UNIT 1 TRANSPORT SERVICES AND MECHANISM

Structure **Page Nos.**

1.0 INTRODUCTION

The transport layer is the core of the OSI model. It is not just another layer but the heart of the whole protocol hierarchy. Protocols at this layer oversee the delivery of data from an application program on one device to an application program on another device. It is the first end-to-end layer in the OSI model. It provides its services to the upper layer to use the services of the network layer and other lower layer protocols.

You must be aware that an Internet is comprised of several physical networks such as the LANs, MANs, and WANs that are connected together through a variety of media (wired as well wireless) to facilitate the transfer of data from a computer on one network to another computer on another network. As a transmission moves from one network to another, the data on the way may be transformed i.e., it may be encapsulated in different types and lengths of packets.

The upper-layer protocols are unaware of the intricacy of the physical networks the transport layer. To the upper layers, the individual physical networks are a simple homogeneous network that somehow takes data and delivers it to its destination, reliably. For example, even if an Ethernet in the LAN part of internet is replaced by a FDDI, the upper layers remain unaware of it. To them, the Internet is a single and essentially unchanging network. The transport layer provides this transparency.

Examples of transport layer protocols are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

In this unit, we will first discuss types of services we might expect from a transport layer. Next, we will examine several mechanisms to support these services.

1.1 OBJECTIVES

After going through this unit, you should be able to:

- list and describe transport services, and
- list and describe transport mechanisms.

1.2 TRANSPORT SERVICES

We begin by looking at the kinds of services that a transport protocol can or should provide to upper layer protocols. *Figure 1* outlines the environment for transport services. There is a transport entity that provides services to the upper layer users, which might be an application process such as http, telnet etc. This local transport entity communicates with some remote-transport entity, based on the services provided by the network layer.

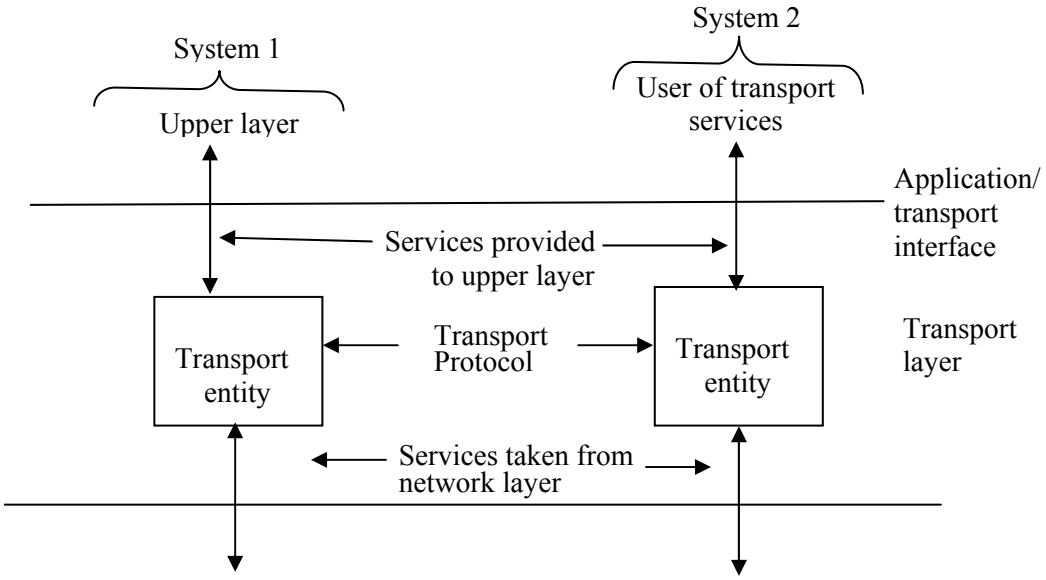


Figure 1: Transport entity environment

As discussed earlier the general service provided by a transport protocol is the end-to-end transport of data in a way that shields the upper layer user from the details of the underlying networks infrastructure. The following categories of services are generally used for describing the services provided by the transport layers.

