Chapter Four

The OSI and TCP/IP Models

Communication and Layer Architecture

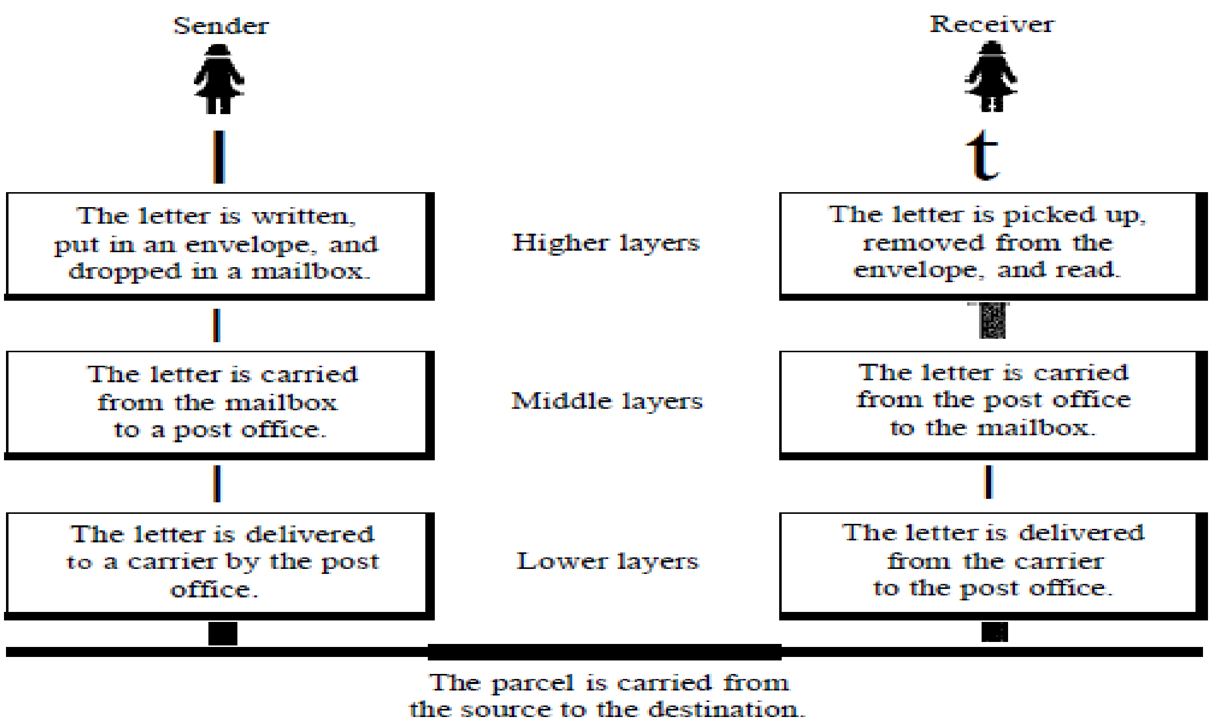
- A network is a combination of **hardware** and **software** that sends data from one location to another.
- The **hardware** consists of the **physical equipment** that **carries signals** from **one point** of the **network** to **another**.
- The **software** consists of **instruction sets** that 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**

Communication and Layer Architecture----

- 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**.
  - ✓ As an **example**, 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.**



The need for Standards

- **Without** agreed on common **standards**, **devices** made by **different manufacturers** or **vendors** would **not** be able to **communicate**.

✓ A **protocol** is a **set** of **rules** that determines how **two devices** will **communicate**.

- **Networking standards** define the **rules** for **data communications** that are needed for **interoperability** of **networking technologies** and **processes**.

The need for Standards-----

- **Standards** help in

  creating and **maintaining open markets** and

  allow different vendors to **compete** on the basis of the **quality** of their **products** while being **compatible** with **existing market products**.

- Over the past years many of the **networks** that were **built** used different **hardware** and **software implementations**,

  as a result they were **incompatible** and it became **difficult** for **networks** using **different specifications** to **communicate** with each other.

The need for Standards-----

- To address the **problem** of **networks** being **incompatible** and **unable** to **communicate** with **each other**,
  the **International Standardisation for Organisation** (ISO) researched various **network schemes**.
- The **ISO** recognised there was a need to create a **NETWORK MODEL** that would help **vendors** **create interoperable network implementations**.

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**.

Contd.

- **Separating communication** functions adds **flexibility**, for example, we do **not need to modify** the entire **host software** to include more **communication devices**.
- **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**.
- The term **"open"** denotes the **ability** to **connect** any **two systems** which conform to the **reference model** and associated **standards**.

Open Systems Interconnection (OSI)-----

- 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.**

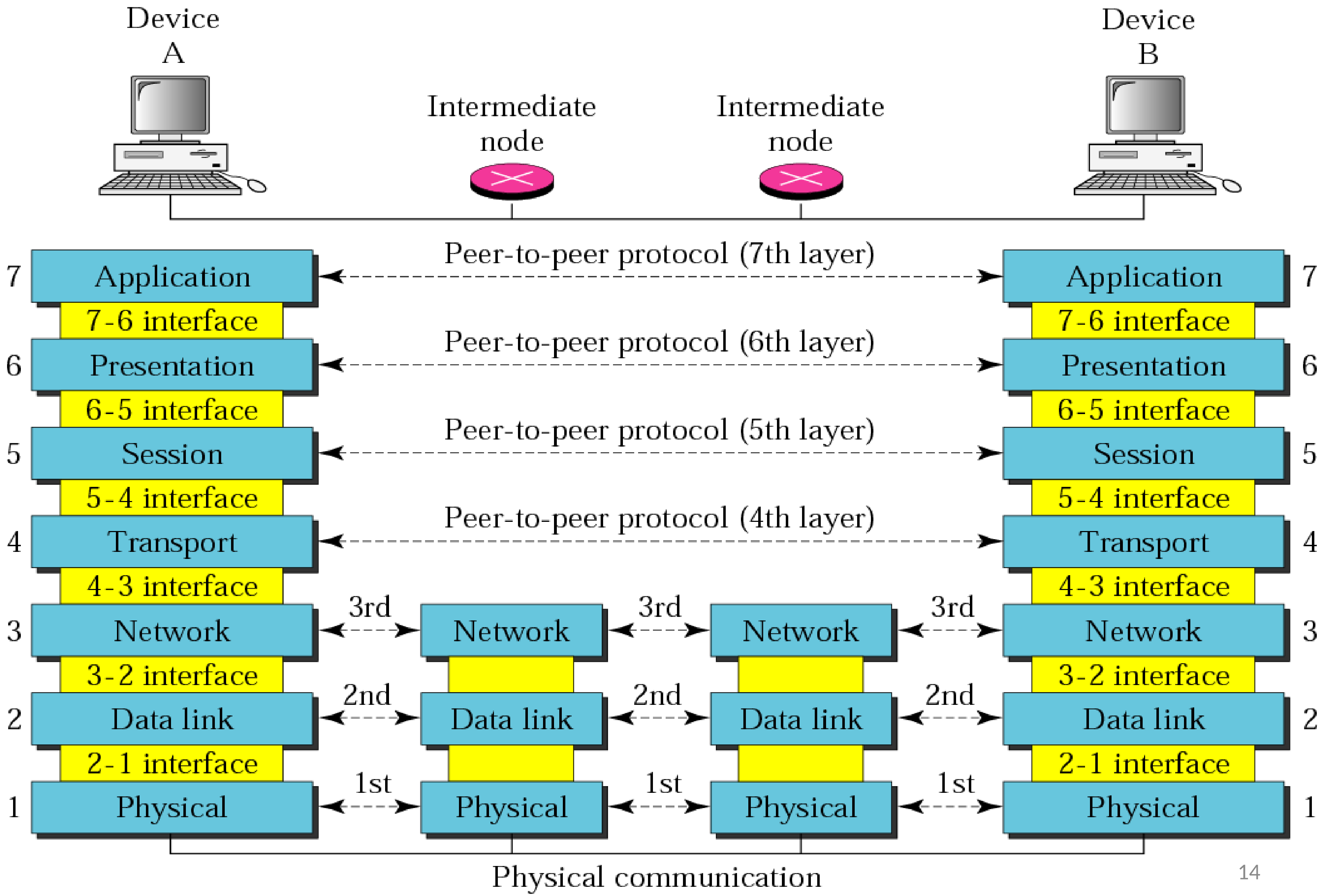
The Seven Layers

Layer 7	Application	• File, print, message, database, and application services
Layer 6	Presentation	• Data encryption, compression, and translation services
Layer 5	Session	• Dialog control
Layer 4	Transport	• End-to-end connection
Layer 3	Network	• Routing
Layer 2	Data Link	• Framing
Layer 1	Physical	• Physical topology

Contd.

- 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.


Device A
Device B
Intermediate node
Intermediate node
Peer-to-peer protocol (7th layer)
Peer-to-peer protocol (6th layer)
Peer-to-peer protocol (5th layer)
Peer-to-peer protocol (4th layer)
7 Application
7-6 interface
6 Presentation
6-5 interface
5 Session
5-4 interface
4 Transport
4-3 interface
3 Network
3-2 interface
2 Data link
2-1 interface
1 Physical
3rd
2nd
1st
Network
Data link
Physical
Physical communication

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**.

Contd.

- 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.
- **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**.

Cont----

- 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.**

Peer-to-Peer Processes------

- 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.

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**.

Interfaces Between Layers---

- 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)

Cont----

- **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.

Contd.

- 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**

- The f**igure** 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.

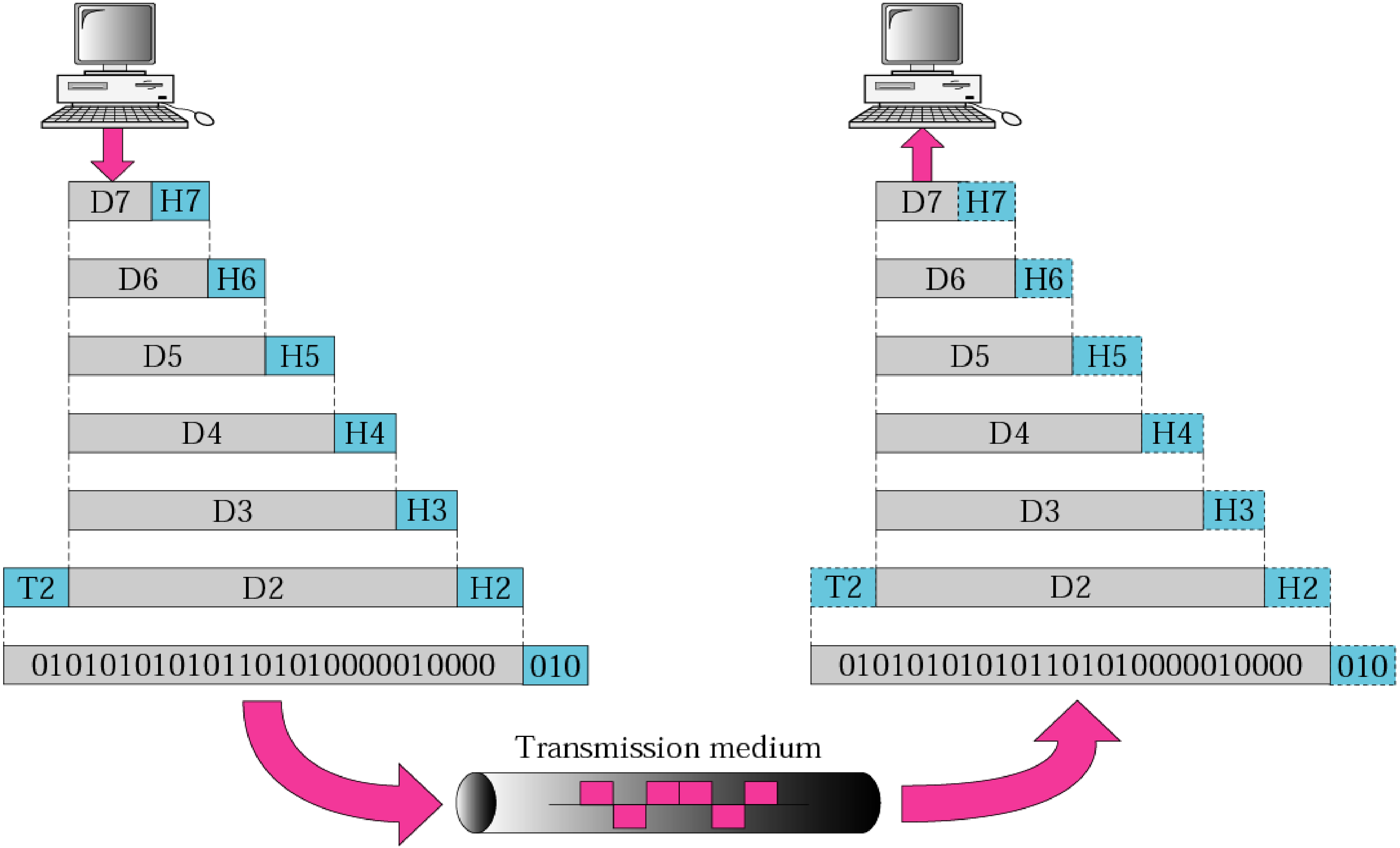
✓ The **process** starts at **layer 7** (the **application layer**), then **moves** from **layer** to **layer** in **descending sequential order**

Contd.

- 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 pac**ket at **level 6** is **encapsulated** in a **packet** at **level 5**, and so on.
- In other words, 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.**

Benefits of OSI Model

1. Provide a wide variety of choice

- Customers have a wide variety of choice since **software/ hardware** from **different manufactures** work together in **harmony**.
- **OSI model** can fit to any **compatible software/hardware** from different **users** in other parts of the **world**.

2. It does not rely on a specific operating system

- **OSI** is convenient since **errors** are dealt with at **each level**, as **different levels** operate **automatically independent** of **each other**.

Benefits of OSI Model -----

- This makes it **easier** to **troubleshoot problems** that may arise at each stage, by **separating** the **networks** into **small manageable pieces**.

3. To understand the common terms used in networking

- **OSI model** also help the **user** to **understand different networking terms** and **functional relationship applied** on **multiple networks**.
- In addition, the **user** also **understand** how **new technologies** are **developed** in the **existing networks**.

Benefits of OSI Model-----

4. Interprets product functionality at each stage

- The **OSI model** simply uses **different stages** of **functionality**.
- For instance, **each stage** has **specific functions** to **ensure** all **networks** operate without **technical hitches**.
- Also, each **layer** has it **own interface specifications** and a well-defined connector.

5. Encrypt data for security purposes

- **Decryption** and **encryption services** are also available for **security purposes**.
- **Expansion** and **compression** of messages is simplified to **ensure** it **travels** from one **system** to **another efficiently**.

