Module 4

Transport Layer:
Transport layer services provided to the upper layers elements of transport protocol addressing connection establishment, Connection release, Error Control & Flow Control.
Internet Transport Protocols: UDP vs TCP, The TCP Service Model, The TCP Segment Header, Connection Establishment, Connection Release. TCP Sliding Window and Congestion Control Algorithm.

SERVICES PROVIDED BY TRANSPORT LAYER

The Transport Layer, the fourth layer in the OSI (Open Systems Interconnection) model and the TCP/IP protocol suite, provides essential services for data transport and communication between devices on a network. Its primary functions and services include:

1. Segmentation and Reassembly: The Transport Layer divides large messages or data streams into smaller segments for efficient transmission over the network. It ensures that these segments are reassembled correctly at the destination.

2. End-to-End Communication: It enables communication between two devices, regardless of their location in the network, by addressing data to specific processes or services on each device.

3. Connection Establishment and Termination: Depending on the transport protocol in use (e.g., TCP), the Transport Layer manages the establishment and termination of connections between devices. It ensures that data is reliably delivered between endpoints.

4. Flow Control: The Transport Layer manages the rate of data exchange between sender and receiver to prevent overwhelming the receiver with more data than it can handle. Flow control mechanisms help balance data flow.

5. Error Detection and Correction: It performs error detection and correction, especially in connection-oriented protocols like TCP. This ensures data integrity during transmission.

6. Reliability: Transport Layer protocols, such as TCP, provide a reliable data delivery service by guaranteeing that data is delivered correctly and in the correct order. This is crucial for applications like web browsing and file transfers.

7. Multiplexing and Demultiplexing: Multiple applications or services on the same device can use the Transport Layer simultaneously. The Transport Layer uses port numbers to distinguish between different services (multiplexing) and directs incoming data to the appropriate service (demultiplexing).

8. Quality of Service (QoS): The Transport Layer can prioritize certain types of traffic over others to ensure that critical data, such as voice or video streams, is delivered with minimal delay and jitter.

9. Session Management: For some transport protocols (e.g., SCTP - Stream Control Transmission Protocol), the Transport Layer provides session management capabilities, allowing multiple streams of data to be managed within a single connection.

10. Virtual Circuits: In connection-oriented protocols, the Transport Layer establishes virtual circuits that provide logical connections between devices. This is important for maintaining session state and data sequencing.

11. Port Addressing: Transport Layer protocols use port numbers to address specific services or processes on a device. Ports help direct incoming data to the appropriate application or service.

12. Congestion Control: Transport Layer protocols, especially TCP, implement congestion control mechanisms to manage network congestion and prevent it from negatively impacting data transmission.

The two most common transport layer protocols in use are TCP (Transmission Control Protocol), which provides a reliable and connection-oriented service, and UDP (User Datagram Protocol), which provides a simpler, connectionless service with minimal overhead. The choice between these protocols depends on the specific requirements of the application or service being used.

Transport Layer

The transport layer is a 4th layer from the top.

1. The main role of the transport layer is to provide the communication services directly to the application processes running on different hosts.
2. 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.
3. The transport layer protocols are implemented in the end systems but not in the network routers.
4. 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.
5. All transport layer protocols provide multiplexing/demultiplexing service. It also provides other services such as reliable data transfer, bandwidth guarantees, and delay guarantees.
6. 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.

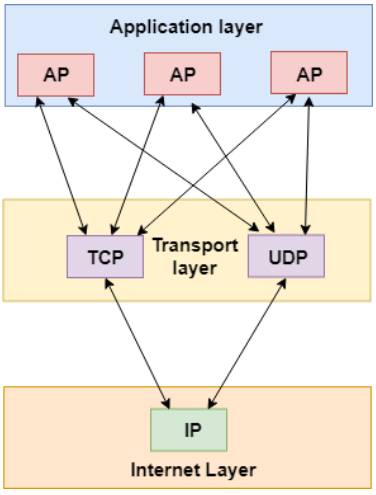

Fig 4.0 Layer Details

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:

1. End-to-end delivery
2. Addressing

3. Reliable delivery
4. Flow control
5. Multiplexing

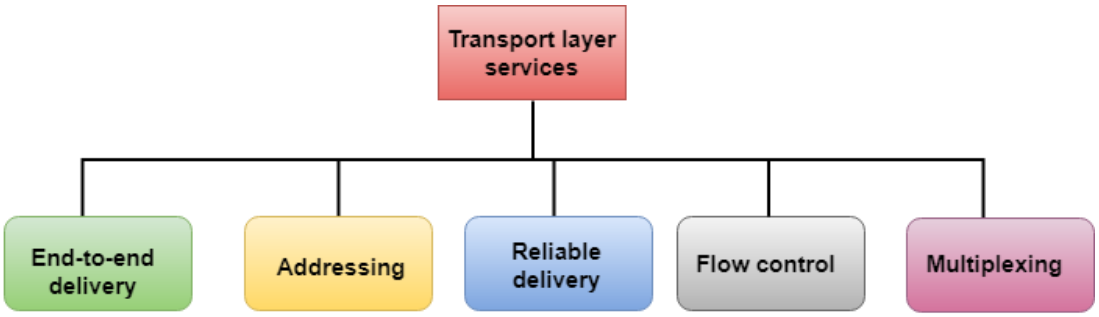

Fig 4.1 Service Provided By Transport layer

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:

1. Error control
2. Sequence control
3. Loss control
4. Duplication control

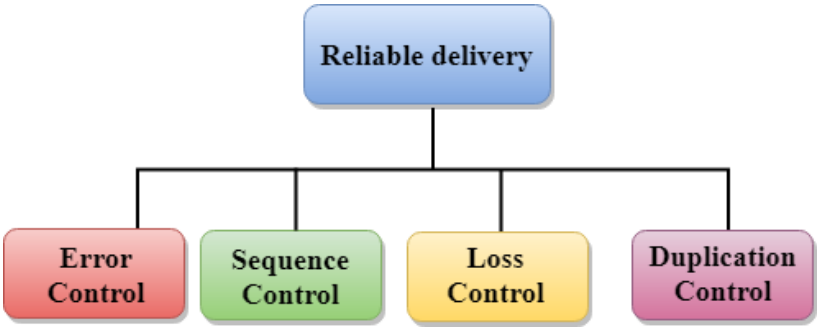

Fig 4.3 Components essential for reliable for delivery of packet

Error Control

1. 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.
2. 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.
3. 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.

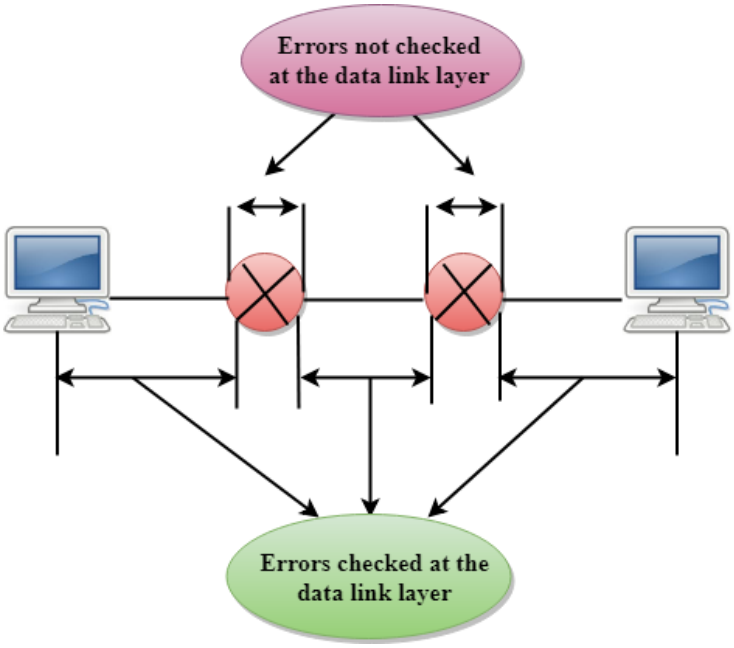

Fig 4.4 error control

Sequence Control

1. The second aspect of the reliability is sequence control which is implemented at the transport layer.
2. 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 receiver?s 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.