- Types of services
- Quality of service
- Data transfer
- Connection management
- Expedited delivery

1.2.1 Types of Services

There are generally two basic types of services: (i) Connection-oriented and (ii) Connectionless, or datagram services provided by the Internet. A connection-oriented service is the most common type of protocol service available. It provides the establishment, maintenance, and termination of a logical connection between users. This is also **called handshaking** procedures. The connection-oriented service generally implies that the service is reliable which includes features such as flow control, error control and sequence delivery. Flow control makes sure that neither side of a connection overwhelms the other side by sending too many pockets too quickly.

Connection-Oriented: The Internet connection oriented service is accomplished using TCP (Transmission Control Protocol). Reliable transport, flow control and congestion control are types of services that are provided to an application by TCP. These services are acknowledged. The TCP's congestion control mechanism is used to maintain throughput.

Connectionless Service: There is no handshaking here before sending the packet. There are also no flow control and congestion control mechanisms. Since there is no handshaking procedure, data can be transferred faster. But there is no reliable data transfer either as these services are acknowledged. The Internets connectionless service is called UDP (User Datagram Protocol). Some of the applications of connectionless service are internet telephony and video conferencing.

The connection-oriented transport service is similar to the connection oriented network service. So the question is, why do we require two different layers to provide the same type of service. The answer to this question is that the transport code runs on the user's machine whereas the network layer runs mostly on the routers, which are controlled by the carrier which provide poor service. Therefore, the only possibility is to put another layer on top of the network layer, that improves the quality of service.

It is due to the transport layer that application programmers can write programs according to standard sets of library procedure calls and have these programs run on networks without worrying about dealing with different subnet interfaces and unreliable transmission.

Thus, there is a place at the transport level for both a connection-oriented and a connectionless type of service.

1.2.2 Quality of Service

The transport protocol entity should allow the upper layer protocol users to specify the types of quality of transmission service to be provided. Based on these specifications the transport entity attempts to optimise the use of the underlying link, network, and other resources to the best of its ability, so as to provide the collective requested services. But these services are limited to the internet capabilities of the network layer services.

Examples of services that might be requested are:

- Expedited delivery of packet
- Acceptable error and loss levels of packet
- Desired average and maximum delay in transmission of a packet
- Desired average and minimum throughout
- Priority levels.

You are aware that IP is a standard protocol for the network layer. IP does provide a quality-of-service parameter such as priority as well as a binary specification for normal or low delay, normal or high throughput, and normal or high reliability. Thus, the transport entity can make a request to the internetwork entity. The network may alter flow control parameters and the amount of network resources allocated on a virtual circuit to achieve desired throughput. The transport layer may also split one transport connection among multiple virtual circuits to enhance throughput. This will be explained in section 1.3.2.

You must be aware that different applications may require different quality of services. For example:

- A file transfer protocol might require high throughput. It may also require high reliability to avoid retransmissions at the file transfer level.
- A transaction protocol (e.g., web browser-web server) may require low delay.
- An electronic mail protocol may require multiple priority levels.

One approach to providing a variety of qualities of service is to include a quality-of-service facility within the protocol; the transport protocols typically follow the same approach. An alternative is to provide a different transport protocol for different

classes of traffic; this is to some extent the approach taken by the ISO-standard family of transport protocols.

An application layer protocols need from the Transport layer [Ref.2]. These are:

(i) Reliable data transfer
(ii) Bandwidth
(iii) Timing/delay.

(i) **Reliable data transfer**: By reliable data transfer means that there is not a single bit of loss of data during transmission. There are certain types of application, which does not tolerate any loss of data, such as financial applications. In particular, a loss of file data, or data in a financial transaction, can have devastating consequences (in the latter case, for either the bank or the customer). Other applications in the category are transfer of web documents, electronic mail, file transfer and remote host access. But there are some applications which can tolerate some amount of data loss. For example, multimedia applications such as real-time audio/video. In these multimedia applications, lost data might result in a small glitch while playing the audio/video. The effects of such a loss on application quality and actual amount of tolerable packet loss, will depend strongly on the application and the coding scheme used.