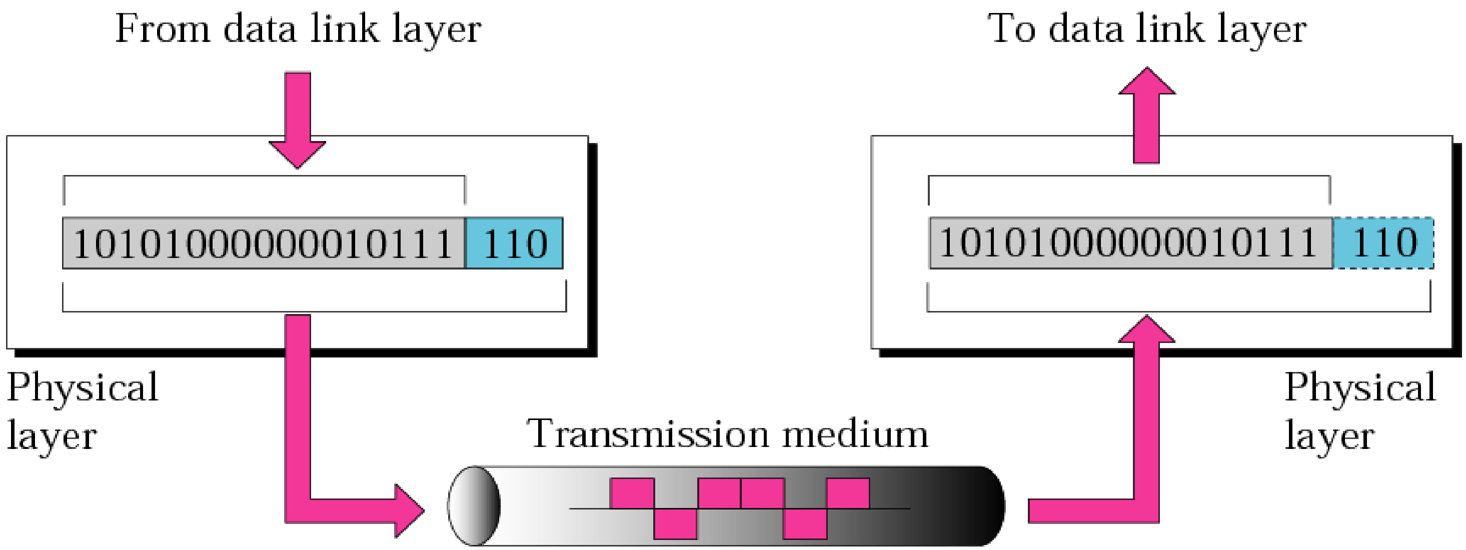
Benefits of OSI Model------

6. It is easier to add multiple network models

- The **OSI model** is designed in such a way that user further **extend new protocols** within the **process**.
- This means you can **use additional layered architecture** other than the existing one.
- Due to its **complexity**, **poor performance** can be obtained in **day** to **day applications**, thereby it requires **great technical know-how**

1. 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.**

1. Physical Layer

- **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**

Contd.

- 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).

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.

Contd.

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**.

5. Line configuration.

- The **physical layer** is concerned with the **connection** of **devices** to the **media** (**point – to – point or multipoint**).

Contd.

6. Physical topology.

- The **physical topology** defines how **devices** are connected **to make a network.**

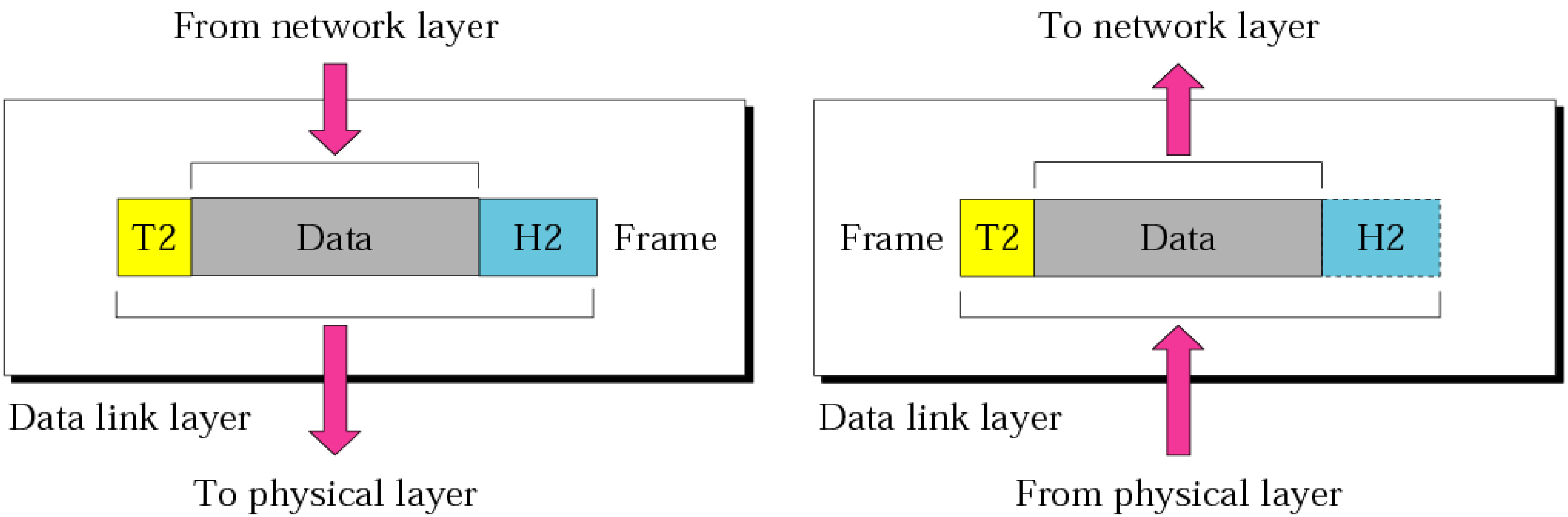
7. Transmission mode.

- The **physical layer** also defines the **direction of transmission** between **two devices**: **simplex**, **half-duplex**, or **full-duplex**.

2. 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 **packets 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**.

Contd.

- 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**.

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**.

Contd.

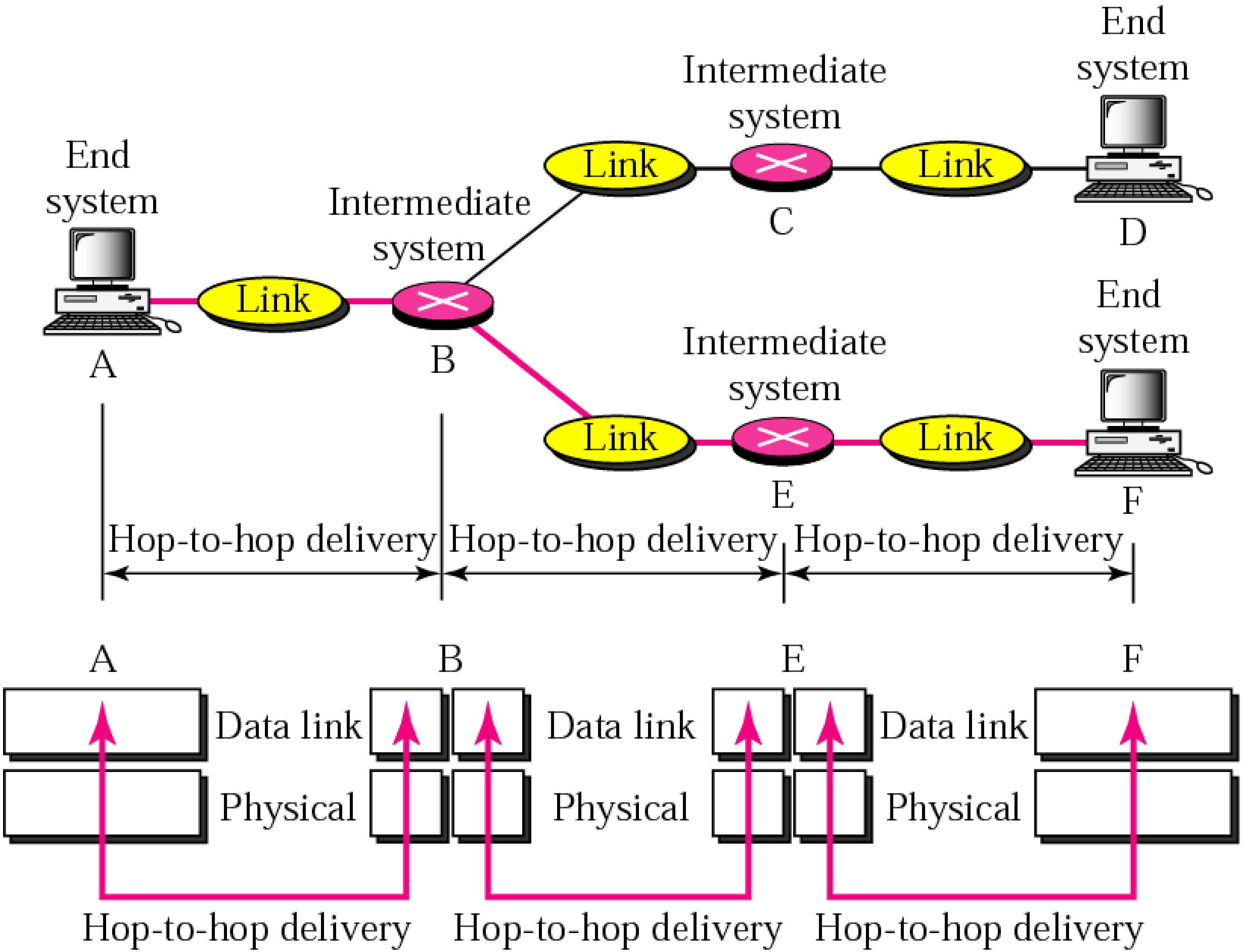
4. Error control

- 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**.

Contd.

- 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.

Cont----

- **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.
- **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**.

Contd.

- **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 size Framing**
- **Variable size Framing**

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

1. Character-Oriented Protocols

- In a **character-oriented protocol**, data to be carried are **8-bit characters** from a **coding system** such as **ASCII**.
- The **header**, which normally carries the **source** and **destination addresses** and other **control information**, and the **trailer**, which carries **error detection or error correction redundant bits,** are also **multiples** of **8 bits**.

Character-Oriented Protocols-----

- To **separate** one **frame** from the **next**, an **8-bit (1-byte) flag** is **added** at the **beginning** and the **end** of a **frame**.
- The **flag**, composed of **protocol-dependent** **special characters**, **signals** the **start** or **end** of a **frame**.
- **Character-oriented** **framing** was popular when **only text** was **exchanged** by the **data link layers**.

✓ The **flag** could be **selected** to be any **character not** used for **text communication**.

- Now, however, we **send** other **types** of **information** such as **graphs**, **audio**, and **video**.

Character-Oriented Protocols-----

- Any **pattern** used for the **flag** could also be **part** of the **information**.

✓ If this happens, the **receiver**, when it **encounters** this **pattern** in the **middle** of the **data**, thinks it has reached the **end** of the **frame**.

- To **fix** this **problem**, a **byte-stuffing strategy** was **added** to **character-oriented framing**.
- In **byte stuffing** (or **character stuffing**),

a special byte is **added** to the **data section** of the **frame** when there is a **character** with the **same pattern** as the **flag.**

- The **data section** is **stuffed** with an **extra byte**.

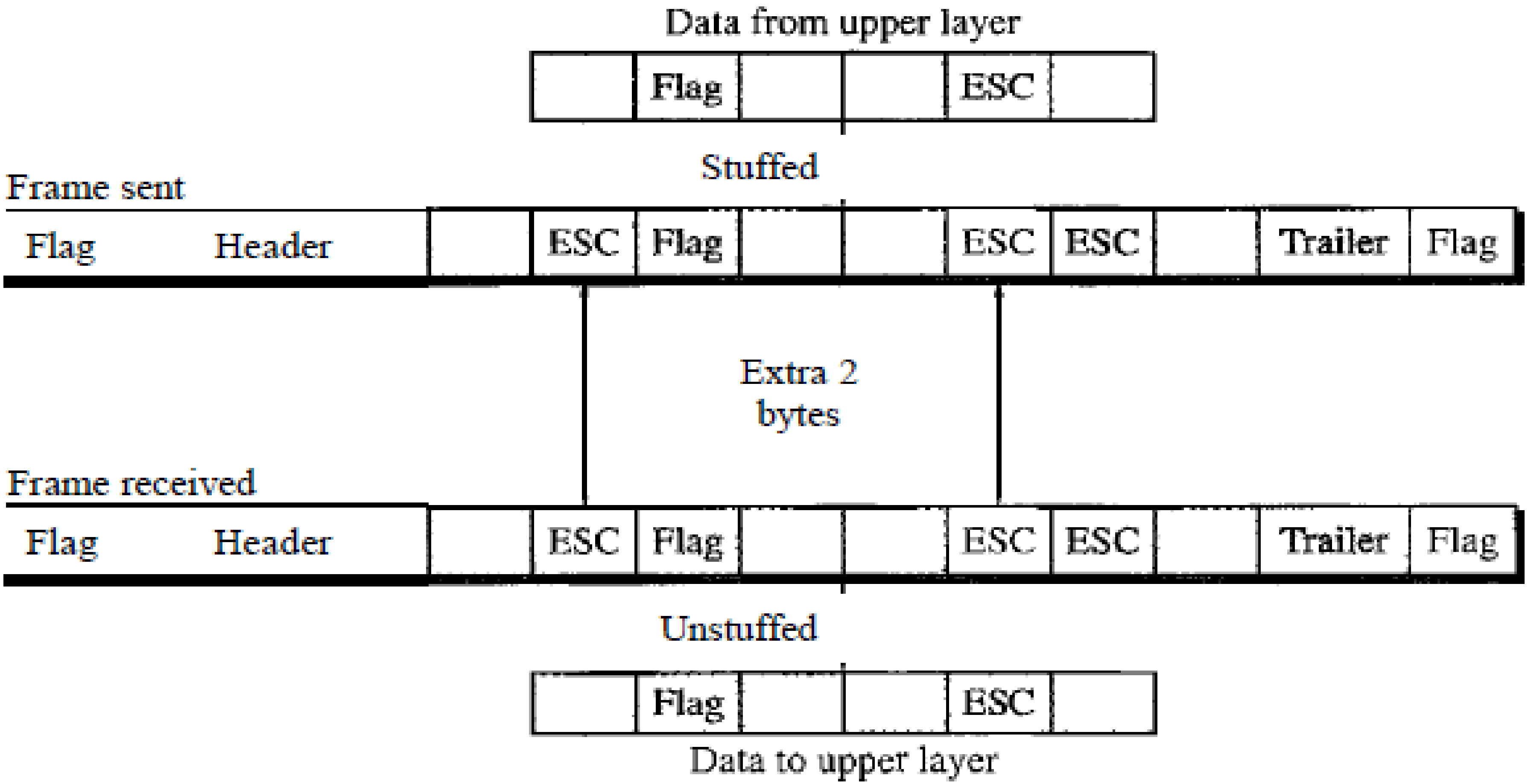
Character-Oriented Protocols-----

- This **byte** is usually called the **escape character (ESC)**, which has a **predefined bit pattern**.
- **Whenever** the **receiver** encounters the **ESC character**, it **removes** it from the **data section** and **treats** the **next character** as **data**, **not** a **delimiting flag**.
- **Byte stuffing** by the **escape character** allows the **presence** of the **flag** in the **data section** of the **frame**, but it **creates another problem**.
- **What happens** if the **text contains** one or more **normal escape characters followed** by a **flag?**

Character-Oriented Protocols-----

- The **receiver removes** the **escape character**, but **keeps** the **flag**, which is **incorrectly interpreted** as the **end** of the **frame**.
- To **solve** this **problem**, the **escape characters** that are **part** of the **text** must also be marked by **another escape character**.
- In other words, if the **escape character** is **part** of the **text**, an **extra one** is added to show that the **second one** is part of the **text**.

