned with getting packets from **source to destination**.
- The network layer must know the topology of the **subnet** and **choose appropriate paths** through it.
- When source and destination are in **different networks**, the network layer must deal with these differences.
- The network layer is responsible for the **source-to-destination delivery of a packet**, possibly across multiple networks.
- Whereas the data link layer oversees the delivery of the packet between two **systems on the same network**, the network layer ensures that each packet gets from its point **of origin** to its **final destination**.
- If two systems are connected to the **same local network**, there is usually **no need for a network layer**. However, if the two systems are attached to different networks with connecting devices between the networks, there is often a need for the network layer to accomplish **source-to-destination delivery**.

## Contd.

- The network layer is responsible for the delivery of individual packets from the **source host to the destination host**.

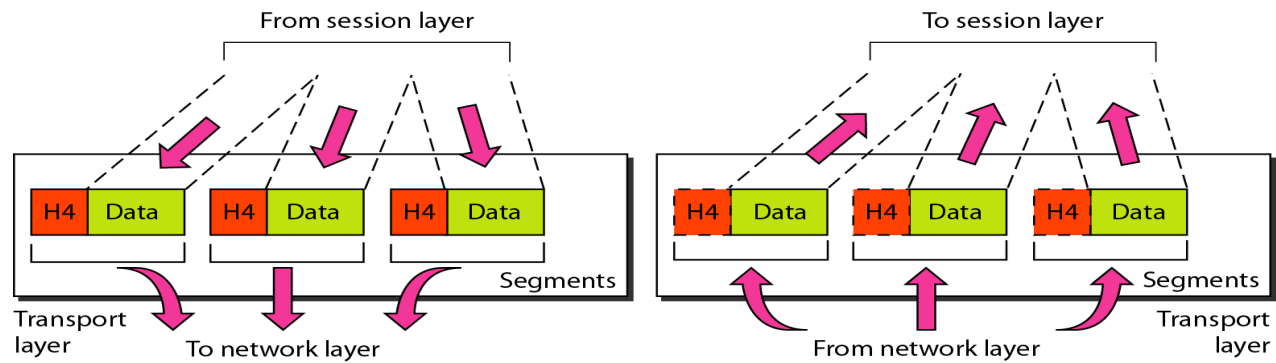
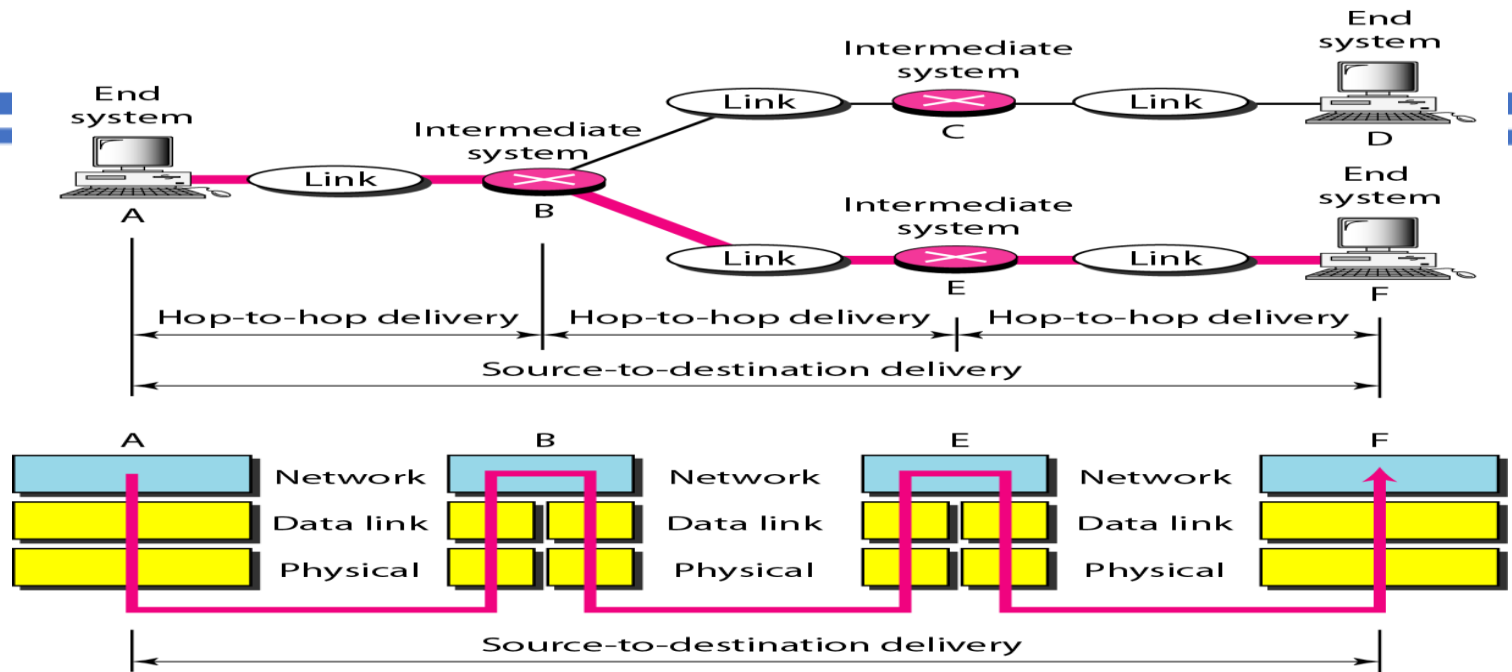


## Contd.

Other responsibilities of the network layer include the following:

- 1. Logical addressing:-** The physical addressing implemented by the data link layer handles the **addressing problem locally**.
  - If a packet passes the **network boundary**, we need another addressing system to help distinguish the source and destination systems.
  - The network layer **adds a header** to the packet coming from the upper layer that, among other things, includes the **logical addresses** of the **sender and receiver**. We discuss logical addresses later in chapter Six.
- 2. Routing:-** When **independent** networks or links are connected to create **internetworks** (network of networks) or a large network, the connecting devices (called **routers or switches**) **route or switch the packets to their final destination**.
  - ❖ One of the functions of the network layer is to provide this mechanism.

