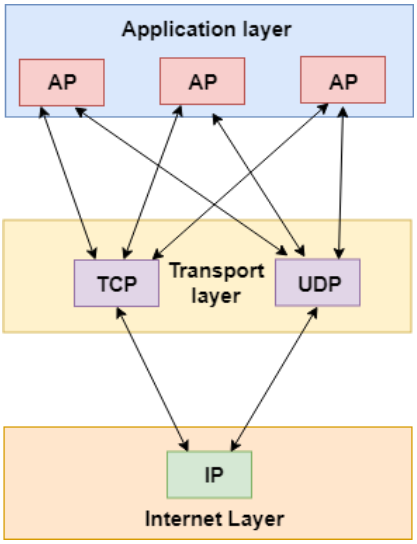
UNIT – 4

# TRANSPORT LAYER

*Introduction – Transport Layer Protocols – Services – Port Numbers – User Datagram Protocol – Transmission Control Protocol – SCTP.*

## 4.0 TRANSPORT LAYER

- The transport layer is a 4th layer from the top.
- The main role of the transport layer is to provide the communication services directly to the application processes running on different hosts.
- The transport layer provides a logical communication between application processes running on different hosts. Although the application processes on different hosts are not physically connected, application processes use the logical communication provided by the transport layer to send the messages to each other.
- The transport layer protocols are implemented in the end systems but not in the network routers.
- A computer network provides more than one protocol to the network applications. For example, TCP and UDP are two transport layer protocols that provide a different set of services to the network layer.
- All transport layer protocols provide multiplexing/demultiplexing service. It also provides other services such as reliable data transfer, bandwidth guarantees, and delay guarantees.
- Each of the applications in the application layer has the ability to send a message by using TCP or UDP. The application communicates by using either of these two protocols. Both TCP and UDP will then communicate with the internet protocol in the internet layer. The applications can read and write to the transport layer. Therefore, we can say that communication is a two-way process.




## 4.1 TRANSPORT LAYER PROTOCOLS

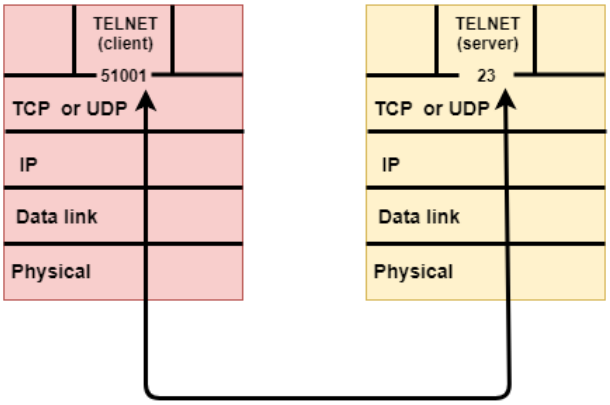
The transport layer is represented by two protocols: TCP and UDP.

The IP protocol in the network layer delivers a datagram from a source host to the destination host.

Nowadays, the operating system supports multiuser and multiprocessing environments, an executing program is called a process. When a host sends a message to other host means that source process is sending a process to a destination process. The transport layer protocols define some connections to individual ports known as protocol ports.

An IP protocol is a host-to-host protocol used to deliver a packet from source host to the destination host while transport layer protocols are port-to-port protocols that work on the top of the IP protocols to deliver the packet from the originating port to the IP services, and from IP services to the destination port.

Each port is defined by a positive integer address, and it is of 16 bits.



## UDP

- UDP stands for User Datagram Protocol.
- UDP is a simple protocol and it provides nonsequenced transport functionality.
- UDP is a connectionless protocol.
- This type of protocol is used when reliability and security are less important than speed and size.
- UDP is an end-to-end transport level protocol that adds transport-level addresses, checksum error control, and length information to the data from the upper layer.
- The packet produced by the UDP protocol is known as a user datagram.
- User Datagram Format
- The user datagram has a 16-byte header which is shown below:
- UDP packets, called user data grams, have a fixed-size header of 8 bytes. Figure shows the format of a user datagram

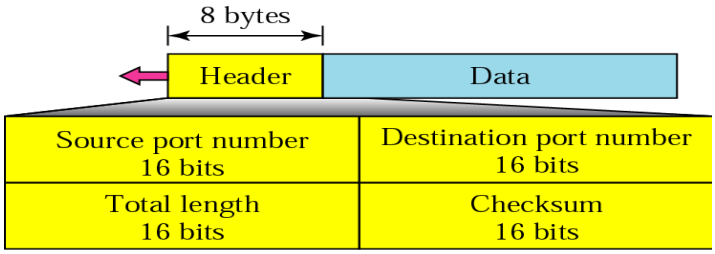



***Fig: UDP Structure***

### 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.
- If the source host is the client (a client sending a request), the port number, in most cases, is an ephemeral port number requested by the process and chosen by the UDP software running on the source host.
- If the source host is the server (a server sending a response), the port number, in most cases, is a well-known port number.

### Destination port number.

- 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.

### Checksum.

- This field is used to detect errors over the entire user datagram (header plus data).

### UDP Operation

- Connectionless Services
- Flow and Error Control
- Encapsulation and Decapsulation
- Queuing

### Use of UDP

- 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.
- UDP is used for management processes such as SNMP
- UDP is used for some route updating protocols such as Routing Information Protocol (RIP)

### TCP

- TCP stands for Transmission Control Protocol.
- It provides full transport layer services to applications.
- It is a connection-oriented protocol means the connection established between both the ends of the transmission. For creating the connection, TCP generates a virtual circuit between sender and receiver for the duration of a transmission.
- Features Of TCP protocol
- Stream data transfer: TCP protocol transfers the data in the form of contiguous stream of bytes. TCP group the bytes in the form of TCP segments and then passed it to the IP layer for transmission to the destination. TCP itself segments the data and forward to the IP.