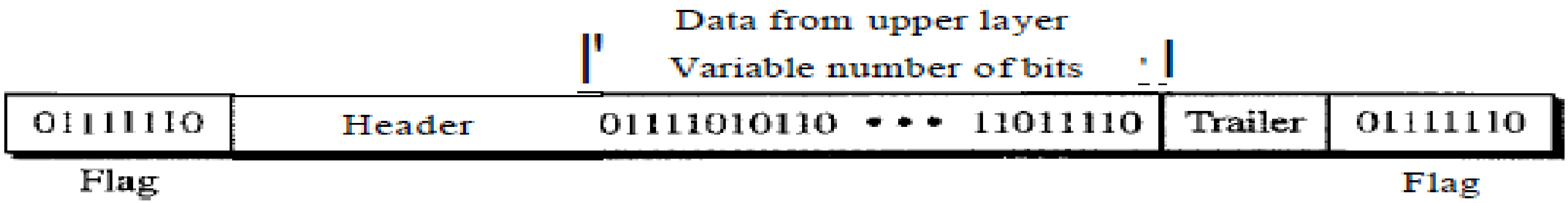
Contd.



- **Byte stuffing** is the **process** of **adding 1 extra byte** whenever there is a **flag** or **escape character** in the **text**.

2. Bit-Oriented Protocols

- In a **bit-oriented protocol**, the **data section** of a **frame** is a **sequence** of **bits** to be **interpreted** by the **upper layer** as **text**, **graphic**, **audio**, **video**, and so on.
- However, in **addition** to **headers** (and **possible trailers**), we still need a **delimiter** to separate one **frame** from the **other**.
- Most **protocols** use a **special 8-bit pattern flag 01111110** as the **delimiter** to define the **beginning** and the **end** of the **frame**, as shown in the figure.



Contd.

- This **flag** can create the same **type** of **problem** we saw in the **byte-oriented protocols**.
- That is, if the **flag pattern** appears in the **data**, we need to somehow **inform** the **receiver** that this is **not** the end of the **frame**.
- We do this by **stuffing 1 single bit (instead of 1 byte)** to **prevent** the **pattern** from looking like a **flag**.

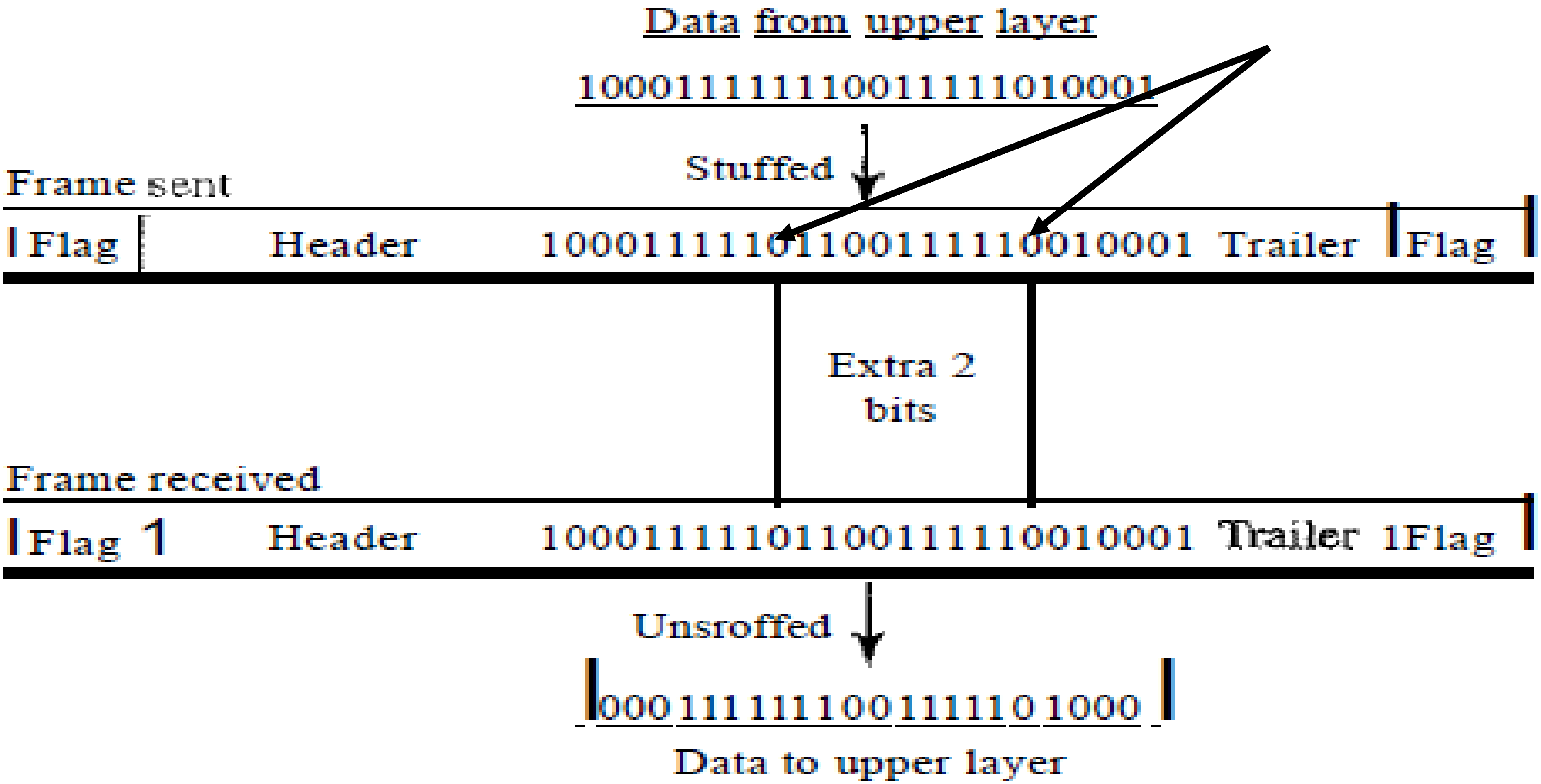
✓ The **strategy** is called **bit stuffing**.

- In **bit stuffing**, if a **0** and **five consecutive 1 bits** are **encountered**, an **extra 0** is **added**.

Contd.

- This **extra stuffed bit** is eventually removed from the **data** by the **receiver**.
- **Note** that the **extra bit** is **added** after **one 0** followed by **five 1s** regardless of the **value** of the **next bit.**
- This **guarantees** that the **flag field sequence** does **not** inadvertently appear in the **frame**.

Contd.



- **Bit stuffing** is the **process** of **adding one extra 0** whenever **five consecutive 1s** follow a **0** in the **data**, so that the **receiver** does **not mistaken** the **pattern 0111110** for a **flag**.

Contd.

- The above figure shows **bit stuffing** at the **sender** and **bit removal** at the **receiver**.
- Note that even if we have a **0 after five 1s**, we still **stuff** a **0**.

✓ The **0** will be removed by the **receiver**.

- This means that if the **flag** like **pattern 01111110** appears in the **data**, it will change to **011111010** (**stuffed**) and is **not mistaken** as a **flag** by the **receiver**.
- The **real flag 01111110** is **not stuffed** by the **sender** and is recognized by the **receiver** as a **flag**.

2. Flow Control

- Flow control **coordinates** the amount of data that can be sent before **receiving** an acknowledgment and is one of the most **important duties** of the **data link layer**.
- In most **protocols**, **flow control** is a **set of procedures** that tells the **sender** how much data it can **transmit** before it must wait for an **acknowledgment** from the **receiver**.
- The flow of data must not be allowed to **overwhelm** the **receiver**.
- Any **receiving device** has a **limited speed** at which it can **process incoming data** and a **limited amount of memory** in which to **store incoming data**.

Contd.

- The **receiving device** must be able to **inform** the **sending device** before those limits are **reached** and to **request** that the **transmitting device** send fewer **frames** or **stop temporarily**.
- **Incoming data** must be **checked** and **processed** before they can be used.
- **The rate of such processing is often slower than the rate of transmission.**

Contd.

- For this reason, each **receiving device** has a **block of memory**, called a **buffer**, reserved for **storing incoming data** until they are **processed**.
- If the **buffer** begins to **fill up**, the **receiver** must be able to tell the **sender** to **halt transmission** until it is once again able to **receive**.

3. Media Access control

- The **data link layer** can further be divided in to **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**,

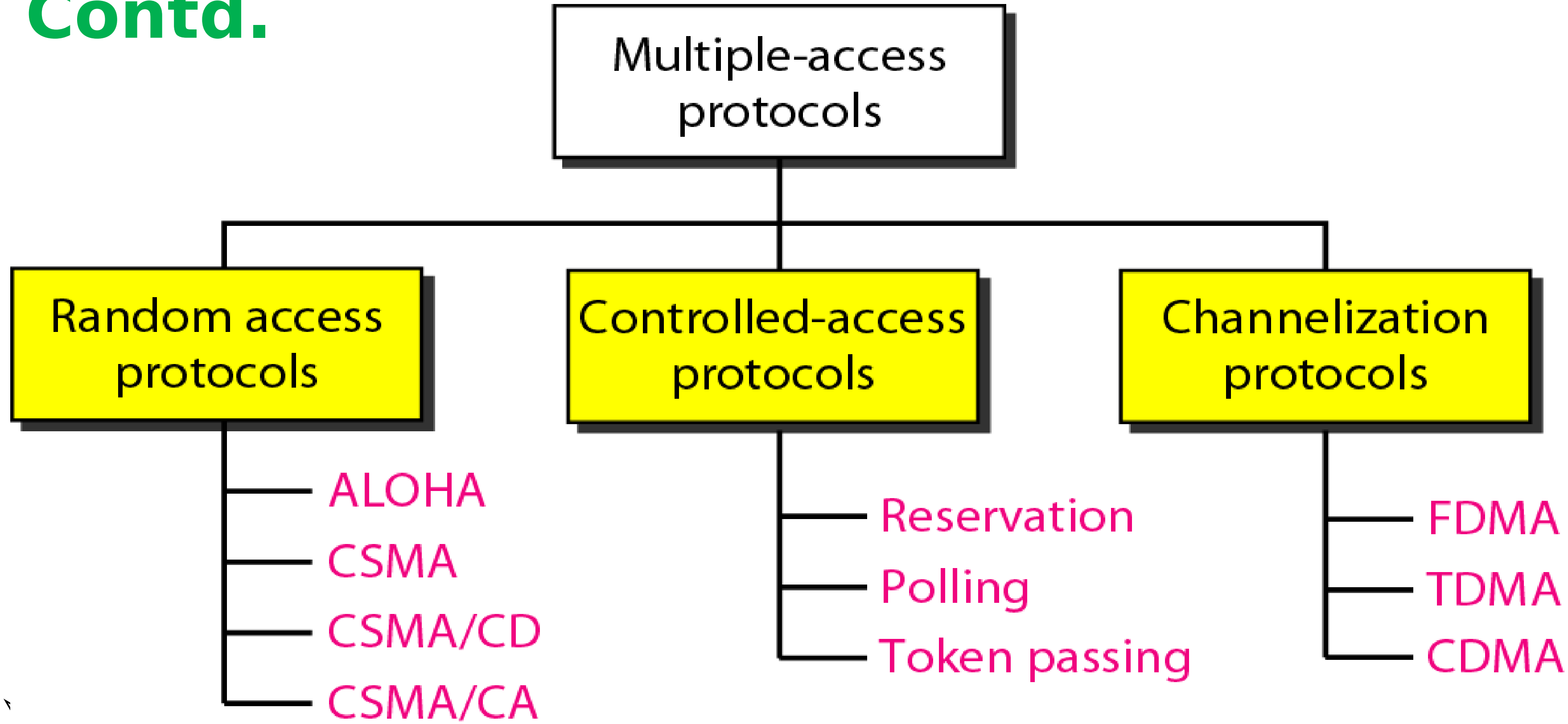
  we need a **multiple-access protocol** to coordinate access to the link.

3. Media Access control-----

- 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 **channelization protocols:**
  - Frequency Division Multiple Access (FDMA), Time Division Multiple Access (TDMA), and Code Division Multiple Access (CDMA)

3.1 Random Access

- In **random access or contention** methods, **no station** is **superior** to another station and **none** is **assigned** the **control over another**.

- **No station** permits, or does **not permit**, another **station** to **send**.

- **At each instance**, a **station** that has **data** to send **uses** a **procedure** defined by the **protocol** to **make** a **decision** on whether or **not** to **send**.

- This **decision depends** on the **state** of the **medium** (idle or busy).

Contd.

- In other words, each station can **transmit** when it **desires** on the condition that it **follows** the **predefined procedure**, including the **testing** of the **state** of the **medium**.
- In a **random access method**, **each station** has the **right** to the **medium** without being **controlled** by any other **station**.
- However, if **more** than one **station** tries to **send**, there is an access **conflict-collision**-and the **frames** will be either **destroyed** or **modified**.
- To **avoid access conflict** or to **resolve** it when it happens, each **station** follows a **procedure** that answers the **following questions**:

Contd.

- **When** can the **station access** the **medium**?
- **What** can the **station** do if the **medium** is **busy**?
- **How** can the **station determine** the **success** or **failure** of the **transmission**?
- **What** can the **station** do if there is an **access conflict?**

- The **random access** methods have evolved from a very interesting **protocol** known as **ALOHA**, which used a very simple procedure called **multiple access (MA)**.
- The **method** was **improved** with the **addition** of a **procedure** that forces the **station** to **sense** the **medium** before **transmitting**.

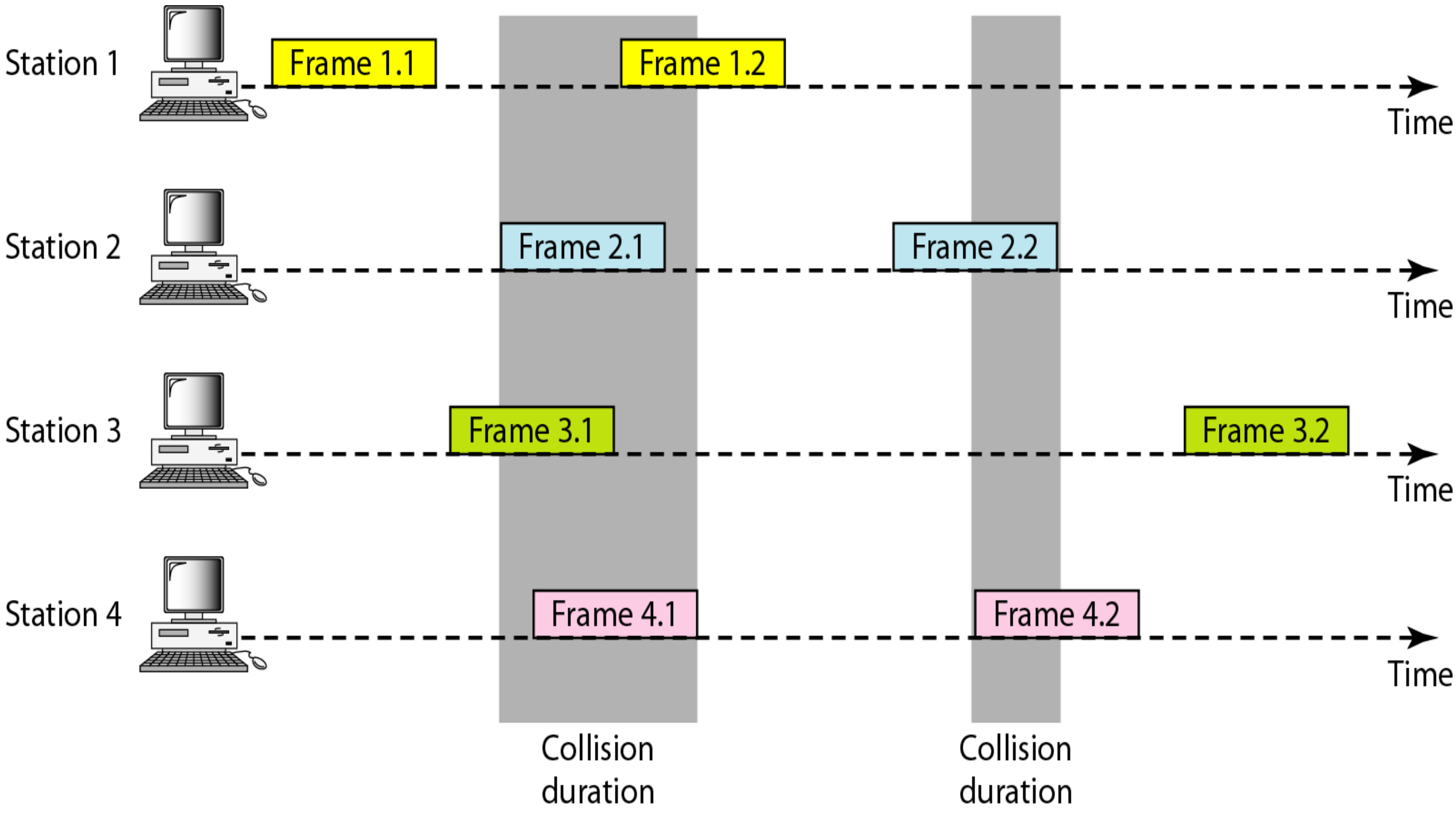
Contd.