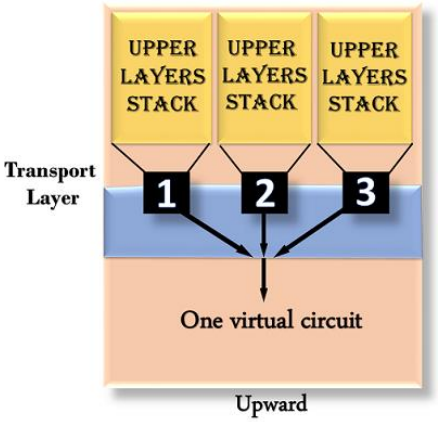

Fig 4.5 role of transport layer in Virtual circuit

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.

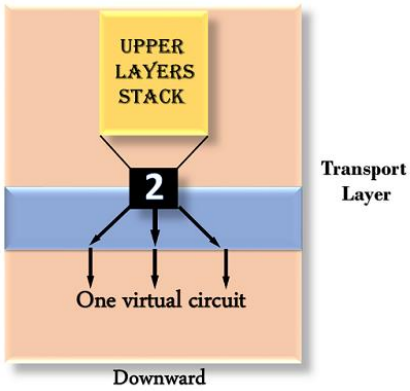

Fig 4.6 role of transport layer in Virtual circuit

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.

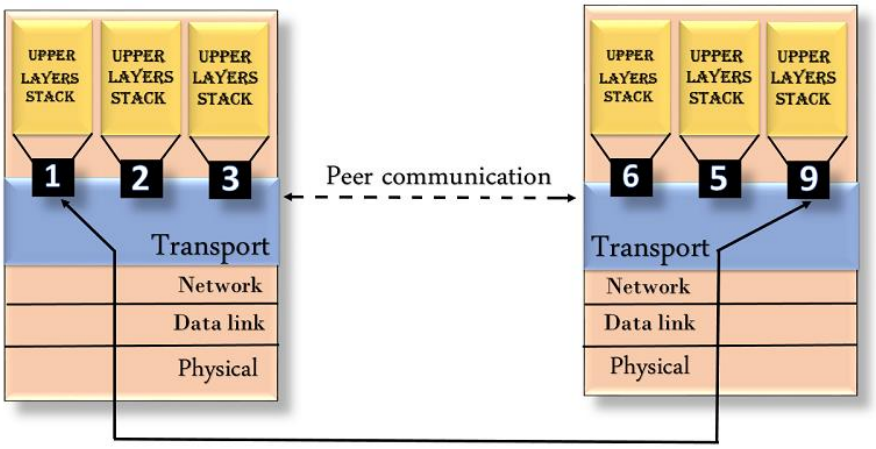

Fig 4.7 upper layer protocol

Transport Layer protocols

1. The transport layer is represented by two protocols: TCP and UDP.
2. The IP protocol in the network layer delivers a datagram from a source host to the destination host.
3. 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.
4. 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.
5. Each port is defined by a positive integer address, and it is of 16 bits.

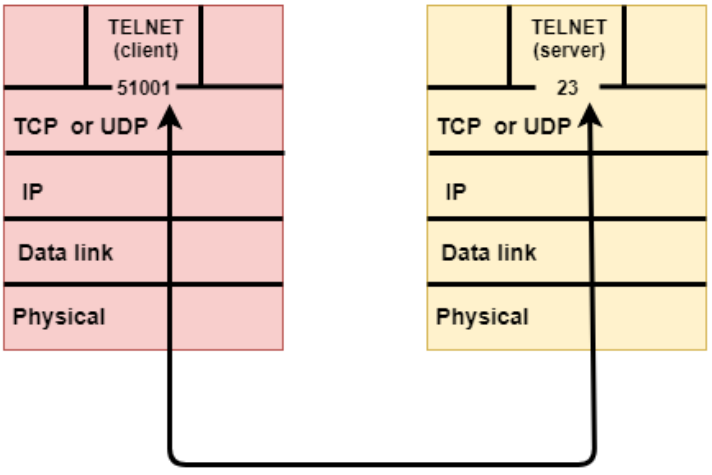

Fig 4.8 Transport Layer protocols

UDP

1. UDP stands for **User Datagram Protocol**.
2. UDP is a simple protocol and it provides nonsequenced transport functionality.
3. UDP is a connectionless protocol.
4. This type of protocol is used when reliability and security are less important than speed and size.
5. 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.
6. 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:

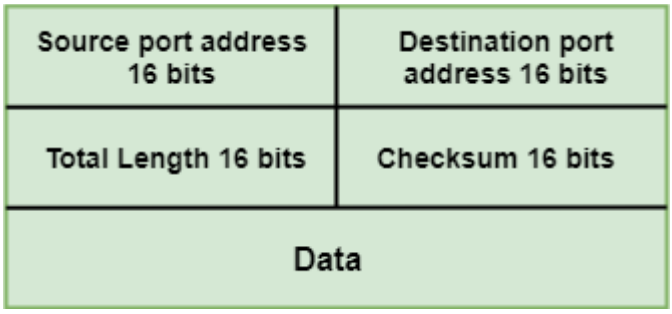

Fig 4.9 UDP Datagram

Where,

1. **Source port address:** It defines the address of the application process that has delivered a message. The source port address is of 16 bits address.
2. **Destination port address:** It defines the address of the application process that will receive the message. The destination port address is of a 16-bit address.
3. **Total length:** It defines the total length of the user datagram in bytes. It is a 16-bit field.
4. **Checksum:** The checksum is a 16-bit field which is used in error detection.

Disadvantages of UDP protocol

1. UDP provides basic functions needed for the end-to-end delivery of a transmission.
2. It does not provide any sequencing or reordering functions and does not specify the damaged packet when reporting an error.
3. UDP can discover that an error has occurred, but it does not specify which packet has been lost as it does not contain an ID or sequencing number of a particular data segment.

TCP

1. TCP stands for Transmission Control Protocol.
2. It provides full transport layer services to applications.
3. 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

1. **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.
2. **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.
3. **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.
4. **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.
5. **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.
6. **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.

TCP Segment Format

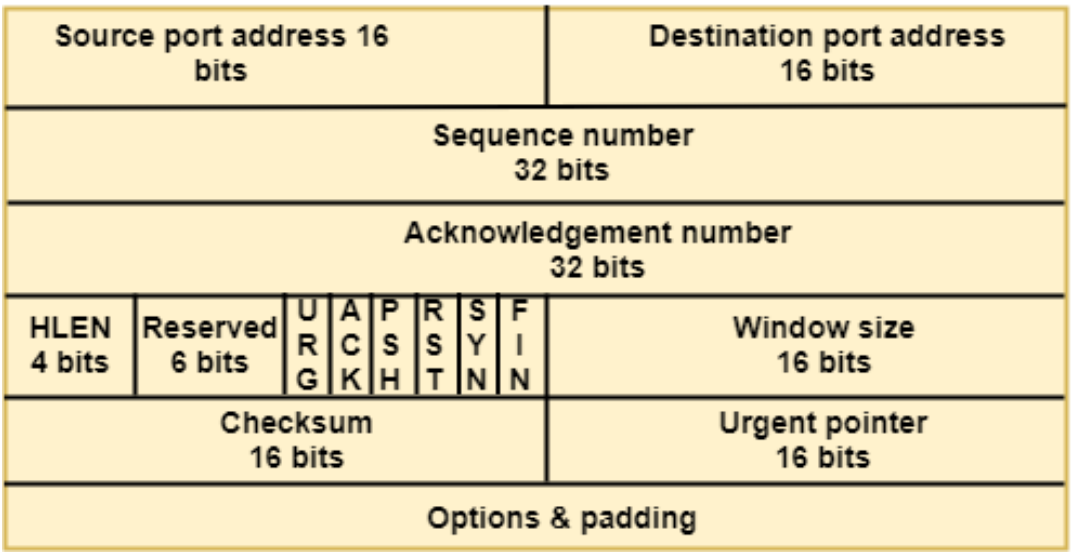

Fig 4.10 TCP Segment format

Where,

1. **Source port address:** It is used to define the address of the application program in a source computer. It is a 16-bit field.
2. **Destination port address:** It is used to define the address of the application program in a destination computer. It is a 16-bit field.
3. **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.
4. **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.
5. **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.
6. **Reserved:** It is a six-bit field which is reserved for future use.
7. **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:

1. **URG:** The URG field indicates that the data in a segment is urgent.
2. **ACK:** When ACK field is set, then it validates the acknowledgement number.
3. **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.
4. **RST:** The reset bit is used to reset the TCP connection when there is any confusion occurs in the sequence numbers.
5. **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.
6. **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 that 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.

https://www.youtube.com/watch?v=jE_FcgpQ7Co

TCP Three-Way Handshake

Three-Way HandShake or a TCP 3-way handshake is a process which is used in a TCP/IP network to make a connection between the server and client. It is a three-step process that requires both the client and server to exchange synchronization and acknowledgment packets before the real data communication process starts.

Three-way handshake process is designed in such a way that both ends help you to initiate, negotiate, and separate TCP socket connections at the same time. It allows you to transfer multiple TCP socket connections in both directions at the same time.

TCP message types

Message	Description
Syn	Used to initiate and establish a connection. It also helps you to synchronize sequence numbers between devices.
ACK	Helps to confirm to the other side that it has received the SYN.
SYN-ACK	SYN message from local device and ACK of the earlier packet.
FIN	Used to terminate a connection.

TCP Three-Way Handshake Process

TCP traffic begins with a three-way handshake. In this TCP handshake process, a client needs to initiate the conversation by requesting a communication session with the Server:

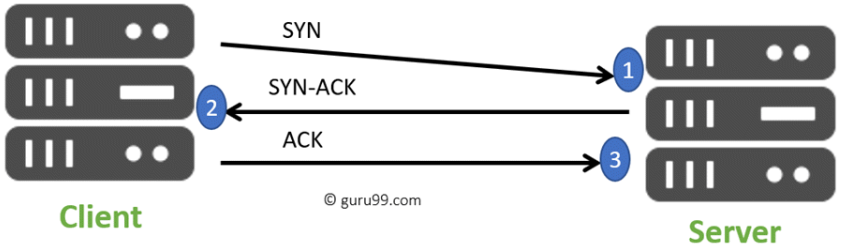

Fig 4.11 Three way hand shake

Step 1: In the first step, the client establishes a connection with a server. It sends a segment with SYN and informs the server about the client should start communication, and with what should be its sequence number.

Step 2: In this step server responds to the client request with SYN-ACK signal set. ACK helps you to signify the response of segment that is received and SYN signifies what sequence number it should able to start with the segments.

Step 3: In this final step, the client acknowledges the response of the Server, and they both create a stable connection will begin the actual data transfer process.

Real-world Example

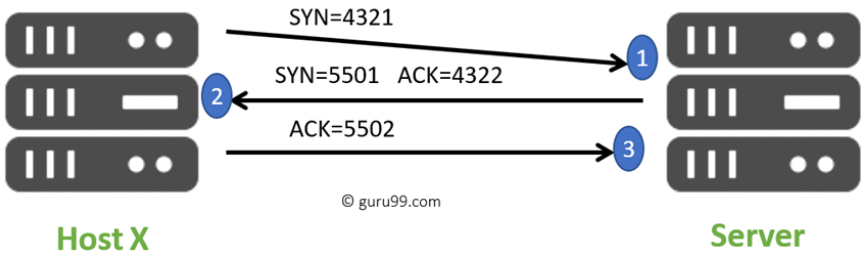

Fig 4.11 Three way hand shake (Real world example)

Here is a simple example of the three-way handshake process that is consists of three steps:

1. Host X begins the connection by sending the TCP SYN packet to its host destination. The packets contain a random sequence number (For example, 4321) that indicates the beginning of the sequence numbers for data that the Host X should transmit.
2. After that, the Server will receive the packet, and it responds with its sequence number. It's response also includes the acknowledgment number, that is Host X's sequence number incremented with 1 (Here, it is 4322).
3. Host X responds to the Server by sending the acknowledgment number that is mostly server's sequence number that is incremented by 1.

