Contents

Unit 5
The Transport Layer

Structure of the Unit

5.1 Unit Outcomes

After the successful completion of this unit, the student will be able to:

- List the important services provided by the transport layer
- Compare Connectionless and Connection-oriented transport layer protocols
- Explain the working of the connectionless UDP transport protocol

5.2 Introduction

The important service provided by the transport layer is to allow two application programs or processes running on different hosts to communicate. To provide this service to the application layer, the transport layer in turn uses the services from the network layer. By providing process-to-process communication, this layer brings down the burden on the application layer protocols.

The first important function of the transport layer is to extend the network layer functionality of delivering packets from the source computer to the destination computer to reorder the packets and then deliver the packets to the appropriate application processes running on these computers. The second function of this layer is to transport data reliably without data loss and corruption as this service is not provided by the network layer. The third function is to control the data transmission rate to avoid network congestion or recover from it. The above services are implemented in the form of two widely used protocols: Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

The size and complexity of these protocols depend on the capability of the network layer below it. If the network layer is built with reliability features, these protocols are less complex otherwise they are heavy in terms of size and complexity requiring error detection and recovery.
Let us understand the transport layer functionality with the help of Fig.5.1 showing the same scenario discussed in Section 3.2 with appropriate changes showing the communication at the transport layer. In the figure, Jay's host who is working for the India Research company gets connected logically with Ram's host and gives the feeling of a physical connection between them. The figure also shows that only the end systems use the services of this layer and the intermediate devices use only the three lower-layer services.

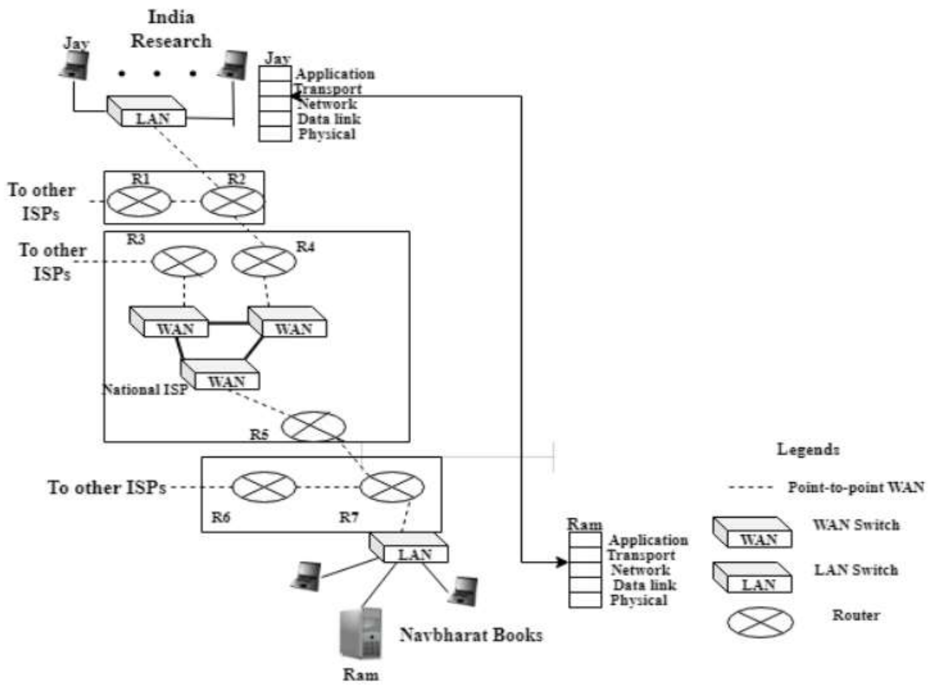

Fig. 5.1: Logical Connection at the Transport Layer

At the transmitter, the messages created by the application layer are broken into smaller divisions or parts, and a header is added to each division to convert them into transport layer data units called segments. These segments are passed to the network layer, which will encapsulate them within the network layer packets and then send them to the receiver (discussed in section 2.3.3). The routers in the path do not examine the transport layer fields. They are extracted only at the transport layer of the destination host. After processing, the data is extracted from the segment and passed to the appropriate application process at the receiver.