- This was called **carrier sense multiple access**.
- This method later evolved into **two parallel methods**:
  - ✓ **Carrier Sense Multiple Access** with **Collision Detection (CSMA/CD)** and
  - ✓ **Carrier Sense Multiple Access** with **Collision Avoidance (CSMA/CA).**
- **CSMA/CD** tells the **station** what to do when a **collision** is **detected**.
- **CSMA/CA** tries to **avoid** the **collision**.

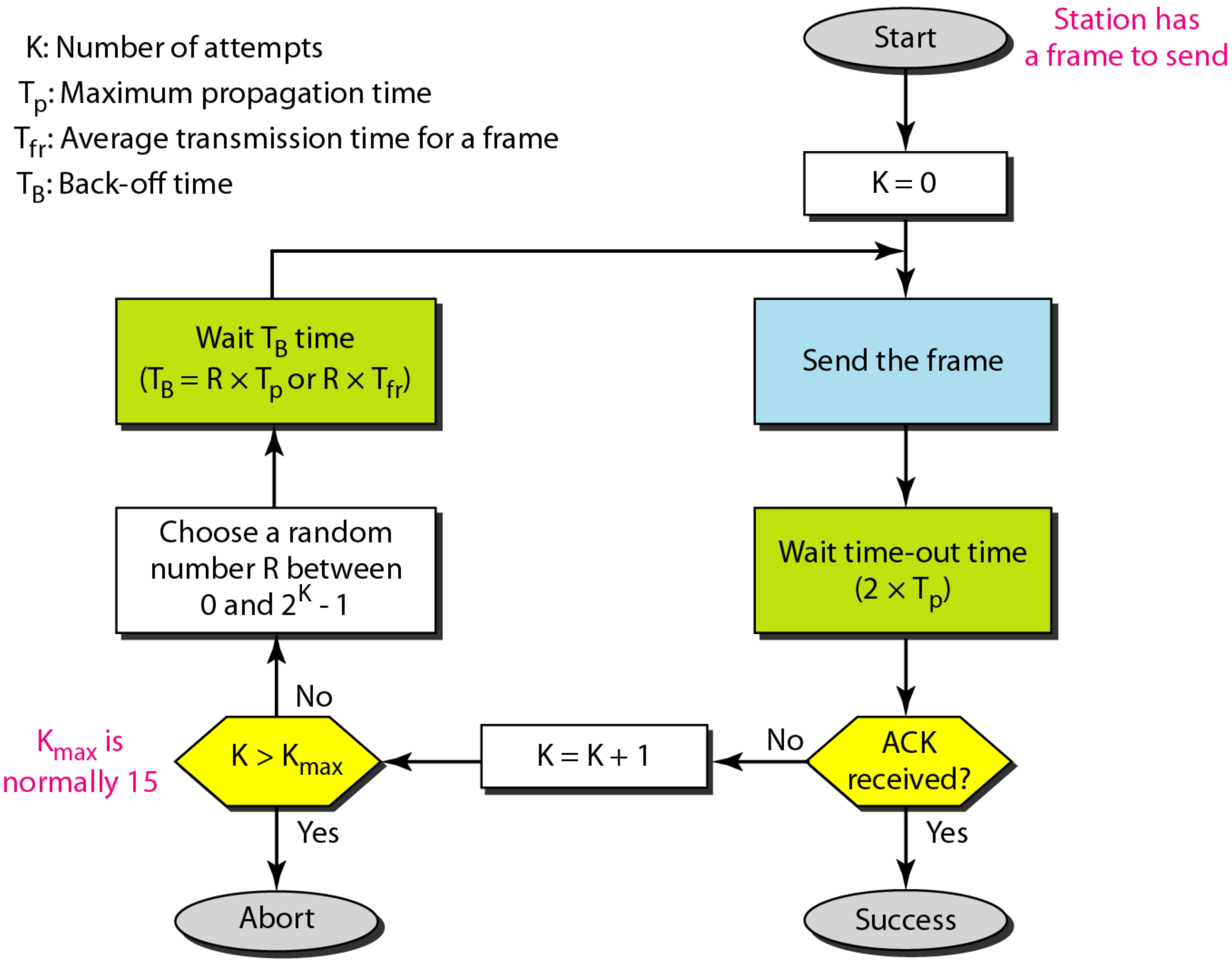
Pure ALOHA

- **ALOHA** is the **simplest technique** in **multiple accesses**.
- Basic idea of this **mechanism** is a **user** can **transmit** the **data whenever** they **want**.
- If **data** is **successfully transmitted** then there **isn’t** any **problem**.
- But if **collision occurs** then the station will **transmit again**.
- **Sender** can **detect** the **collision** if it **doesn’t receive** the **acknowledgement** from the **receiver**.

Procedure for pure ALOHA protocol



Procedure for pure ALOHA protocol



Carrier Sense Multiple Access (CSMA)

- **Protocols** that **listen** for a **carrier** and **act** accordingly are called **carrier sense protocols**.
- **Carrier sensing** allows the station to detect whether the medium is currently being used.
- Schemes that use a carrier sense circuits are classed together as carrier sense multiple access or CSMA schemes.
- There are two variants of CSMA. CSMA/CD and CSMA/CA
- The simplest CSMA scheme is for a station to sense the medium, sending packets immediately if the medium is idle.
- If the station waits for the medium to become idle it is called persistent otherwise it is called non persistent.

Persistent

- **Persistent:-wait** if **busy** and **transmit only** when the **media becomes idle again (not transmission after a triggered timer expire)**
- When a station has the data to send, it first listens the channel to **check** if **anyone** else is transmitting data or not.
  - If it senses the channel **idle**, station **starts transmitting** the **data**.
- If it **senses** the **channel busy** it waits **until** the **channel** is idle, by **continuously sensing** the **channel**.

Non-Persistent

- **Non persistent CSMA** is **less aggressive** compared to **persistent protocol**.
- In this **protocol**, before **sending** the **data**, the **station senses** the **channel** and if the **channel** is **idle** it **starts transmitting** the **data**.
- But if the **channel** is **busy**, the **station** does **not continuously sense** it but instead of that it **waits** for **random amount** of **time** and **repeats** the **algorithm**.
- Here the **algorithm** leads to **better channel utilization** but also results in **longer delay** compared to **persistent**.

Carrier Sense Multiple Access/Collision Detection (CSMA/CD)

- **CSMA/CD** is a technique for **multiple access protocols**.

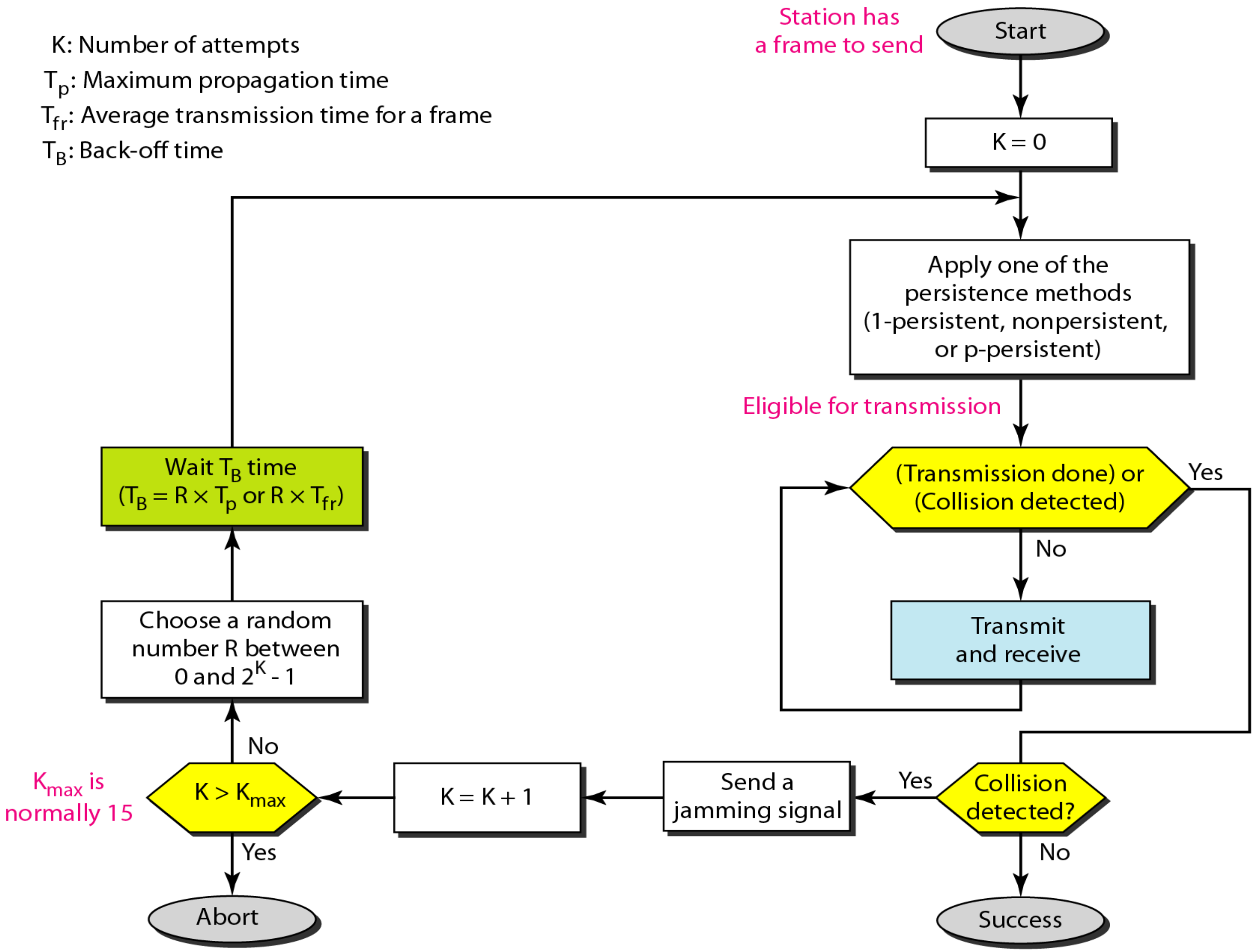
✓ If **no transmission** is taking place at the time, the particular station can transmit.

✓ If **two stations attempt** to **transmit** **simultaneously**, this causes a **collision**, which is **detected** **by all** **participating stations**.

- After a **random time interval**, the **stations** that **collided** attempt to **transmit again**.
- If another **collision** occurs, the **time intervals from** which the **random waiting time** is selected are **increased step** by **step**.

  ✓ This is known as **exponential back off.**

Flow diagram for the CSMA/CD



- **Reading Assignment:** **Read About CSMA/CA and understand the difference with CSMA/CD**

3.2 Controlled access

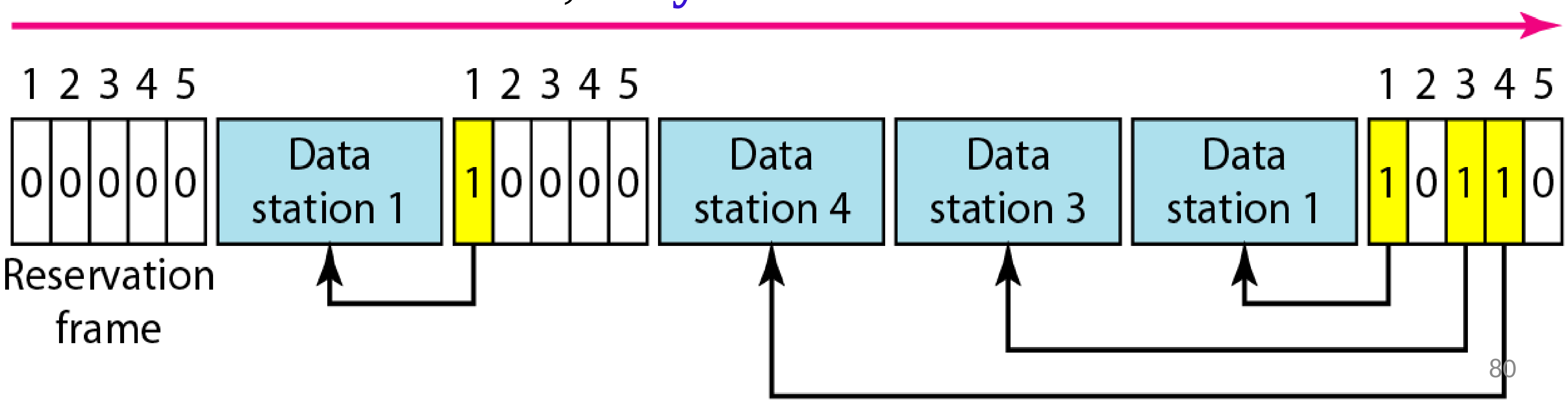
- In **controlled access,** the stations **consult one another** to **find** which **station** has the **right** to **send**.
- A **station cannot send** unless it has been **authorized** by other **stations**.

Reservation

- In the **reservation method**, a **station** needs to make a **reservation** before **sending data**.
- **Time** is divided into **intervals**.
- In each **interval**, a **reservation frame** precedes the **data frames sent** in that **interval**.
- If there are ***N* stations** in the **system**, there are exactly ***N* reservation minis lots** in the **reservation frame**.
- Each **mini slot** belongs to a **station**.
- When a station needs to send a data frame, it makes a reservation in its own minis lot.

Contd.