# Contd.



## Transport layer (4<sup>th</sup> OSI layer)

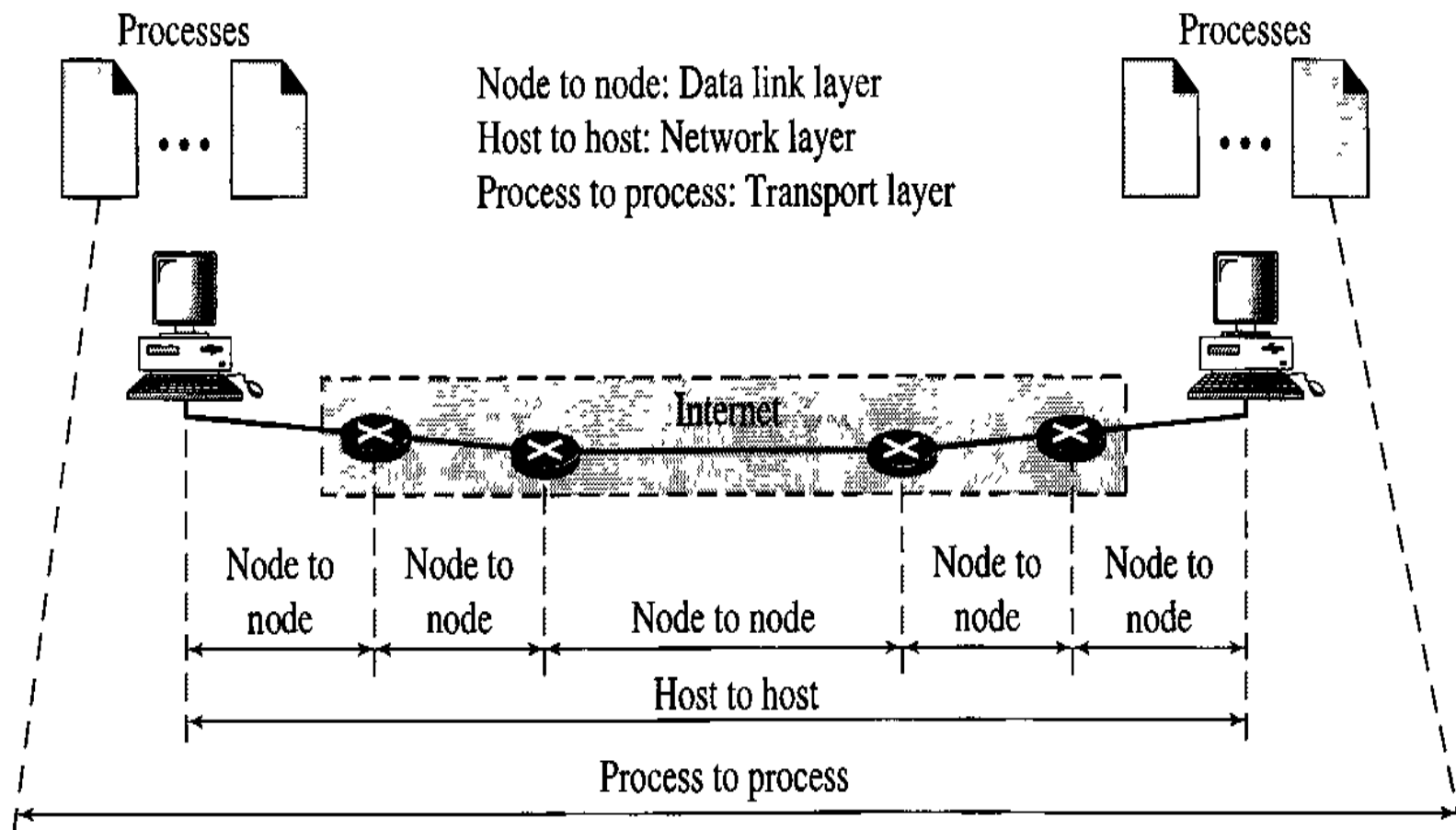
- ✉ The transport layer is responsible for **process-to-process delivery** of the entire message.
- ✉ A process is an **application program** running on a host.
- ✉ Whereas the network layer oversees **source-to-destination** delivery of individual packets, it does **not recognize any relationship between those packets**.
- ✉ The network layer treats each packet **independently**, as though each piece belonged to a **separate message**, whether or not it does.
- ✉ The transport layer, on the other hand, **ensures** that the whole **message arrives intact** and **in order**, overseeing both **error control** and **flow control** at the **source-to-destination** level.

## Major functions of the transport layer

➤ The Transport layer encompasses these functions:

1. Enables multiple applications to communicate over the network at the same time on a single device
2. Ensures that, if required, all the data is received reliably and in order by the correct application
3. Employs error handling mechanisms

# Contd.





# Responsibilities of the transport layer include the following:

1. **Service-point addressing:-** Computers often run several programs at the same time. For this reason, **source-to-destination delivery** means delivery not only **from one computer to the next** but also from a **specific process** (running program) 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** (or **port address**).
  - *The network layer gets each packet to the **correct computer**; the transport layer gets the entire message to the **correct process** on that computer.*
2. **Segmentation and reassembly:-** A message is divided into transmittable segments, with each segment containing a **sequence number**.
  - These numbers enable the transport layer to **reassemble** the message correctly upon arriving at the destination and to identify and replace packets that were **lost** in transmission.

## Contd.

3. **Flow control:-** Like the data link layer, the transport layer is responsible for flow control.
  - 📢 However, flow control at this layer is performed **end to end** rather than across a **single link**.
4. **Error control:-** Like the data link layer, the transport layer is responsible for error control.
  - 📢 However, error control at this layer is performed **process-to-process** rather than **across a single link**.
  - 📢 The sending transport layer makes sure that the entire message arrives at the receiving transport layer without error (damage, loss, or duplication).
  - 📢 Error correction is usually achieved through **retransmission**.

## Contd.

5. **Connection control:-** The transport layer can be either **connectionless or connection-oriented**.

**A. Connectionless** transport layer treats each segment as an **independent packet** and delivers it to the transport layer at the destination machine.

**B. Connection-oriented** transport layer makes a connection with the transport layer at the destination machine first before delivering the packets.