**Reliability**: TCP assigns a sequence number to each byte transmitted and expects a positive acknowledgement from the receiving TCP. If ACK is not received within a timeout interval, then the data is retransmitted to the destination.

The receiving TCP uses the sequence number to reassemble the segments if they arrive out of order or to eliminate the duplicate segments.

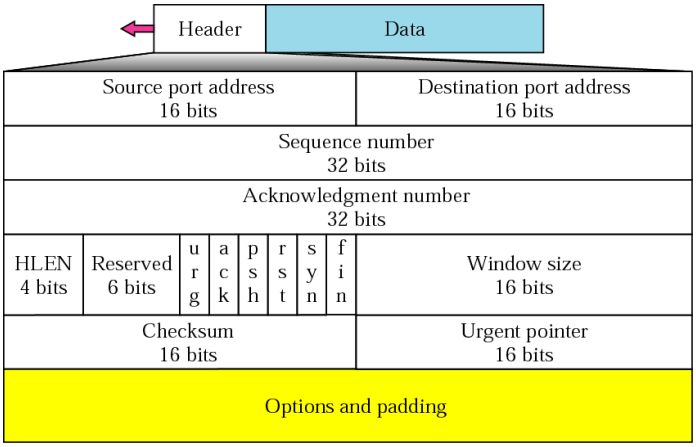
**Flow Control**: When receiving TCP sends an acknowledgement back to the sender indicating the number the bytes it can receive without overflowing its internal buffer. The number of bytes is sent in ACK in the form of the highest sequence number that it can receive without any problem. This mechanism is also referred to as a window mechanism.

**Multiplexing:** Multiplexing is a process of accepting the data from different applications and forwarding to the different applications on different computers. At the receiving end, the data is forwarded to the correct application. This process is known as demultiplexing. TCP transmits the packet to the correct application by using the logical channels known as ports.

**Logical Connections**: The combination of sockets, sequence numbers, and window sizes, is called a logical connection. Each connection is identified by the pair of sockets used by sending and receiving processes.

**Full Duplex:** TCP provides Full Duplex service, i.e., the data flow in both the directions at the same time. To achieve Full Duplex service, each TCP should have sending and receiving buffers so that the segments can flow in both the directions. TCP is a connection-oriented protocol. Suppose the process A wants to send and receive the data from process B. The following steps occur:

- Establish a connection between two TCPs.
- Data is exchanged in both the directions.
- The Connection is terminated.




Where,

**Source port address**: It is used to define the address of the application program in a source computer. It is a 16-bit field.

**Destination port address**: It is used to define the address of the application program in a destination computer. It is a 16-bit field.

Sequence number: A stream of data is divided into two or more TCP segments. The 32-bit sequence number field represents the position of the data in an original data stream.

**Acknowledgement number**: A 32-field acknowledgement number acknowledge the data from other communicating devices. If ACK field is set to 1, then it specifies the sequence number that the receiver is expecting to receive.

**Header Length (HLEN):** It specifies the size of the TCP header in 32-bit words. The minimum size of the header is 5 words, and the maximum size of the header is 15 words. Therefore, the maximum size of the TCP header is 60 bytes, and the minimum size of the TCP header is 20 bytes.

**Reserved:** It is a six-bit field which is reserved for future use.

**Control bits**: Each bit of a control field functions individually and independently. A control bit defines the use of a segment or serves as a validity check for other fields.

There are total six types of flags in control field:

**URG:** The URG field indicates that the data in a segment is urgent.

**ACK:** When ACK field is set, then it validates the acknowledgement number.

**PSH:** The PSH field is used to inform the sender that higher throughput is needed so if possible, data must be pushed with higher throughput.

**RST:** The reset bit is used to reset the TCP connection when there is any confusion occurs in the sequence numbers.

**SYN:** The SYN field is used to synchronize the sequence numbers in three types of segments: connection request, connection confirmation ( with the ACK bit set ), and confirmation acknowledgement.

**FIN:** The FIN field is used to inform the receiving TCP module that the sender has finished sending data. It is used in connection termination in three types of segments: termination request, termination confirmation, and acknowledgement of termination confirmation.

**Window Size:** The window is a 16-bit field that defines the size of the window.

**Checksum:** The checksum is a 16-bit field used in error detection.

**Urgent pointer:** If URG flag is set to 1, then this 16-bit field is an offset from the sequence number indicating that it is a last urgent data byte.

**Options and padding:** It defines the optional fields t hat convey the additional information to the receiver.

## Differences b/w TCP & UDP

| Basis for Comparison | TCP | UDP |
|---|---|---|
| Definition | TCP establishes a virtual circuit before transmitting the data. | UDP transmits the data directly to the destination computer without verifying whether the receiver is ready to receive or not. |

| Connection Type | It is a Connection-Oriented protocol | It is a Connectionless protocol |
|---|---|---|
| Speed | slow | high |
| Reliability | It is a reliable protocol. | It is an unreliable protocol. |
| Header size | 20 bytes | 8 bytes |
| acknowledgement | It waits for the acknowledgement of data and has the ability to resend the lost packets. | It neither takes the acknowledgement, nor it retransmits the damaged frame. |

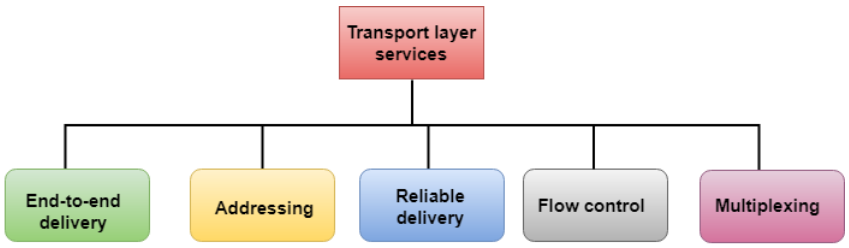
## 4.3 SERVICES

Services provided by the Transport Layer

- The services provided by the transport layer are similar to those of the data link layer.
- The data link layer provides the services within a single network while the transport layer provides the services across an internetwork made up of many networks.
- The data link layer controls the physical layer while the transport layer controls all the lower layers.