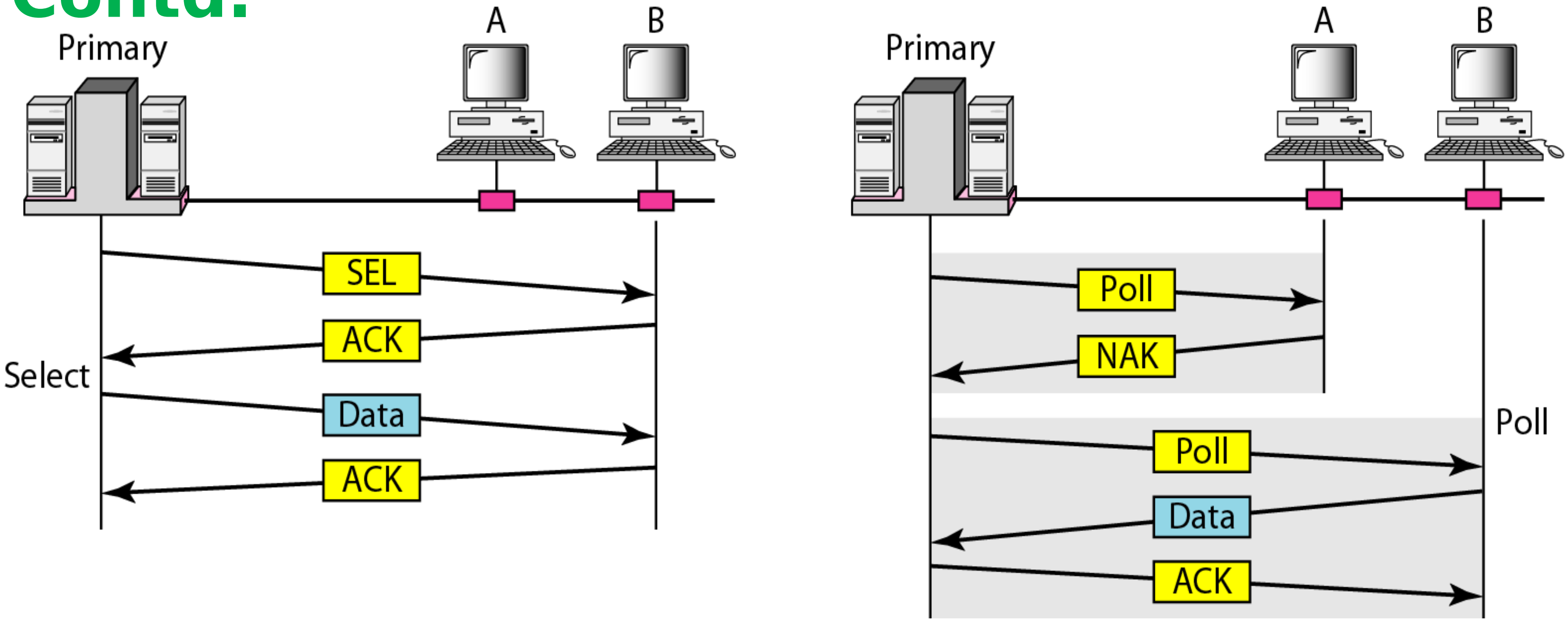
- The **stations** that have made **reservations** can **send** their **data frames** after the **reservation frame**.
- The following figure shows a situation with **five stations** and a **five-minis lot reservation frame**.
- In the **first interval**, **only stations 1, 3, and 4** have made **reservations**.
- In the second interval, **only station 1** has made a reservation.



Polling

- **Polling** works with **topologies** in which **one device** is designated as a **primary station** and the other **devices** are **secondary stations**.
- All **data** exchanges must be made through the **primary device** even when the **ultimate destination** is a **secondary device**.
- The **primary** device **controls** the **link**; the **secondary devices** follow its **instructions**.
- It is up to the **primary device** to determine which **device** is allowed to use the **channel** at a given **time**.
- The **primary device**, therefore, is always the **initiator of a session.**

Contd.



- If the **primary** wants to **receive data**, it asks the **secondary devices** if they have anything to **send**; this is called **poll function**.
- If the **primary** wants to **send data**, it tells the **secondary** to get **ready** to **receive**; this is called **select function**.

Token Passing

- In the **token-passing method**, the **stations** in a **network** are organized in a **logical ring.**
- In other words, for **each station**, there is a ***predecessor*** and a ***successor***.
- The **predecessor** is the **station** which is **logically** **before** the **station** in the ring;

  the **successor** is the **station** which is **after** the **station** in the **ring**.
- The **current station** is the one that is **accessing** the **channel** now.

Contd.

- The **right** to this access has been passed from the **predecessor** to the **current** **station**.
- The **right** will be **passed** to the **successor** when the **current station** has **no more data** to **send**.
- But **how** is the **right** to **access** the **channel passed** from one **station** to **another**?
- In this **method**, a **special packet** called a **token circulates** through the **ring**.
- The **possession** of the **token** gives the **station** the **right** to **access** the **channel** and **send** its **data**.

Contd.

- When a **station** has some **data** to **send**, it **waits** until it **receives** the **token** from its **predecessor**.
- It then **holds** the **token** and **sends** its **data**.
- When the **station** has **no** more **data** to **send**, it **releases** the **token**, **passing** it to the **next logical station** in the **ring**.
- The **station cannot send data until** it **receives** the **token again** in the **next round**.

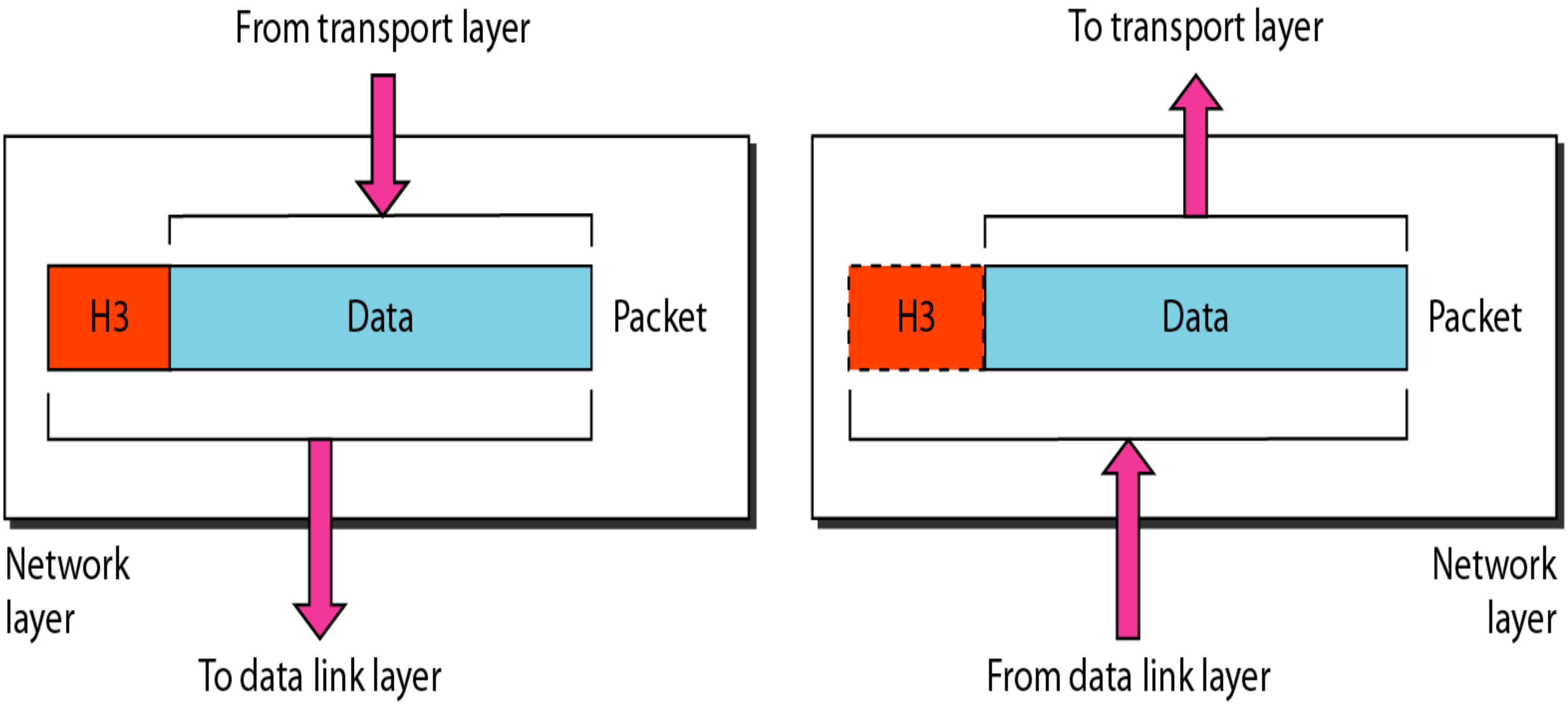
3. Network Layer (3rd OSI Layer)

- 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**.

3. Network Layer (3rd OSI Layer)-------

- 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**.

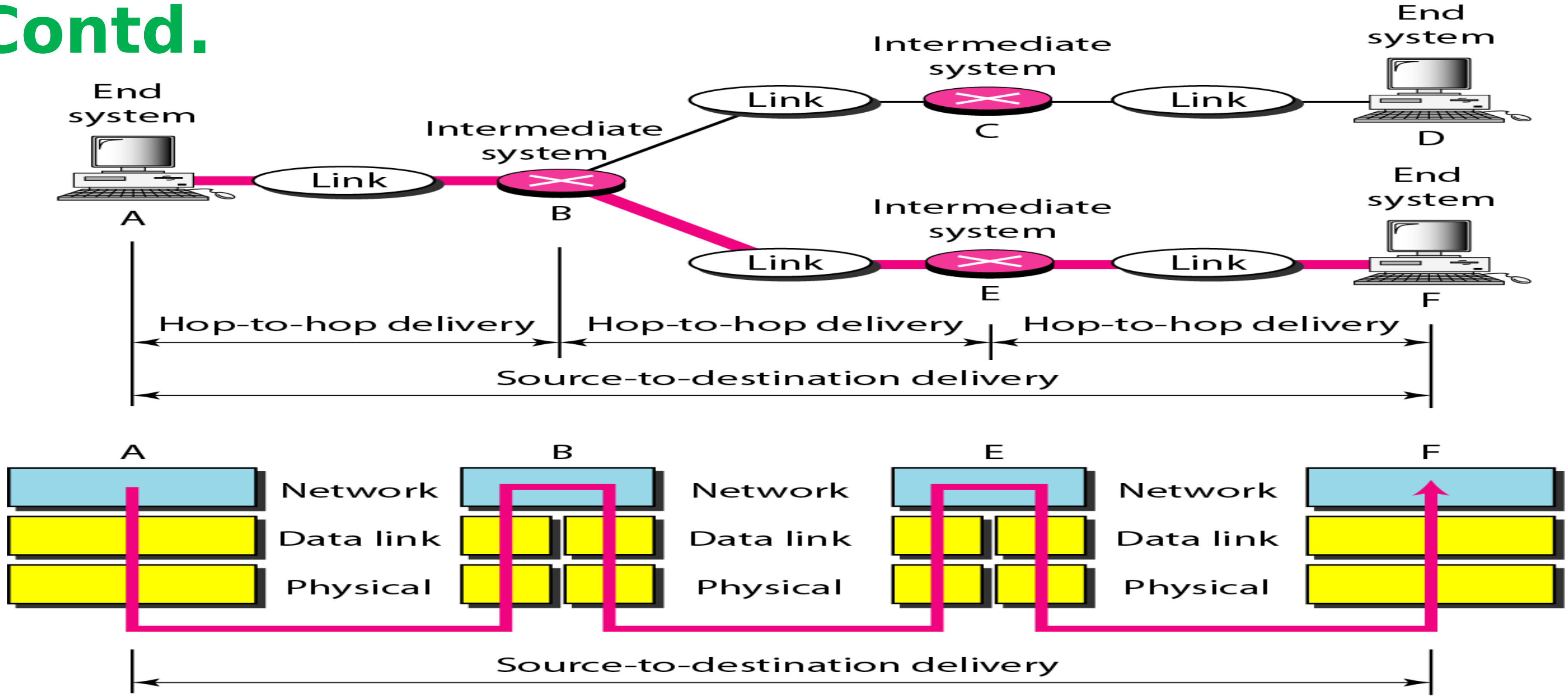
Contd.

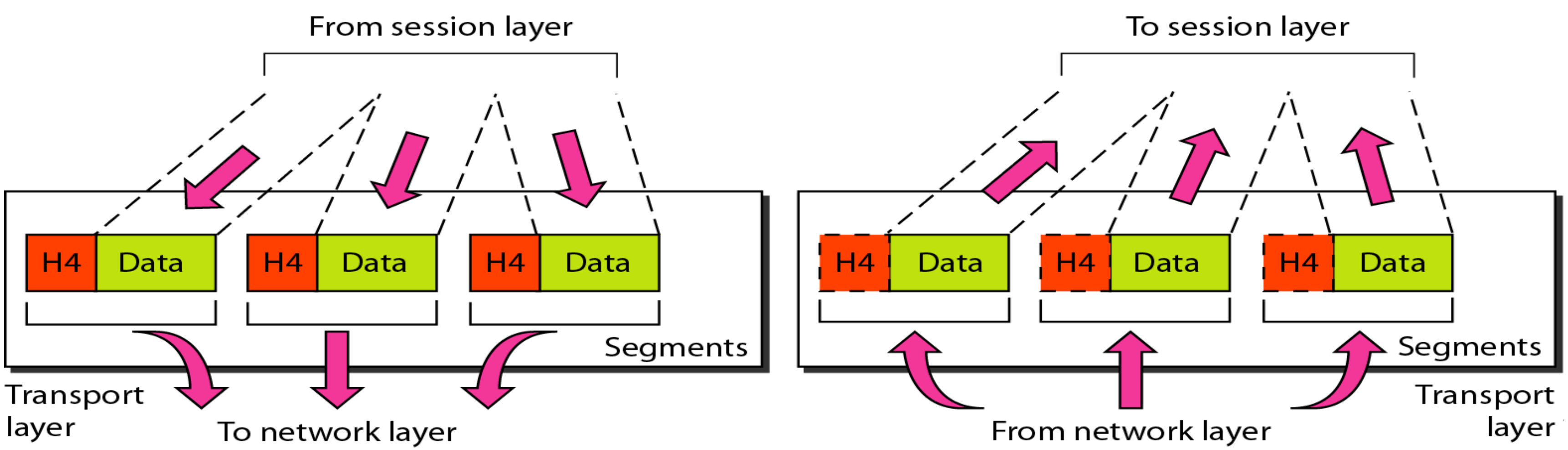
2. Routing:-

- 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.