➤ After all the data are transferred, the **connection is terminated**.

# Process-to-Process Delivery

- As a revision, the data link layer is responsible for delivery of frames between two **neighboring nodes** over a link. This is called *node-to-node delivery*.
- The network layer is responsible for delivery of datagrams between two hosts.* This is called *host-to-host 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**). We need process-to-process delivery.
- However, 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.**
- The transport layer is responsible for *process-to-process* delivery-the delivery of a packet, part of a message, from one process to another.
  - Two processes communicate in a **client/server relationship**.

# Transport layer addressing

- 🔊 Whenever we need to deliver something to one specific destination among many, we need an **address**.
- 🔊 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.

## Contd.

- 🔊 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**.
- In the Internet model, the port numbers are **16-bit** integers between **0 and 65,535**.
- The client program defines itself with a port number, chosen **randomly** by the transport layer software running on the client host. This is the **ephemeral (temporal) port number**. (next slide, more on port number)

# Connectionless Versus Connection-Oriented Service

- A transport layer protocol can either be **connectionless** or **connection-oriented**.

## 1. *Connectionless Service*

- In a connectionless service, the packets are sent from one party to another with **no need for connection establishment** or **connection release**.
- The packets are **not numbered**; they may be **delayed or lost or may arrive out of sequence**. There is **no acknowledgment** either.
- UDP is connectionless.

## 2. *Connection-Oriented Service*

- a. In a connection-oriented service, a **connection** is **first established** between the sender and the receiver.
- b. Data are transferred. At the end, the **connection is released**.
- c. TCP and SCTP are connection-oriented protocols.

# 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.
- ❖ In the Internet, there are two common different transport layer protocols, as we have already mentioned. **UDP is connectionless and unreliable; TCP and SCTP are connection-oriented and reliable protocols**. These three can respond to the demands of the application layer programs.

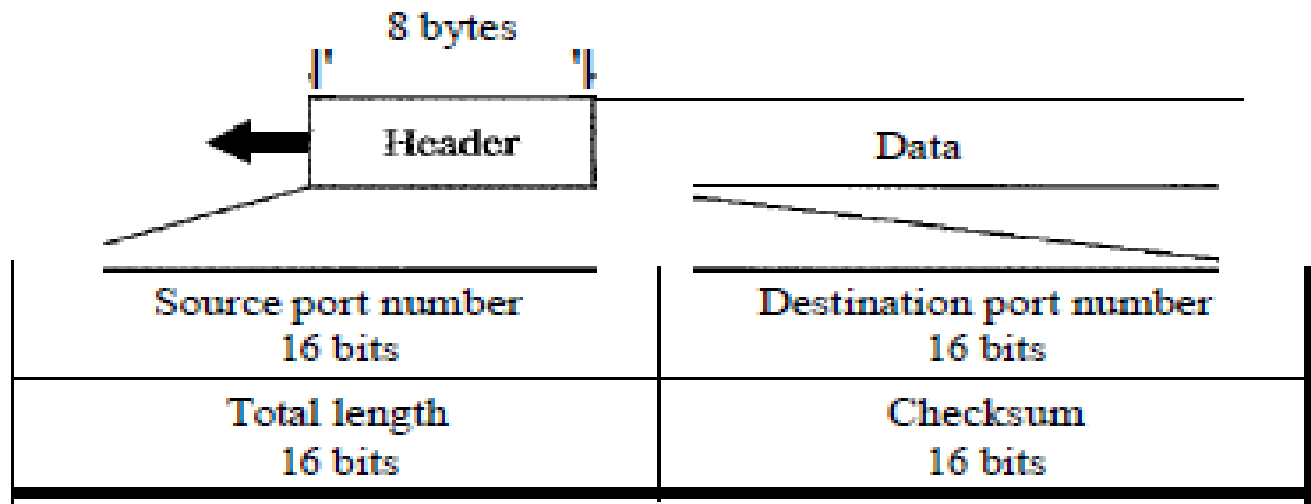


# USER DATAGRAM PROTOCOL (UDP)

- 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.
- Also, it performs very limited error checking. **If UDP is so powerless, why would a process want to use it?**
- With the disadvantages come some 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.

## Contd.

- 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.
- **Destination port number**. This is the port number used by the process running on the destination host.
- **Length**. This is a 16-bit field that defines the total length of the user datagram, header plus data.



## Contd.

- UDP does not perform:
  - ❖ Flow control
  - ❖ Error control
  - ❖ Connection control
    - All these functions are done by the processes (application layer programs) using UDP
  - ❖ UDP is **not capable of segmenting and reassembling frames** and **does not implement sequence numbers**
- But UDP, like TCP, performs:
  - ❖ Service point addressing
- UDP can transmit only **small portions of data** at a time because it is not capable of segmenting and reassembling frames and does not implement sequence numbers

## Use of UDP

1. 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
2. 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.
3. UDP is a suitable transport protocol for **multicasting**. Multicasting capability is embedded in the UDP software but not in the TCP software.
4. UDP is used for management processes such as **SNMP**.
5. UDP is used for some **route updating** protocols such as Routing Information Protocol (RIP).

# Transmission Control Protocol (TCP)

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

## TCP Services

1. ***Process-to-Process Communication:-*** Like UDP, TCP provides process-to-process communication using port numbers.
2. ***Stream Delivery Service***
3. ***Full-Duplex Communication***
4. ***Connection-Oriented Service***
5. ***Reliable Service***

## Stream Delivery Service

- TCP, unlike UDP, is a **stream-oriented protocol**.
- In UDP, a process (an application program) sends messages, with predefined boundaries, to UDP for delivery.
- UDP adds its own header to each of these messages and delivers them to IP for transmission.
- Each message from the process is called a user datagram and becomes, eventually, **one IP datagram**.
- **Neither IP nor UDP recognizes any relationship between the datagrams.**
- Each application using UDP must send small data to fit into one user datagram as it is. (UDP does not segment/reassemble)