The services provided by the transport layer protocols can be divided into five categories:

- End-to-end delivery
- Addressing
- Reliable delivery
- Flow control
- Multiplexing




### End-to-end delivery:

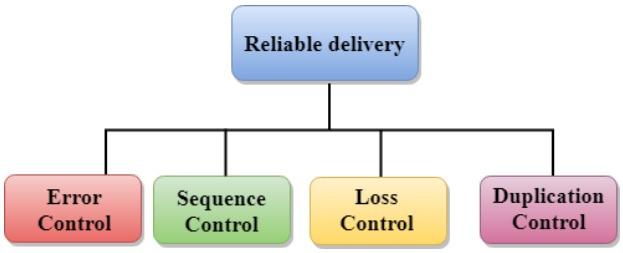
The transport layer transmits the entire message to the destination. Therefore, it ensures the end-to-end delivery of an entire message from a source to the destination.

### Reliable delivery:

The transport layer provides reliability services by retransmitting the lost and damaged packets.

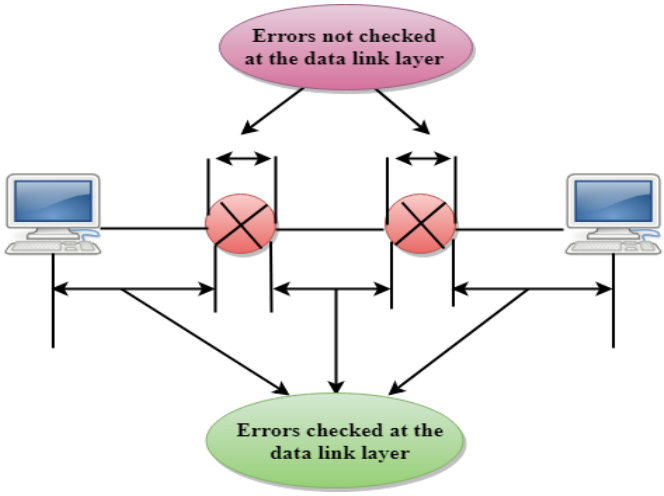
The reliable delivery has four aspects:

- Error control
- Sequence control
- Loss control
- Duplication control




### *Error Control*

- The primary role of reliability is **Error Control**. In reality, no transmission will be 100 percent error-free delivery. Therefore, transport layer protocols are designed to provide error-free transmission.
- The data link layer also provides the error handling mechanism, but it ensures only node-to-node error-free delivery. However, node-to-node reliability does not ensure the end-to-end reliability.
- The data link layer checks for the error between each network. If an error is introduced inside one of the routers, then this error will not be caught by the data link layer. It only detects those errors that have been introduced between the beginning and end of the link. Therefore, the transport layer performs the checking for the errors end-to-end to ensure that the packet has arrived correctly.




### *Sequence Control*

- The second aspect of the reliability is sequence control which is implemented at the transport layer.

- On the sending end, the transport layer is responsible for ensuring that the packets received from the upper layers can be used by the lower layers. On the receiving end, it ensures that the various segments of a transmission can be correctly reassembled.

### *Loss Control*

Loss Control is a third aspect of reliability. The transport layer ensures that all the fragments of a transmission arrive at the destination, not some of them. On the sending end, all the fragments of transmission are given sequence numbers by a transport layer. These sequence numbers allow the receivers transport layer to identify the missing segment.

### *Duplication Control*

Duplication Control is the fourth aspect of reliability. The transport layer guarantees that no duplicate data arrive at the destination. Sequence numbers are used to identify the lost packets; similarly, it allows the receiver to identify and discard duplicate segments.

### *Flow Control*

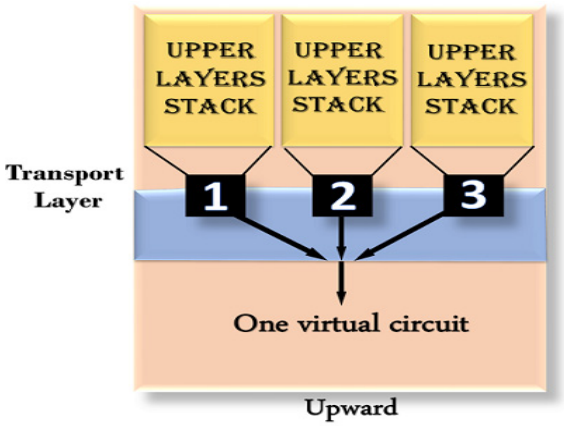
Flow control is used to prevent the sender from overwhelming the receiver. If the receiver is overloaded with too much data, then the receiver discards the packets and asking for the retransmission of packets. This increases network congestion and thus, reducing the system performance. The transport layer is responsible for flow control. It uses the sliding window protocol that makes the data transmission more efficient as well as it controls the flow of data so that the receiver does not become overwhelmed. Sliding window protocol is byte oriented rather than frame oriented.

### *Multiplexing*

The transport layer uses the multiplexing to improve transmission efficiency.

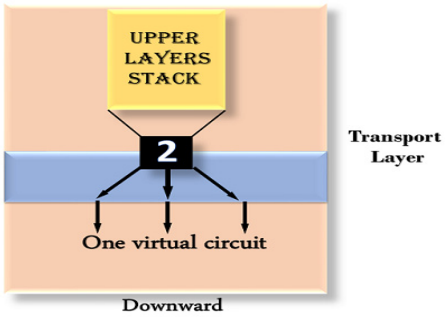
### Multiplexing can occur in two ways:

- **Upward multiplexing:** Upward multiplexing means multiple transport layer connections use the same network connection. To make more cost-effective, the transport layer sends several transmissions bound for the same destination along the same path; this is achieved through upward multiplexing.