End system
Intermediate system
Link
Link
Intermediate system
Link
Link
End system
A
B
C
D
Intermediate system
Link
Link
End system
E
F
Hop-to-hop delivery
Hop-to-hop delivery
Hop-to-hop delivery
Source-to-destination delivery
A
B
E
F
Network
Network
Network
Data link
Data link
Data link
Physical
Physical
Physical
Source-to-destination delivery


From session layer
H4
Data
H4
Data
H4
Data
Segments
Transport layer
To network layer
To session layer
H4
Data
H4
Data
H4
Data
Segments
From network layer
Transport layer

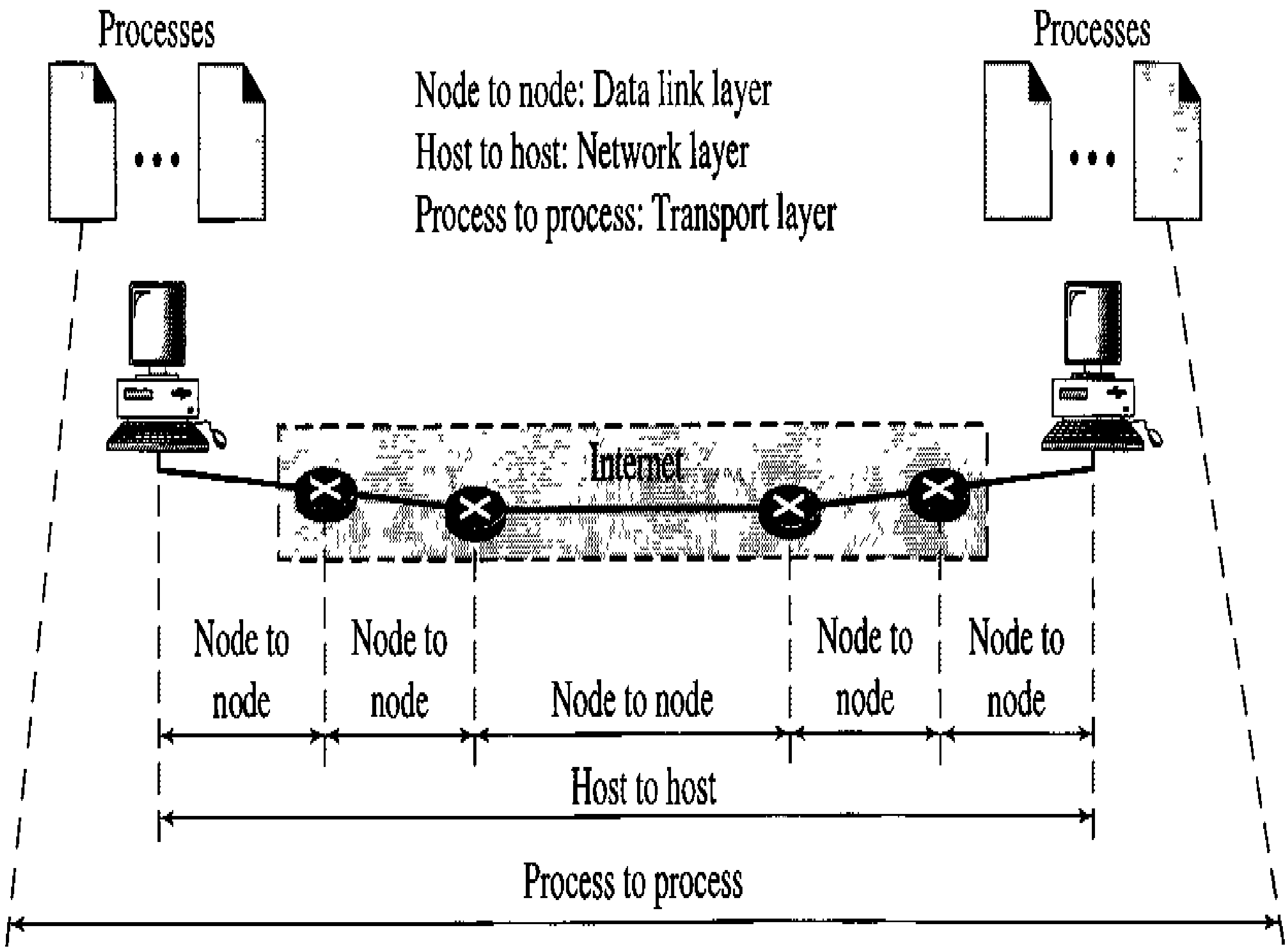
4. Transport layer (4th OSI layer)

- 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.

- 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.



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

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***)*.

Responsibilities Transport Layer-------

- 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.**

Contd.

- 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**.

5. Connection control

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

Contd.

A. Connectionless

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

A. 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**.

1. 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 **packets** 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).**

1. Process-to-Process Delivery---

- 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.**

2. 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**.
- 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**.

3. Identifying Applications (Processes)

- In order to **pass data streams** to the **proper applications**, the **Transport layer** must identify the **target application**.
- To accomplish this, the **Transport layer assigns** an **application identifier** called a **port number**.
- Each **software process** that needs to **access** the **network** is assigned a **port number unique** in that **host**
- This **port number** is used in the **transport layer header** to indicate to which **application** that **piece** of **data** is associated.

➢ The **server process** must also define itself with a **port number**.

- This **port number,** however, **cannot be chosen randomly**.

Contd.

- If the **computer** at the **server site runs** a **server process** and **assigns** a **random number** as the **port number**,

  the **process** at the **client site** that wants to **access** that **server** and use its **services** **will not know the port number.**

- Of course, one **solution** would be to **send** a **special packet** and **request** the **port number** of a specific **server**, but this requires more **overhead**.
- The **Internet** has decided to use **universal port numbers** for **servers**; these are called **well-known port numbers.**
- **Example** of well known **port numbers**: **21** for **FTP, 23 telnet, 25 SMTP**, **80 HTTP**, etc

Internet Assigned Number Authority (IANA) Ranges

- There are some **exceptions** to this **rule**; for **example**, there are **clients** that are **assigned** **well-known port numbers.**
- Every **client process** knows the well-known **port number** of the corresponding **server process**.

1. Well-known ports.

- ✓ The **ports** ranging from **0 to 1023** are assigned and controlled by **IANA**. These are the **well-known ports**.

2. Registered ports.

- ✓ The **ports** ranging from **1024 to 49,151** are **not assigned** or controlled by **IANA**.

Internet Assigned Number Authority (IANA) Ranges

- ✓ They can only be **registered** with **IANA** to **prevent duplication**.

3. Dynamic ports.

- The **ports** ranging from **49,152 to 65,535** are **neither controlled nor registered**.
- ✓ They can be used by **any process**.
- ✓ These are the **ephemeral ports** **(temporary ports).**

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**.

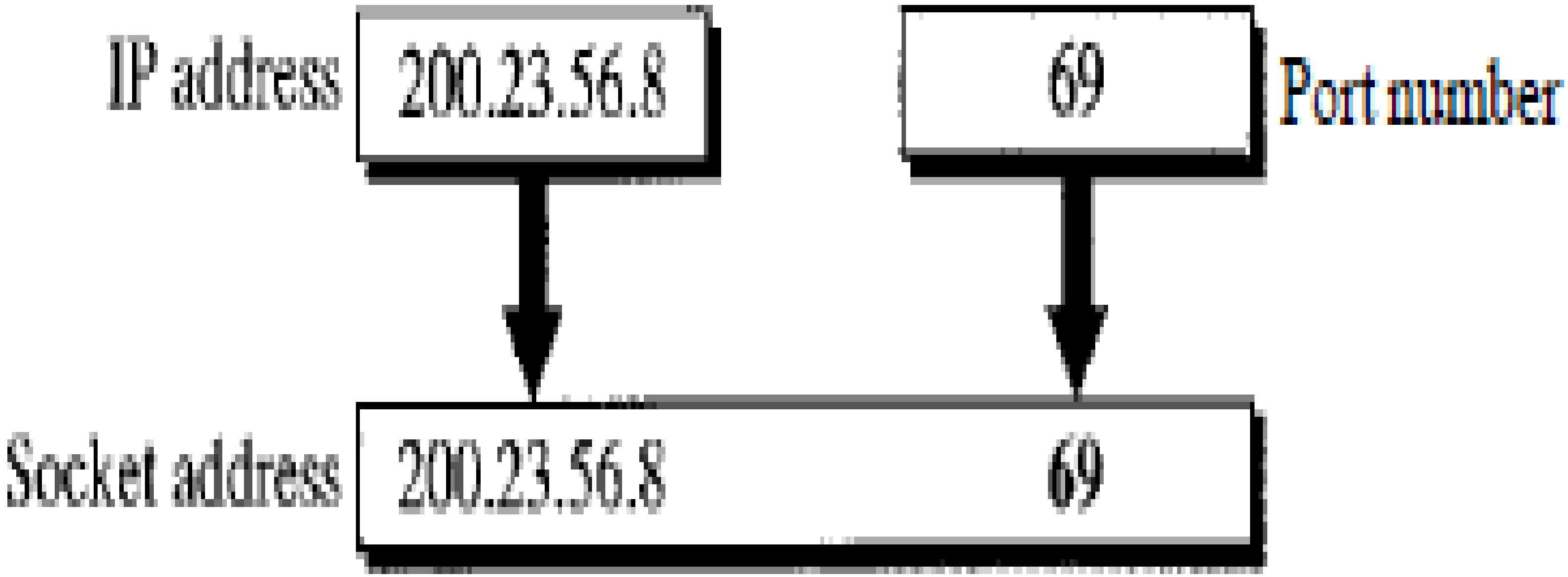
✓ The **client socket address** defines the **client process uniquely** just as the **server socket address** defines the **server process uniquely**.

- A **transport layer protocol** needs a pair of **socket addresses**: the **client socket address** and the **server socket address**.

Socket Addresses

- 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.**



Transport Layer Protocols

- 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),

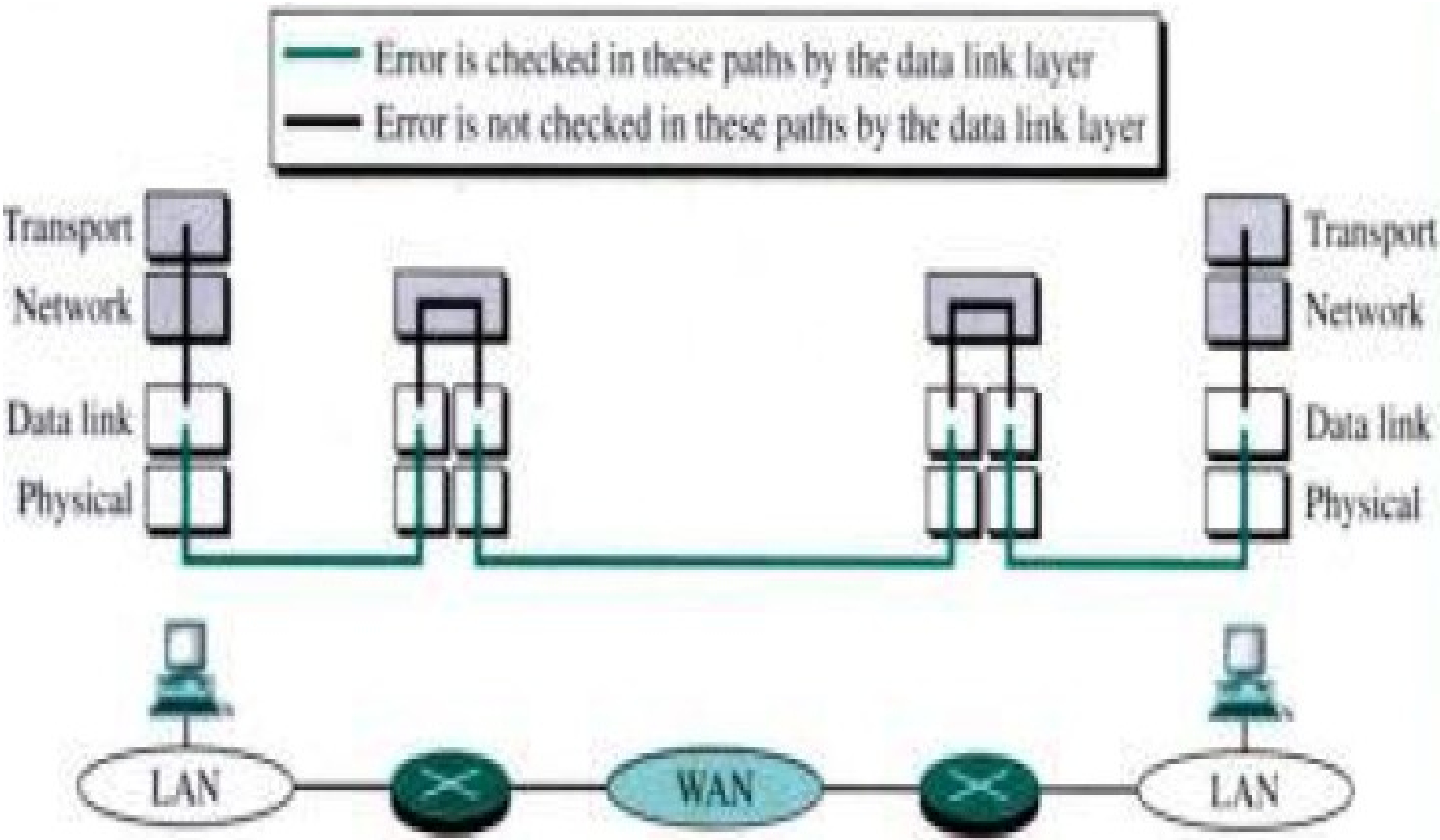
  then an unreliable protocol can be used

Reliable Versus Unreliable

- 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**.

Why flow control and error control at the transport layer as we have it at the data link layer???



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**.
- **UDP packets**, called user **datagrams**, have a **fixed-size** header of **8 bytes**

1. Source port number.

✓ This is the port number used by the **process running** on the **source host**.

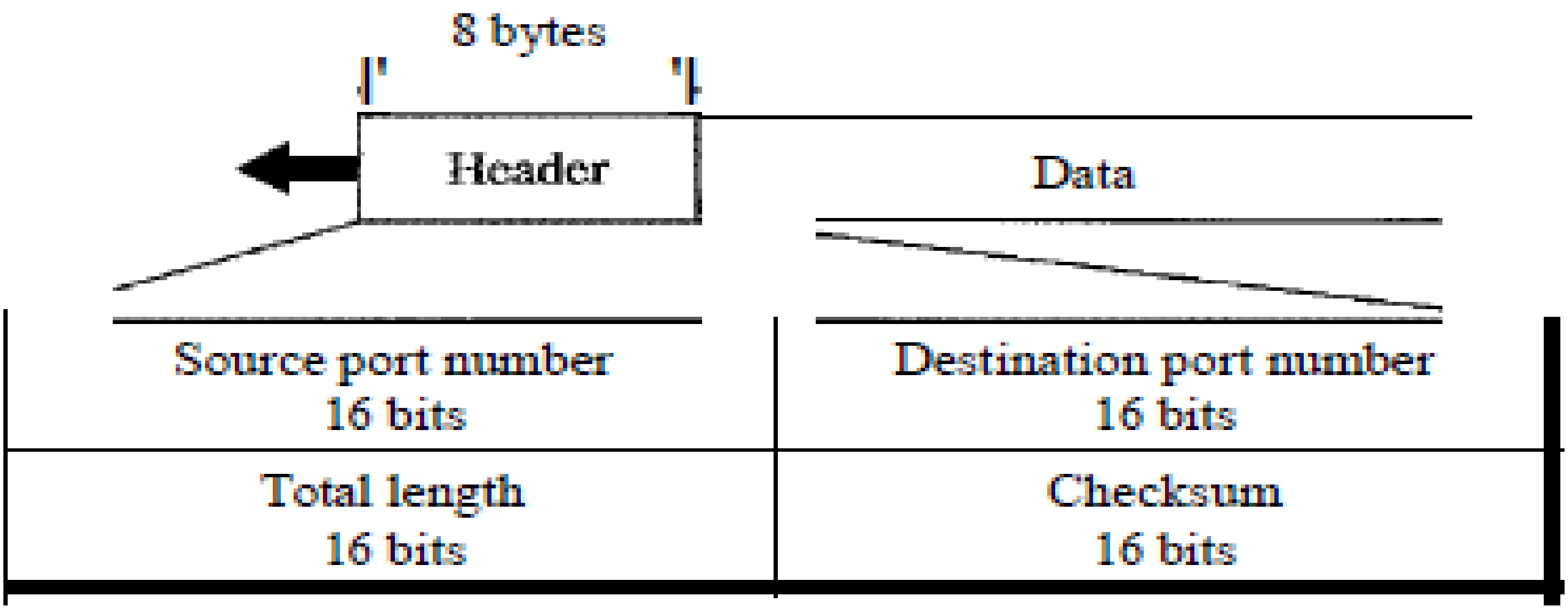
2. Destination port number.