(ii) **Bandwidth:** The concept of bandwidth has been explained in the first block. Just to recall, higher the bandwidth more the channel capacity. There are certain applications, which are **bandwidth sensitive**. For example, the Internet telephony, requires a given amount of bandwidth. But there are some other types of application called elastic application which can make use of as much or as little bandwidth as happens to be available. Electronic mail, file transfer, and Web transfers are all elastic application [Ref.2] of course; the more bandwidth, and the better would be transport capacity.

(iii) **Timing/delay:** The final service requirement is that of timing. There are certain applications, which require tight timing constraints on data delivery in order to be effective. For example, interactive real-time applications, such as Internet telephony, virtual environments, teleconferencing, and multiplayer games. Many of these applications require that end-to end delays be on the order of a few hundred milliseconds or less. Long delays in Internet telephony, and a long delay between taking an action and judging the response from the environment in a multiplicity game tends to result in unnatural pauses in the conversation.

1.2.3 Data Transfer

The whole purpose, of course, of a transport protocol is to transfer data between two transport entities. Both user data and control data must be transferred from one end to the other end either on the same channel or separate channels. Full-duplex service must be provided. Half-duplex and simplex modes may also be offered to support peculiarities of particular transport users.

Here, you may ask the question if the network layer does a similar task, why it is necessary at the transport layer. The network layer overseas the hop by hop delivery of the individual packet but does not see any relationship between those packets even those belonging to a single message. Each packet is treated as an independent entity. The transport layer on the other hand makes sure that not just a single packet but the entire sequence of packets.

1.2.4 Connection Management

When connection-oriented service is provided, the transport entity is responsible for establishing and terminating connections. Both symmetric and asymmetric procedure for connecting establishment may be provided.
Connection termination can be either *abrupt* or *graceful*. With an abrupt termination, data in transmit may be lost. A graceful termination prevents either side from shutting down until all data has been delivered.

1.2.5 Expedited Delivery

A service similar to that provided by priority classes is the expedited delivery of data. Some data submitted to the transport service may supersede data submitted previously. The transport entity will endeavour to have the transmission facility transfer the data as rapidly as possible. At the receiving end, the transport entity will interrupt the transport service user to notify it of the receipt of urgent data. Thus, the expedited data service is in the nature of an interrupted mechanism, and is used to transfer occasional urgent data, such as a break character from a terminal or an alarm condition. In contrast, a priority service might dedicate resources and adjust parameters such that, on average, higher priority data are delivered more quickly.

☞ Check Your Progress 1

1) List the three types of services provided by Transport layer to Applications layer.

   ……

   ……

   ……

2) Describe the presence of data loss and time sensitiveness for the following applications:

   (a) File Transfer
   (b) E-mail
   (c) Web surfing
   (d) Stored audio/video.

1.3 ELEMENTS OF TRANSPORT LAYER PROTOCOLS

The services provided by the transport layer are implemented by the transport layer protocols such as TCP and UDP. We will talk about them in the next unit. In this section, we will take up important components of the transport layer for discussion. A transport layer deals with hop-by-hop processes.

1.3.1 Addressing

The issue concerned with addressing is simply this-how a user process will set up a connection to a remote user process? How the address of a remote process should be specified? The method generally used is to define transport address to which the process can listen for connection requests. In Internet jargon these end points are called **port or TSAP** (Transport Services Access Points). Similarly, these end points at the network layer are then called NSAP (Network Services Access Points). The following illustration (*Figure 2)* shows the connection between a user processes as host 1 with a remote process (server) as host 2.