In this unit, the first section briefly discusses the transport layer protocol services, the types of transport layer protocols, and some general concepts about them. It also discusses the concepts of connection-oriented and connectionless protocols and their differences. The second section discusses the User Datagram Protocol (UDP), which is the popular connectionless transport layer protocol on the Internet.

5.3 Transport Layer Services

The three functions of the transport layer are implemented in the form of six services which are discussed in this section. They are Process-to-process communication, Addressing, Encapsulation and decapsulation, Multiplexing and demultiplexing, Flow control, Error control, and Congestion Control.

5.3.1 Process-to-Process Communication

To understand this concept, let us understand the difference between the service provided by the network layer and the transport layer. The network layer provides an end-to-end delivery for the packets. This is incomplete because the packet is only delivered to the correct destination host, but not to the correct process. So the transport layer does the job of delivering the packet to the correct process. This is done by different methods, but the most popular one is through the client-server paradigm (discussed in section 3.3.1.1) as discussed below.

Fig.5.2 shows the services at the network layer versus the transport layer.

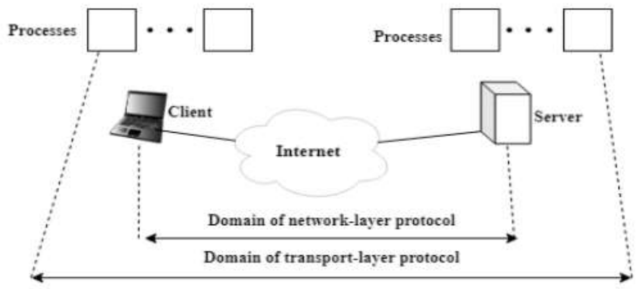
Processes
Processes
Client
Server
Internet
Domain of network-layer protocol
Domain of transport-layer protocol

Fig. 5.2: Network Layer versus Transport Layer

5.3.2 Process Addressing: Port Numbers

A process running on a local host called the client, requests services from processes on the remote hosts called the server. Today, operating systems support multitasking, multiprogramming, and multiuser environments. So there are multiple client and server processes running on them simultaneously. For communication purposes, these processes should be identified by addresses. The remote and local hosts are identified by IP addresses and the individual **processes** running on them are identified by port numbers.

Ports are 16-bit numbers (ranging from 0 to 65535) used to identify specific services and applications. The first 1024 port numbers are reserved for specific services. Port numbers are not random numbers, but TCP/IP uses universal port numbers for servers and are called well-known port numbers. Sometimes clients are also assigned these well-known port numbers, but otherwise, a client process identifies itself with a port number called the **ephemeral port number** or an **unused** port number for communication. These are not fixed they change and are always greater than 1023.

Some of the popular application layer services and their corresponding port numbers are listed in Table 5.1.

Table 5.1: Popular TCP and UDP ports

TCP		UDP	
FTP	20,21	DNS	53
SSH	22	BooTPS/DHCP	67
Telnet	23	TFTP	69
SNMP	25	NTP	123
DNS	53	SNMP	161
HTTP	80		
POP3	110		
IMAP4	143		
HTTPS	443		

On a TCP/IP network, the two protocols TCP and UDP together specify three important pieces of information in their header: the source and destination IP addresses and the transport layer protocol in the IP packet header, the source and destination port numbers as a part of their segment header.

The well-known port numbers of the server process are known by every client process. Fig. 5.3 demonstrates this concept with an example. Here, the Daytime client process uses 5000 as a temporary port number to identify itself. This process can request the Daytime server process for the date and time on the fixed port number 13.