- **Downward multiplexing:** Downward multiplexing means one transport layer connection uses the multiple network connections. Downward multiplexing allows the

transport layer to split a connection among several paths to improve the throughput. This type of multiplexing is used when networks have a low or slow capacity.

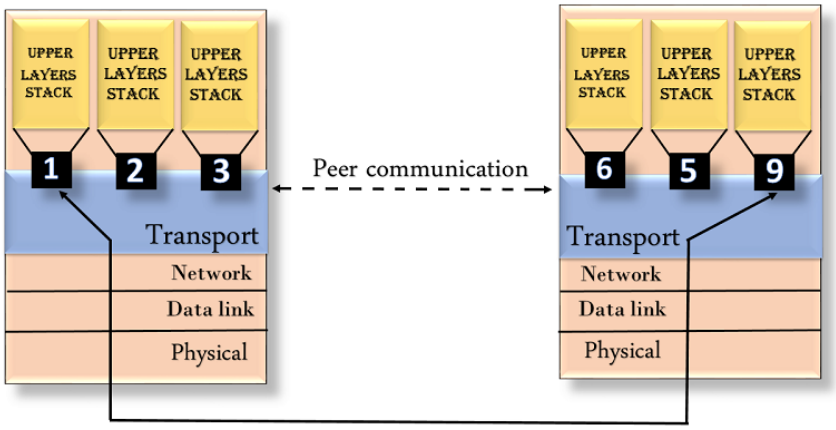



## Addressing

According to the layered model, the transport layer interacts with the functions of the session layer. Many protocols combine session, presentation, and application layer protocols into a single layer known as the application layer. In these cases, delivery to the session layer means the delivery to the application layer. Data generated by an application on one machine must be transmitted to the correct application on another machine. In this case, addressing is provided by the transport layer.

The transport layer provides the user address which is specified as a station or port. The port variable represents a particular TS user of a specified station known as a Transport Service access point (TSAP). Each station has only one transport entity.

The transport layer protocols need to know which upper-layer protocols are communicating.




## 4.4 PORT NUMBERS

In computer networking, a port is an endpoint of communication. Physical as well as wireless connections are terminated at ports of hardware devices.

At the software level, within an operating system, a port is a logical construct that identifies a specific process or a type of network service. Ports are identified for each protocol and address combination by 16-bit unsigned numbers, commonly known as the port number.

Inbound packets are received, and the port number in the header is used to decide which application is to be passed the packets.

Ports provide a multiplexing service for multiple services or multiple communication sessions at one network address. In the client–server model of application architecture, a multiplexing service is established, so that multiple simultaneous communication sessions may be initiated for the same service.

The most commonly used protocols that use ports are the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP).

Specific port numbers are commonly reserved to identify specific services. The lowest numbered 1024 port numbers are called the well-known port numbers. Higher-numbered ports are available for general use by applications and are known as ephemeral ports.

A port number is a 16-bit unsigned integer, thus ranging from 0 to 65535. For TCP, port number 0 is reserved and cannot be used, while for UDP, the source port is optional and a value of zero means no port. A process associates its input or output channels via an Internet socket, which is a type of file descriptor, with a transport protocol, an IP address, and a port number. This is known as binding, and enables the process to send and receive data via the network.

## Common port numbers

The Internet Assigned Numbers Authority (IANA) is responsible for the global coordination of the DNS Root, IP addressing, and other Internet protocol resources. This includes the registration of commonly used port numbers for well-known Internet services.

The port numbers are divided into three ranges: the well-known ports, the registered ports, and the dynamic or private ports.

The well-known ports (also known as system ports) are those from 0 through 1023. The requirements for new assignments in this range are stricter than for other registrations, examples include:

20: File Transfer Protocol (FTP) Data Transfer

21: File Transfer Protocol (FTP) Command Control

22: Secure Shell (SSH) Secure Login

23: Telnet remote login service, unencrypted text messages

25: Simple Mail Transfer Protocol (SMTP) E-mail routing

53: Domain Name System (DNS) service

80: Hypertext Transfer Protocol (HTTP) used in the World Wide Web

110: Post Office Protocol (POP3)

119: Network News Transfer Protocol (NNTP)

123: Network Time Protocol (NTP)

143: Internet Message Access Protocol (IMAP) Management of digital mail

161: Simple Network Management Protocol (SNMP)

194: Internet Relay Chat (IRC)

443: HTTP Secure (HTTPS) HTTP over TLS/SSL

The registered ports are those from 1024 through 49151. IANA maintains the official list of well-known and registered ranges.[3] The dynamic or private ports are those from 49152 through 65535. One common use for this range is for ephemeral ports.

## 4.5 TRANSMISSION CONTROL PROTOCOL

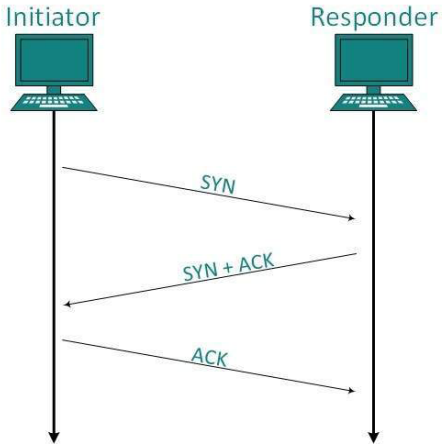
- It is a more sophisticated transport protocol is one that offers a reliable, connection-oriented, byte-stream service.
- Transmission Control Protocol (TCP) is probably the most widely used protocol.
- It also includes a flow-control. Like UDP, TCP supports a demultiplexing mechanism that allows multiple application programs on any given host to simultaneously carry on a conversation with their peers.
- TCP also implements a highly tuned congestion-control mechanism.
- End to End Issues
- At the heart of TCP is the sliding window algorithm.
- TCP supports logical connections between processes that are running on any two computers in the Internet.
- This means that TCP needs an explicit connection establishment phase during which the two sides of the connection agree to exchange data with each other.
- This difference is analogous to having to dial up the other party, rather than having a dedicated phone line.
- TCP also has an explicit connection teardown phase.
- One of the things that happens during connection establishment is that the two parties establish some shared state to enable the sliding window algorithm to begin.
- connection teardown is needed so each host knows it is OK to free this state.
- Second, whereas a single physical link that always connects the same two computers has a fixed RTT, TCP connections are likely to have widely different round-trip times.