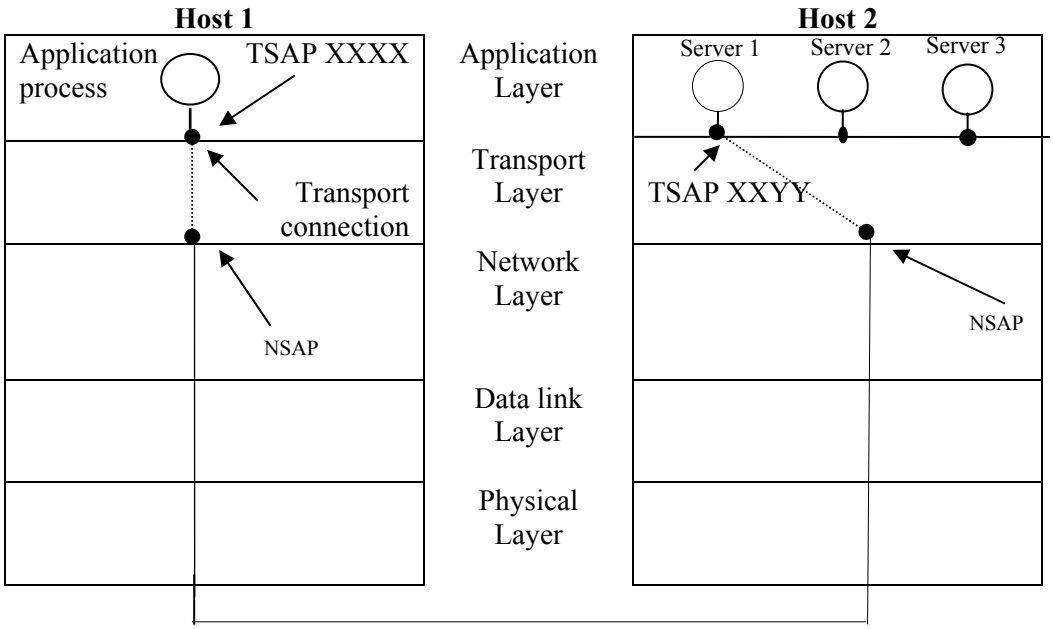


Figure 2: Transport connections between a client and a server

The following steps may be needed for establishing a transport connection [Ref. 1]:

1) Just assume that an application process running on 1 wants to find out the next day's weather forecast, so, it issues a CONNECT request specifying its transport connection point number TSAP XXXX as well as the TSAP number XXYY of the weather forecasting server which will establish transport connection with the destination. This action ultimately results in a transport connection between the application process on host 1 and a server 1 on host 2.

2) Assume that a weather forecasting server is attached to a transport connection at XXYY.

3) The application process then sends a request for the next 10 days weather report.

4) The weather server process responds with the current weather report.

5) The transport connection is then released.