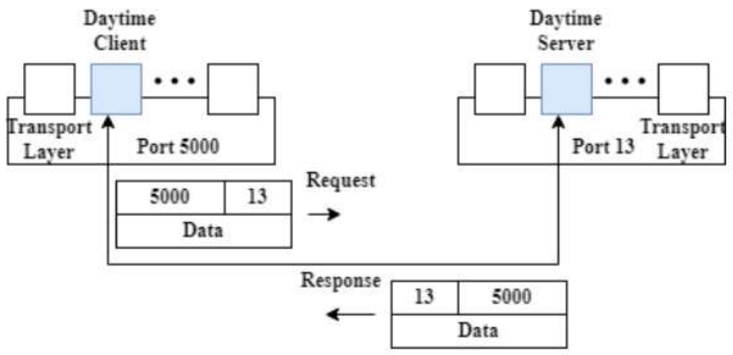

Fig.5.3: Port Numbers

Both, the IP address and port numbers are used to select the destination for data. The IP address selects a host out of many hosts on the global internet, while the port number selects a specific process out of the many processes running on that host. This concept is demonstrated in Fig. 5.4, where port number 13 identifies the Daytime process on the Server host which is identified by the IP address 193.14.26.7

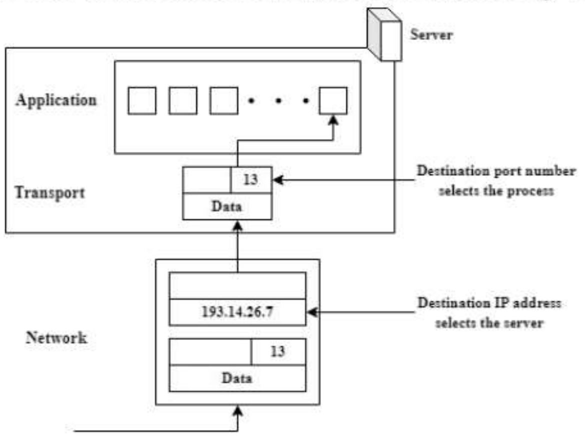
Server
Application
Transport
13
Data
Destination port number selects the process
Network
193.14.26.7
13
Data
Destination IP address selects the server

Fig. 5.4: IP address versus Port Numbers

5.3.2.1 Socket Addresses

To make an end-to-end connection by the transport layer protocol, it needs both an IP address and port number. This pair of information: IP address and port number is known as a socket address. On the internet, the clients and servers have their own socket addresses (defining their processes) to use the services of the transport layer. The IP address is a part of the network-layer packet header and the port number is defined in the transport-layer header.

Sockets are a means in an IP network for communicating between two machines. Different types of application processes can send and receive messages through them and they are built to work with different types of networks.

Sockets are mechanisms provided by operating systems to the user to access a network through a program called socket program.

Socket programming involves Server-side and client-side programming.
It involves six important steps:

1. Create a socket using the socket system call.
2. Identify the socket with an address
3. Wait for an incoming connection on the Server side
4. Connect to the server's socket on the Client side.
5. Send and Receive message
6. Close the socket

A socket address structure consists of three members defined in the header file socket.h: the length of the address of type int (POSIX compliant), a family of the socket address of type unsigned int (currently we can use TCP or UDP), and a protocol-specific address of type char.
The socket address is then used by bind(), connect(), listen(), accept(), and other socket functions.

5.3.3 Encapsulation and Decapsulation

The encapsulation process is performed at the source host and a reverse process called decapsulation is performed at the destination host. (Discussed in section 2.3.3). At the source host, the transport layer receives the message from the application layer which contains a pair of socket addresses and other control information. This layer then adds its own header and forms the transport layer packets.
At the destination host, the transport layer removes the header from the user datagram (or segment), and the message is passed to the application layer.

5.3.4 Multiplexing and Demultiplexing

These two processes are performed when an object receives items from many sources (multiplexing) or an object sends items to many destinations (demultiplexing) (discussed in section 2.3.5). This is demonstrated in Fig. 5.5, where communication between one client and two servers is shown.
There are three processes P1, P2, and P3 running on the client side requesting services from server1

and server2. P1 and P3 send requests to the corresponding processes running on server1 and P2 sends requests to a corresponding process running on server2. On the client side, the transport layer creates three packets 1, 2, and 3, from P1, P2, and P3 respectively, by a process called multiplexing. Using the same logical channel, packets 1 and 3 reach the transport layer of server1 which demultiplexes and distributes them to different application processes.
Using another logical channel, packet 2 reaches the transport layer of server2 and is passed on to the corresponding process at the application process. A single process will also be demultiplexed.