### Addressing

TCP communication between two remote hosts is done by means of port numbers (TSAPs). Ports numbers can range from 0 – 65535 which are divided as:

- System Ports (0 – 1023)
- User Ports ( 1024 – 49151)
- Private/Dynamic Ports (49152 – 65535)
- Connection Management

TCP communication works in Server/Client model. The client initiates the connection and the server either accepts or rejects it. Three-way handshaking is used for connection management.




## Establishment:

Client initiates the connection and sends the segment with a Sequence number. Server acknowledges it back with its own Sequence number and ACK of client's segment which is one more than client's Sequence number. Client after receiving ACK of its segment sends an acknowledgement of Server's response.

## Release:

Either of server and client can send TCP segment with FIN flag set to 1. When the receiving end responds it back by Acknowledging FIN, that direction of TCP communication is closed and connection is released.

## Bandwidth Management:

CP uses the concept of window size to accommodate the need of Bandwidth management. Window size tells the sender at the remote end, the number of data byte segments the receiver at this end can receive. TCP uses slow start phase by using window size 1 and increases the window size exponentially after each successful communication.

For example, the client uses windows size 2 and sends 2 bytes of data. When the acknowledgement of this segment received the windows size is doubled to 4 and next sent the segment sent will be 4 data bytes long. When the acknowledgement of 4-byte data segment is received, the client sets windows size to 8 and so on.

If an acknowledgement is missed, i.e. data lost in transit network or it received NACK, then the window size is reduced to half and slow start phase starts again.

## Three-Way Handshake

The following scenario occurs when a TCP connection is established:

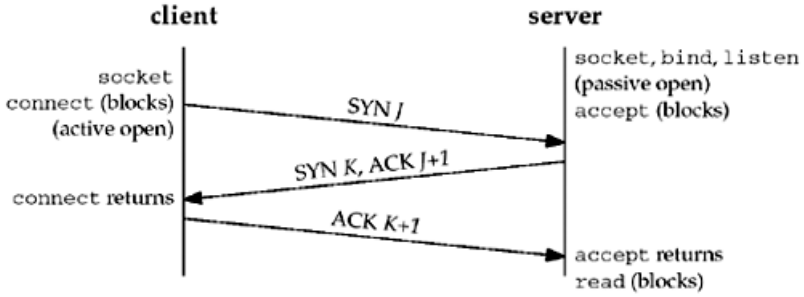
The server must be prepared to accept an incoming connection. This is normally done by calling socket, bind, and listen and is called a passive open.

The client issues an active open by calling connect. This causes the client TCP to send a "synchronize" (SYN) segment, which tells the server the client's initial sequence number for the data that the client will send on the connection. Normally, there is no data sent with the SYN; it just contains an IP header, a TCP header, and possible TCP options (which we will talk about shortly).

The server must acknowledge (ACK) the client's SYN and the server must also send its own SYN containing the initial sequence number for the data that the server will send on the connection. The server sends its SYN and the ACK of the client's SYN in a single segment.

The client must acknowledge the server's SYN.

The minimum number of packets required for this exchange is three; hence, this is called TCP's three-way handshake. We show the three segments in Figure




We show the client's initial sequence number as J and the server's initial sequence number as K. The acknowledgment number in an ACK is the next expected sequence number for the end sending the ACK. Since a SYN occupies one byte of the sequence number space, the acknowledgment number in the ACK of each SYN is the initial sequence number plus one. Similarly, the ACK of each FIN is the sequence number of the FIN plus one.

An everyday analogy for establishing a TCP connection is the telephone system The socket function is the equivalent of having a telephone to use. bind is telling other people your telephone number so that they can call you. listen is turning on the ringer so that you will hear when an incoming call arrives. connect requires that we know the other person's phone number and dial it. accept is when the person being called answers the phone.

Having the client's identity returned by accept (where the identify is the client's IP address and port number) is similar to having the caller ID feature show the caller's phone number. One difference, however, is that accept returns the client's identity only after the connection has been established, whereas the caller ID feature shows the caller's phone number before we choose whether to answer the phone or not. If the DNS is used, it provides a service analogous to a telephone book. getaddrinfo is similar to looking up a person's phone number in the phone book. getnameinfo would be the equivalent of having a phone book sorted by telephone numbers that we could search, instead of a book sorted by name.

## TCP Options

Each SYN can contain TCP options. Commonly used options include the following:

**MSS option**. With this option, the TCP sending the SYN announces its maximum segment size, the maximum amount of data that it is willing to accept in each TCP segment, on this connection. The sending TCP uses the receiver's MSS value as the maximum size of a segment that it sends.

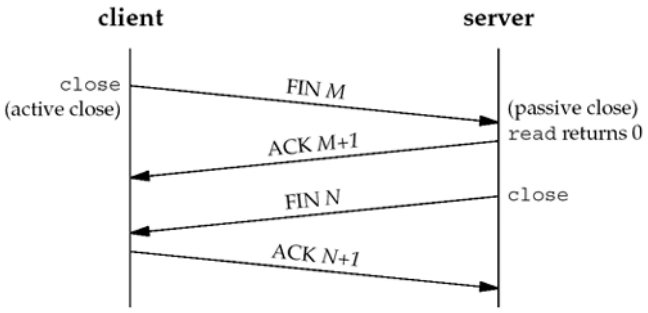
**Window scale option.** The maximum window that either TCP can advertise to the other TCP is 65,535, because the corresponding field in the TCP header occupies 16 bits. But, high-speed connections, common in today's Internet (45 Mbits/sec and faster, as described in RFC 1323 [Jacobson, Braden, and Borman 1992]), or long delay paths (satellite links) require a larger window to obtain the maximum throughput possible. This newer option specifies that the advertised window in the TCP header must be scaled (left-shifted) by 0–14 bits, providing a maximum window of almost one gigabyte (65,535 x 214). Both end-systems must support this option for the window scale to be used on a connection.

