**COMPUTER NETWORKS**

* A **network** is a collection of computers, servers, mainframes, network devices, peripherals, or other devices connected to one another to allow the sharing of data.
* An excellent example of a network is the Internet, which connects millions of people all over the world.

**What was the first computer network?**

* One of the first computer networks to use [packet switching](https://www.computerhope.com/jargon/p/packetsw.htm), [ARPANET](https://www.computerhope.com/jargon/a/arpanet.htm) was developed in the mid-1960s and is considered to be the direct predecessor of the modern Internet.
* The first ARPANET message was sent on October 29, 1969.

# The OSI Model - Features, Principles and Layers

* There are n numbers of users who use computer network and are located over the world.
* So to ensure, national and worldwide data communication, systems must be developed which are compatible to communicate with each other ISO has developed a standard.
* ISO stands for **International organization of Standardization**.
* This is called a model for **Open System Interconnection** (OSI) and is commonly known as OSI model.
* The ISO-OSI model is a seven layer architecture. It defines seven layers or levels in a complete communication system. They are:
1. Application Layer
2. Presentation Layer
3. Session Layer
4. Transport Layer
5. Network Layer
6. Datalink Layer
7. Physical Layer



Feature of OSI Model

1. Big picture of communication over network is understandable through this OSI model.
2. We see how hardware and software work together.
3. We can understand new technologies as they are developed.
4. Troubleshooting is easier by separate networks.
5. Can be used to compare basic functional relationships on different networks.

## Principles of OSI Reference Model

1. A layer should be created where a different abstraction is needed.
2. Each layer should perform a well-defined function.
3. The function of each layer should be chosen with an eye toward defining internationally standardized protocols.
4. The layer boundaries should be chosen to minimize the information flow across the interfaces.
5. The number of layers should be large enough that distinct functions need not be thrown together in the same layer out of necessity and small enough that architecture does not become unwieldy.

Introduction to Transport Layer

* The transport layer in the TCP/IP suite is located between the application layer and the network layer.
* It provides services to the application layer and receives services from the network layer.
* The transport layer acts as a liaison between a client program and a server program, a process-to-process connection.
* The transport layer is the heart of the OSI protocol suite.
* It is the end-to-end logical vehicle for transferring data from one point to another in the Internet.

We have divided this chapter into two sections:

* We normally require from the transport layer, such as process-to-process communication, addressing, multiplexing and demultiplexing , error, flow, and congestion control.
* We then show that the transport-layer protocols are divided into two categories:
	1. connectionless
	2. connection-oriented.
* These protocols concentrate on flow and error control services provided by an actual transport layer protocol.
* Understanding these protocols helps us better understand the design of the transport-layer protocols in the Internet, such as UDP, TCP, and SCTP.
* It provides a process-to-process communication between two application layers, one at the local host and the other at the remote host.
* Communication is provided using a logical connection, which means that the two application layers, which can be located in different parts of the globe, assume that there is an imaginary direct connection through which they can send and receive messages.



PROCESS TOPROCESS COMMUNICATION

* The first duty of a transport-layer protocol is to provide **process-to-process communication.**
* A process is an application-layer entity (running program) that uses the services of the transport layer.
* The network layer is responsible for communication at the computer level (host-to-host communication).
* A network-layer protocol can deliver the message only to the destination computer.
* However, this is an incomplete delivery.The message still needs to be handed to the correct process. This is where a transport-layer protocol takes over.
* A transport-layer protocol is responsible for delivery of the message to the appropriate process.



**Addressing: Port Numbers**

* A process on the local host, called a *client,* needs services from a process usually on the remote host, called a *server.*
* The local host and the remote host are defined using IP addresses.
* To define the processes, we need second identifiers, called ***port numbers.*** In the TCP/IP protocol suite, the port numbers are integers between 0 and 65,535 (16 bits).
* The client program defines itself with a port number, called the ***ephemeral port number.*** The word *ephemeral* means “short-lived” and is used because the life of a client is normally short.
* TCP/IP has decided to use universal port numbers for servers; these are called ***well-known port numbers.***



* **IP address** is address of the system in the Network.
* **Port** is address of the service within the System.

**Socket Addresses**

* A transport-layer protocol in the TCP suite needs both the 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.

****

* To use the services of the transport layer in the Internet, we need a pair of socket addresses: the client socket address and the server socket address.

**Encapsulation and Decapsulation**

* **Encapsulation** happens at the sender site.
* When a process has a message to send, it passes the message to the transport layer along with a pair of socket addresses and some other pieces of information, which depend on the transport-layer protocol.
* The transport layer receives the data and adds the transport-layer header.
* The packets at the transport layer in the Internet are called *user datagrams, segments,* or *packets,* depending on what transport-layer protocol we use.
* **Decapsulation** happens at the receiver site.
* When the message arrives at the destination transport layer, the header is dropped and the transport layer delivers the message to the process running at the application layer.
* The sender socket address is passed to the process in case it needs to respond to the message received.