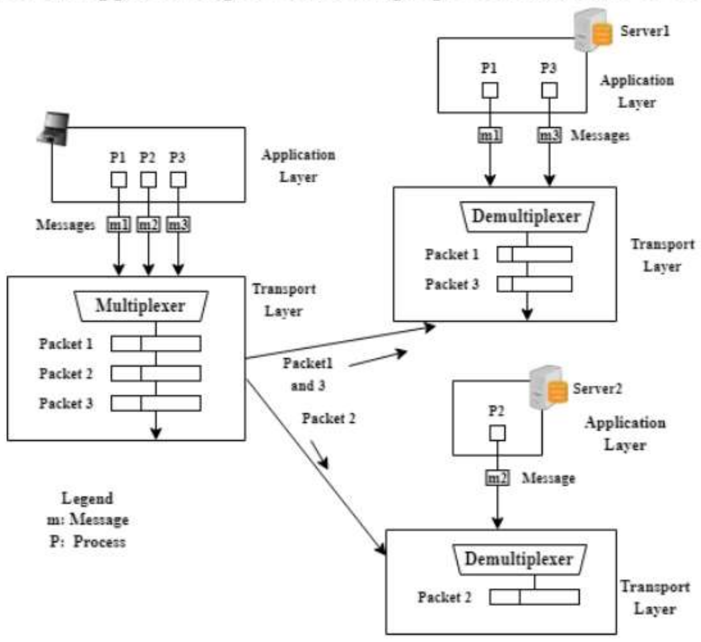

Fig. 5.5 Multiplexing and Demultiplexing

5.3.5 Flow Control at Transport Layer

Flow control is a technique used to regulate the data transfer between two hosts in a computer network. If the rate of sending data is more than the receiving rate, the receiver discards some packets which results in a loss of data. To reduce this there should be control in the flow by feedback to warn the sender. If the rate of sending data is less than the receiving rate, then the system efficiency reduces.
To address the first issue of flow control, there are many techniques used. One of them is them is the use of Buffers. They can be used in many ways. One way is to have one buffer at the sender and one at the receiver. Buffers are memory locations that can hold packets. Before we understand the use of buffers and their location at the transport layer, let us understand the important entities at this layer.

Communication at the transport layer involves four entities, two each at the sending and receiving ends. Two processes and two transport layers one at each end. There is a need for two cases of flow control as shown in Fig. 5.6.
Case 1: When the sending side buffer is full, the transport layer informs the application layer to stop sending messages.
Case 2: When the receiving side buffer is full, the transport layer of the receiver informs the transport layer of the sender to stop sending messages.

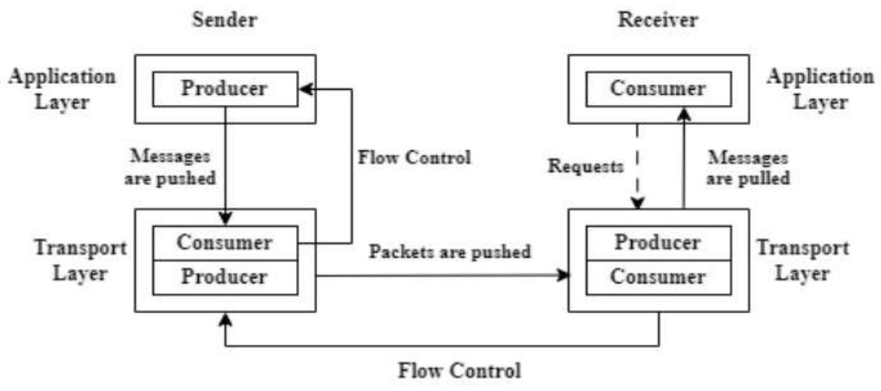

Fig. 5.6 Flow Control at the Transport Layer

5.3.6 Error Control

To achieve error control in the transmission of datagrams, the transport layer does the following operations:

1. The corrupted packets are detected and discarded.
2. The discarded packets are kept track of and resent.
3. The duplicate packets are recognized and discarded.
4. The out-of-order packets are stored in the buffer till the missing packets arrive.