To provide interoperability with older implementations that do not support this option, the following rules apply. TCP can send the option with its SYN as part of an active open. But it can scale its windows only if the other end also sends the option with its SYN. Similarly, the server's TCP can send this option only if it receives the option with the client's SYN. This logic assumes that implementations ignore options that they do not understand, which is required and common, but unfortunately, not guaranteed with all implementations.

**Timestamp option:** This option is needed for high-speed connections to prevent possible data corruption caused by old, delayed, or duplicated segments.

## TCP Connection Termination

- While it takes three segments to establish a connection, it takes four to terminate a connection.
- One application calls close first, and we say that this end performs the active close. This end's TCP sends a FIN segment, which means it is finished sending data.
- The other end that receives the FIN performs the passive close. The received FIN is acknowledged by TCP. The receipt of the FIN is also passed to the application as an end-of-file (after any data that may have already been queued for the application to receive), since the receipt of the FIN means the application will not receive any additional data on the connection.
- Sometime later, the application that received the end-of-file will close its socket. This causes its TCP to send a FIN.
- The TCP on the system that receives this final FIN (the end that did the active close) acknowledges the FIN.
- Since a FIN and an ACK are required in each direction, four segments are normally required. We use the qualifier "normally" because in some scenarios, the FIN in Step 1 is sent with data. Also, the segments in Steps 2 and 3 are both from the end performing the passive close and could be combined into one segment. We show these packets



- A FIN occupies one byte of sequence number space just like a SYN. Therefore, the ACK of each FIN is the sequence number of the FIN plus one.
- Between Steps 2 and 3 it is possible for data to flow from the end doing the passive close to the end doing the active close. This is called a half-close
- The sending of each FIN occurs when a socket is closed. We indicated that the application calls close for this to happen, but realize that when a Unix process terminates, either voluntarily (calling exit or having the main function return) or involuntarily (receiving a signal that terminates the process), all open descriptors are closed, which will also cause a FIN to be sent on any TCP connection that is still open.
- Although we show the client performing the active close, either end—the client or the server—can perform the active close. Often the client performs the active close, but with some protocols (notably HTTP/1.0), the server performs the active close.

## TCP State Transition Diagram

The operation of TCP with regard to connection establishment and connection termination can be specified with a state transition diagram.