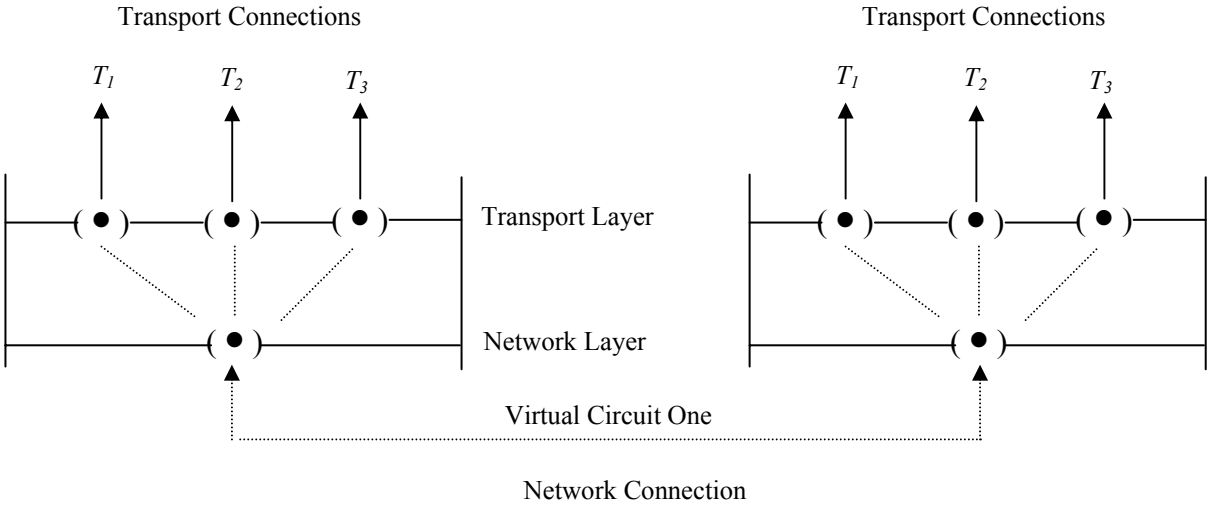
The question, that needs to be addressed: How does the host user know that the address of the destination server process is attached to a particular transport connection. To resolve this issue various strategies are suggested [Refer 1]:

1) **Initial connection protocol:** In the scheme a process server acts as a proxy server for less heavily used servers. Instead of every server listening at a well known port address waiting for a connection request, a process server listens to sets of ports at the same time and informs a particular server to act upon getting a connecting request to perform the required work*. The new server then does the request work, while the process server goes back to listening for new request. Therefore, through a process server a rarely used server should not be listening to ports all the time.

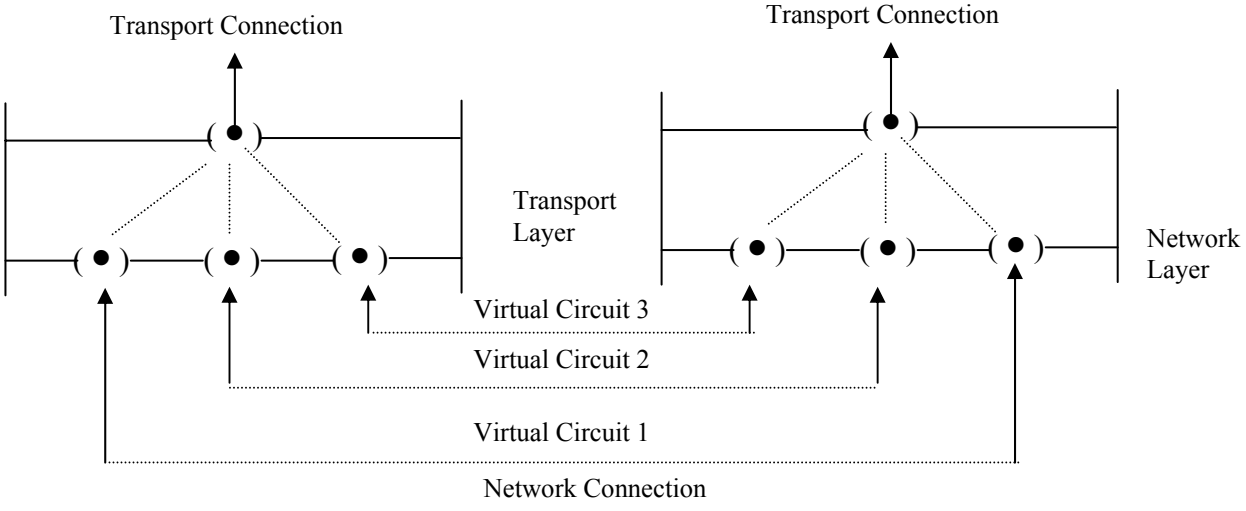
2) Some commonly used services are assigned "well-known address" (for example, time sharing and word processing).
3) A name server or sometimes a directory server is provided. The Transport Service user requests a service through some generic or global name. The request is sent to the name server, which does a directory lookup and returns an address. The transport entity then proceeds with the connection. This service is useful for commonly used applications that change location from time to time. It is also used for those stations in which services do exist independently of a process server. For example, a file server that needs to run on a special hardware (a machine with disk) that cannot be created dynamically when someone want to talk to it [Ref. 1].