Most of the time the transport layer at the receiver takes the above responsibilities by informing the sender regarding the problems.

5.3.6.1 Sequence Numbers

In order to perform the above operations, error control uses sequence numbers for the packets for identification of corrupted, missing, or duplicate packets. Packets are numbered in a sequential manner. If there are m bits assigned for including a sequence number in the packet header, then the range of sequence numbers is from 0 to $2^m - 1$.

5.3.6.2 Acknowledgement

This is a signal sent by a receiver to the sender for a set of packets that arrived without any error. The sender uses a timer to detect lost packets. The sender can choose a time duration to wait for an acknowledgment. If a time-out occurs and no acknowledgment is received for a packet, then this can indicate that the packet is lost. The sender will then retransmit this packet.

The receiver usually discards the corrupted and duplicate packets. Out-of-order packets are also sometimes discarded to be treated as lost packets or buffered until the missing packets arrive.

5.3.7 Congestion Control

In the context of communication networks, Congestion is a condition when the number of packets sent into the network is more than what it can handle. In other words, it is an undesirable situation when the load on a network is more than its capacity. These occur when the interconnecting network devices like switches and routers cannot process the packets at the rate at which they arrive and the buffers are

overloaded with the packets and congestion builds up. As routers and switches operate at the network layer, congestion starts here and results in congestion at the transport layer too.
There are mechanisms implemented at the transport layer using TCP to maintain the load in the network below its capacity and control congestion.

5.4 Connectionless and Connection-Oriented Protocols at the Transport Layer

Communication between two or more devices can be established in two ways: connectionless and connection-oriented. On the Internet, both transport and network layers can offer these two different types of services to the upper layers for transferring data.

5.4.1 Connectionless services

These services do not require any connection creation and termination processes for transferring data. It is analogous to a postal service. This system does not require the creation of a virtual path between a sender and a receiver and authentication from the receiver.

The connectionless protocol at the network layer takes care of the different physical paths that the datagrams take. It assumes that there is a logical connection between the two transport layers (sender and the receiver). But at the transport layer, connectionless means non-dependence between the packets, and connection-oriented means dependency between the packets. At the transport layer, the protocol used for connectionless service is the User Datagram Protocol (UDP).

The application program divides a message into smaller parts of data depending on the size of the data that the transport layer can handle and delivers them to it. Every part of data is encapsulated in a packet and then the transport layer treats each packet independently. The packets may arrive at the destination in a different order than that is sent (out of order) and so will be delivered in that order to the server process.

For a better understanding, let us consider this process with an example scenario as shown in Fig. 5.7.
Here, a client process has three chunks of messages to be sent to a server. The client process passes these chunks to the connectionless transport layer protocol in order of 0,1,2. Though the packets are delivered to the transport layer instantaneously, a timeline is used to show the movement of the packets. Packet 1 is shown to arrive late due to the delay in its transportation, so they may reach the server process out of order and the server may receive a strange message.

Also, as the messages are not numbered, it may be a worse situation if one of the packets gets lost. This is because the transport layer at the receiver has no idea about it. The above issues arise as the transport layers do not coordinate with each other so flow control, congestion control, and error control cannot be effectively implemented in the connectionless service and protocols.