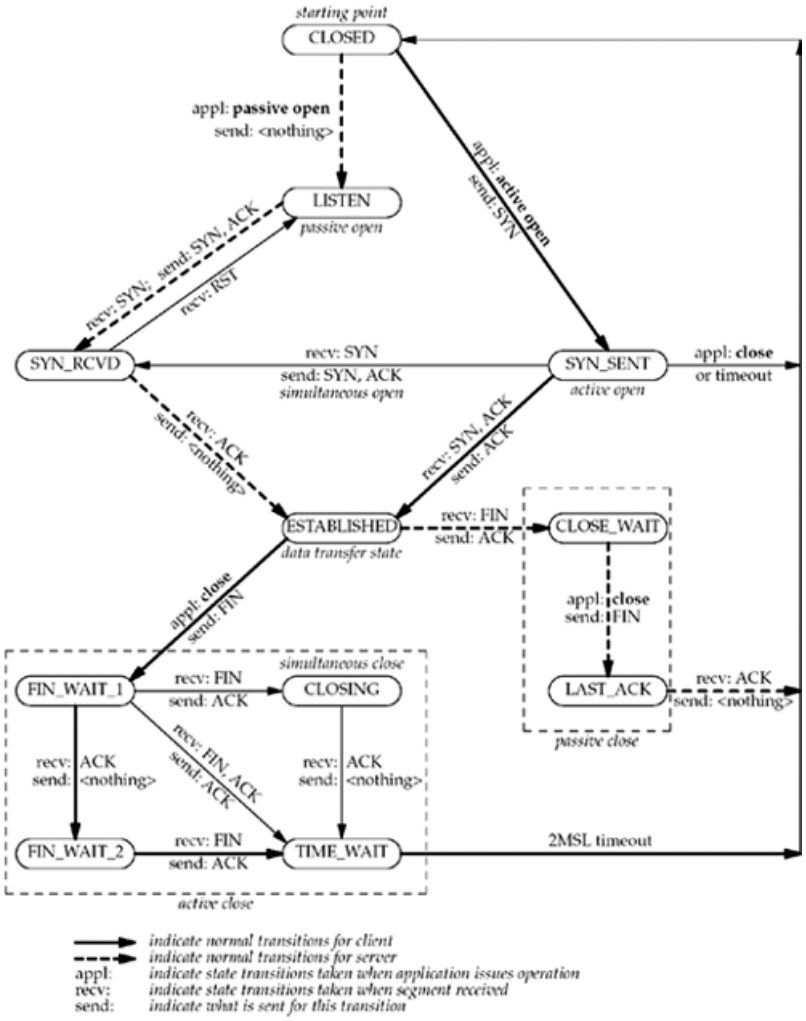


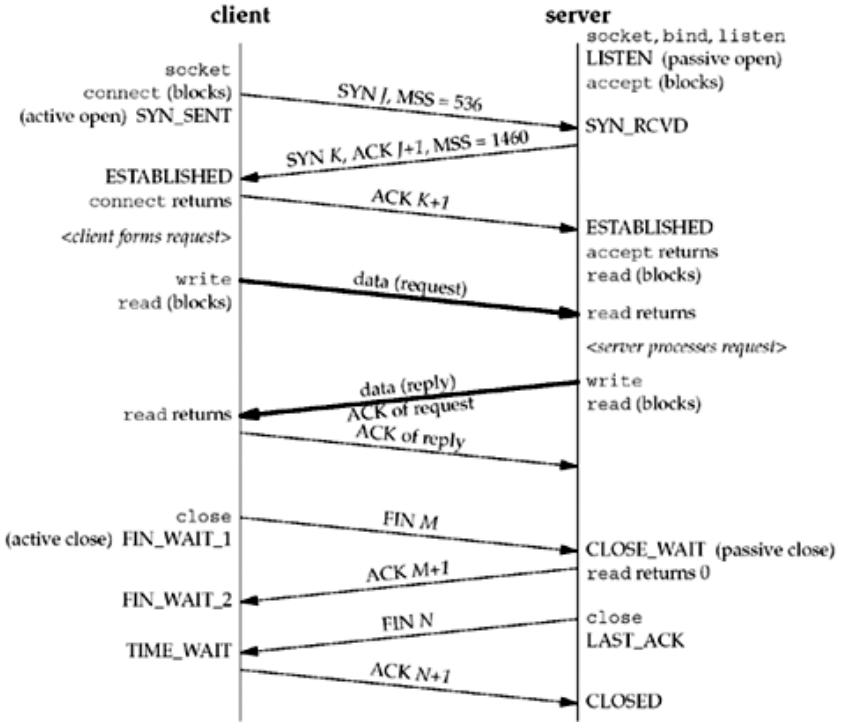

***Fig : TCP state transition diagram***

There are 11 different states defined for a connection and the rules of TCP dictate the transitions from one state to another, based on the current state and the segment received in that state. For example, if an application performs an active open in the CLOSED state, TCP sends a SYN and the new state is SYN_SENT. If TCP next receives a SYN with an ACK, it sends an ACK and the new state is ESTABLISHED. This final state is where most data transfer occurs.

The two arrows leading from the ESTABLISHED state deal with the termination of a connection. If an application calls close before receiving a FIN (an active close), the transition is to the FIN_WAIT_1 state. But if an application receives a FIN while in the ESTABLISHED state (a passive close), the transition is to the CLOSE_WAIT state.

## Watching the Packets

The actual packet exchange that takes place for a complete TCP connection: the connection establishment, data transfer, and connection termination.




The client in this example announces an MSS of 536 (indicating that it implements only the minimum reassembly buffer size) and the server announces an MSS of 1,460 (typical for IPv4 on an Ethernet).

Once a connection is established, the client forms a request and sends it to the server. We assume this request fits into a single TCP segment (i.e., less than 1,460 bytes given the server's announced MSS). The server processes the request and sends a reply, and we assume that the reply fits in a single segment (less than 536 in this example). We show both data segments as bolder arrows. Notice that the acknowledgment of the client's request is sent with the server's reply. This is called piggybacking and will normally happen when the time it takes the server to process the request and generate the reply is less than around 200 ms. If the server takes longer, say one second, we would see the acknowledgment followed later by the reply.

It is important to notice in Figure that if the entire purpose of this connection was to send a one-segment request and receive a one-segment reply, there would be eight segments of overhead involved when using TCP. If UDP was used instead, only two packets would be exchanged: the request and the reply. But switching from TCP to UDP removes all the reliability that TCP provides to the application, pushing lots of these details from the transport layer (TCP) to the UDP application.

Another important feature provided by TCP is congestion control, which must then be handled by the UDP application. Nevertheless, it is important to understand that many applications are built using UDP because the application exchanges small amounts of data and UDP avoids the overhead of TCP connection establishment and connection termination.

## 4.6 STREAM CONTROL TRANSMISSION PROTOCOL

The Stream Control Transmission Protocol (SCTP) is a computer networking communications protocol which operates at the transport layer and serves a role similar to the popular protocols TCP and UDP

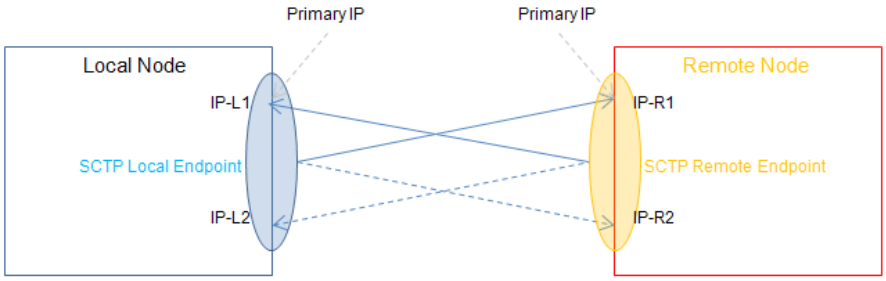
SCTP provides some of the features of both UDP and TCP: it is message-oriented like UDP and ensures reliable, in-sequence transport of messages with congestion control like TCP. It differs from those protocols by providing multi-homing and redundant paths to increase resilience and reliability.

### Features of SCTP include:

- Reliable transmission of both ordered and unordered data streams.
- Multihoming support in which one or both endpoints of a connection can consist of more than one IP address, enabling transparent fail-over between redundant network paths.
- Delivery of chunks within independent streams eliminates unnecessary head-of-line blocking, as opposed to TCP byte-stream delivery.
- Explicit partial reliability.
- Path selection and monitoring to select a primary data transmission path and test the connectivity of the transmission path.
- Validation and acknowledgment mechanisms protect against flooding attacks and provide notification of duplicated or missing data chunks.
- Improved error detection suitable for Ethernet jumbo frames.

