# COMPUTER COMMUNICATION AND NETWORKS

BCA III SEM(NEP)

NOTES

Prepared By

Mr. Prabhu Kichadi, BE, MTECH

9880437187

# UNIT – V

**The Transport Layer and Application Layer: Elements of Transport service, Elements of Transport, protocols, Internet transport protocols (TCP & UDP), DNS, Electronic Mailing, and World Wide Web.**

## Transport Layer:

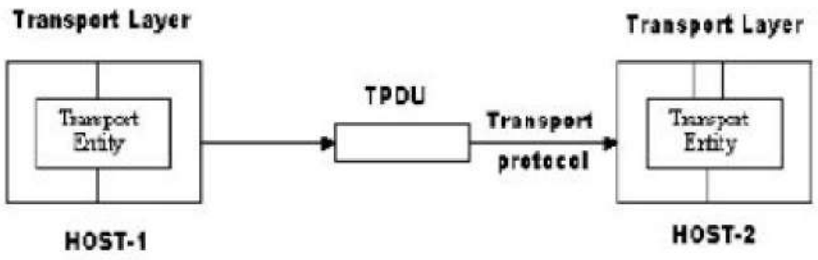
The network layer provides end-to-end packet delivery using data-grams or virtual circuits. The transport layer builds on the network layer to provide data transport from a process on a source machine to a process on a destination machine with a desired level of reliability that is independent of the physical networks currently in use. It provides the abstractions that applications need to use the network.

**Transport Entity:** The hardware and/or software which make use of services provided by the network layer, (within the transport layer) is called transport entity.

**Transport Service Provider:** Layers 1 to 4 are called Transport Service Provider. Transport Service User: The upper layers i.e., layers 5 to 7 are called Transport Service User.

**Transport Service Primitives**: Which allow transport users (application programs) to access the transport service.

**TPDU (Transport Protocol Data Unit):** Transmissions of message between 2 transport entities are carried out by TPDU. The transport entity carries out the transport service primitives by blocking the caller and sending a packet the service. Encapsulated in the payload of this packet is a transport layer message for the server's transport entity. The task of the transport layer is to provide reliable, cost-effective data transport from the source machine to the destination machine, independent of physical network or networks currently in use.



### TRANSPORT SERVICE

#### 1.Services Provided to the Upper Layers

The ultimate goal of the transport layer is to provide efficient, reliable, and cost-effective data transmission service to its users, normally processes in the application layer. To achieve this, the transport layer makes use of the services pro-vided by the network layer. The software and/or hardware within the transport layer that does the work is called the transport entity. The transport entity can be located in the operating system kernel, in a library package bound into network applications, in a separate user process, or even on the network interface card.

## Transport Service Primitives

- To allow users to access the transport service, the transport layer must provide some operations to application programs, that is, a transport service interface. Each transport service has its own interface.
- The transport service is similar to the network service, but there are also some important differences.
- The main difference is that the network service is intended to model the service offered by real networks. Real networks can lose packets, so the network service is generally unreliable.
- The (connection-oriented) transport service, in contrast, is reliable

As an example, consider two processes connected by pipes in UNIX. They assume the connection between them is perfect. They do not want to know about acknowledgements, lost packets, congestion, or anything like that. What they want is a 100 percent reliable connection. Process A puts data into one end of the pipe, and process B takes it out of the other.

A second difference between the network service and transport service is whom the services are intended for. The network service is used only by the transport entities. Consequently, the transport service must be convenient and easy to use.

***Table:4.1 - The primitives for a simple transport service.***

| Primitive | Packet sent | Meaning |
|---|---|---|
| LISTEN | (none) | Block until some process tries to connect |
| CONNECT | CONNECTION REQ. | Actively attempt to establish a connection |
| SEND | DATA | Send information |
| RECEIVE | (none) | Block until a DATA packet arrives |
| DISCONNECT | DISCONNECTION REQ. | This side wants to release the connection |