After the data transmission process is over, TCP automatically terminates the connection between two separate endpoints.

TCP Service Model in Computer Network

In TCP service, the sender and receiver needs to create endpoints called sockets. Each socket has an address which is made up of two parts.

- ✓ An IP address of the host.
- ✓ A port number that is 16 bit local to host (source or destination)

Both are collectively called socket addresses. A port is the TCP name for TSAP (Transport Service Access Point). It is essential to create a link between the sockets of the sender & receiver. Connections are used as identifiers at both ends. It can use the same socket for greater than one connection at a time.

We explain some of the sockets calls with their respectful meaning in the following table. It also should know that TCP does not support multicasting & broadcasting.

Different Socket Calls

The table given below explains the socket calls in TCP service model −

Socket Calls	Meaning
Socket	It creates a new socket call connection.
Bind	It gives a local location to a socket.
Listen	In response to making a new connection, and it shows a willingness to accept new connections.
Accept	It can block the caller unit when a connection attempt arrives.
Send	It can send data over the connection.
Receive	It can receive data over the connection.
Connect	It can attempt to make a connection.
Close	It can release the connection.

PUSH Flag

The message boundaries are not maintained end to end. When an application reaches information to TCP, and an application needs that data to be shared directly, it sets the PUSH Flag, forcing the TCP to send information without any interruption.

Buffering

But when TCP doesn't send the data received from the above application layer, it is collected for some time before sending. That is called Buffering.

Urgent Data

The sending application puts some regulation data in the data flow and provides it to TCP, and then an urgent Flag is set. Therefore, TCP will break buffering information and send it directly.

On reaching the urgent data at destination, the receiving application is disrupted, and the urgent data flow is displayed to it. The last end of urgent data is always indicated for the application to understand that no further is urgent data.

Congestion Control Algorithm.

Congestion causes choking of the communication medium. When too many packets are displayed in a method of the subnet, the subnet's performance degrades. Hence, a network's communication channel is called congested if packets are traversing the path and experience delays mainly over the path's propagation delay.

There is two congestion control algorithm which is as follows:

Leaky Bucket

The leaky bucket algorithm discovers its use in the context of network traffic shaping or rate-limiting. The algorithm allows controlling the rate at which a record is injected into a network and managing burstiness in the data rate.

A leaky bucket execution and a token bucket execution are predominantly used for traffic shaping algorithms. This algorithm is used to control the rate at which traffic is sent to the network and shape the burst traffic to a steady traffic stream.

The figure shows the leaky bucket algorithm.

