Unit I
Chapter 2

Reference: Data Communication and Networking, Behrouz A Forouzan**,**
McGraw Hill, 5th Edition,
2008.

2.1 PROTOCOL LAYERING

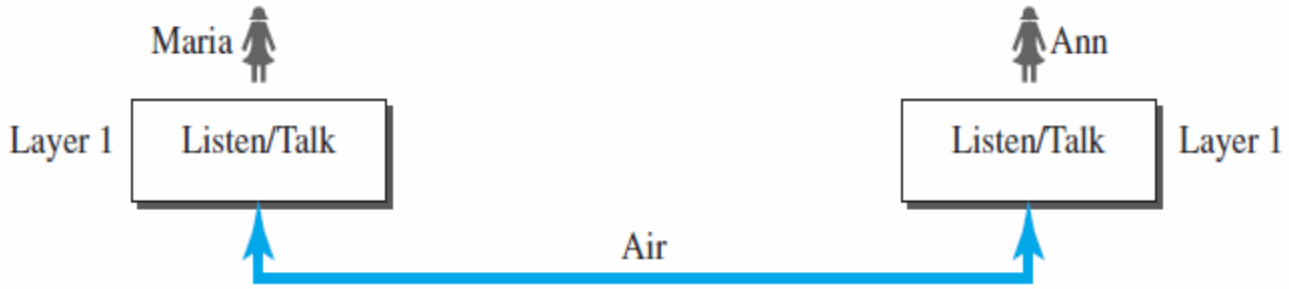
- In data communication and networking, a **protocol defines the rules that both the sender and receiver and all intermediate devices need to follow** to be able to communicate effectively.

- When communication is **simple**, we may need only one **simple protocol**; when the communication is **complex**, we may need to divide the task between different layers, in which case we need a protocol at each layer, or **protocol layering.**

2.1.1 Scenarios

- ***First Scenario***
- In the first scenario, communication is so **simple** that it can occur in only one layer.
- Assume Maria and Ann are **neighbors** with a lot of common ideas.
- Communication between Maria and Ann takes place in **one layer, face to face**, in the same language, as shown in Figure 2.1.
- Even in this simple scenario, we can see that a set of rules needs to be followed.
  - First, Maria and Ann know that they should **greet each other** when they meet.
  - Second, they know that they should confine their **vocabulary** to the level of their **friendship**.

Figure 2.1 *A single-layer protocol*



2.1.1 Scenarios Contd.

  - Third, each party knows that she should **refrain from speaking when the other party is speaking.**
  - Fourth, each party knows that the conversation **should be a dialog**, not a monolog: both should have the opportunity to talk about the issue.
  - Fifth, they should exchange some **nice words when they leave**.

- We can see that the protocol used by Maria and Ann is **different** from the **communication** between a **professor and the students** in a lecture hall.

- The communication in the second case is mostly **monolog**; the **professor talks most of the time** unless a student has a question, a situation in which the protocol dictates that she should **raise his/her hand** and wait for **permission to speak**.

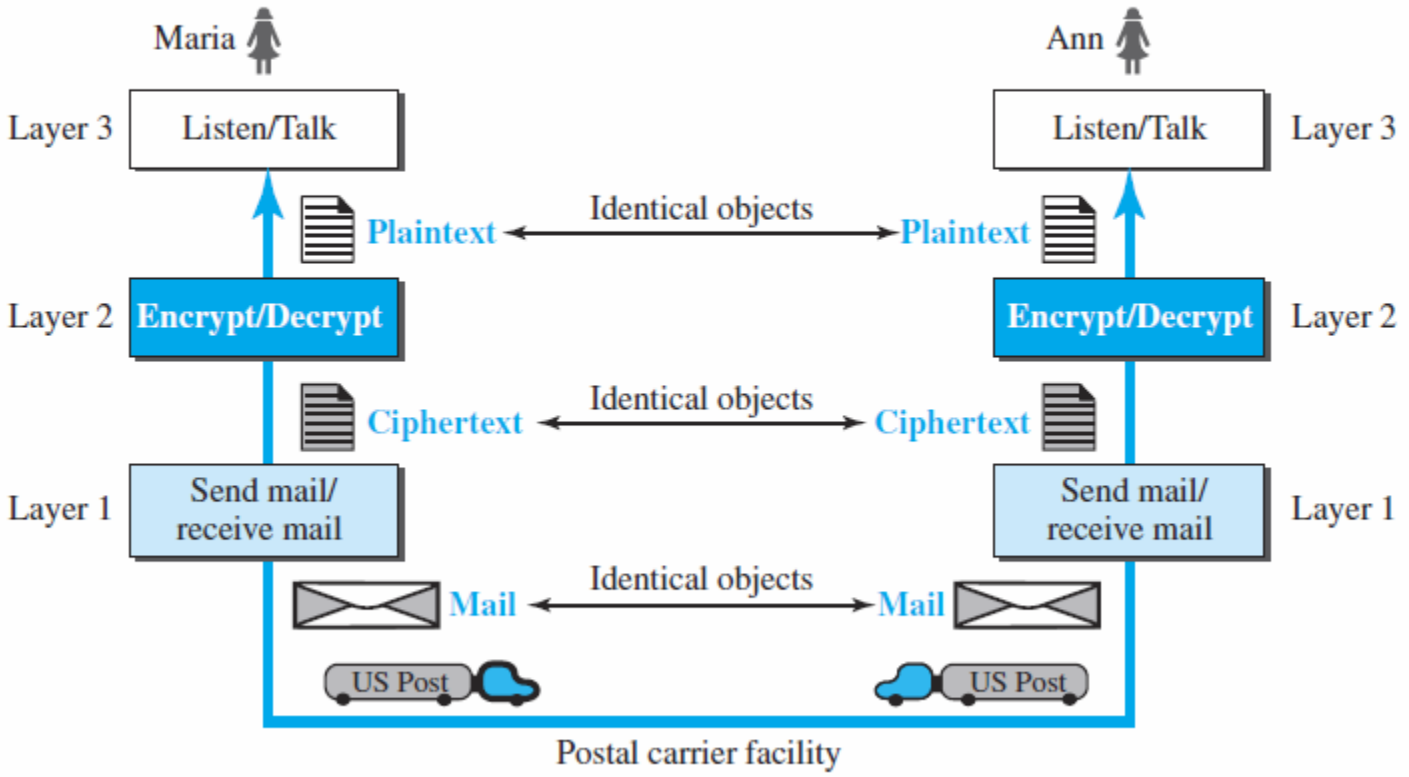
- In this case, the communication is normally very **formal** and limited to the subject being taught.