✓ Used by the **process running** on the **destination host**.

Cont------

3. Length

- This is a **16-bit field** that defines the **total length** of the **user datagram**, **header** plus **data**.



- **UDP** does **not** perform:
- **Flow control, Error control and 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**.

Use of 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

5. Session Layer

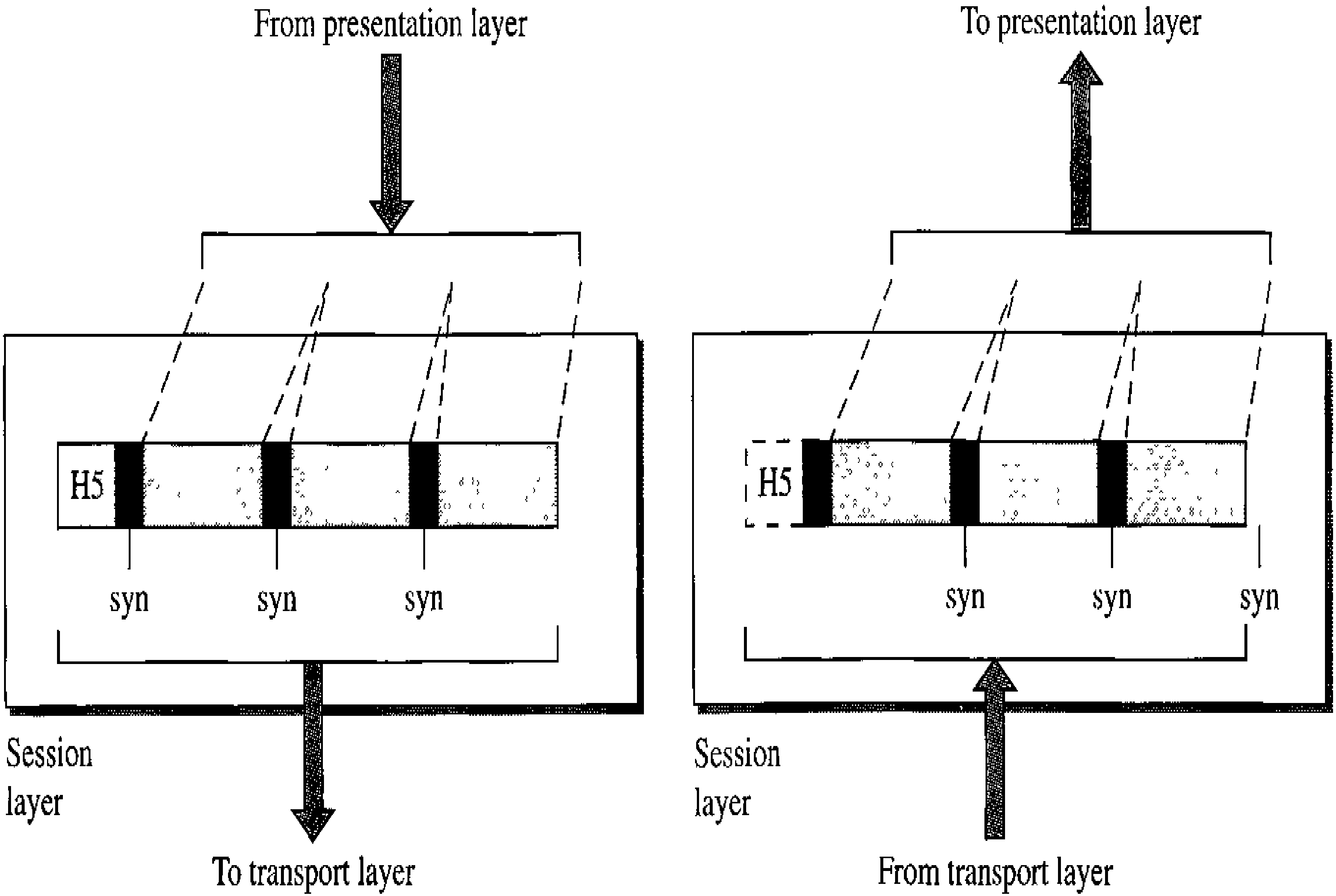
- The **lowest four layers** of the **OSI model** (**Physical, Data link, Network**, and **Transport**) provide the means for the **reliable exchange** of **data** and **provide** a **fast data service**.
- For example, a **remote terminal** access application might require a **half-duplex dialogue**.
- 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.

Cont----

- 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.



From presentation layer
H5
syn
syn
syn
Session
layer
To transport layer
To presentation layer
H5
syn
syn
syn
Session
layer
From transport layer

Key Services provided by the Session Layer

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**.

Key Services provided by the Session Layer

3. Recovery.

- The **session layer** can provide a **check pointing mechanism**,

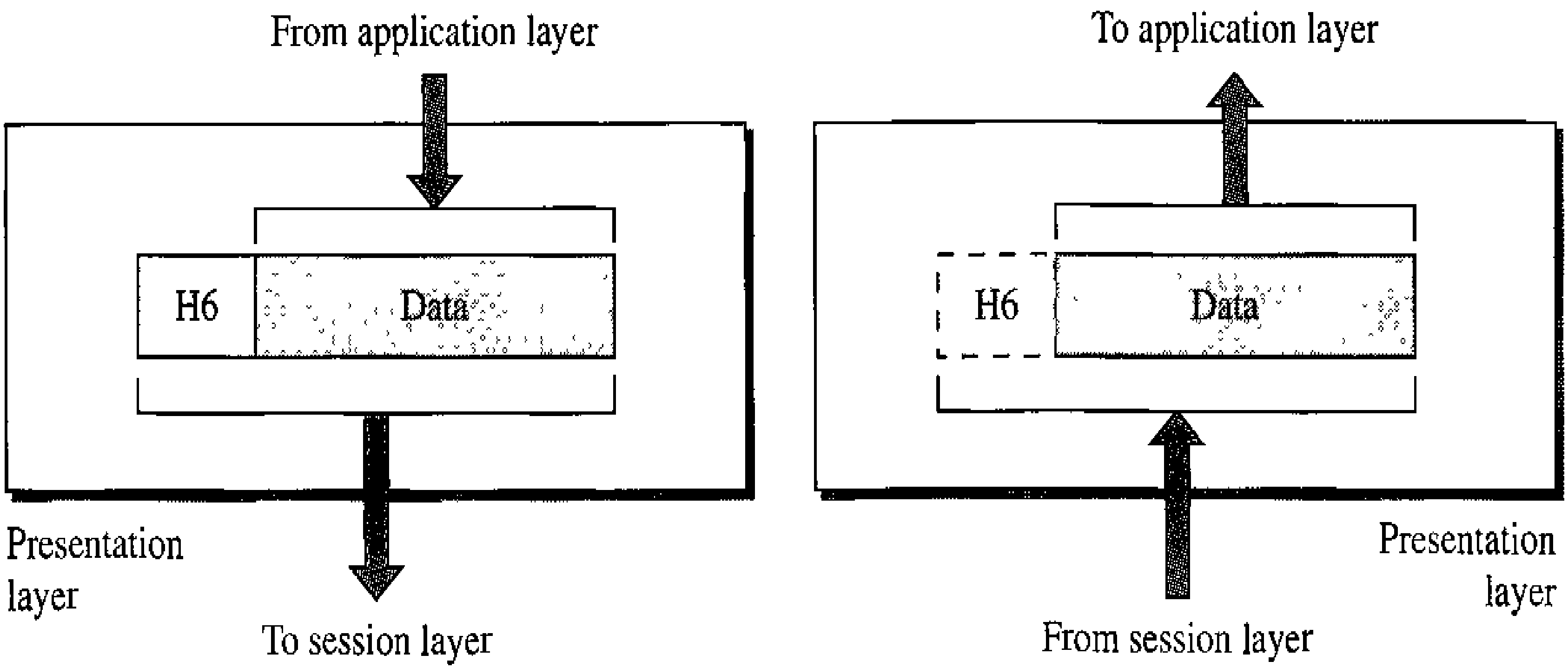
  so that if a **failure** of some **sort occurs** between **checkpoints**, the **session entity** can **retransmit** all **data** since the **last checkpoint.**

- 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**.

6. 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**.

Specific responsibilities of the presentation layer :

- 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**.

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.**

Specific responsibilities of the presentation layer :

- **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**.

7. 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.

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

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**

DNS continued...

- **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**

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

2. 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**

3. E-mail

- 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:

- **Simple Mail Transfer Protocol (SMTP):**

✓ To **send** mail from **client** to **server** or **server** to **server**

- **Post Office Protocol (POP3):**

✓ To **download email** from **server** to **client**, and the **server deletes** the **mail**

- **Internet Message Access Protocol (IMAP4):**

✓ To **download email** from **server** to **client**, and the **server** does **not delete** (**keeps**) the **mail**

Activity

1. Which one of the following **is not correct** about POP3 and SMTP?

    A. A sender can send email using SMTP

    B. Both are application layer protocols

    C. Both are Network Layer protocols

    D. A sender can receive email using POP3

2. Which type of network is the largest as compared to the rest?

    A. Metropolitan Area Network

    B. The Internet

    C. Local Area Network

    D. Wide Area Network

Activity------

3. A university owns a number of private local area and wide area networks, which are designed for an access by its academic staff, students and administrative workers. Which term best describes the University's network?

   A. Wide Area Network

   B. Metropolitan Area Network

   C. The Internet

   D. Local Area Network

4. Which layer uses port number to identify applications?

   A. Application layer

   B. Physical layer

   C. Network Layer

   D. Transport Layer

Activity-------

5. At which layer of the OSI model devices such as bridges, switches and Network Interface Cards **(NICs)** are used?

   A. Application layer

   B. Physical layer

   C. Network layer

   D. Data link layer

6. Write the difference and similarities between UDP and TCP transport Layer Protocols?

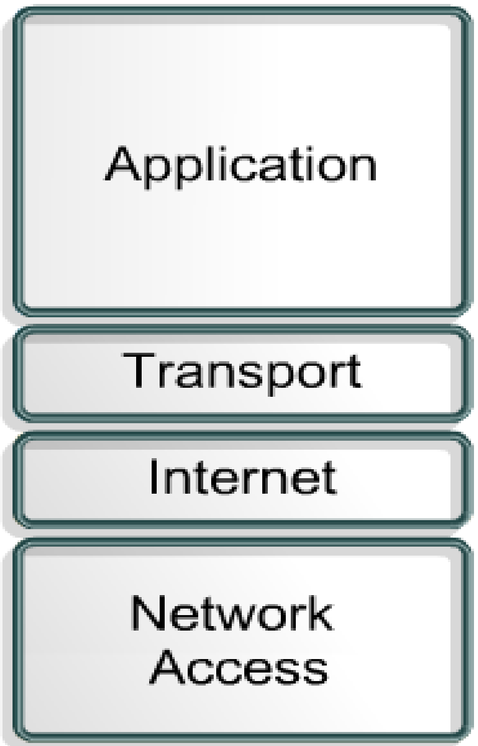
2. TCP/IP Reference Model

- **TCP/IP Reference Model** is a **four-layered** suite of **communication protocols**.
- It is named after the two **main protocols** that are used in the model, namely, **TCP** **TCP/IP** stands for **Transmission Control Protocol / Internet Protocol**
- It was developed by the **DoD (Department of Defence)** in the **1960s** as a **model** able to **withstand** intense **military attack** and **not fail**.
- **Data transmission** was possible to any **destination** on the **network** under any circumstances.
- **Standardized** in **1981**, the **TCP/IP model** is now the **standard** on which the **Internet** is **based**.

2. TCP/IP Reference Model------

- It has **four layers**

- There are **similarities** and **differences** between the **TCP/IP model** and the seven **layer OSI model.**



1. TCP/IP Application Layer

- Ensures that the **data** is properly **packaged** before **being passed** on.
- Handles high-level **protocols**, **representation**, **encoding**, and **dialog control**.
- There are **lots** of **protocols** defined at this layer:
  - ✓ **Simple Network Management Protocol (SNMP)** – allows network managers to manage **configurations**, **statistics**, **performance**, and **security**.
  - ✓ **Domain Name System (DNS)** – used to **translate domain names** into **IP addresses**.

145

1. TCP/IP Application Layer------

- Has protocols to support **file transfer, e-mail**, and **remote login**.

- **File Transfer:**

1. **Trivial File Transfer Protocol (TFTP)**

- **Unreliable**, **connectionless User Datagram Protocol (UDP)** service used to **transfer configuration files**, **Cisco IOS images**, and to **transfer files** in a LAN.

2. **File Transfer Protocol (FTP)**

- **Reliable**, **connection-oriented service** that uses **TCP** to **transfer files** between **systems**

146

3. **Network File System (NFS)**

- Allows **file access** to a **remote storage device** such as a **hard disk**

1. TCP/IP Application Layer------

➢E-mail:

1. **Simple Mail Transfer Protocol (SMTP)**

- **Administers** the **transmission** of **plain text e-mail** between **email servers**.

2. Post Office Protocol (POP3) and IMAP4

- Handles **email transfer** between a **mail server** and **client machine**

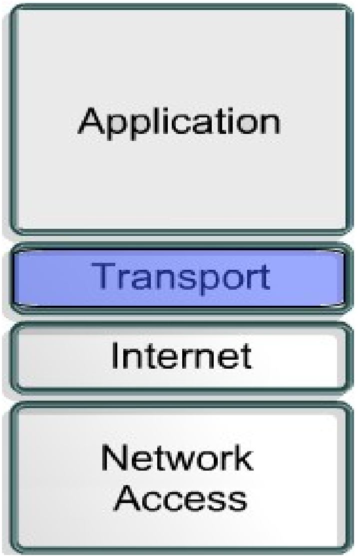
2. Remote Access:

- **Telnet**

✓ A **protocol** that **remotely access** a **computer**, **enabling** a **user** to **log** into an **Internet host** and **execute commands**.

✓ A **Telnet client** is called a **local host**. A **Telnet server** is called a **remote host**.