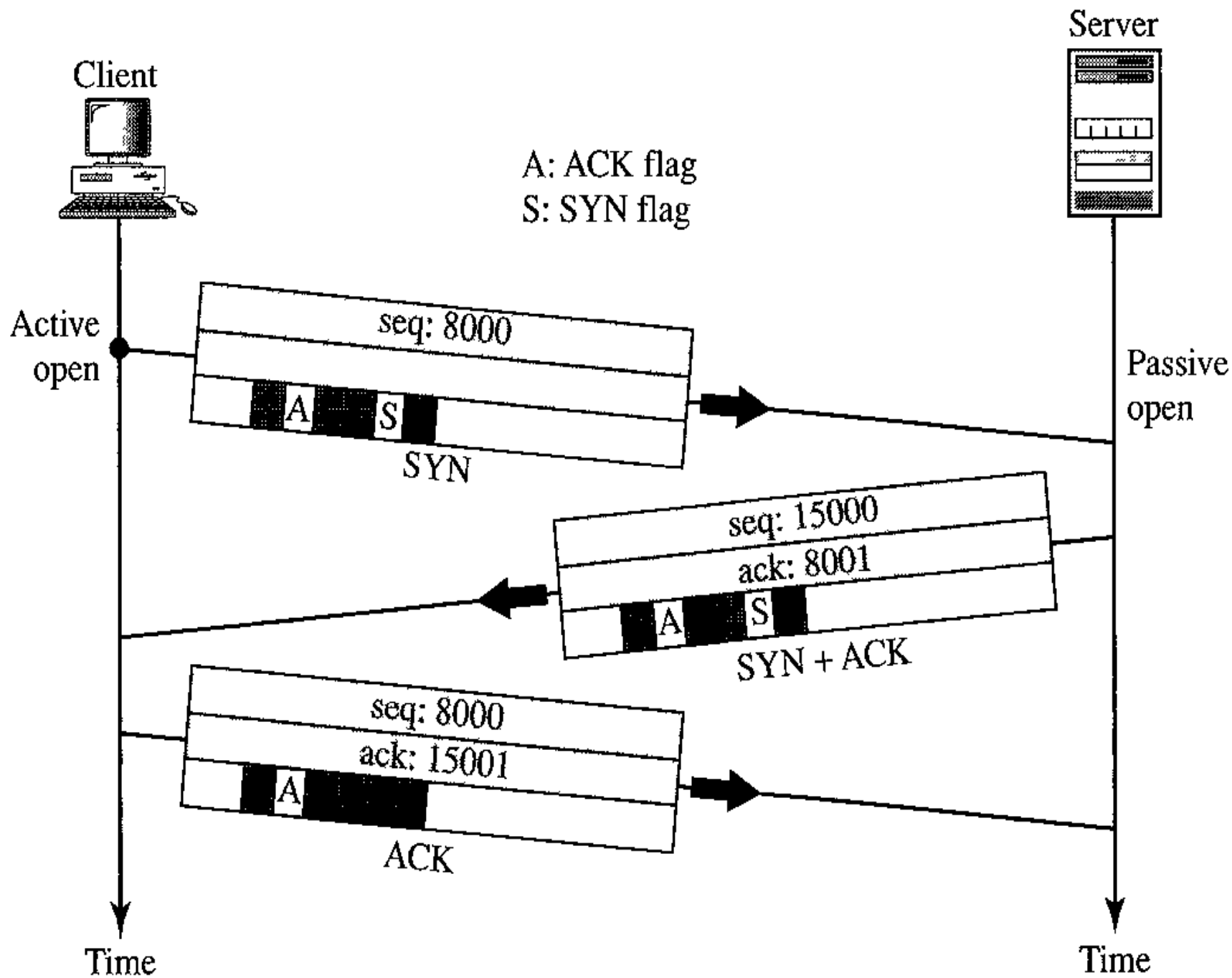
## Contd.

- TCP, on the other hand, 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 (Fig. next slide)



# Connection-Oriented Service

- 🔊 TCP, unlike UDP, 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 occurs:
  - 1. The two TCPs establish a connection between them.**
  - 2. Data are exchanged in both directions.**
  - 3. The connection is terminated.**
- 🔊 Note that this is a **virtual connection**, not a physical connection. The TCP segment is encapsulated in an **IP datagram** and can be sent out of order, or lost, or corrupted, and then resent.
- 🔊 Each may use a **different path to reach the destination**. There is no physical connection.
- 🔊 TCP creates a stream-oriented environment in which it accepts the responsibility of delivering the bytes in order to the other site.