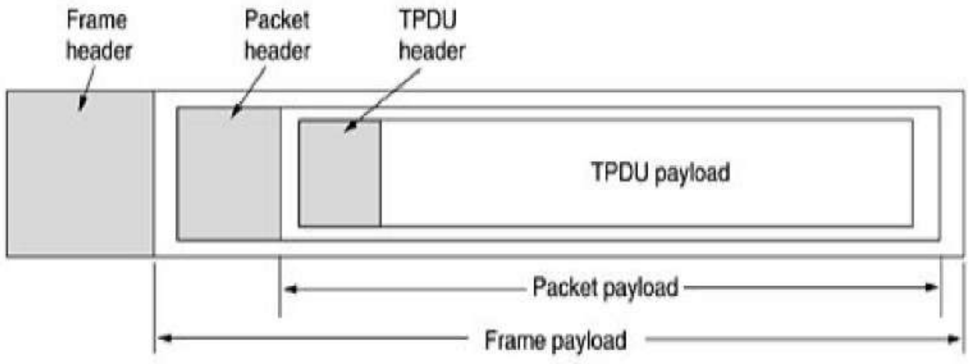


***Figure 4.2 - Nesting of TPDUs, packets, and frames***

- The term segment for messages sent from transport entity to transport entity.
- TCP, UDP and other Internet protocols use this term. Segments (exchanged by the transport layer) are contained in packets (exchanged by the network layer).
- These packets are contained in frames(exchanged by the data link layer).When a frame arrives, the data link layer processes the frame header and, if the destination address matches for local delivery, passes the contents of the frame payload field up to the network entity.

- The network entity similarly processes the packet header and then passes the contents of the packet payload up to the transport entity. This nesting is illustrated in Fig. 4.2

In fig. 4.3 each transition is triggered by some event, either a primitive executed by the local transport user or an incoming packet. For simplicity, we assume here that each TPDU is separately acknowledged. We also assume that a symmetric disconnection model is used, with the client going first. Please note that this model is quite unsophisticated. We will look at more realistic models later on.

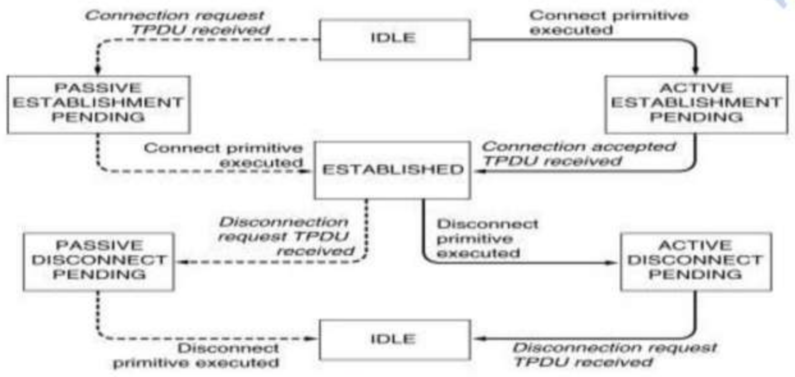

IDLE
Connection request TPDU received
Connect primitive executed
PASSIVE ESTABLISHMENT PENDING
ACTIVE ESTABLISHMENT PENDING
Connect primitive executed
ESTABLISHED
Connection accepted TPDU received
Disconnection request TPDU received
Disconnect primitive executed
PASSIVE DISCONNECT PENDING
ACTIVE DISCONNECT PENDING
IDLE
Disconnect primitive executed
Disconnection request TPDU received

***Figure 4.3 - A state diagram for a simple connection management scheme. Transitions labelled in italics are caused by packet arrivals. The solid lines show the client's state sequence. The dashed lines show the server's state sequence.***

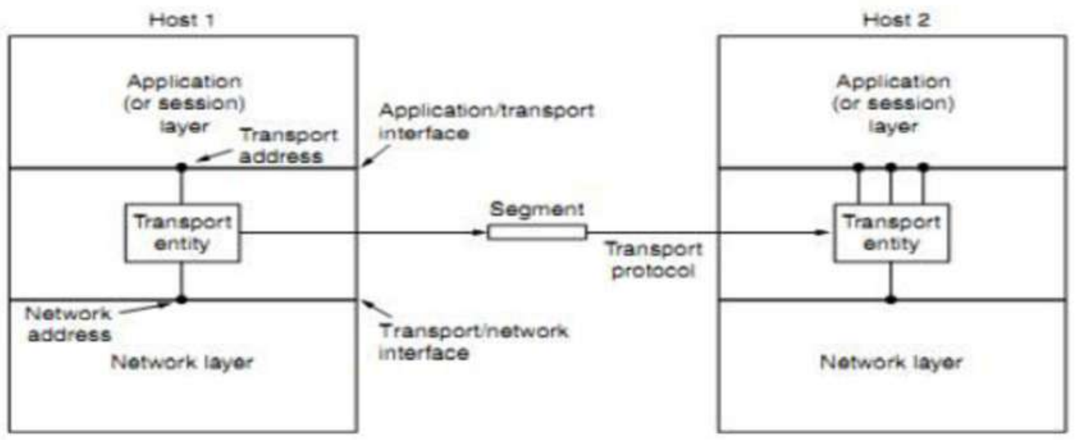


Fig The network, Application and transport layer

There are two types of network service

o Connection-oriented

o Connectionless Similarly,

there are also two types of transport service. The connection-oriented transport service is similar to the connection-oriented network service in many ways. In both cases, connections have three phases:

o Establishment

o Data transfer

o Release.