2.1.1 Scenarios Contd.

- ***Second Scenario***
- In the second scenario, we assume that Ann is offered a higher-level position in her company, but needs to **move to another branch** located in a city very far from Maria.
- The two friends still want to continue their communication and **exchange ideas** because they have come up with an **innovative project to start a new business** when they both retire.
- They decide to continue their conversation using **regular mail** through the post office.
- However, they do not want their **ideas to be revealed by other people** if the letters are intercepted.
- They agree on an **encryption/decryption** technique.
- The **sender** of the letter **encrypts** it to make it unreadable by an intruder; the **receiver** of the letter **decrypts** it to get the original letter.
- Assume that Maria and Ann use one technique that makes it hard to decrypt the letter if one does not have the key for doing so.
- Now we can say that the communication between Maria and Ann takes place in **three layers**, as shown in Figure 2.2.
- We assume that Ann and Maria each have **three machines** (or robots) that can perform the task at each layer.

2.1.1 Scenarios Contd.

Figure 2.2 *A three-layer protocol*



2.1.1 Scenarios Contd.

- Let us assume that Maria sends the **first letter** to Ann.
- Maria talks to the machine at the **third layer** as though the machine is Ann and is listening to her.
- The third layer machine **listens** to what Maria says and creates the **plaintext** (a letter in English), which is passed to the second layer machine.
- The **second layer machine** takes the plaintext, encrypts it, and creates the **ciphertext**, which is passed to the first layer machine.
- The **first layer machine**, presumably a robot, takes the ciphertext, puts it in an envelope, adds the sender and receiver addresses, and **mails it.**
- At Ann's side, the **first layer** machine picks up the letter from Ann's **mail box**, recognizing the letter from Maria by the sender address. The machine takes out the ciphertext from the envelope and delivers it to the second layer machine.
- The **second layer machine decrypts** the message, creates the plaintext, and passes the plaintext to the third-layer machine.
- The **third layer machine** takes the plaintext and **reads** it as though Maria is speaking.

2.1.1 Scenarios Contd.

- **Protocol layering** enables us to **divide a complex task** into several smaller and simpler tasks.
- For example, in Figure 2.2, we could have used **only one machine to do the job of all three machines.**
- However, if Maria and Ann decide that the **encryption/ decryption** done by the machine is **not enough to protect** their secrecy, they would have to **change the whole machine.**
- In the present situation, they need to **change only the second layer machine**; the other two can remain the same.
- This is referred to as ***modularity***.
- Modularity in this case means **independent layers**.
- A layer (**module**) can be defined as a **black box** with inputs and outputs, without concern about how inputs are changed to outputs.
- If two machines provide the same outputs when given the same inputs, they can **replace** each other.

2.1.1 Scenarios Contd.

- For example, Ann and Maria **can buy the second layer machine from two different manufacturers**.
- As long as the two machines create the same ciphertext from the same plaintext and vice versa, they do the job.
- One of the **advantages** of protocol layering is that it allows us to **separate the services from the implementation.**
- A layer needs to be able to receive a set of services from the lower layer and to give the services to the upper layer; we don't care about **how the layer is implemented.**
- For example, Maria may decide not to buy the machine (robot) for the first layer; she can do the job herself. As long as Maria can do the tasks provided by the first layer, in both directions, the communication system works.
- Communication does not always use only two end systems; there are **intermediate systems** that need only some layers, but not all layers.
- If we did not use protocol layering, we would have to make each intermediate system as complex as the end systems, which makes the whole system more expensive.

2.1.1 Scenarios Contd.

- Is there any **disadvantage** to protocol layering?
- One can argue that **having a single layer makes the job easier.**
- There is no need for each layer to provide a service to the upper layer and give service to the lower layer.
- For example, Ann and Maria could find or build one machine that could do all three tasks.
- However, as mentioned above, if one day they found that their code was broken, each would have to **replace the whole machine** with a new one instead of just changing the machine in the second layer.