1.3.2 Multiplexing

The transport entity may also perform a multiplexing function with respect to the network services that it uses. There are two types of multiplexing techniques that define **upward multiplexing** as the multiplexing of multiple transport connections on a single network connection, and **downward multiplexing** as the splitting of a single transport connection among multiple lower-level connections. This is illustrated in *Figure 3*.



(a) Upward Multiplexing



(b) Downward Multiplexing

Figure 3: Multiplexing

Consider, for example, a transport entity making use of an X.25 service. X.25 is a network layer protocol like IP. Why should the transport entity employ upward multiplexing? There are, after all, 4905 virtual circuits available. In the typical case, this is a more than enough to handle all active Transport Service user. However most X.25 networks base part of their charge on virtual-circuit connect time, as each virtual circuit consumes some node buffer resources. Thus, if a single virtual circuit provides sufficient throughput for multiple TS users, upward multiplexing may be used [*Figure 3*].

On the other hand, **downward multiplexing** or splitting might be used to provide more bandwidth than a single Virtual Circuit can manage. A way out is to open multiple network connections and distribute traffic among them on a round-robin basis, as indicated in *Figure (3b).* This *modus operandi* is called **downward multiplexing.** With *k* network connections open, the effective bandwidth is increased by a factor of *k*. A common example of downward multiplexing occurs with home users who have an ISDN line. This line provides for two separate connections of 64 kbps each. Using both of them to call an Internet provider and dividing the traffic over both lines makes it possible to achieve an effective bandwidth of 128 kbps.

1.3.3 Flow Control and Buffering

There are similarities as well as differences between data link layer and transport layer in order to support flow control mechanism. The similarity is in using the sliding window protocol. The main difference is that a router usually has relatively few lines, whereas a host may have numerous connections. Due to this reason it is impractical to implement the data link buffering strategy in the transport layer.

Whereas flow control is a relatively simple mechanism at the data link layer, it is a rather complex mechanism at the transport layer, for two main reasons:

- Flow control at the transport level involves the interaction of three components: TS users, transport entities, and the network service.
- The transmission delay between transport entities is variable and generally long compared to actual transmission time.

The following delay may arise during the interaction of the two transport entities A and B:

(i) Waiting time to get permission from its own transport entity (interface flow control).

(ii) Waiting time by A to have permission to send the data to B.

(iii) Waiting time as network layer services.

In any case, once the transport entity has accepted the data, it sends out a segment. Some time later, it receives an acknowledgement that the data has been received at the remote end. It then sends a confirmation to the sender.

Now let us come to the flow control problem.

First, we present two ways of coping with the flow control requirement [Ref.3] by using a fixed sliding-window protocol and by using a credit scheme.

The first mechanism is already familiar to us from our discussions of data link layer protocols. The key ingredients are:

- The use of sequence numbers on data units.

- The use of a window of fixed size.
- The use of acknowledgements to advance the window.

This scheme really works well. For example, consider a protocol with a window size of 3. Whenever the sender receives an acknowledgement from a particular segment, it is automatically authorised to send the succeeding seven segments. (Of course, some may already have been sent). Now, when the receiver's buffer capacity comes down to 7 segments, it can withhold acknowledgement of incoming segments to avoid overflow. The sending transport entity can send, at most, seven additional segments and then must stop. Because the underlying network service is reliable, the sender will not time-out and retransmit. Thus, at some point, a sending transport entity may have a number of segments outstanding, for which no acknowledgement has been received. Because we are dealing with a reliable network, the sending transport entity can assume that the segments will come through and that the lack of acknowledgement is a flow control tactic. Such a strategy would not work well in an unreliable network, as the sending transport entity would not know whether the lack of acknowledgement is due to flow control or a lost segment.