• Addressing and flow control are also similar in both layers. Furthermore, the connectionless transport service is also very similar to the connectionless network service.

• The bottom four layers can be seen as the transport service provider, whereas the upper layer(s) are the transport service user

## BERKLEY SOCKETS

These primitives are socket primitives used in Berkley UNIX for TCP. The socket primitives are mainly used for TCP. These sockets were first released as part of the Berkeley UNIX 4.2BSD software distribution in 1983. They quickly became popular.

The primitives are now widely used for Internet programming on many operating systems, especially UNIX -based systems, and there is a socket-style API for Windows called "winsock."

| Primitive | Meaning |
|---|---|
| SOCKET | Create a new communication end point |
| BIND | Attach a local address to a socket |
| LISTEN | Announce willingness to accept connections; give queue size |
| ACCEPT | Block the caller until a connection attempt arrives |
| CONNECT | Actively attempt to establish a connection |
| SEND | Send some data over the connection |
| RECEIVE | Receive some data from the connection |
| CLOSE | Release the connection |

***Figure 4.4 - The socket primitives for TCP.***

The first four primitives in the list are executed in that order by servers.
**The SOCKET** primitive creates a new endpoint and allocates table space for it within the transport entity. The parameter includes the addressing format to be used, the type of service desired and the protocol. Newly created sockets do not have network addresses.
**The BIND** primitive is used to connect the newly created sockets to an address. Once a server has bound an address to a socket, remote clients can connect to it.
**The LISTEN** call, which allocates space to queue incoming calls for the case that several clients try to connect at the same time.
The server executes an **ACCEPT** primitive to block waiting for an incoming connection.

## Elements of Transport, protocols:

The transport service is implemented by a transport protocol used between the two transport entities. The transport protocols resemble the data link protocols. Both have to deal with error control, sequencing, and flow control, among other issues.

The difference transport protocol and data link protocol depends upon the environment in which they are operated.

These differences are due to major dissimilarities between the environments in which the two protocols operate, as shown in Fig.

At the data link layer, two routers communicate directly via a physical channel, whether wired or wireless, whereas at the transport layer, this physical channel is replaced by the entire network. This difference has many important implications for the protocols.

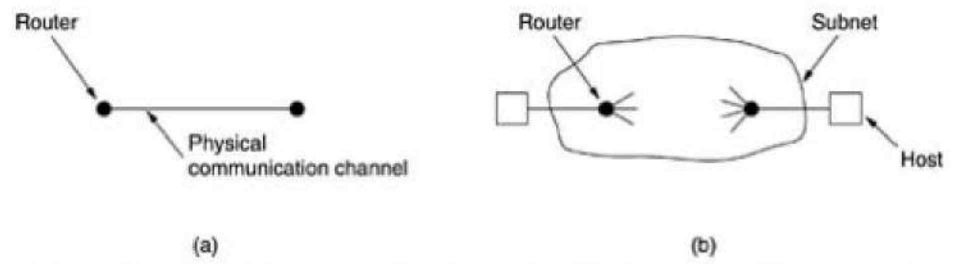


***Figure (a) Environment of the data link layer. (b) Environment of the transport layer.***

The elements of transport protocols are:

**1. Addressing**
**2. Connection Establishment.**
**3. Connection Release.**
**4. Error Control and Flow Control**
**5. Multiplexing.**

**1. Addressing:** When an application (e.g., a user) process wishes to set up a connection to a remote application process, it must specify which one to connect to. The method normally used is to define transport addresses to which processes can listen for connection requests. In the Internet, these endpoints are called ports.

There are two types of **Access Points.**

**TSAP (Transport Service Access Point)** to mean a specific endpoint in the transport layer.

The analogous endpoints in the network layer (i.e., network layer addresses) are not surprisingly **called NSAPs (Network Service Access Points).** IP addresses are examples of NSAPs.

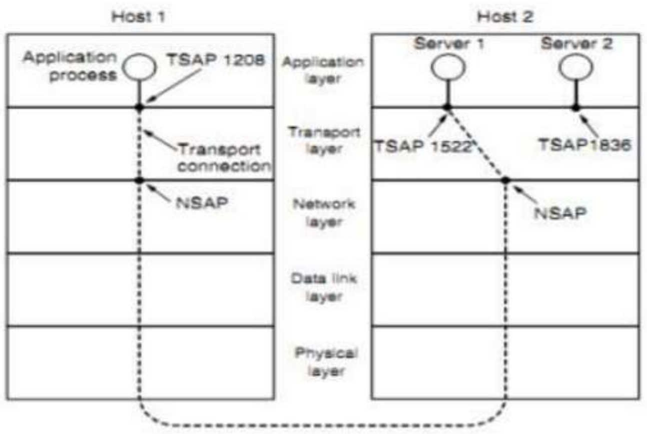


*Fig 4.5: TSAP and NSAP network connections*

## 2. Connection Establishment:

With packet lifetimes bounded, it is possible to devise a fool proof way to establish connections safely.