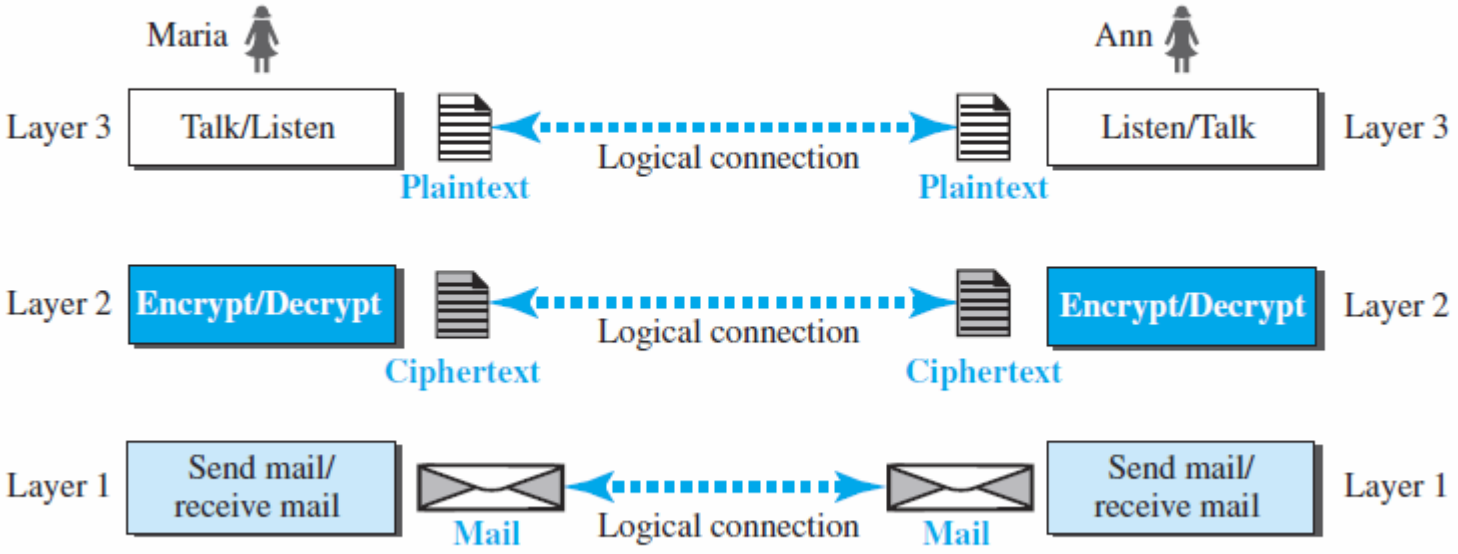
2.1.2 Principles of Protocol Layering

- ***First Principle***
- If we want bidirectional communication, we need to make each layer so that it is able to perform **two opposite tasks**, one in each direction.
  - For example, the third layer task is to **listen** (in one direction) and ***talk*** *(in the other* direction).
  - The second layer needs to be able to **encrypt and decrypt**.
  - The first layer needs to **send and receive mail.**
- ***Second Principle***
- The two objects under each layer at both sites should be **identical**.
  - For example, the object under layer 3 at both sites should be a **plaintext letter**.
  - The object under layer 2 at both sites should be a **ciphertext letter**.
  - The object under layer 1 at both sites should be a **piece of mail.**

2.1.3 Logical Connections

- After following the above two principles, we can think about logical connection between each layer as shown in Figure 2.3.
- This means that we have **layer-to-layer communication**.
- Maria and Ann can think that there is a logical (imaginary) connection at each layer through which they can send the object created from that layer.

Figure 2.3 *Logical connection between peer layers*



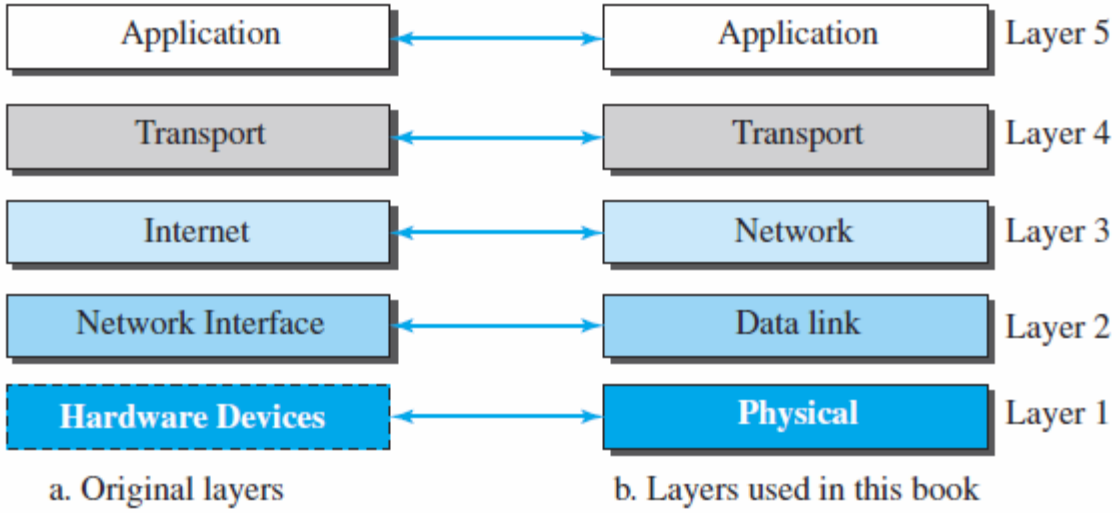
2.2 TCP/IP PROTOCOL SUITE

- **TCP/IP is a protocol suite** (a set of protocols organized in different layers) used in the Internet today.
- It is a hierarchical protocol made up of **interactive modules**, each of which provides a specific functionality.

- The term **hierarchical** means that each **upper level protocol is supported by the services provided by** one or more **lower level protocols**.

- TCP/IP is thought of as a **five-layer model.** Figure 2.4 shows both configurations.