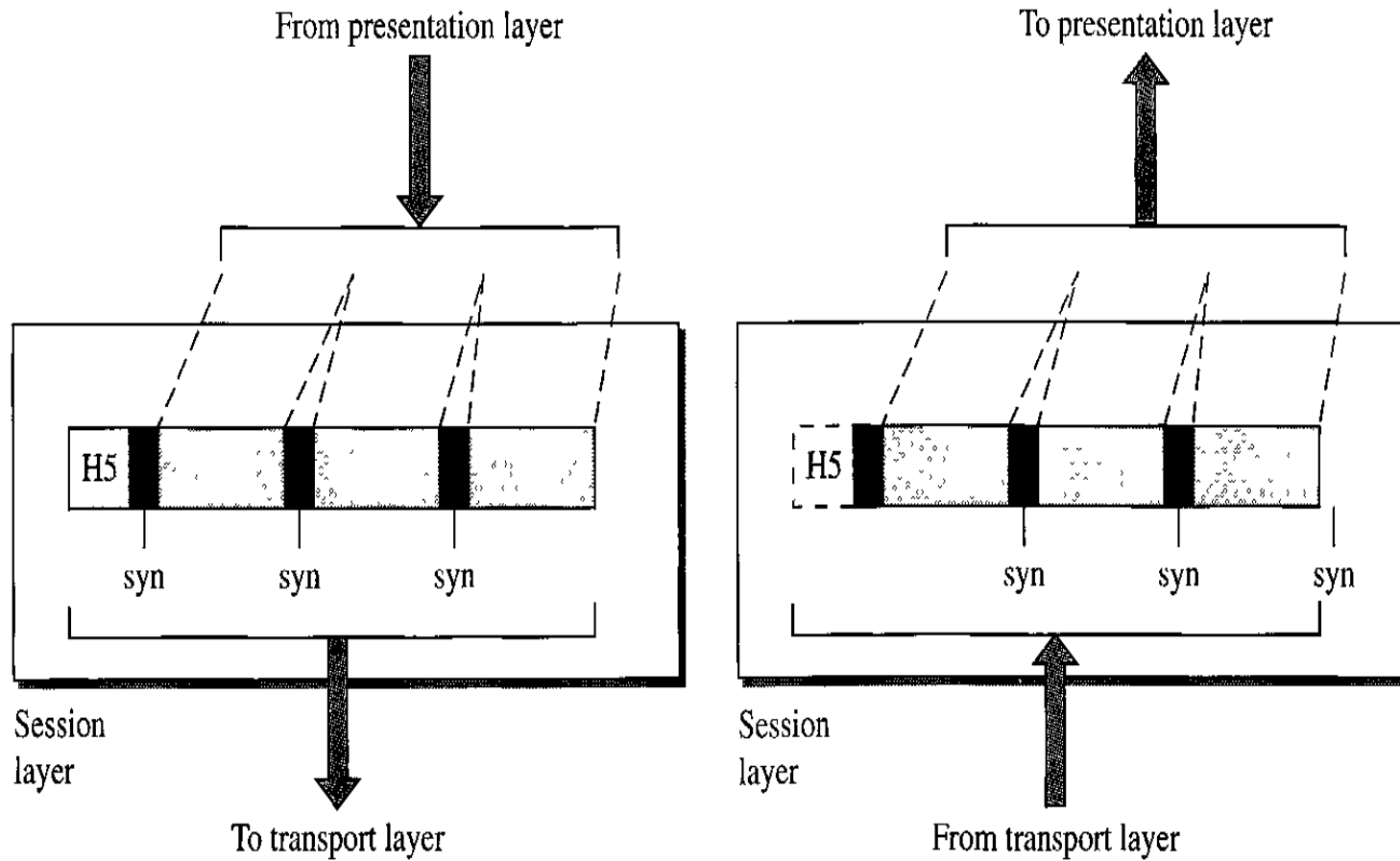
# Session Layer

- The lowest four layers of the OSI model (Physical, Data link, Network, and Transport) provide the means for the **reliable exchange** of data and provide a fast data service.
- For example, a **remote terminal** access application might require a **half-duplex dialogue**.
- A transaction-processing application might require **checkpoints** in the data-transfer stream to permit **backup and recovery**.
- A message processing application might require the ability to **interrupt a dialogue in order to prepare a new portion of a message** and later to resume the dialogue where it was left off.

## Contd.

- ⦿ All these capabilities could be embedded in specific applications at **layer 7**.
- ⦿ However, because these types of dialogue-structuring tools have widespread applicability, it makes sense to organize them into a **separate layer**: the session layer.
- ⦿ The session layer provides the mechanism for **controlling the dialogue** between applications in end systems.
- ⦿ In many cases, there will be little or no need for session-layer services, but for some applications, such services are used.

# Contd.



# The key services provided by the session layer include

- The primary job of session layer protocols is to provide the means necessary to **set up, manage, and end sessions**
- 1. **Dialogue discipline.** This can be two-way simultaneous (full duplex) or two way alternate (half duplex) communication between processes.
- 2. **Grouping.** The flow of data can be marked to define groups of data.
  - For example, if a retail store is transmitting sales data to a regional office, the data can be marked to indicate the end of the sales data for each department; this would signal the host computer to finalize running totals for that department and start new running counts for the next department.

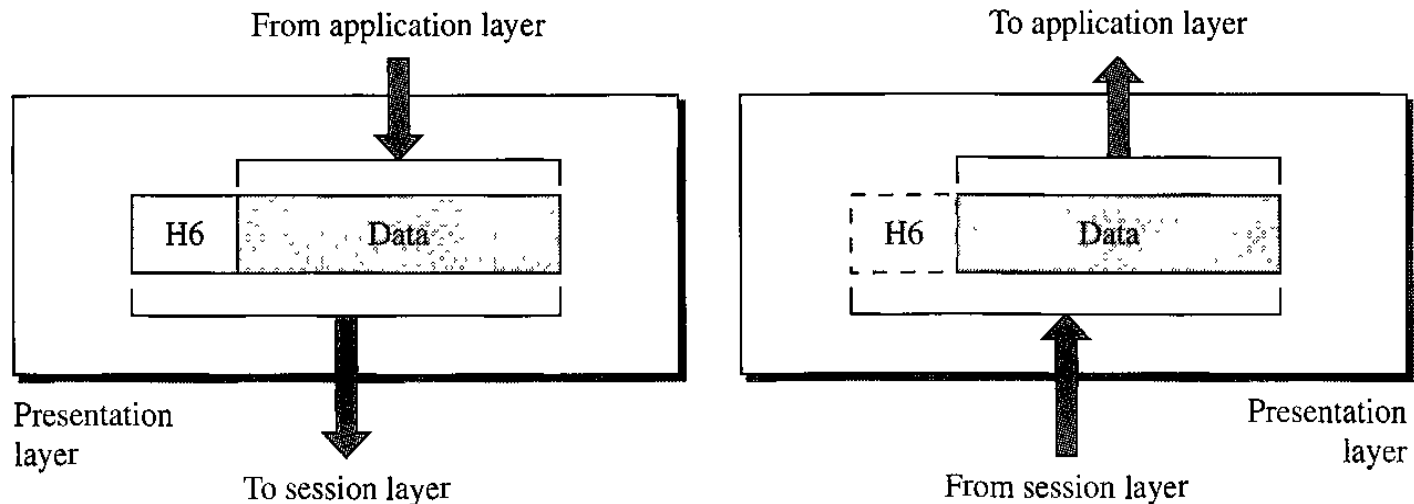
## Contd.

**3. Recovery.** The session layer can provide a check pointing mechanism, so that if a failure of some sort occurs between checkpoints, the session entity can retransmit all data since the last checkpoint, as in the example shown below:

- For example, if a system is sending a file of 2000 pages, it is advisable to insert checkpoints after every 100 pages to ensure that each 100-page unit is received and acknowledged independently.
- In this case, if a crash happens during the transmission of page 523, the only pages that need to be resent after system recovery are pages 501 to 523. Pages previous to 501 need not be resent.