****

**Multiplexing and Demultiplexing**

* Whenever an entity accepts items from more than one source, this is referred to as ***multiplexing*** (many to one).
* whenever an entity delivers items to more than one source, this is referred to as ***demultiplexing*** (one to many).
* The transport layer at the source performs multiplexing; the transport layer at the destination performs demultiplexing.



* It shows communication between a client and two servers.
* Three client processes are running at the client site, P1, P2, and P3.
* The processes P1 and P3 need to send requests to the corresponding server process running in a server.
* The client process P2 needs to send a request to the corresponding server process running at another server.
* The transport layer at the client site accepts three messages from the three processes and creates three packets.
* It acts as a *multiplexer*. The packets 1 and 3 use the same logical channel to reach the transport layer of the first server.
* When they arrive at the server, the transport layer does the job of a *demultiplexer* and distributes the messages to two different processes.
* The transport layer at the second server receives packet 2 and delivers it to the corresponding process.

**Flow Control**

* Whenever an entity produces items and another entity consumes them, there should be a balance between production and consumption rates.
* If the items are produced faster than they can be consumed, the consumer can be overwhelmed and may need to discard some items.
* If the items are produced more slowly than they can be consumed, the consumer must wait, and the system becomes less efficient.
* Flow control is related to the first issue. We need to prevent losing the data items at the consumer site.

**Pushing or Pulling**

* Delivery of items from a producer to a consumer can occur in one of two ways: pushing or pulling.
* If the sender delivers items whenever they are produced⎯without a prior request from the consumer⎯the delivery is referred to as pushing.
* If the producer delivers the items after the consumer has requested them, the delivery is referred to as pulling.

****

The following is the content of a UDP header in hexadecimal format.

 **CB84000D001C001C**

**a.** What is the source port number?

The source port number is the first four hexadecimal digits (CB84)16, which means that

the source port number is 52100.

**b.** What is the destination port number?

The destination port number is the second four hexadecimal digits (000D)16, which means that the destination port number is 13.

**c.** What is the total length of the user datagram?

The third four hexadecimal digits (001C)16 define the length of the whole UDP packet as 28 bytes.

**d.** What is the length of the data?

The length of the data is the length of the whole packet minus the length of the header, or 28 8 20 bytes.

**UDP Services**

**Process-to-Process Communication**

* UDP provides process-to-process communication using **socket addresses,** a combination of IP addresses and port numbers.

**Connectionless Services**

* UDP provides a *connectionless service.*
* This means that each user datagram sent by UDP is an independent datagram.
* Each user datagram can travel on a different path.

***Error Control***

* There is no *error control* mechanism in UDP except for the checksum.

***Checksum***

* UDP checksum calculation includes three sections: a pseudoheader, the UDP header, and the data coming from the application layer
* If the checksum does not include the pseudoheader, a user datagram may arrive safe and sound. However, if the IP header is corrupted, it may be delivered to the wrong host.

**UDP Features**

* UDP is a connectionless protocol. Each UDP packet is independent from other packets sent by the same application program.
* This feature can be considered as an advantage or disadvantage depending on the application requirements.
* It is an advantage if, for example, a client application needs to send a short request to a server and to receive a short response.
* If the request and response can each fit in a single user datagram, a connectionless service may be preferable.
* The overhead to establish and close a connection may be significant in this case.
* In the connection oriented service, to achieve the above goal, at least 9 packets are exchanged between the client and the server; in connectionless service only 2 packets are exchanged.
* The connectionless service provides less delay; the connection-oriented service creates more delay. If delay is an important issue for the application, the connectionless service is preferred.

**Typical Applications**

* 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
* 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.
* UDP is a suitable transport protocol for multicasting. Multicasting capability is embedded in the UDP software but not in the TCP software.
* UDP is used for management processes such as SNMP.
* UDP is used for some route updating protocols such as Routing Information Protocol (RIP).
* UDP is normally used for interactive real-time applications that cannot tolerate uneven delay between sections of a received message.

**TRANSMISSION CONTROL PROTOCOL**

* **Transmission Control Protocol (TCP)** is a connection-oriented, reliable protocol.
* TCP explicitly defines connection establishment, data transfer, and connection teardown phases to provide a connection-oriented service.
* TCP uses a combination of GBN (Go-Back-N ARQ) and SR(selective repeat) protocols to provide reliability.
* To achieve this goal, TCP uses checksum (for error detection), retransmission of lost or corrupted packets, cumulative and selective acknowledgments, and timers.

**TCP SERVICES**

* TCP provides process-to-process communication using port numbers.
* TCP, on the other hand, allows the sending process to deliver data as a stream of bytes and allows the receiving process to obtain data as a stream of bytes.
* TCP creates an environment in which the two processes seem to be connected by an imaginary “tube” that carries their bytes across the Internet.
* The sending process produces (writes to) the stream and the receiving process consumes (reads from) it.