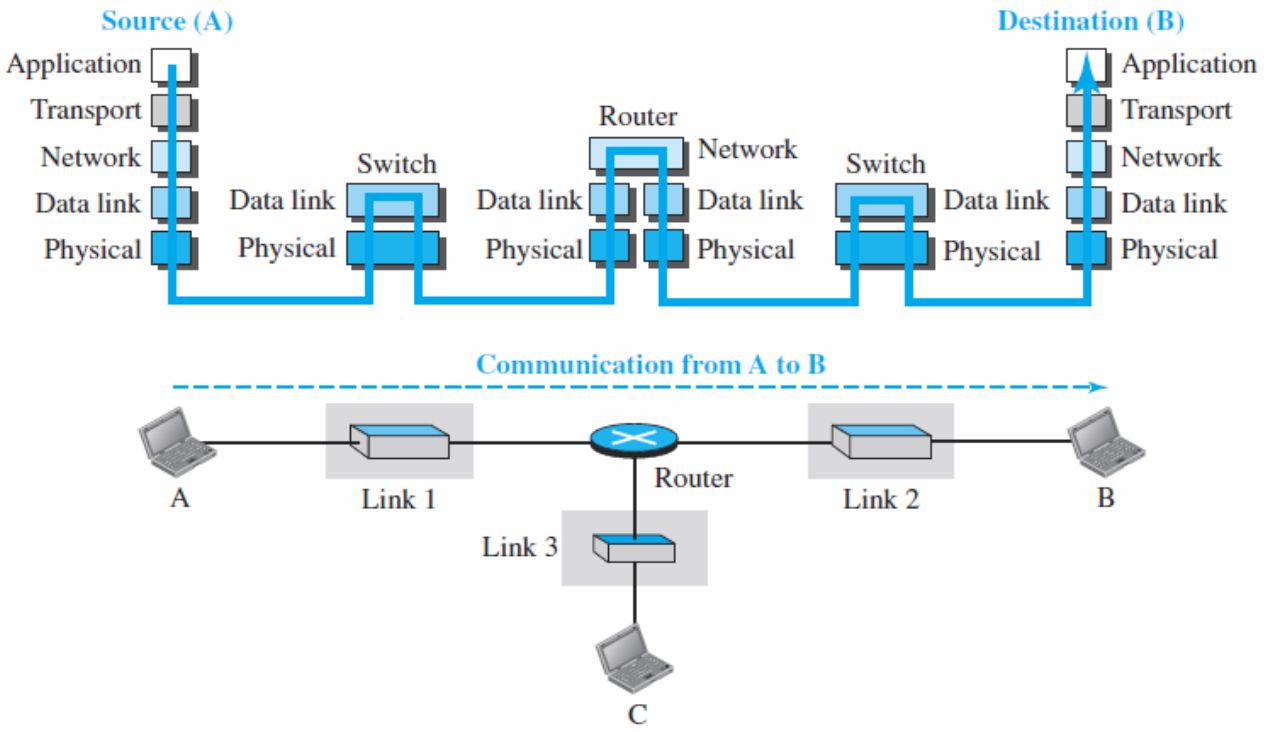
Figure 2.4 *Layers in the TCP/IP protocol suite*



2.2.1 Layered Architecture

- To show how the layers in the TCP/IP protocol suite are involved in communication between two hosts, we assume that we want to use the suite in a small internet made up of three LANs (links), each with a link-layer switch. We also assume that the links are connected by one router, as shown in Figure 2.5.

Figure 2.5 *Communication through an internet*



2.2.1 Layered Architecture Contd.

- Let us assume that computer A communicates with computer B.
- As the figure shows, we have **five communicating devices** in this communication: source host (computer A), the link-layer switch in link 1, the router, the link-layer switch in link 2, and the destination host (computer B).
- Each device is involved with a set of layers depending on the role of the device in the internet.
- The **two hosts are involved in all five layers;** the **source host** needs to **create a message** in the application layer and send it down the layers so that it is physically sent to the destination host.
- The **destination host** needs to **receive** the communication at the physical layer and then deliver it through the other layers to the application layer.

2.2.1 Layered Architecture Contd.

- The **router is involved in only three layers**; there is no transport or application layer in a router as long as the router is used only for routing.

- Although a router is always involved in one network layer, it is **involved in n combinations of link** and physical layers in which n is the number of links the router is connected to.

- The reason is that each link may use its own data-link or physical protocol.

- For example, in the above figure, the router is involved in three links, but the message sent from source A to destination B is involved in two links.

- **Each link may be using different link-layer and physical-layer protocols**; the router needs to receive a packet from link 1 based on one pair of protocols and deliver it to link 2 based on another pair of protocols.

- **A link-layer switch** in a link, however, is involved only in **two layers**, data-link and physical.

- Although each switch in the above figure has two different connections, the connections are in the same link, which **uses only one set of protocols**. This means that, unlike a router, a link-layer switch is involved only in one data-link and one physical layer.

2.2.3 Description of Each Layer

- ***Physical Layer***
- The physical layer is responsible for **carrying individual bits** in a frame across the link.

- Although the physical layer is the lowest level in the TCP/IP protocol suite, the communication between two devices at the physical layer is still a **logical communication because there is another, hidden layer, the transmission media**, under the physical layer.

- Two devices are connected by a **transmission medium (cable or air).**

- We need to know that the transmission medium does not carry bits; it carries **electrical or optical signals**.

- So the bits received in a frame from the data-link layer are transformed and sent through the transmission media, but we can think that the logical unit between two physical layers in two devices is a **bit**.

- There are several protocols that **transform a bit to a signal**.

2.2.3 Description of Each Layer Contd.

- ***Data-link Layer***
- The routers are responsible for choosing the *best links.* However, when the next link to travel is determined by the router, the **data-link layer is responsible for taking the datagram and moving it across the link.**

- The **link can** be a **wired LAN** with a link-layer switch, a **wireless LAN**, a **wired WAN**, or a **wireless WAN**.

- We can also have **different protocols** used with any link type.

- In each case, the data-link layer is responsible for **moving the packet through the link.**

- The data-link layer takes a datagram and encapsulates it in a packet called a ***frame***.

- Each **link-layer protocol** may provide a **different service.**

- Some link-layer protocols provide complete **error detection and correction**, some provide only error correction.

2.2.3 Description of Each Layer Contd.

- ***Network Layer***
- The network layer is responsible for creating a connection between the source computer and the destination computer.

- The communication at the network layer is **host-to-host**.

- However, since there can be several routers from the source to the destination, **the routers in the path are responsible for choosing the best route for each packet**.