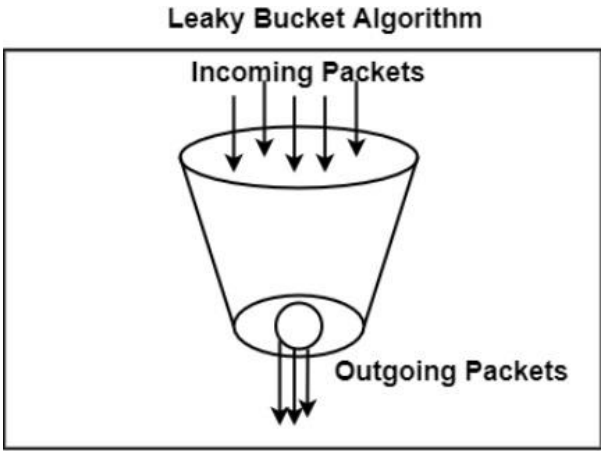

Fig 4.12 Leaky Bucket

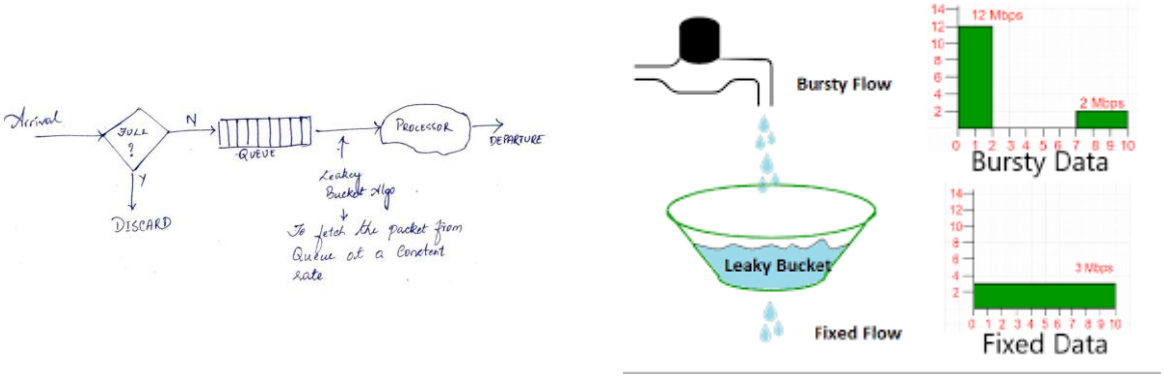

Fig 4.13 leaky bucket illustration

In this algorithm, a bucket with a volume of, say, b bytes and a hole in the Notes bottom is considered. If the bucket is null, it means b bytes are available as storage. A packet with a size smaller than b bytes arrives at the bucket and will forward it. If the packet's size increases by more than b bytes, it will either be discarded or queued. It is also considered that the bucket leaks through the hole in its bottom at a constant rate of r bytes per second.

The outflow is considered constant when there is any packet in the bucket and zero when it is empty. This defines that if data flows into the bucket faster than data flows out through the hole, the bucket overflows.

The disadvantages compared with the leaky-bucket algorithm are the inefficient use of available network resources. The leak rate is a fixed parameter. In the case of the traffic, volume is deficient, the large area of network resources such as bandwidth is not being used effectively. The leaky-bucket algorithm does not allow individual flows to burst up to port speed to effectively consume network resources when there would not be resource contention in the network.

Token Bucket Algorithm

The leaky bucket algorithm has a rigid output design at the average rate independent of the bursty traffic. In some applications, when large bursts arrive, the output is allowed to speed up. This calls for a more flexible algorithm, preferably one that never loses information. Therefore, a token bucket algorithm finds its uses in network traffic shaping or rate-limiting.

It is a control algorithm that indicates when traffic should be sent. This order comes based on the display of tokens in the bucket. The bucket contains tokens. Each of the tokens defines a packet of predetermined size. Tokens in the bucket are deleted for the ability to share a packet.

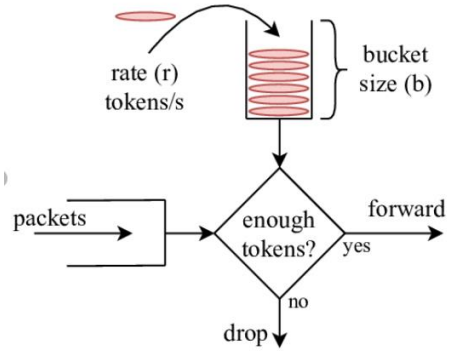

Fig 4.14 Token flow chart

When tokens are shown, a flow to transmit traffic appears in the display of tokens. No token means no flow sends its packets. Hence, a flow transfers traffic up to its peak burst rate in good tokens in the bucket.

Thus, the token bucket algorithm adds a token to the bucket each 1 / r seconds. The volume of the bucket is b tokens. When a token appears, and the bucket is complete, the token is discarded. If a packet of n bytes appears and n tokens are deleted from the bucket, the packet is forwarded to the network.

When a packet of n bytes appears but fewer than n tokens are available. No tokens are removed from the bucket in such a case, and the packet is considered non-conformant. The non-conformant packets can either be dropped or queued for subsequent transmission when sufficient tokens have accumulated in the bucket.

They can also be transmitted but marked as being non-conformant. The possibility is that they may be dropped subsequently if the network is overloaded.