****

Sending and Receiving Buffers

* TCP needs buffers for storage.
* There are two buffers, the sending buffer and the receiving buffer, one for each direction.
* The figure shows the movement of the data in one direction.
* At the sender, the buffer has three types of chambers.
* The white section contains empty chambers that can be filled by the sending process (producer).
* The colored area holds bytes that have been sent but not yet acknowledged.
* The TCP sender keeps these bytes in the buffer until it receives an acknowledgment.
* The shaded area contains bytes to be sent by the sending TCP.
* The operation of the buffer at the receiver is simpler.
* The circular buffer is divided into two areas (shown as white and colored).
* The white area contains empty chambers to be filled by bytes received from the network.
* The colored sections contain received bytes that can be read by the receiving process.
* When a byte is read by the receiving process, the chamber is recycled and added to the pool of empty chambers.



***Segments***

******

* The network layer, as a service provider for TCP, needs to send data in packets, not as a stream of bytes.
* At the transport layer, TCP groups a number of bytes together into a packet called a *segment*.
* TCP adds a header to each segment (for control purposes) and delivers the segment to the network layer for transmission.
* The segments are encapsulated in an IP datagram and transmitted. This entire operation is transparent to the receiving process.
* All of these are handled by the TCP receiver with the receiving application process unaware of TCP’s activities.

****

***Source port address*.** This is a 16-bit field that defines the port number of the application program in the host that is sending the segment.

***Destination port address*.** This is a 16-bit field that defines the port number of the application program in the host that is receiving the segment.

***Sequence number.*** This 32-bit field defines the number assigned to the first byte of data contained in this segment. As we said before, TCP is a stream transport protocol. To ensure connectivity, each byte to be transmitted is numbered. The sequence number tells the destination which byte in this sequence is the first byte in the segment.

During connection establishment (discussed later) each party uses a random number generator to create an **initial sequence number** (ISN), which is usually different in each direction.

***Acknowledgment number.*** This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party. If the receiver of the segment has successfully received byte number *x* from the other party, it returns *x* 1 as the acknowledgment number. Acknowledgment and data can be piggybacked together.

***Header length.*** This 4-bit field indicates the number of 4-byte words in the TCP header. The length of the header can be between 20 and 60 bytes. Therefore, the value of this field is always between 5 (5 4 20) and 15 (15 4 60).

***Control.*** This field defines 6 different control bits or flags, as shown in Figure 24.8. One or more of these bits can be set at a time. These bits enable flow control, connection establishment and termination, connection abortion, and the mode of



***Window size.*** This field defines the window size of the sending TCP in bytes. Note that the length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes. This value is normally referred to as the receiving window (*rwnd*) and is determined by the receiver. The sender must obey the dictation of the receiver in this case.

❑ ***Checksum.*** This 16-bit field contains the checksum. The calculation of the checksum for TCP follows the same procedure as the one described for UDP. However, the use of the checksum in the UDP datagram is optional, whereas the use of the checksum for TCP is mandatory. The same pseudoheader, serving the same purpose, is added to the segment.

**A TCP Connection**

* TCP is connection-oriented.
* A connection-oriented transport protocol establishes a logical path between the source and destination. All of the segments belonging to a message are then sent over this logical path.
* Unlike TCP, IP is unaware of this retransmission. If a segment arrives out of order, TCP holds it until the missing segments arrive; IP is unaware of this reordering.
* In TCP, connection-oriented transmission requires three phases: connection establishment, data transfer, and connection termination.

**Connection Establishment**

* TCP transmits data in full-duplex mode. When two TCPs in two machines are connected, they are able to send segments to each other simultaneously.
* This implies that each party must initialize communication and get approval from the other party before any data are transferred.

**Three-Way Handshaking**

* The connection establishment in TCP is called ***three-way handshaking.***
* In our example, an application program, called the *client,* wants to make a connection with another application program, called the *server,*
* The process starts with the server. The server program tells its TCP that it is ready to accept a connection. This request is called a *passive open*.
* The client program issues a request for an *active open*. A client that wishes to connect to an open server tells its TCP to connect to a particular server.



* The client sends the first segment, a SYN segment, in which only the SYN flag is set.
* This segment is for synchronization of sequence numbers. The client in our example chooses a random number as the first sequence number and sends this number to the server.
* This sequence number is called the *initial sequence number (ISN).*
* Note that the SYN segment is a control segment and carries no data.
* However, it consumes one sequence number because it needs to be acknowledged.
* The server sends the second segment, a SYN ACK segment with two flag bits set as: SYN and ACK.
* This segment has a dual purpose. First, it is a SYN segment for communication in the other direction.
* The server uses this segment to initialize a sequence number for numbering the bytes sent from the server to the client.
* The server also acknowledges the receipt of the SYN segment from the client by setting the ACK flag and displaying the next sequence number it expects to receive from the client.
* It also needs to define the receive window size, *rwnd* (to be used by the client),
* The client sends the third segment. It acknowledges the receipt of the second segment with the ACK flag and acknowledgment number field.
* Note that the ACK segment does not consume any sequence numbers if it does not carry data, but some implementations allow this thirdsegment in the connection phase to carry the first chunk of data from the client.