- The network layer is responsible for host-to-host communication and routing the packet through possible routes.

- why we need a separate network layer?
  - **Separation of different tasks** between different layers.
  - The **routers do not need the application and transport layers**.
  - Separating the tasks allows us to **use fewer protocols** on the routers.

2.2.3 Description of Each Layer Contd.

- The network layer in the Internet includes the main protocol, **Internet Protocol** (IP), that defines the format of the packet, called a datagram at the network layer.
- IP also defines the **format** and the **structure of addresses** used in this layer.
- IP is also responsible for **routing** a packet from its source to its destination, which is achieved by each router forwarding the datagram to the next router in its path.
- IP is a **connectionless protocol** that provides **no flow control, no error control, and no congestion control services.**
- This means that if any of theses services is required for an application, the application should rely only on the transport-layer protocol.
- The network layer also includes **unicast** (one-to-one) and **multicast** (one-to-many) routing protocols.
- A routing protocol creates **forwarding tables** for routers to help them in the routing process.

2.2.3 Description of Each Layer Contd.

- The network layer also has some auxiliary protocols that help IP in its delivery and routing tasks.
- The **Internet Control Message Protocol (ICMP)** helps IP to report some problems when routing a packet.
- The **Internet Group Management Protocol (IGMP**) is another protocol that helps IP in multicasting.
- **The Dynamic Host Configuration Protocol (DHCP)** helps IP to get the network-layer address for a host.
- The **Address Resolution Protocol (ARP**) is a protocol that helps IP to find the link-layer address of a host or a router when its network-layer address is given.

2.2.3 Description of Each Layer Contd.

- ***Transport Layer***
- The **logical connection** at the transport layer is also **end-to-end**.

- The transport layer at the source host gets the message from the application layer, encapsulates it in a transport layer packet (called a ***segment*** *or a* ***user datagram*** *in different protocols) and sends it,* through the logical connection, to the transport layer at the destination host.

- In other words, the transport layer is responsible for **giving services to the application layer** to get a message from an application program running on the source host and deliver it to the corresponding application program on the destination host.

- why we need an end-to-end transport layer ?
  - The reason is the **separation of tasks and duties.**
  - The transport layer should be **independent** of the application layer.
  - Each application program can **use the protocol that best matches its requirement**.

2.2.3 Description of Each Layer Contd.

- The main protocol, **Transmission Control Protocol (TCP**), is a connection-oriented protocol that first establishes a logical connection between transport layers at two hosts before transferring data.

- It creates a **logical pipe** between two TCPs for transferring a stream of bytes.

- TCP provides
  - **flow control** (matching the sending data rate of the source host with the receiving data rate of the destination host to prevent overwhelming the destination),
  - **error control** (to guarantee that the segments arrive at the destination without error and resending the corrupted ones), and
  - **congestion control** to reduce the loss of segments due to congestion in the network.

- The other common protocol, **User Datagram Protocol (UDP),** is a connectionless protocol that transmits user datagrams without first creating a logical connection.

2.2.3 Description of Each Layer Contd.

- ***Application Layer***
- As Figure 2.6 shows, the logical connection between the two application layers is **end-to- end**.

- Communication at the application layer is between two ***processes*** *(two programs* running at this layer).

- To communicate, a process sends a request to the other process and receives a response.

- **Process-to-process communication** is the duty of the application layer.

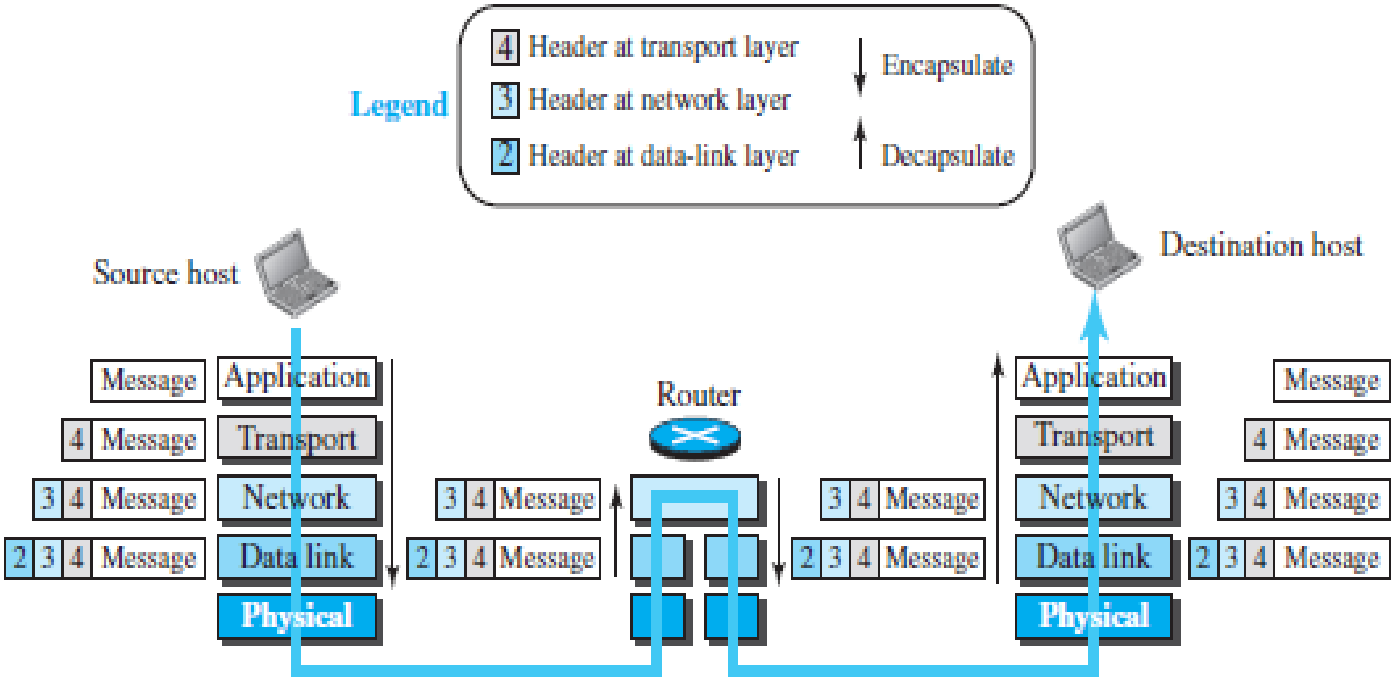
- The application layer in the Internet includes many **predefined protocols**, but a **user can also create a pair of processes** to be run at the two hosts.