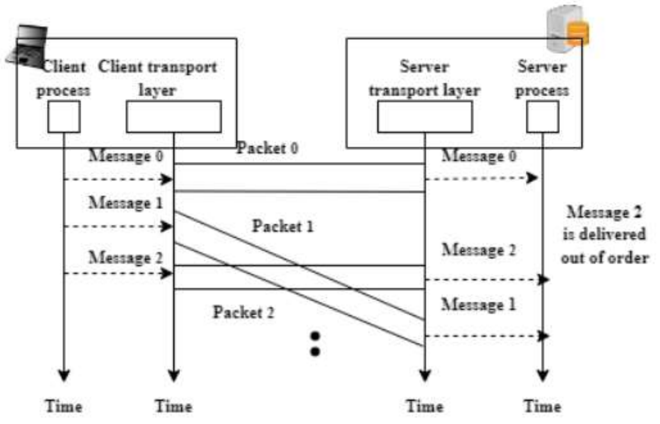

Fig. 5.7: Connectionless service

5.4.2 Connection-oriented services

These services involve the establishment of the connection using a handshake method before transmission and termination of the connection after transmission. It is analogous to a telephone service. At the transport layer, the protocol used for this service is Transmission Control Protocol (TCP). In this service, a logical connection is established between the client and the server before data transmission and the connection is torn down after the transmission is completed.

In the connection-oriented service at the network layer, there is coordination between the sending and receiving hosts and all the routers in the transmission path and as a virtual circuit established between the sender and the receiver, all the packets follow the same path. But at the transport layer, the connection-oriented service involves only the end hosts. So a connection-oriented protocol at the transport layer may run over a connectionless or connection-oriented protocol at the network layer.

To understand the connection-oriented service, the same scenario considered in connectionless service is considered and shown in Fig. 5.8. The figure shows the connection establishment, data transfer, and the tear-down phases. Also, the out-of-order packets at the receiver are reordered by the transport layer and passed to the application layer.

Also, in a connection-oriented protocol, flow control, error control, and congestion control can be effectively implemented.

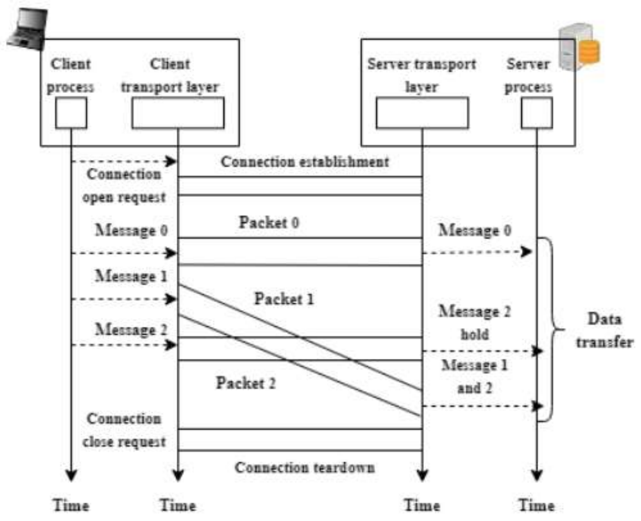

Fig. 5.8: Connection-oriented service

5.5 User Datagram Protocol (UDP)

This is an unreliable and connectionless transport layer protocol. Other than providing process-to-process communication, this protocol does not add any feature to the IP protocol at the network layer. But UDP has an advantage in that it is simple to implement with very few overheads.
For Eg: This protocol can be used to send messages through the mail where reliability is not an important requirement. Messages sent using UDP require very few interactions between the sender and receiver.

5.5.1 UDP Segment Structure

The UDP segment format and the header format are shown in Fig. 5.9 a and Fig. 5.9 b respectively.

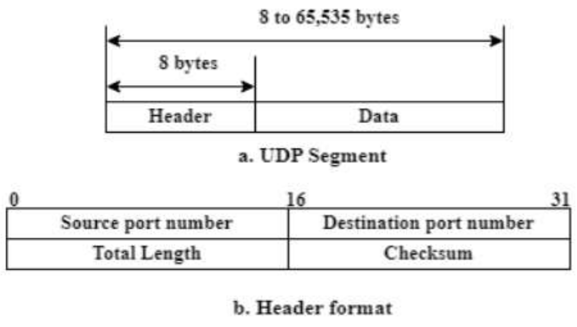

Fig. 5.9: UDP Segment Structure

The header length is 8 bytes, divided into four fields, each of 2 bytes. The first two bytes specify the source and destination port numbers. These port numbers help in the multiplexing and demultiplexing functions at the sending and receiving hosts respectively. At the destination, the destination port number allows the destination host to pass the application data to the correct process.