2. TCP/IP Transport Layer

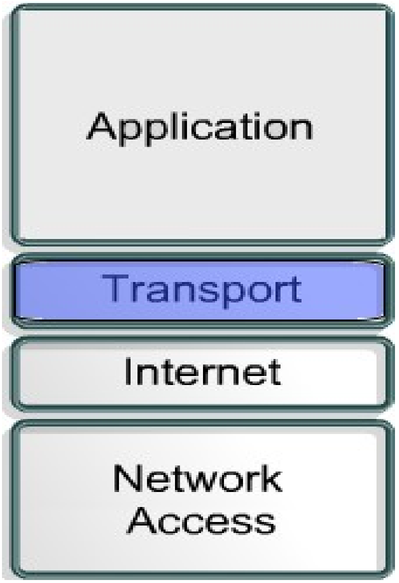


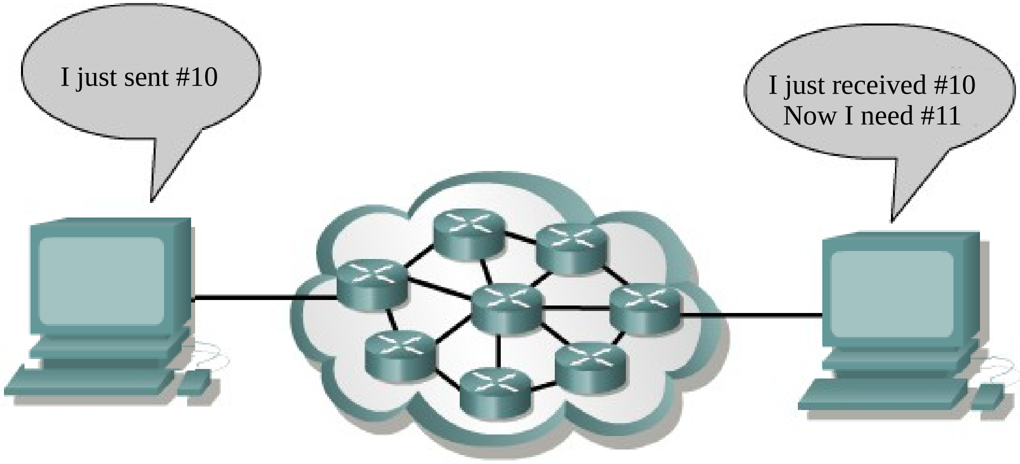
- Provides a **logical connection** between a **source host** and a **destination host**.
- **Transport Layer protocols** **segment** and **reassemble data sent** by **applications**, into the **same data stream**, between **end points**.
- Provides **end-to-end control** and **reliability** as **data travels** through the **cloud**, **accomplished through**:
  - ✓ **sequence numbers**, **acknowledgments** and **sliding windows**.

148

2. TCP/IP Transport Layer-------

Transport



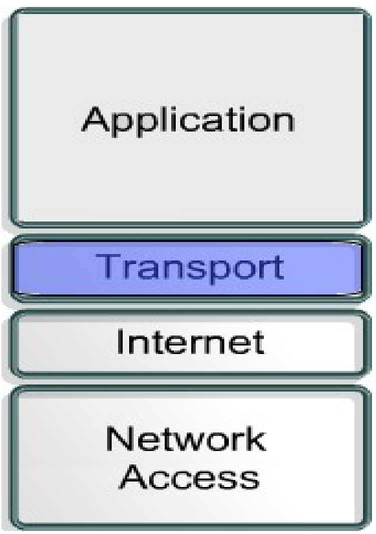


This shows sequence numbers and acknowledgements.

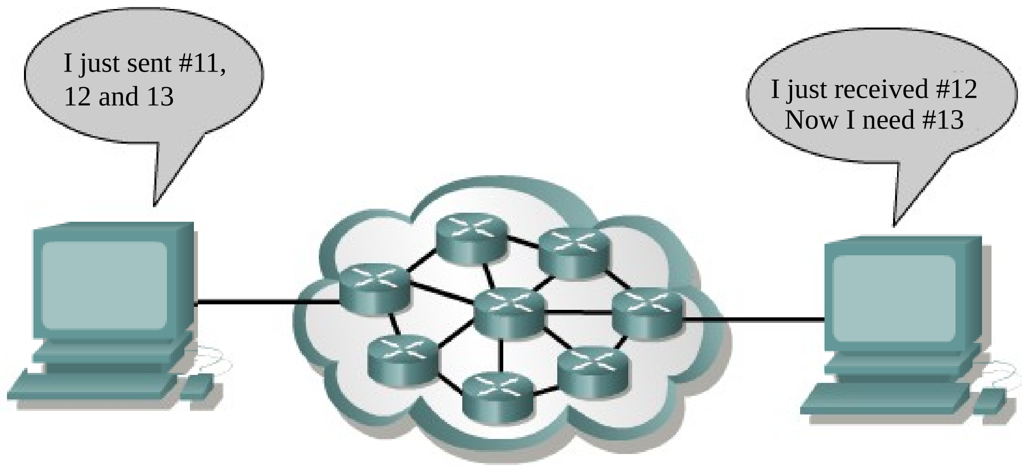
149

2. TCP/IP Transport Layer-------

Transport



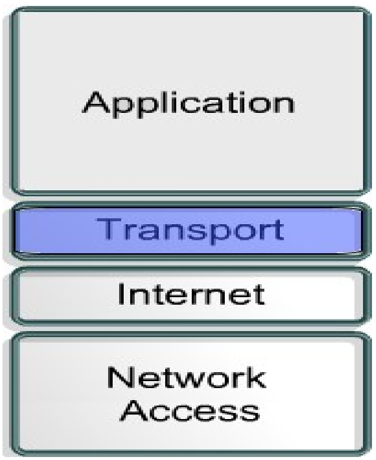
Sliding Windows



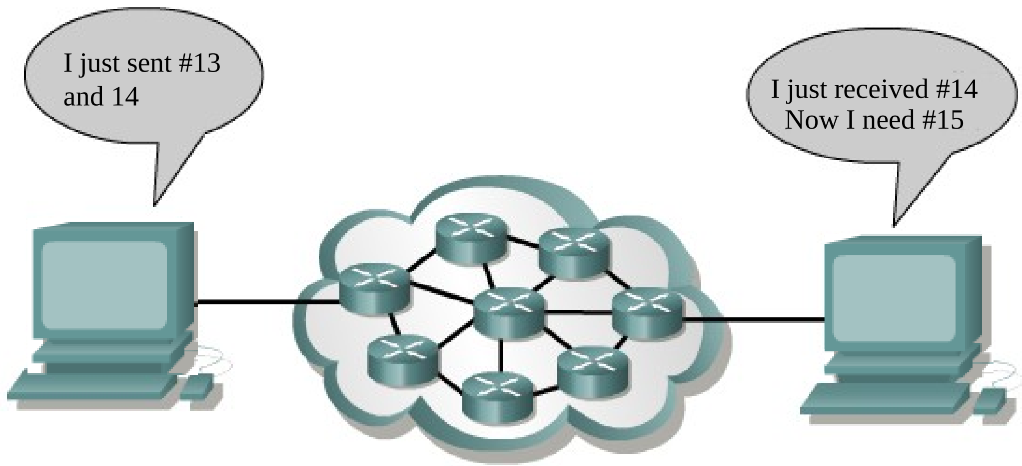
This indicates that packet 13 either did not arrive, or arrived with errors, and needs retransmission.

2. TCP/IP Transport Layer-------

Transport



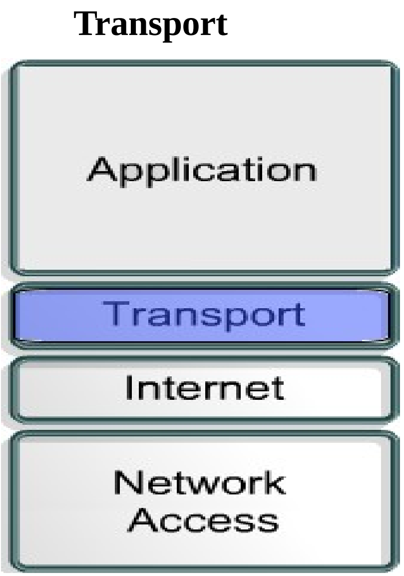
Sliding Windows



The sliding window has worked as the last packet sent has arrived.

151

2. TCP/IP Transport Layer-------



- The only **Transport** layer **protocols** are **TCP** and **UDP**.

1. Transmission Control Protocol (TCP)

- **Connection-oriented protocol**
- **End-to-end operation**
- **Flow control – sliding windows**
- **Reliability**–**sequence numbers** and **acknowledgments**

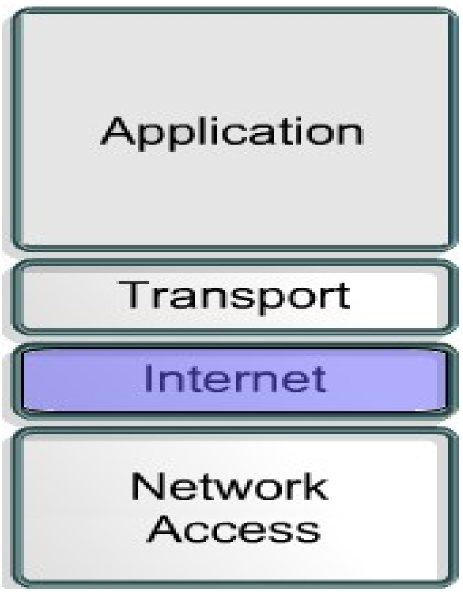
2. User Datagram Protocol (UDP)

- **Connectionless**
- **Unreliable** (**no acknowledgments** or **error checking**)

3. TCP/IP Internet Layer

- **Two purposes** are - **determining** the **best path** and **packet-switching**.
- **No error checking or correction**

Internet

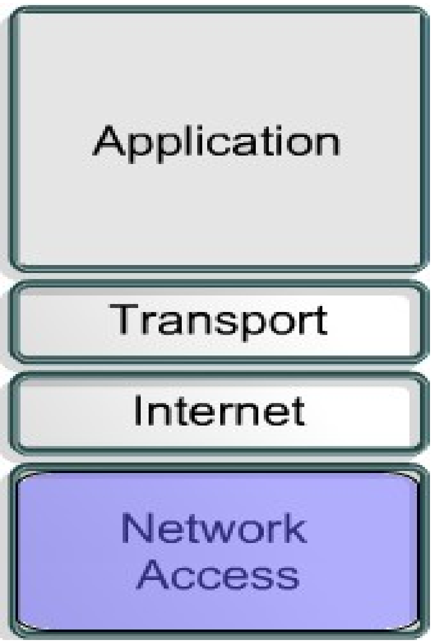


➢ **Protocols**:

- **Internet Protocol (IP) - connectionless**, **best-effort delivery routing** of **packets**; **determines** best **path** to **destination**
- **Internet Control Message Protocol (ICMP)** – **control** and **messaging**
- **Address Resolution Protocol (ARP) - determines** the **MAC address**, for a **known IP address**.
- **Reverse Address Resolution Protocol (RARP) - determines** the **IP address** for a known **MAC address**.

4. TCP/IP Network Access Layer

Network Access

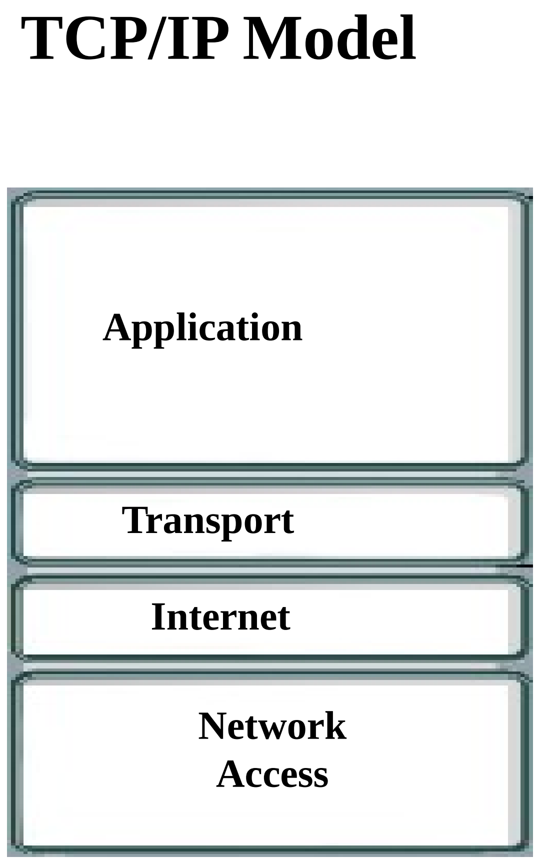


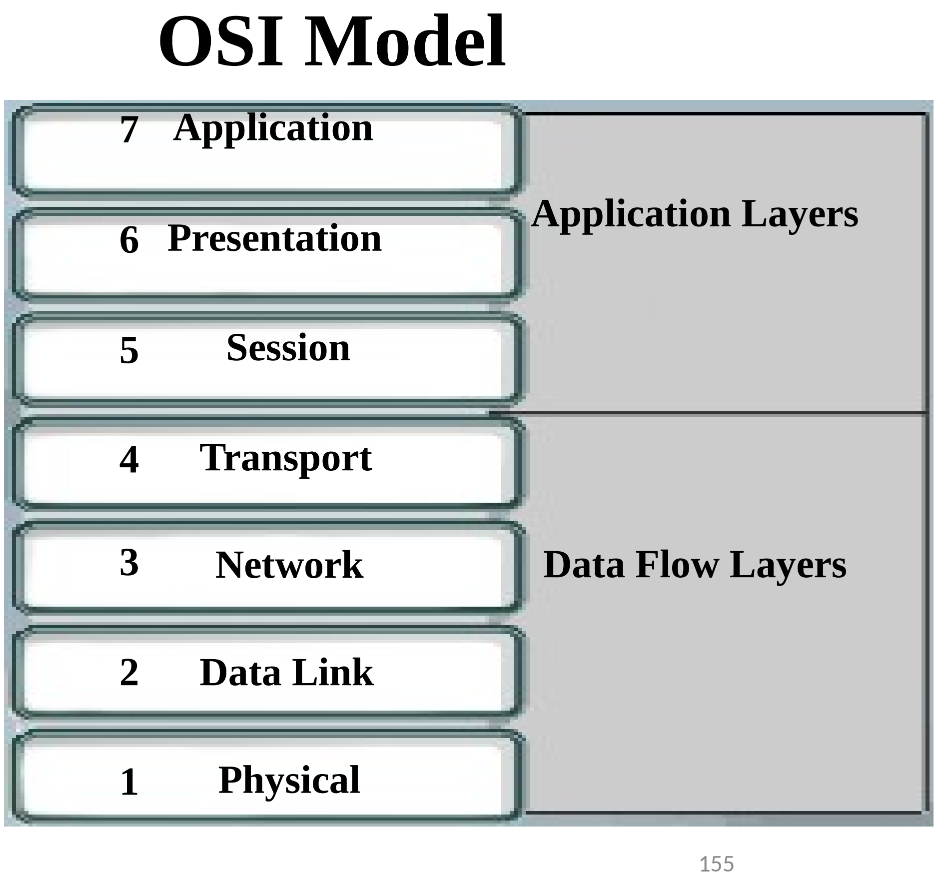
- Allows an **IP packet** to make a **physical link** to the **network media**
- Maps **IP addresses** to **MAC addresses**
- **Encapsulates IP packets** into **frames**
- **Drivers** for **modem cards**, and other **devices operate** at the **network access layer**.
- **Serial Line Internet Protocol (SLIP)** and **Point-to-Point Protocol (PPP)** provide **network access**.

- **ARP** and **RARP** also work at this layer.

✓ **Note: This layer is not defined by TCP/IP.**

Comparing TCP/IP and OSI





Activity

1. Write the advantages and disadvantages of OSI Model
2. Write the advantages and disadvantages of TCP/IP Model
3. Write the difference and similarities between OSI and TCP/IP models
4. Are OSI and TCP concepts outdated?
5. Can TCP work without OSI?
6. Which layers of OSI are relevant to TCP?
7. Write the difference between Circuit-switching and Packet-switching?
8. Explain the term IP packet?