# Presentation Layer

- ❖ The presentation layer is concerned with the **syntax and semantics** of the information exchanged between two systems.





# Specific responsibilities of the presentation layer :

1. **Translation:-** The processes (running programs) in two systems are usually exchanging information in the form of character strings, numbers, and so on.
  - The information must be changed to bit streams before being transmitted. Because different computers use different encoding systems, the presentation layer is responsible for interoperability between these different encoding methods.
  - The presentation layer at the sender changes the information from its **sender-dependent format into a common format**.
  - The presentation layer at the receiving machine changes the common format into its receiver-dependent format.

## Contd.

2. **Encryption.** To carry sensitive information, a system must be able to ensure privacy.
  - 📢 Encryption means that the sender transforms the original information to another form and sends the resulting message out over the network.
  - 📢 Decryption reverses the original process to transform the message back to its original form.
3. **Compression.** Data compression reduces the number of bits contained in the information.
  - 📢 Data compression becomes particularly important in the transmission of multimedia such as text, audio, and video.

# Application Layer

- 🔊 The application layer enables the user, whether human or software, to **access the network**.
- 🔊 It provides **user interfaces** and **support for services** such as **electronic mail, remote file access and transfer, shared database management**, and other types of distributed information services.
- 🔊 Application layer is where users actually communicate to the computer.
  - Take the case of Internet Explorer (IE).
- 🔊 It is also responsible for identifying and establishing the availability of the intended communication partner

## Contd.

- Typical application layer protocols
  - Domain Name System (DNS)
  - Hyper Text Transfer Protocol (HTTP)
  - File Transfer Protocol (FTP)
  - E-mail (SMTP,POP,IMAP)

### 1. Domain Name System (DNS)

- ✉ Thousands of servers, installed in many different locations, provide the services we use over the Internet.
- ✉ Each of these servers is assigned a unique IP address
- ✉ It would be impossible to remember all of the IP addresses

## Contd.

- ✉ DNS provides a way for hosts to use this name to request the IP address of a specific server.
- ✉ DNS names are registered and organized on the Internet within specific high level groups, or domains.
  - Some of the most common high level domains on the Internet are .com, .edu, and .net
- ✉ A DNS server contains a table that associates hostnames in a domain with corresponding IP addresses

## 2. Web client and web server

- ✎ A web client first receives the IP address of a web server from DNS server
- ✎ Then the client browser uses that IP address and port 80 to request web services
- ✎ This request is sent to the server using the Hypertext Transfer Protocol (HTTP)
- ✎ The information content of a web page is encoded using specialized 'mark-up' languages.
  - ✎ E.g. HTML (Hypertext Mark-up Language)
- ✎ Many different web servers and web clients from many different manufactures work together seamlessly because of HTTP and HTML

### 3. File Transfer Protocol (FTP)

- ✉ FTP is another common service used across the Internet that allows users to transfer files
- ✉ A host running FTP client software can access an FTP server to perform various file management functions including file uploads and downloads
- ✉ FTP service uses two different ports to communicate between client and server
  - Requests to begin an FTP session are sent to the server using destination port 21.
  - Once the session is opened, the server will change to port 20 to transfer the data files
- ✉ FTP client software is built into computer operating systems and into most web browsers

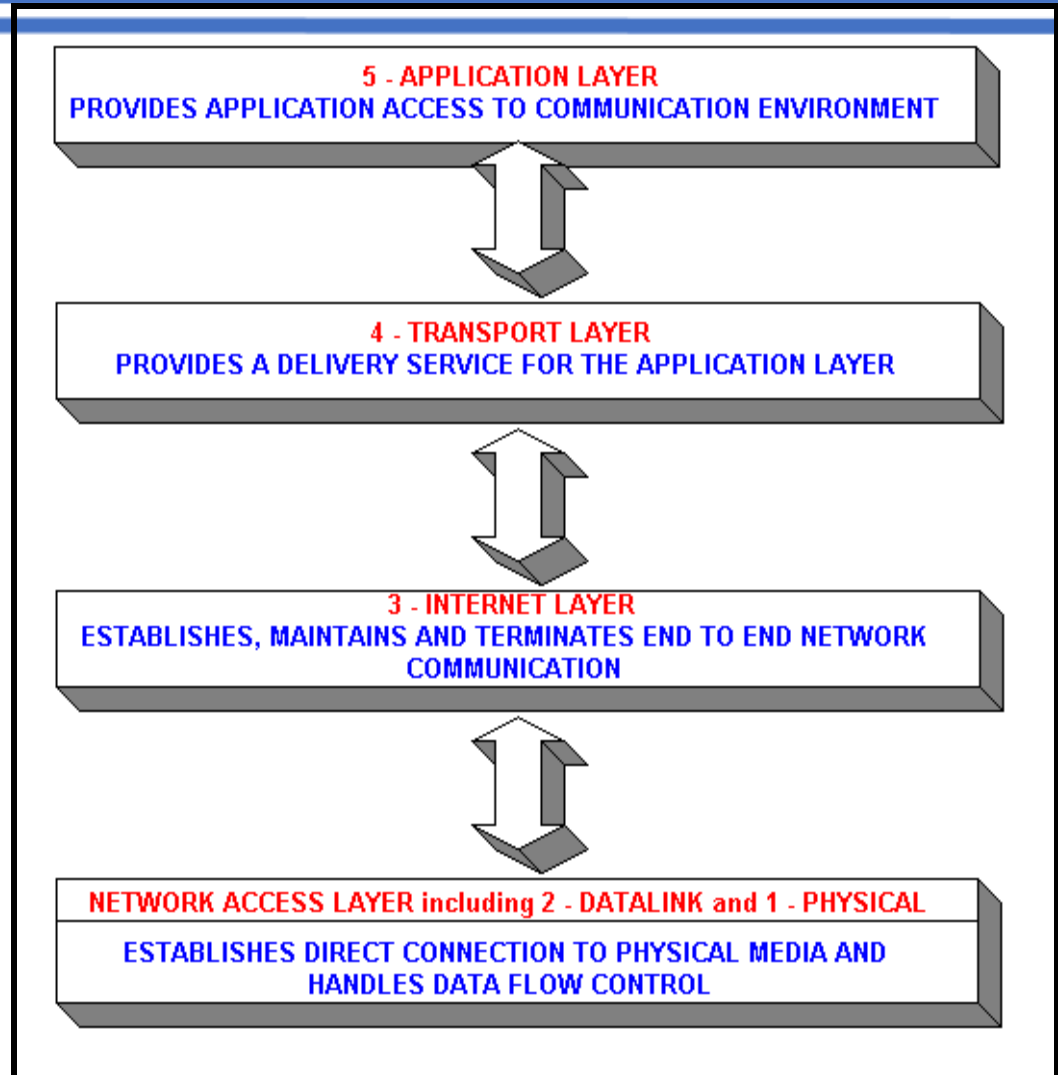
## 4. Email Servers (SMTP, POP3 and IMAP4)