Though the length field of 16 bits specifies the maximum length of the datagram as 65,535 bytes, the length has to be less than that, as it will be encapsulated inside an IP datagram whose total length will be 65,535 bytes.

The last field specifies the checksum bytes which is optional. But if it is added, it helps the destination host to check if any errors have been introduced in the segment during transmission. This checksum is calculated over some fields of the IP header along with the UDP segment.

The data field in the UDP segment structure consists of application data such as a query message or a response message in the case of DNS or an audio sample in the case of an audio streaming application.

5.5.2 UDP Services

The following services are provided by the UDP protocol:

1. Provides process-to-process communication using socket addresses.
2. Bigger messages are divided into smaller chunks of sizes less than 65,507 bytes (i.e. total length of 65,535 bytes – 8 bytes (UDP header) – 20 bytes (IP header)) and each chunk of message travels independently and follows a different path and there is no connection establishment and tear down phase.
3. The protocol does not provide flow control, so if there is a need for this, the UDP process should provide this feature.
4. The protocol does not provide error control, so if there is a need for this, the UDP process should provide this feature. However, the receiver uses a checksum to detect an error in the reception of a datagram and discards the datagram in case of an error.
5. Calculation of UDP checksum is optional, but if it is calculated, it is done over two headers and the data as shown in Fig. 5.10.

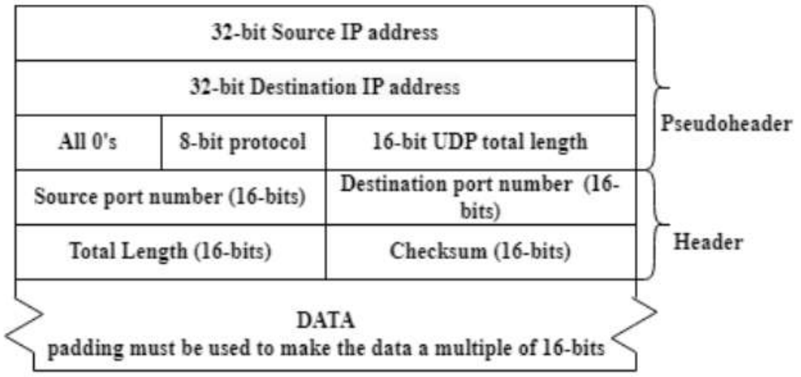
32-bit Source IP address
32-bit Destination IP address
All 0's
8-bit protocol
16-bit UDP total length
Pseudoheader
Source port number (16-bits)
Destination port number (16-bits)
Total Length (16-bits)
Checksum (16-bits)
Header
DATA
padding must be used to make the data a multiple of 16-bits

Fig. 5.10: Pseudoheader for checksum calculation

The checksum includes the pseudo-header, because if the IP header is corrupted and is not discarded, then the datagram may be delivered to the wrong destination host.

The protocol field for the UDP is 17. During checksum calculation, if this value in a packet is detected as erroneous, then the packet is discarded instead of delivering it to the wrong protocol.

6. As the UDP packets are small and occur at irregular time intervals, there is no need for congestion control.
7. Messages are encapsulated and decapsulated by the UDP protocol to send them from one process to another.

8. UDP uses multiplexing and demultiplexing to encapsulate a data unit from various higher-layer protocols at the source host, decapsulate a data unit, and deliver it to the various higher-layer protocols at the destination host.

5.5.3 UDP Checksum

The 16 bits of the checksum field are used to detect errors by finding out if any of the bits in the UDP segment have been altered during transmission due to noise in the links or when they are stored in the routers.
The checksum is calculated by the sending host by performing the 1's complement of the sum of all the 16-bit words of the UDP segment and the UDP pseudo-header. The overflow is wrapped around if it is generated while summing.
This can be explained with an example given below.

Eg: Consider the three 16-bit words in the UDP segment along with the pseudo-header as 0110011001100000, 0101010101010101 and 1000111100001100

At the Sending site:

- The sum of the three 16-bit words with a wrapped-around overflow is 0100101011000010
- The 1's complement of the sum is 101101010100111101
- This sum is added to the UDP segment in the checksum field and transmitted.