The second alternative, a credit scheme, provides the receiver with a greater degree of control over data flow.

The credit scheme decouples acknowledgement from flow control. In fixed sliding-window protocols, used as the data Line layer, the two are synonymous. In a credit scheme, a segment may be acknowledged without granting a new credit, and vice versa. *Figure 4* illustrates the protocol. For simplicity, we have depicted data flow in one direction only. In this example, data segments are numbered sequentially module 8 (e.g., SN 0 = segment with sequence number 0). Initially, through the connection-establishment process, the sending and receiving sequence numbers are synchronised, and A is granted a credit allocation of 7. A advances the trailing edge of its window each time that it transmits, and advances the leading edge only when it is granted a credit.

Figure 4 shows a view of this mechanism from the sending and receiving sides; of course, both sides take both views because data may be exchanged in both directions.

From the receiving point of view, the concern is for received data and for the window of credit that has been allocated. Note that the receiver is not required to immediately acknowledge incoming segments, but may wait and issue a cumulative acknowledgement for a number of segments; this is true for both TCP and the ISO transport protocol.

In both the credit allocation scheme and the sliding window scheme, the receiver needs to adopt some policy concerning the amount of data it permits the sender to transmit. The conservative approach is only to allow new segments up to the limit of available buffer space.

A conservative flow control scheme may limit the throughput of the transport connection in long-delay situations. The receiver could potentially increase throughput by optimistically granting credit for space it does not have. This is called dynamic buffer allocation scheme. For example, if a receiver's buffer is full but it anticipates that it can release space for two segments within a round-trip propagation time, it could immediately send a credit of 2. If the receiver can keep up with the sender, this scheme may increase throughput and can do no harm. If the sender is faster than the receiver, however, some segments may be discarded, necessitating a retransmission. Because retransmissions are not otherwise necessary with a reliable network service, an optimistic flow control scheme will complicate the protocol.

Figure 4: Credit based flow control

The optimum trade-off between source buffering and destination buffering depends on the type of traffic carried out by the connection. For low bandwidth bursty traffic, such as that produced by an interactive terminal, it is better not to dedicate any buffers, but rather to acquire them dynamically at both ends. Since the sender cannot be sure the receiver will be able to acquire a buffer, the sender must retain a copy of the TPDU(Transport Protocol Data Unit) until it is acknowledged. On the other hand, for file transfer and other high bandwidth traffic, it is better if the receiver dedicates a full window of buffers, to allow the data to flow at maximum speed. Thus, for low-bandwidth bursty traffic, it is better to buffer at the sender, and for high-bandwidth smooth traffic, it is better to buffer at the receiver.

As connections are opened and closed and as the traffic pattern changes, the sender and receiver needs to dynamically adjust their buffer allocations. Consequently, the transport protocol should allow a sending host to request buffer space at the other end. Buffers could be allocated per connection, or collectively, for all the connections running between the two hosts. Alternatively, the receiver, knowing its buffer station (but not knowing the offered traffic) could tell the sender "I have reserved X buffers for you". If the number of open connections should increase, it may be necessary for an allocation to be reduced, so the protocol should provide for this possibility.

In summary, a reasonably general way to manage dynamic buffer allocation is to decouple the buffering from the acknowledgements, in contrast to the sliding window protocols of data link layer.

1.3.4 Connection Establishment

End-to-end delivery can be accomplished in either of two modes: connection-oriented or connectionless. Of these two, the connection-oriented mode is the more commonly used. A connection-oriented protocol establishes a virtual circuit or pathway through the Internet between the sender and receiver. All of the packets belonging to a message are then sent over this same path. Using a single pathway for the entire message facilities, the acknowledgement process and retransmission of damaged or lost frames. Connection-oriented services, therefore, are generally considered reliable.