2.2.3 Description of Each Layer Contd.

- The **Hypertext Transfer Protocol (HTTP**) is a vehicle for accessing the World Wide Web (WWW).
- The **Simple Mail Transfer Protocol (SMTP**) is the main protocol used in electronic mail (e-mail) service.
- The **File Transfer Protocol (FTP)** is used for transferring files from one host to another.
- The **Terminal Network (TELNET)** and Secure Shell (SSH) are used for accessing a site remotely.
- The **Simple Network Management Protocol (SNMP)** is used by an administrator to manage the Internet at global and local levels.
- The **Domain Name System (DNS)** is used by other protocols to find the network-layer address of a computer.
- The **Internet Group Management Protocol (IGMP)** is used to collect membership in a group.

2.2.4 Encapsulation and Decapsulation

Figure 2.8 *Encapsulation/Decapsulation*



2.2.4 Encapsulation and Decapsulation Contd.

- Figure 2.8 shows the concept of encapsulation and decapsulation for small internet.
- In Figure 2.8, we show the encapsulation in the source host, decapsulation in the destination host, and encapsulation and decapsulation in the router.

- **Encapsulation at the Source Host**
- At the source, we have only encapsulation.
  - 1. At the application layer, the data to be exchanged is referred to as a message. A message normally does not contain any header or trailer, but if it does, we refer to the whole as the **message**. The message is passed to the transport layer.

  - 2. The transport layer takes the message as the payload, the load that the transport layer should take care of. It adds the transport layer header to the payload, which contains the identifiers of the source and destination application programs that want to communicate plus some more information that is needed for the end-to-end delivery of the message, such as information needed for flow, error control, or congestion control. The result is the transport-layer packet, which is called the **segment** (in TCP) and the user **datagram** (in UDP). The transport layer then passes the packet to the network layer.

2.2.4 Encapsulation and Decapsulation Contd.

- 3. The network layer takes the transport-layer packet as data or payload and adds its own header to the payload. The header contains the addresses of the source and destination hosts and some more information used for error checking of the header, fragmentation information, and so on. The result is the network-layer packet, called a **datagram**. The network layer then passes the packet to the data-link layer.

- 4. The data-link layer takes the network-layer packet as data or payload and adds its own header, which contains the link-layer addresses of the host or the next hop (the router). The result is the link-layer packet, which is called a frame. The **frame** is passed to the physical layer for transmission.

2.2.4 Encapsulation and Decapsulation Contd.

- **Decapsulation and Encapsulation at the Router**

- At the router, we have both decapsulation and encapsulation because the router is connected to two or more links.

  - 1. After the set of bits are delivered to the data-link layer, this layer decapsulates the datagram from the frame and passes it to the network layer.

  - 2. The network layer only inspects the source and destination addresses in the datagram header and consults its forwarding table to find the next hop to which the datagram is to be delivered. The contents of the datagram should not be changed by the network layer in the router unless there is a need to fragment the datagram if it is too big to be passed through the next link. The datagram is then passed to the data-link layer of the next link.

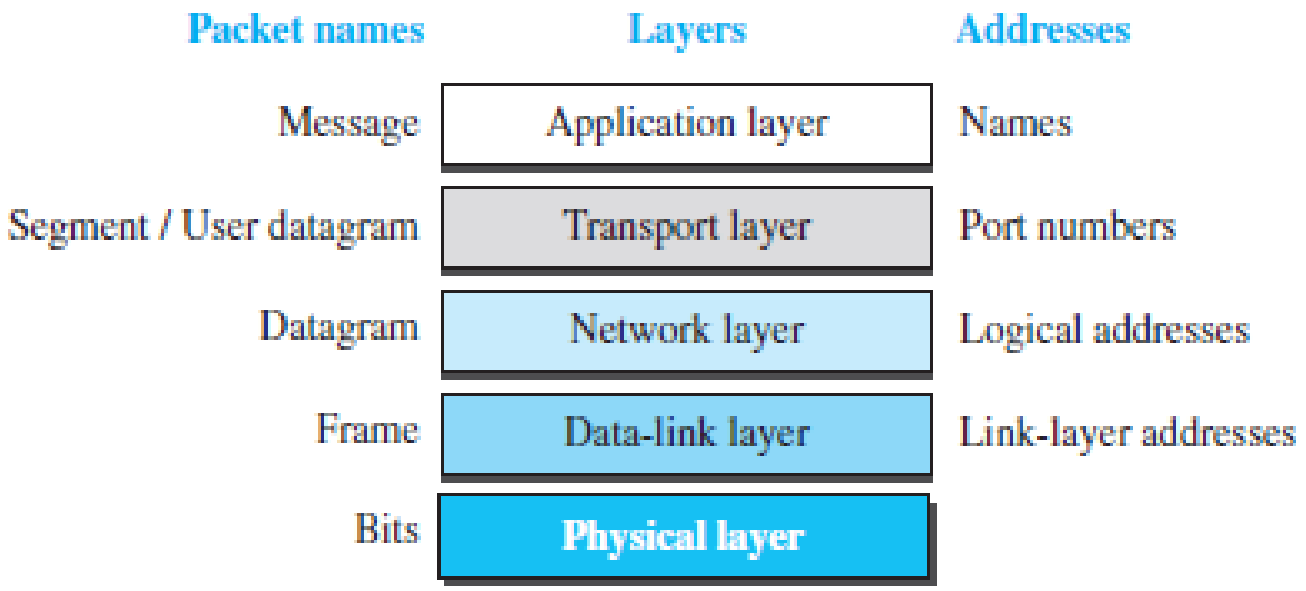
  - 3. The data-link layer of the next link encapsulates the datagram in a frame and passes it to the physical layer for transmission.

2.2.4 Encapsulation and Decapsulation Contd.

- **Decapsulation at the Destination Host**
  - At the destination host, each layer only decapsulates the packet received, removes the payload, and delivers the payload to the next-higher layer protocol until the message reaches the application layer. Decapsulation in the host involves error checking.

2.2.5 Addressing

Figure 2.9 *Addressing in the TCP/IP protocol suite*



2.2.5 Addressing Contd.

- Any communication that involves two parties needs two addresses: source address and destination address.
- The physical layer does not need addresses; the unit of data exchange at the physical layer is a bit, which definitely cannot have an address.
- Figure 2.9 shows the addressing at each layer.
- As the figure shows, there is a relationship between the layer, the address used in that layer, and the packet name at that layer.
- At the application layer, we normally use names to define the site that provides services, such as someorg.com, or the e-mail address, such as somebody@coldmail.com.

2.2.5 Addressing Contd.

- At the transport layer, addresses are called port numbers, and these define the application-layer programs at the source and destination.

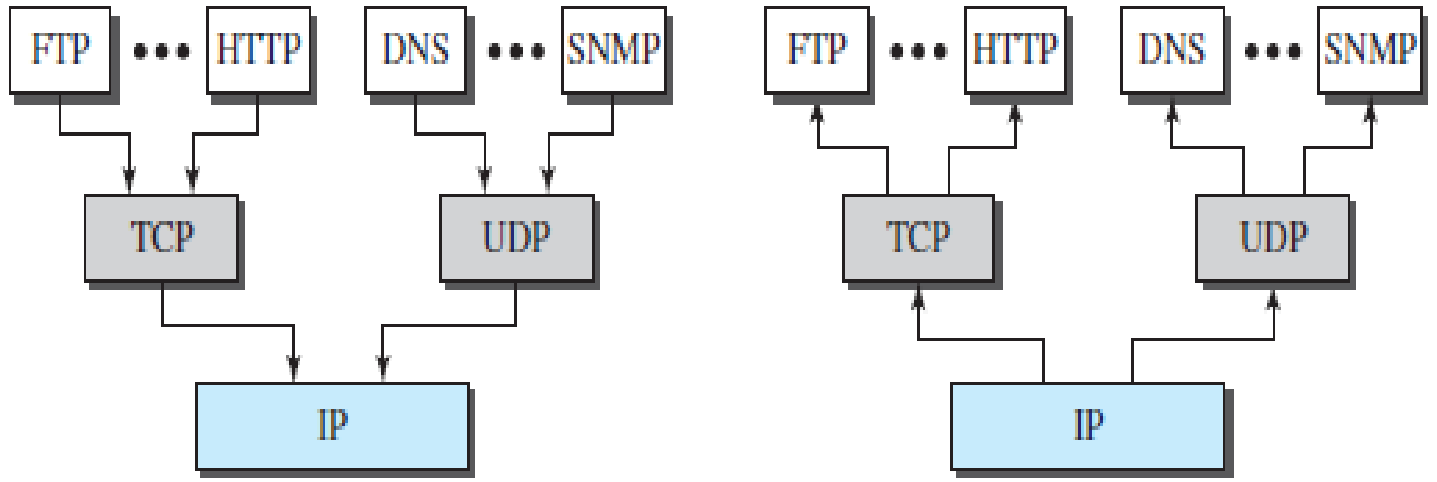
- Port numbers are local addresses that distinguish between several programs running at the same time.

- At the network-layer, the addresses are global, with the whole Internet as the scope.

- A network-layer address uniquely defines the connection of a device to the Internet. The link-layer addresses, sometimes called MAC addresses, are locally defined addresses, each of which defines a specific host or router in a network (LAN or WAN).

2.2.6 Multiplexing and Demultiplexing

Figure 2.10 *Multiplexing and demultiplexing*



a. Multiplexing at source

b. Demultiplexing at destination

2.2.6 Multiplexing and Demultiplexing Contd.

- Since the TCP/IP protocol suite uses several protocols at some layers, we can say that we have multiplexing at the source and demultiplexing at the destination.

- Multiplexing in this case means that a protocol at a layer can encapsulate a packet from several next-higher layer protocols (one at a time)

- Demultiplexing means that a protocol can decapsulate and deliver a packet to several next-higher layer protocols (one at a time).

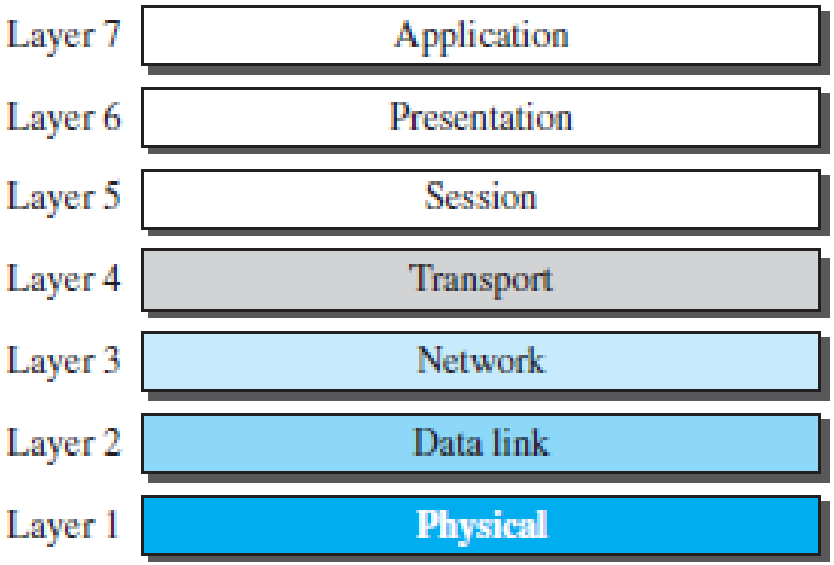
- Figure 2.10 shows the concept of multiplexing and demultiplexing at the three upper layers.

2.2.6 Multiplexing and Demultiplexing Contd.

- To be able to multiplex and demultiplex, a protocol needs to have a field in its header to identify to which protocol the encapsulated packets belong.
- At the transport layer, either UDP or TCP can accept a message from several application-layer protocols.
- At the network layer, IP can accept a segment from TCP or a user datagram from UDP.
- IP can also accept a packet from other protocols such as ICMP, IGMP, and so on.
- At the data-link layer, a frame may carry the payload coming from IP or other protocols such as ARP.

2.3 THE OSI MODEL

Figure 2.11 *The OSI model*



2.3 THE OSI MODEL Contd.

- Although, when speaking of the Internet, everyone talks about the TCP/IP protocol suite, this suite is not the only suite of protocols defined.
- Established in 1947, the **International Organization for Standardization (ISO)** is a multinational body dedicated to worldwide agreement on international standards.
- Almost three-fourths of the countries in the world are represented in the ISO.
- An ISO standard that covers all aspects of network communications is the **Open Systems Interconnection (OSI) model**.
- It was first introduced in the late 1970s.
- An open system is a set of protocols that allows any two different systems to communicate regardless of their underlying architecture.

2.3 THE OSI MODEL Contd.

- The purpose of the OSI model is to show how to facilitate communication between different systems without requiring changes to the logic of the underlying hardware and software.

- The OSI model is not a protocol; it is a model for understanding and designing a network architecture that is flexible, robust, and interoperable.

- The OSI model was intended to be the basis for the creation of the protocols in the OSI stack.

- The OSI model is a layered framework for the design of network systems that allows communication between all types of computer systems.

- It consists of seven separate but related layers, each of which defines a part of the process of moving information across a network .

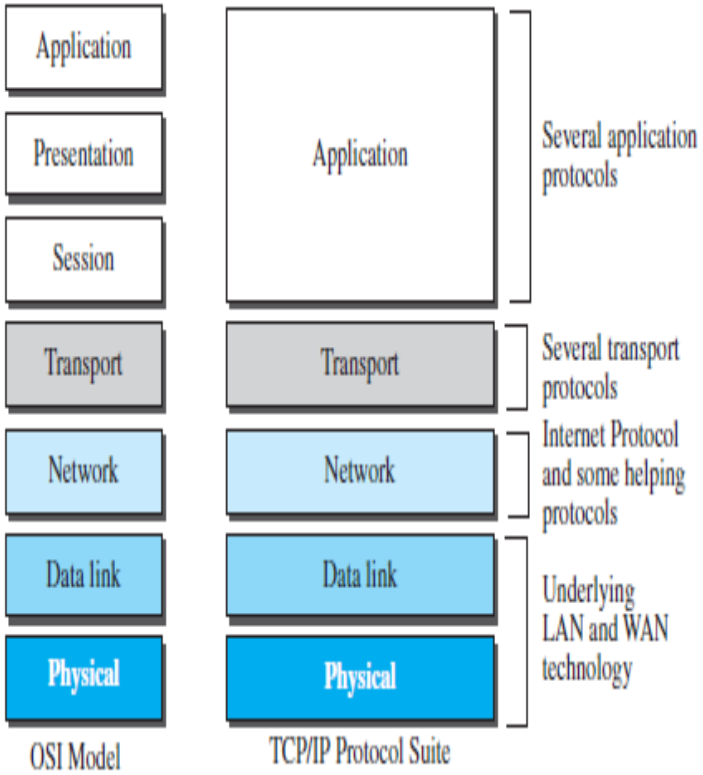
2.3.1 OSI versus TCP/IP

- The application layer in TCP/IP protocol suite is usually considered to be the combination of three layers in the OSI model, as shown in Figure 2.12

- Two reasons were mentioned for this decision. First, TCP/IP has more than one transport-layer protocol. Some of the functionalities of the session layer are available in some of the transport-layer protocols.

- Second, the application layer is not only one piece of software. Many applications can be developed at this layer. If some of the functionalities mentioned in the session and presentation layers are needed for a particular application, they can be included in the development of that piece of software.

Figure 2.12 *TCP/IP and OSI model*



2.3.2 Lack of OSI Model's Success

- The OSI model appeared after the TCP/IP protocol suite.
- Most experts were at first excited and thought that the TCP/IP protocol would be fully replaced by the OSI model.
- This did not happen for several reasons, but we describe only three, which are agreed upon by all experts in the field.
- First, OSI was completed when TCP/IP was fully in place and a lot of time and money had been spent on the suite; changing it would cost a lot.
- Second, some layers in the OSI model were never fully defined. For example, although the services provided by the presentation and the session layers were listed in the document, actual protocols for these two layers were not fully defined, nor were they fully described, and the corresponding software was not fully developed.
- Third, when OSI was implemented by an organization in a different application, it did not show a high enough level of performance to entice the Internet authority to switch from the TCP/IP protocol suite to the OSI model.

END