At the Receiving site:

- The sum of all the 16-bit words is added along with the sender's checksum.
- If the sum is 1111111111111111, there are no errors introduced in the UDP segment
- If any of the bits in the sum is 0, then the contents of the segment have errors and the packet is discarded.

5.5.4 UDP applications

1. File transfer applications using Trivial File Transfer Protocol (TFTP) can use UDP because error and flow control mechanisms are internally implemented in TFTP.
2. It is a suitable protocol for multicasting as this capability is embedded in the UDP software.
3. Used along with Routing Information Protocols (RIP) at the network layer and Simple Network Management Protocol (SNMP) at the application layer.
4. Used for real-time applications, where delay in transmission cannot be tolerated.

5.6 Self-Assessment Questions

Q1. What are the different services provided by the transport layer? (2 marks, L2)
Q2. Explain addressing in the context of transport layer service. (5 marks, L3)
Q3. Explain the process of Multiplexing and Demultiplexing at the transport layer (6 marks, L3)
Q4. Describe the flow control and error control processes at the transport layer (6 marks, L3)
Q5. Compare Connectionless and connection-oriented protocols (10 marks, L4)

Q6. With a neat diagram, explain the User datagram header format (5 marks, L3)

5.7 Self-Assessment Activities

A1. Find out some more applications of UDP protocols that you have used for communication on the Internet.

A2. Find out the commands used to find the addresses and port numbers by creating an application process.

A3. Using a network packet analyzer tool (Wireshark), capture network traffic, analyze the packets, and identify the transport layer protocols used, source and destination ports, and other control messages.

5.8 Multiple-Choice Questions

Q1. Which of these services is not provided by the transport layer protocols? [1 mark, L1]

A. Addressing
B. Multiplexing
C. Flow control
D. End-to-end delivery

Q2. Processes are identified with the help of, [1 mark, L1]

A. IP address
B. Port number
C. Both of the above
D. None of the above

Q3. Servers are assigned with _____ port numbers [1 mark, L1]

A. Well known
B. Ephemeral
C. Temporary
D. None of the above

Q4. A pair of information IP address and Port number is called ____ [1 mark, L1]

A. Interface
B. Socket
C. Any of the above
D. None of the above

Q5. The transport layer protocol used for email communication is, [1 mark, L1]

A. TCP
B. UDP
C. SMTP
D. TFTP

Q6. Which transport layer protocol is suitable for real-time applications [1 mark, L1]

A. TCP
B. UDP

C. Both the above

D. None of the above

Q7. The connectionless transport layer protocol is. [1 mark, L1]

A. TCP

B. UDP

C. HTTP

D. ICMP

5.9 Keys to Multiple-Choice Questions

Q1.	End-to-end delivery (D)
Q2.	Port number (B)
Q3.	Well-Known (A)
Q4.	Socket (B)
Q5.	SMTP (C)
Q6.	UDP (B)
Q7.	UDP (B)

5.10 Summary of the Unit

This unit covers two important topics related to the transport layer covering three sections: The first section of this unit discusses the important services of the transport layer in the protocol stack. The concept of process-to-process communication, addressing using port numbers, flow, and error control, use of multiplexing and demultiplexing for transferring the transport layer packets to appropriate applications, and vice-versa are studied.

In the second section, the services provided by connectionless and connection-oriented protocols, in general, are discussed.

In the third section, one of the important connectionless and unreliable transport layer protocols UDP is discussed. Its features, services, header format, and application areas are discussed briefly.

5.11 Recommended Learning Resources

[1] James F Kurose and Keith W Ross, Computer Networking, A Top-Down Approach, Sixth Edition, Pearson,2017.

[2] Behrouz A Forouzan, Data and Communications and Networking, Fifth Edition, McGraw Hill, Indian Edition

[3] https://people.cs.rutgers.edu/~pxk/rutgers/notes/sockets/

[4] https://www.informit.com/articles/article.aspx?p=169505&seqNum=2

[5] https://www.ijert.org/research/sockets-and-socket-address-structure-IJERTV1IS8559.pdf