Packet lifetime can be bounded to a known maximum using one of the following techniques:

- Restricted subnet design
- Putting a hop counter in each packet
- Time stamping in each packet

Using a 3-way hand shake, a connection can be established. This establishment protocol doesn't require both sides to begin sending with the same sequence number.

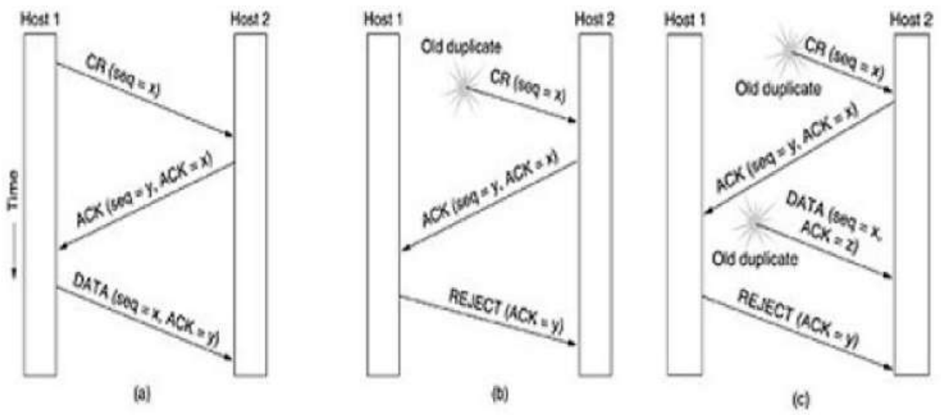


*Fig 4.6: Three protocol scenarios for establishing a connection using a three-way handshake. CR denotes CONNEC TION REQUEST (a) Normal operation. (b) Old duplicate CONNECTION REQUEST appearing out of nowhere. (c) Duplicate CONNECTION REQUEST and duplicate ACK .*

- The **first technique** includes any method that prevents packets from looping, combined with some way of bounding delay including congestion over the longest possible path. It is difficult, given that internets may range from a single city to international in scope.

- The **second method** consists of having the hop count initialized to some appropriate value and decremented each time the packet is forwarded. The network protocol simply discards any packet whose hop counter becomes zero.
- The **third method** requires each packet to bear the time it was created, with the routers agreeing to discard any packet older than some agreed-upon time.

In **fig (A)** Tomlinson (1975) introduced the **three-way handshake**.

- This establishment protocol involves one peer checking with the other that the connection request is indeed current. Host 1 chooses a sequence number, x , and sends a CONNECTION REQUEST segment containing it to host 2. Host 2replies with an ACK segment acknowledging x and announcing its own initial sequence number, y.

- Finally, host 1 acknowledges host 2's choice of an initial sequence number in the first data segment thatit sends

In **fig (B)** the first segment is a delayed duplicate CONNECTION REQUEST from an old connection.

- This segment arrives at host 2 without host 1's knowledge. Host 2 reacts to this segment by sending host1an ACK segment, in effect asking for verification that host 1 was indeed trying to set up a new connection.
- When host 1 rejects host 2's attempt to establish a connection, host 2 realizes that it was tricked by a delayed duplicate and abandons the connection. In this way, a delayed duplicate does no damage.

- The worst case is when both a delayed CONNECTION REQUEST and an ACK are floating around in the subnet.

In **fig (C)** previous example, host 2 gets a delayed CONNECTION REQUEST and replies to it.

- At this point, it is crucial to realize that host 2 has proposed using y as the initial sequence number for host 2 to host 1 traffic, knowing full well that no segments containing sequence number y or acknowledgements to y are still in existence.
- When the second delayed segment arrives at host 2, the fact that z has been acknowledged rather than ytells host 2 that this, too, is an old duplicate.

## 3. Connection Release:

A connection is released using either asymmetric or symmetric variant. But, the improved protocol for releasing a connection is a 3-way handshake protocol.
There are two styles of terminating a connection:

1) Asymmetric release and
2) Symmetric release.

   **Asymmetric release** is the way the telephone system works: when one party hangs up, the connection is broken. **Symmetric release** treats the connection as two separate unidirectional connections and requires each one to be released separately.

## Internet transport protocols (TCP & UDP):

## DNS:

## Electronic Mailing:

## World Wide Web: