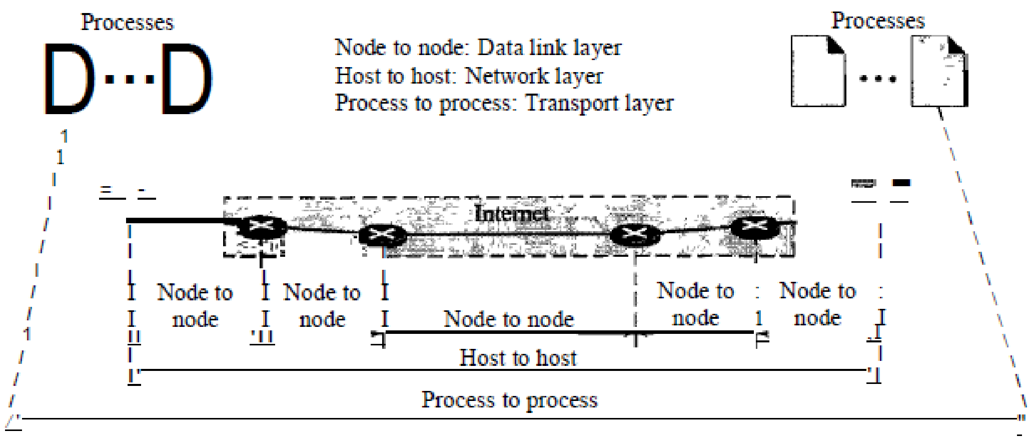
UNIT-III

TRANSPORT LAYER (Layer 4):

- The Transport Layer is the 4th Layer of OSI Model which takes data from Application Layer and sends data to the network layer.
- The transport layer is responsible for process-to-process delivery.
- Whereas, the data link layer is responsible for delivery of frames between two neighboring nodes over a link. This is called *node-to-node delivery.*
- The network layer is responsible for delivery of datagrams between two hosts. This is called *host-to-host delivery.*
- Communication on the Internet is not defined as the exchange of data between two nodes or between two hosts. Real communication takes place between two processes (application programs). We need process-to-process delivery. The transport layer is responsible for process-to-process delivery-the delivery of a packet, part of a message, from one process to another. Two processes communicate in a client/server relationship.
- There three types of deliveries and their domains as shown figure below.

Figure *Types ofdata deliveries*



USER DATAGRAM PROTOCOL (UDP)

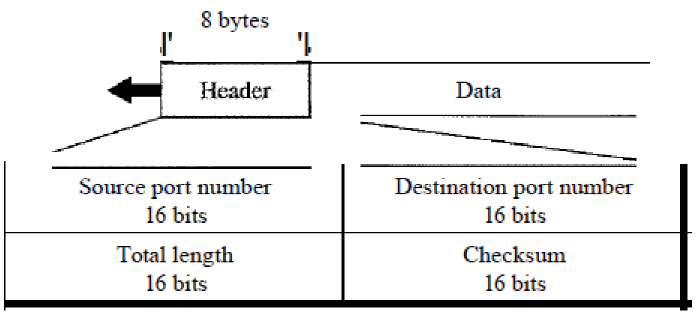
The User Datagram Protocol (UDP) is called a connectionless, unreliable transport protocol. It does not add anything to the services of IP except to provide process-to-process communication instead of host-to-host communication. Also, it performs very limited error checking.

If a process wants to send a small message and does not care much about reliability, it can use UDP. Sending a small message by using UDP takes much less interaction between the sender and receiver than using TCP or SCTP.

User Datagram

UDP packets, called user datagrams, have a fixed-size header of 8 bytes. The below Figure shows the format of a user datagram.

Figure *User datagram format*



The fields are as follows:

Source port number. This is the port number used by the process running on the source host. It is 16 bits long, which means that the port number can range from 0 to 65,535

Destination port number. This is the port number used by the process running on the destination host. It is also 16 bits long. If the destination host is the server (a client sending a request), the port number, in most cases, is a well-known port number. If the destination host is the client (a server sending a response), the port number, in most cases, is an ephemeral port number. In this case, the server copies the ephemeral port number it has received in the request packet.

Length. This is a 16-bit field that defines the total length of the user datagram, header plus data. The 16 bits can define a total length of 0 to 65,535 bytes. However, the total length needs to be much less because a UDP user datagram is stored in an IP datagram with a total length of 65,535 bytes.

UDP length = IP length - IP header's length

Checksum. This field is used to detect errors over the entire user datagram (header plus data). The checksum is discussed next.

TCP(Transport Layer Protocol)

TCP is most widely used for data for data transmission is communication network such as Internet.TCP provides Process-to-Process communication using port numbers.

Features:

- TCP is a process-to-process (program-to-program) protocol.
- Like UDP the TCP also uses port numbers.
- TCP is a connection-oriented protocol; it creates a virtual connection between two TCPs to send data. In addition, TCP uses flow and error control mechanisms at the transport level. In brief, TCP is called a *connection-oriented, reliable* transport protocol. It adds connection-oriented and reliability features to the services of IP.
- Uses Numbering system .ex: byte number, sequence number, acknowledgement number
- Flow control
- Error control
- Congestion control

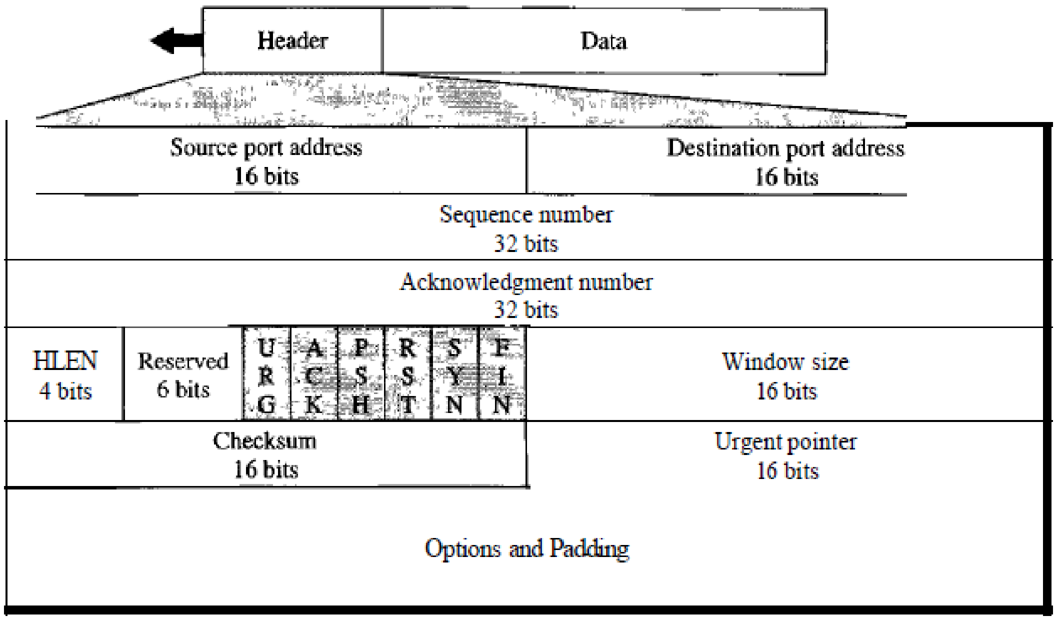
Segment

A packet in TCP is called a segment. The segment consists of a 20- to 60-byte header, followed by data from the application program. The header is 20 bytes if there are no options and up to 60 bytes if it contains options.

Format

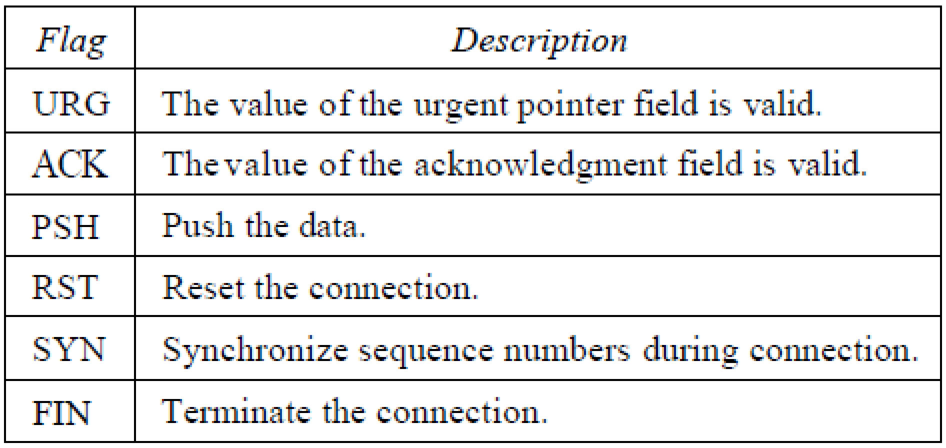
The format of a segment is as shown in below Figure

Figure *TCP segment format*



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. This serves the same purpose as the source port address in the UDP header.
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. This serves the same purpose as the destination port address in the UDP header.
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 comprises the first byte in the segment. During connection establishment, 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 defines *x* + I 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 can be between 5 (5 x 4 =20) and 15 (15 x 4 =60).
Reserved. This is a 6-bit field reserved for future use.
Control. This field defines 6 different control bits or flags as One or more of these bits can be set at a time.

Table : *Description offlags in the controlfield*



Flag	*Description*
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledgment field is valid.
PSH	Push the data.
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.

Window size. This field defines the size of the window, in bytes, that the other party must maintain. 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 inclusion of the checksum in the UDP datagram is optional, whereas the inclusion of the checksum for TCP is

mandatory. The same pseudoheader, serving the same purpose, is added to the segment. For the TCP pseudoheader, the value for the protocol field is 6.
Urgent pointer. This l6-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data. It defines the number that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment.
Options. There can be up to 40 bytes of optional information in the TCP header.

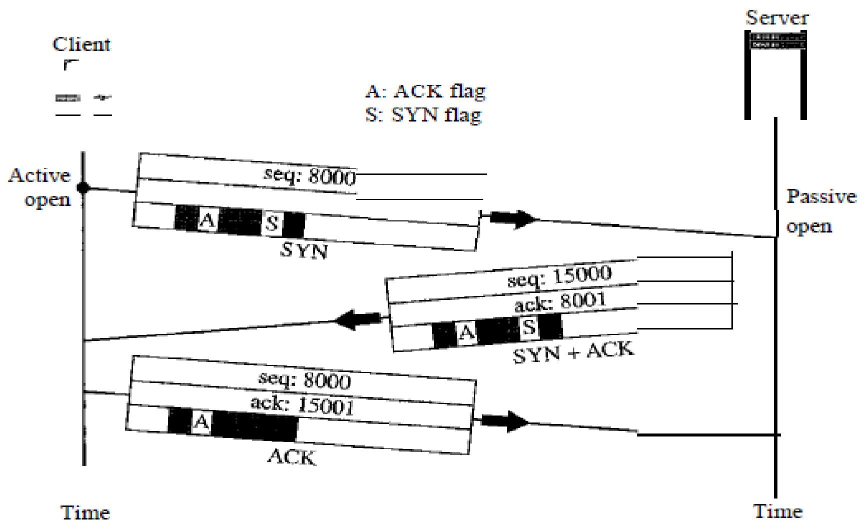
Three-Way Handshaking

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, using TCP as the transport layer protocol.

The process starts with the server. The server program tells its TCP that it is ready to accept a connection. This is called a request for a *passive open.* Although the server TCP is ready to accept any connection from any machine in the world, it cannot make the connection itself.

The client program issues a request for an *active open.* A client that wishes to connect to an open server tells its TCP that it needs to be connected to that particular server. TCP can now start the three-way handshaking process as shown in Figure

Figure *Connection establishment using three-way handshaking*



The sequence number, acknowledgment number, control flags (only those that are set), and the window size, if not empty. The three steps in this phase are as follows.

1. 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. It consumes one sequence number. When the data transfer starts, the sequence number is incremented by 1. We can say that the SYN segment carries no real data, but we can think of it as containing 1 imaginary byte.
   A SYN segment cannot carry data, but it consumes one sequence number.

2. The server sends the second segment, a SYN +ACK segment, with 2 flag bits set: SYN and ACK. This segment has a dual purpose. It is a SYN segment for communication in the other direction and serves as the acknowledgment for the SYN segment. It consumes one sequence number.

    A SYN +ACK segment cannot carry data, but does consume one sequence number.
3. The client sends the third segment. This is just an ACK segment. It acknowledges the receipt of the second segment with the ACK flag and acknowledgment number field. Note that the sequence number in this segment is the same as the one in the SYN segment; the ACK segment does not consume any sequence numbers.

    An ACK segment, if carrying no data, consumes no sequence number.

SCTP (**Stream Control Transmission Protocol)**

Stream Control Transmission Protocol (SCTP) is a new reliable, message-oriented transport layer protocol. SCTP, however, is mostly designed for Internet applications that have recently been introduced.

UDP: UDP is a message-oriented protocol. A process delivers a message to UDP, which is encapsulated in a user datagram and sent over the network. UDP *conserves the message boundaries;* each message is independent of any other message. UDP is unreliable; the sender cannot know the destiny of messages sent. A message can be lost, duplicated, or received out of order. UDP also lacks some other features, such as congestion control and flow control, needed for a friendly transport layer protocol.
TCP is a byte-oriented protocol. It receives a message or messages from a process, stores them as a stream of bytes, and sends them in segments. There is no preservation of the message boundaries. However, TCP is a reliable protocol. The duplicate segments are detected, the lost segments are resent, and the bytes are delivered to the end process in order. TCP also has congestion control and flow control mechanisms.
SCTP combines the best features of UDP and TCP. SCTP is a reliable message oriented protocol. It preserves the message boundaries and at the same time detects lost data, duplicate data, and out-of-order data. It also has congestion control and flow control mechanisms. Later we will see that SCTP has other innovative features unavailable in UDP and TCP.

Hence, SCTP is a *message-oriented, reliable* protocol that combines the best features of UDP and TCP.

SCTP Services

1. *Process-to-Process Communication*
2. *Multiple Streams*
3. *Multihoming*
4. *Full-Duplex Communication*
5. *Connection-Oriented Service*
6. *Reliable Service*

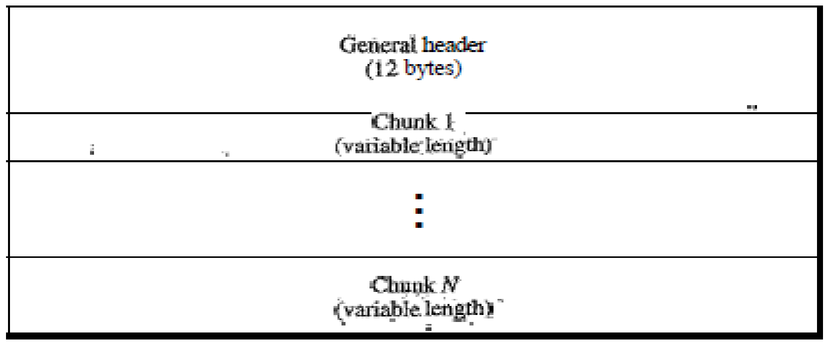
SCTP Features

1. *Transmission Sequence Number*
2. *Stream Identifier*
3. *Stream Sequence Number*
4. *Packets*
5. *Acknowledgment Number*
6. *Flow Control*
7. *Error Control*
8. *Congestion Control*

SCTP format

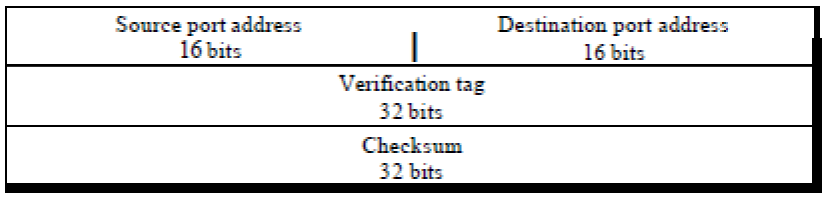
An SCTP packet has a mandatory general header and a set of blocks called chunks. There are two types of chunks: control chunks and data chunks. A control chunk controls and maintains the association; a data chunk carries user data. In a packet, the control chunks corne before the data chunks.
general format of an SCTP packet.

Figure *SCTP packet format*



The general header (packet header) defines the endpoints of each association to which the packet belongs, guarantees that the packet belongs to a particular association, and preserves the integrity of the contents of the packet including the header itself. The format of the general header is shown below figure.

Figure *General header*



There are four fields in the general header:

- **Source port address.** This is a 16-bit field that defines the port number of the process sending the packet.
- **Destination port address.** This is a 16-bit field that defines the port number of the process receiving the packet.
- **Verification tag.** This is a number that matches a packet to an association. This prevents a packet from a previous association from being mistaken as a packet in this association. It serves as an identifier for the association; it is repeated in every packet during the association. There is a separate verification used for each direction in the association.
- **Checksum**. This 32-bit field contains a CRC-32 checksum. Note that the size of the checksum is increased from 16 (in UDP, TCP, and IP) to 32 bits to allow the use of the CRC-32 checksum.

Congestion Control and Quality of Service

DATA TRAFFIC

The main focus of congestion control and quality of service is data traffic. In congestion control we try to avoid traffic congestion. In quality of service, we try to create an appropriate environment for the traffic.