- ✉ Each mail server receives and stores mail for users who have mailboxes configured on the mail server
- ✉ Each user with a mailbox must then use an email client to access the mail server and read these messages
  - ❖ Mailboxes are identified by the format: **user@company.domain**
- ✉ Three application protocols used in processing email include
  1. **Simple Mail Transfer Protocol (SMTP)**:- to send mail from client to server or server to server
  2. **Post Office Protocol (POP3)**:- to download email from server to client, and the server deletes the mail
  3. **Internet Message Access Protocol (IMAP4)**:- to download email from server to client, and the server does not delete (keeps) the mail

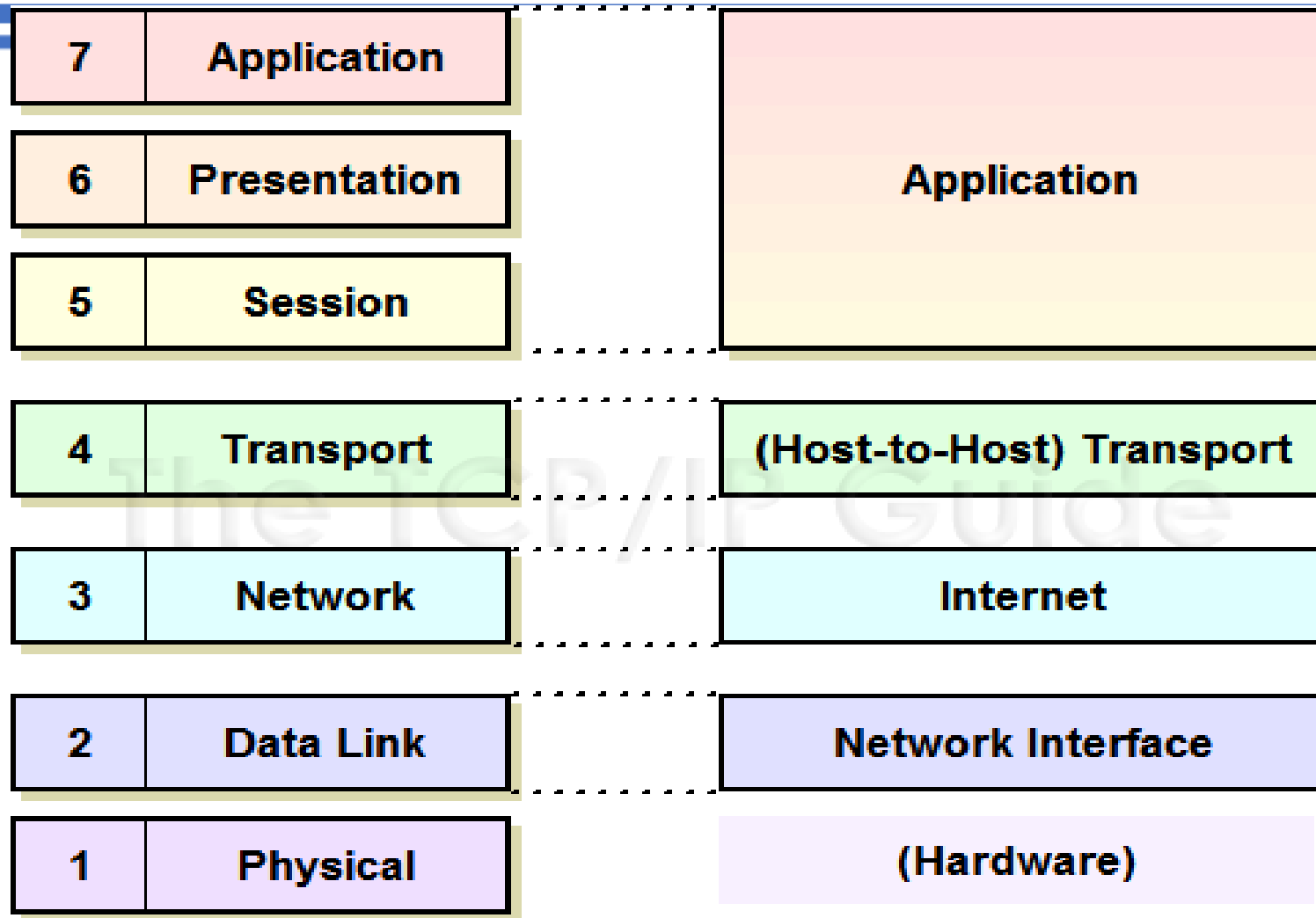


# TCP/IP Network Model Layers

- ✧ As with the OSI model, the TCP/IP suite uses a layered model.
- ✧ TCP/IP model has **four or five** - depending on who **you talk to and which books you read!**
- ✧ Some people call it a four layer suite - **Application, Transport, Internet** and **Network Access**, others split the **Network Access** layer into its **Physical** and **Datalink** components.