Connection-oriented transmission has three stages: connection establishment, data transfer, and connection termination.

The network is generally unreliable and congested. Even with a reliable network service, there is a need for connection establishment and termination procedures to support connection-oriented service. Connection establishment serves three main purposes:

- It allows each end to assure that the other exists.
- It allows negotiation of optional parameters (e.g., maximum segment size, maximum window size, quality of service).
- It triggers allocation of transport entity resources (e.g., buffer space, entry in connection table).

To solve the problems arising out of unreliable and congested networks 3-way handshake procedure is used to establish transport connection, which is generally comprised of three steps.

(i) Connection initiation / requirement

(ii) Connection confirmation

(iii) Acknowledgement of confirmation and data transfer.

1.3.5 Crash Recovery

Similar to the database this is an important issue in computer network. In a network a host as well as a router can crash. Therefore, the issue is its recovery. In case of the loss of Virtual Circuit (In case the network layer provides connection oriented service) due to a crash delivery, a new virtual circuit may be established then probing the transport entity to ask which TPDU it has received and which one it has not received so that the latter can be retransmitted. In case of datagram subnet, the frequency of the lost TPDU is high but the network layer knows how to handle that. The real problem is the recovery from the host crash. Students are requested to read *Computer Network by **Tanenbaum*** on this method and also to relate to how a similar problem is resolved in the database (MCS-043).

☞ Check Your Progress 2

1) List the important multiplexing mechanism at the Transport Layer and also explain how they are different from each other.

……

……

……

2) Describe the similarities as well as differences between Data Link Layer and Transport Layer in order to support Flow Control mechanism.

……

……

……

1.4 SUMMARY

In this unit, we have discussed two important components with respect to transport layer namely, transport services and the elements of transport layer protocols. As a part of transport services we discussed type of services, quality of services, connection mechanism. We also outlined three types of services provided by the transport layer through the applications layer. We listed the various types of protocols (applications layer) and the kind of services. We also differentiated between the data link layer and transport layer with respect to flow control mechanism. Similarly, we also differentiated between two types of multiplexing mechanism at the transport layer.

1.5 SOLUTIONS/ANSWERS

Check Your Progress 1

1) Reliable data transfer, Bandwidth, timing delay

2)

	Application	**Data Loss**	**Time-sensitive**
a.	File Transfer	No Loss	No
b.	e-mail	No Loss	No
c.	Web Surfing	No Loss	No
d.	Stored Audio/Video	Loss tolerant	Yes

Check Your Progress 2

1) There are two multiplexing mechanism in the transport layer namely, upward multiplexing and downward multiplexing. Upward multiplexing as the multiplexing of multiple transport connections on a single network connection, and downward multiplexing as the splitting of a single transport connection among multiple lower-level connections. Downward multiplexing or splitting might be used to provide more bandwidth than a single Virtual Circuit can manage. A way out is to open multiple network connections and distribute the traffic among them on a round-robin basis. This *modus operandi* is called downward multiplexing.

2) The similarity is in using the sliding window protocol. The main difference is that an intermediate node usually has relatively few lines, whereas a host may have numerous connections. Due to this reason it is impractical to implement the data link buffering strategy in the transport layer. Whereas flow control is a relatively simple mechanism at the data link layer, it is a rather complex mechanism at the transport layer, for two main reasons:

   - Flow control at the transport level involves the interaction of three components: TS users, transport entities, and the network service.

   - The transmission delay between transport entities is variable and generally long compared to actual transmission time.

1.6 FURTHER READINGS

1) *Computer Networks,* A.S. Tanenbaum, 4th Edition, Prentice Hall of India, New Delhi, 2002.

2) *Computer Networking, A Top down approach featuring the Internet,* James Kurose and Keith W. Ross, Pearson education, New Delhi.

3) *Data and Computer Communications*, William Stalling, 6th Edition, Pearson Education, New Delhi.