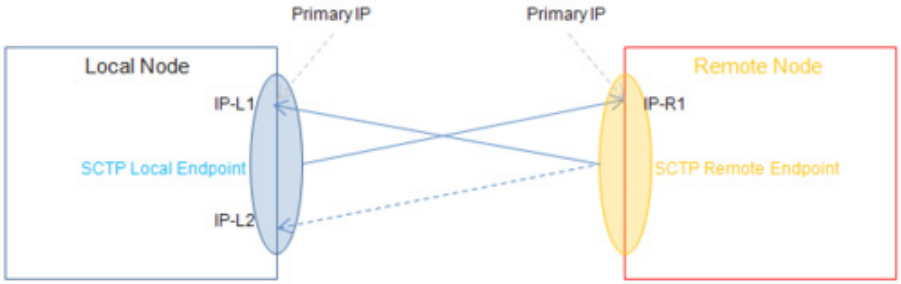
### Multi homing

- SCTP provides redundant paths to increase reliability.
- Each SCTP end point needs to check reachability of the primary and redundant addresses of the remote end point using a heartbeat. Each SCTP end point needs to ack the heartbeats it receives from the remote end point.
- When SCTP sends a message to a remote address, the source interface will only be decided by the routing table of the host (and not by SCTP)

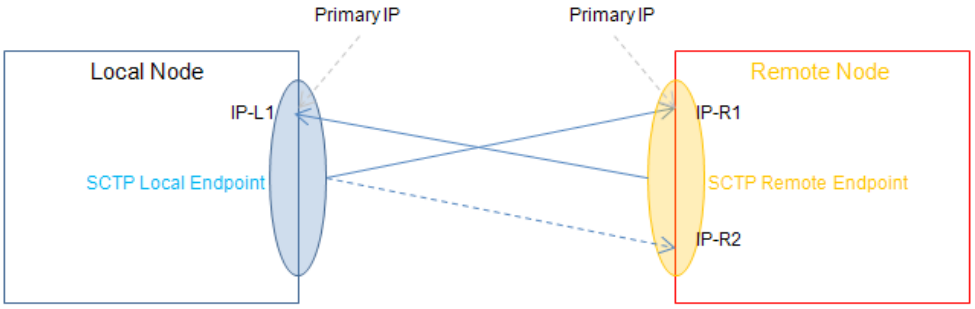



## Asymmetric multi homing

In asymmetric multi homing, one of the two end points does not support multi homing.




Local multi homing - Remote single homing




In Local multi homing and Remote single homing, if the remote primary address is not reachable, the SCTP association fails even if an alternate path is possible.

## Packet structure

Every SCTP packet contains the Common header as seen above. The header contains four different fields and is set for every SCTP packet.

- **Source port** - bit 0-15. This field gives the source port of the packet, which port it was sent from. The same as for TCP and UDP source port.
- **Destination port** - bit 16-31. This is the destination port of the packet, ie., the port that the packet is going to. It is the same as for the TCP and UDP destination port.
- **Verification Tag** - bit 32-63. The verification tag is used to verify that the packet comes from the correct sender. It is always set to the same value as the value received by the other peer in the Initiate Tag during the association initialization, with a few exceptions:
- An SCTP packet containing an INIT chunk must have the Verification tag set to 0.

- A SHUTDOWN COMPLETE chunk with the T-bit set must have the verification tag copied from the verification tag of the SHUTDOWN-ACK chunk.
- Packets containing ABORT chunk may have the verification tag set to the same verification tag as the packet causing the ABORT.
- Checksum - bit 64-95. A checksum calculated for the whole SCTP packet based on the Adler-32 algorithm. Read RFC 2960 - Stream Control Transmission Protocol, appendix B for more information about this algorithm.

| Bits | 0–7 | 8–15 | 16–23 | 24–31 |
|---|---|---|---|---|
| +0 | Source port | | Destination port | |
| 32 | Verification tag | | | |
| 64 | Checksum | | | |
| 96 | Chunk 1 type | Chunk 1 flags | Chunk 1 length | |
| 128 | Chunk 1 data | | | |
| … | … | | | |
| … | Chunk N type | Chunk N flags | Chunk N length | |
| … | Chunk N data | | | |

## SCTP Types

| Chunk Number | Chunk Name |
|---|---|
| 0 | Payload Data (DATA) |
| 1 | Initiation (INIT) |
| 2 | Initiation Acknowledgement (INIT ACK) |
| 3 | Selective Acknowledgement (SACK) |
| 4 | Heartbeat Request (HEARTBEAT) |
| 5 | Heartbeat Acknowledgement (HEARTBEAT ACK) |
| 6 | Abort (ABORT) |
| 7 | Shutdown (SHUTDOWN) |
| 8 | Shutdown Acknowledgement (SHUTDOWN ACK) |
| 9 | Operation Error (ERROR) |
| 10 | State Cookie (COOKIE ECHO) |
| 11 | Cookie Acknowledgement (COOKIE ACK) |
| 12 | Reserved for Explicit Congestion Notification Echo (ECNE) |
| 13 | Reserved for Congestion Window Reduced (CWR) |
| 14 | Shutdown Complete (SHUTDOWN COMPLETE) |
| 15-62 | Reserved for IETF |
| 63 | IETF-defined chunk extensions |
| 64-126 | reserved to IETF |
| 127 | IETF-defined chunk extensions |

| Chunk Number | Chunk Name |
|---|---|
| 128-190 | reserved to IETF |
| 191 | IETF-defined chunk extensions |
| 192-254 | reserved to IETF |
| 255 | IETF-defined chunk extensions |

- ***Chunk Flags*** - bit 8-15. The chunk flags are generally not used but are set up for future usage if nothing else. They are chunk specific flags or bits of information that might be needed for the other peer. According to specifications, flags are only used in DATA, ABORT and SHUTDOWN COMPLETE packets at this moment.
- ***Chunk Length -*** bit 16-31. This is the chunk length calculated in bytes. It includes all headers, including the chunk type, chunk flags, chunk length and chunk value. If there is no chunk value, the chunk length will be set to 4 (bytes).
- ***Chunk Value -*** bit 32-n. This is specific to each chunk and may contain more flags and data pertaining to the chunk type. Sometimes it might be empty, in which case the chunk length will be set to 4.