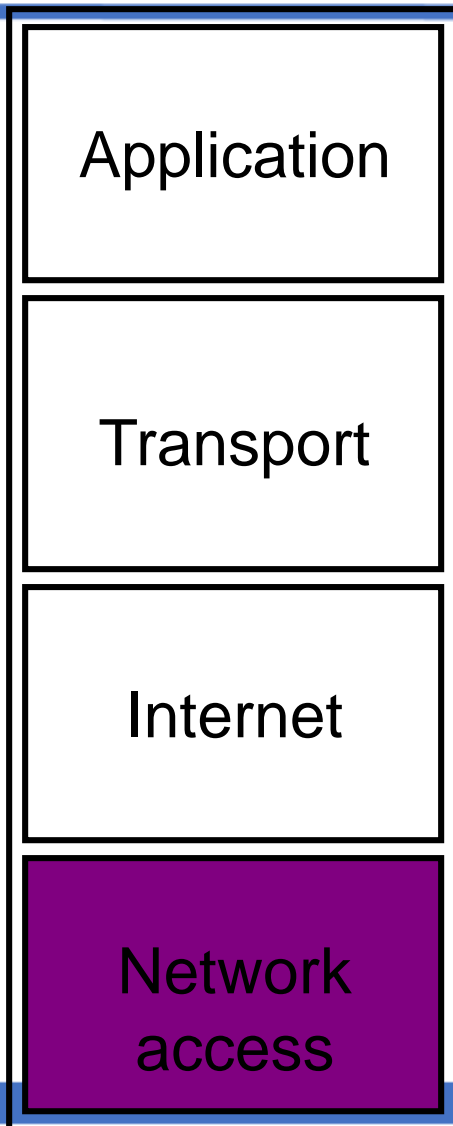
# OSI Model Vs. TCP/IP Model Layers



**OSI Model**

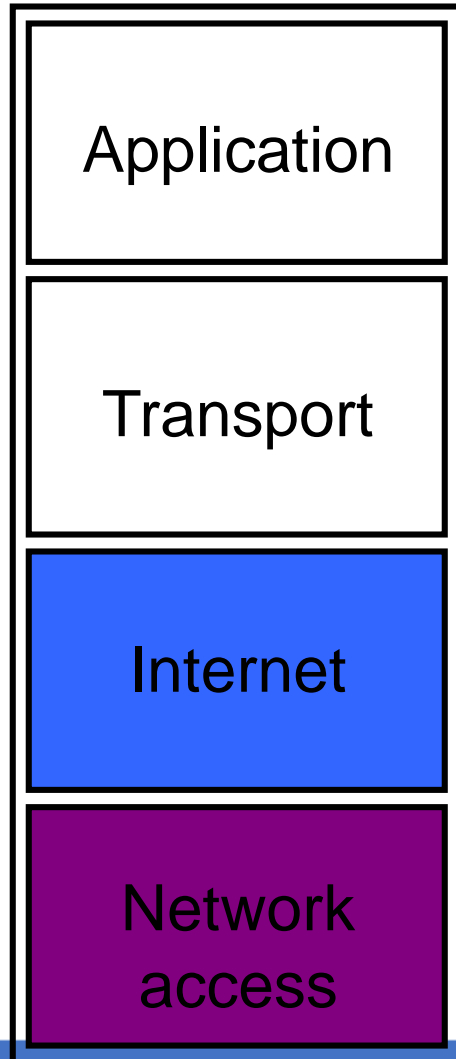
**TCP/IP Model**

# TCP/IP Layers- What does each layer do?



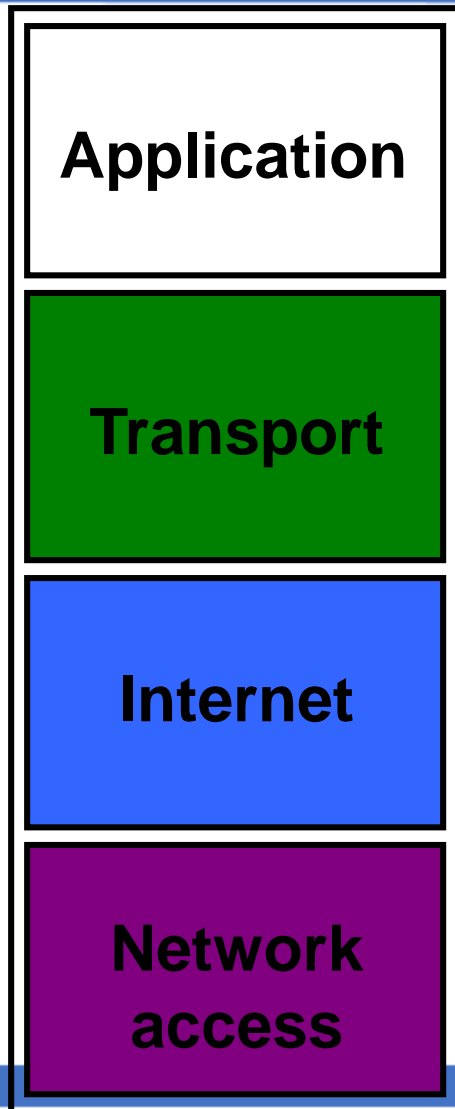
- Network access layer is the combination of **data link** and **physical layers** and it deals with **pure hardware** issues (wires, satellite links, network interface cards, etc.)
- Access methods such as **CSMA/CD** (carrier sensed multiple access with collision detection)
- Ethernet exists at the network access layer - its hardware operates at the physical layer and its medium access control method (CSMA/CD) operates at the data link layer.

# TCP/IP Layers- What does each layer do?



- ❖ The Internet layer is responsible for the **routing and delivery** of data across networks.
- ❖ It allows communication across networks of the **same and different types** and carries out translations to deal with **dissimilar data addressing schemes**.
- ❖ **IP** (Internet Protocol) and **ARP** (Address Resolution Protocol) are both to be found at the Internet layer.

# TCP/IP Layers- What does each layer do?



- The Transport layer is similar to the OSI transport model, but with elements of the OSI session layer functionality.
- The two protocols found at the transport layer are:
  - A. **TCP** (Transmission Control Protocol): reliable, connection-oriented protocol that provides error checking and flow control through a virtual link that it establishes and finally terminates. Examples include FTP and Email
  - B. **UDP** (User Datagram Protocol): unreliable, connectionless protocol that not error check or offer any flow control. Examples include SNMP

# TCP/IP Layers- What does each layer do?

## Application

- 📢 The Application layer is broadly equivalent to the **application, presentation and session layers** of the OSI model.

## Transport

- 📢 It gives an application access to the communication environment.

## Internet

- 📢 Some examples: **FTP, HTTP, SMTP**

## Network access

