UNIT IV
The Transport Layer

Transport Layer

- The Transport Layer is the fourth layer of the OSI model (above the Network Layer and below the Application Layer).
- Its main job is to provide end-to-end communication between applications running on different hosts.
- It ensures that data is delivered reliably, in order, and without errors from one process to another (process-to-process communication).

The Transport Layer Services

1. End-to-end Connection between Hosts
2. Flow Control
3. Multiplexing and Demultiplexing
4. Connection Establishment
5. Connection Termination
6. Reliable Data Delivery

1. End-to-end Connection between Hosts

- The transport layer enables direct communication between two end systems.
- It uses two main protocols:
    - TCP (Transmission Control Protocol)
        - Connection-oriented and reliable
        - Uses the three-way handshake to establish a connection
        - Guarantees ordered delivery, error checking, and retransmission if packets are lost
        - Used for web browsing, emails, file transfers (HTTP, SMTP, FTP)
    - UDP (User Datagram Protocol)
        - Connectionless and unreliable
        - Does not guarantee delivery or order
        - Best for real-time applications like video conferencing, VoIP, or live streaming
        - Often used for multicasting

2. **Flow Control**

- The transport layer provides a flow control mechanism between the adjacent layers of the TCP/IP model. TCP also prevents data loss due to a fast sender and slow receiver by imposing some flow control techniques. It uses the method of sliding window protocol which is accomplished by the receiver by sending a window back to the sender informing the size of data it can receive.

3. Multiplexing and Demultiplexing

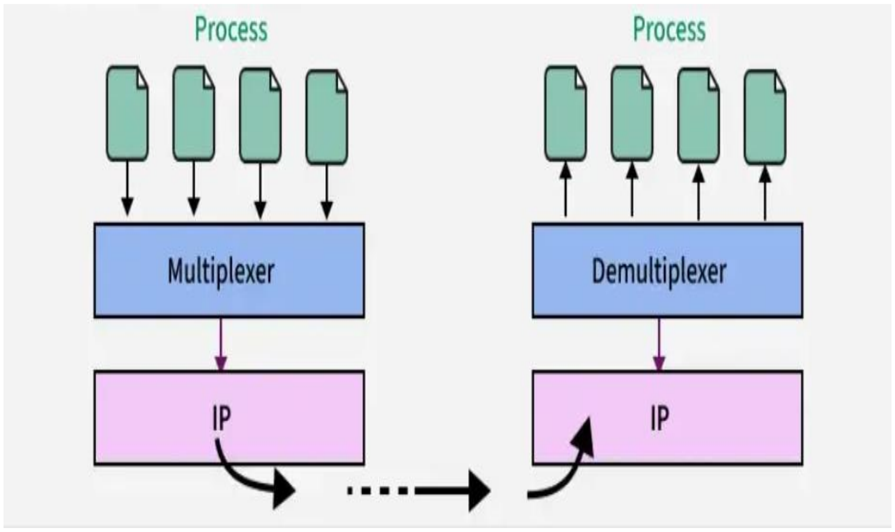
- **Multiplexing (many → one):**
  - Multiple applications (like browser, email, video call) send data simultaneously.
  - The transport layer adds port numbers to identify each application's data.
  - Combines all data into one stream for sending over the network.
- **Demultiplexing (one → many):**
  - At the receiver side, the transport layer reads the destination port number.
  - It delivers the received data to the correct application.

    Example:

    Port 80 → Web server

    Port 25 → Email server

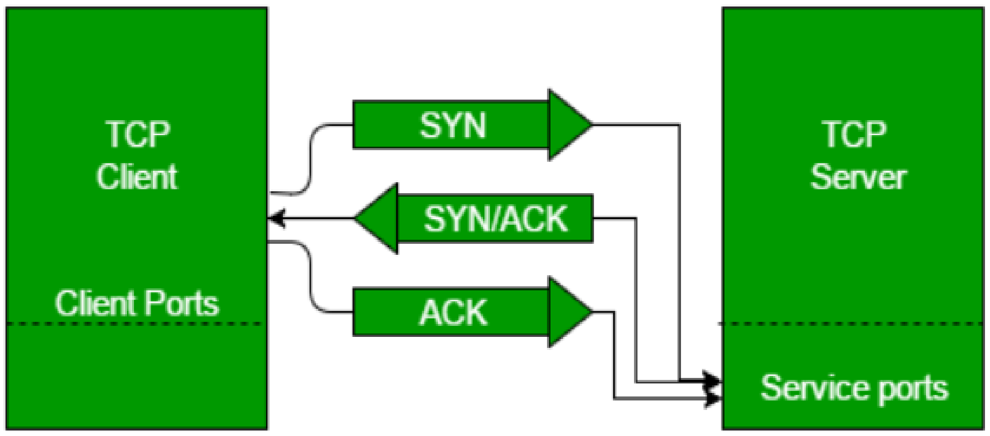
    Port 53 → DNS service



4. Connection Establishment

- When two devices in a network wants to establish a connection using TCP, it is done through 3-way handshake process.
  1. The first computer connects to the second computer by sending a SYN packet to a specified port number.
  2. If the second computer is listening, it will respond with a SYN/ACK.
  3. When the first computer receives the SYN/ACK, it replies with an ACK packet.

  After this, the two devices can communicate normally.



5. Connection Termination

- When communication is done, the TCP connection must be closed
- In a TCP connection, we have two types of termination mechanisms:
  1. Graceful termination:
     Both sides close the connection properly using a four-step process (each side sends a FIN and receives an ACK).
  2. Abrupt termination:
     One side immediately closes the connection — this might cause data loss if packets are still in transit.

6. Reliable Data Delivery

- The Transport Layer checks for errors in the received data using checksums.
- Uses ACK (Acknowledgment) and NACK (Negative Acknowledgment) to confirm data delivery.

- If an error or loss is detected:
    - The sender retransmits the data.
    - Ensures complete and accurate delivery of all data segments.
- Maintains the sequence order of packets so data is reassembled correctly at the destination.

User Datagram Protocol (UDP)

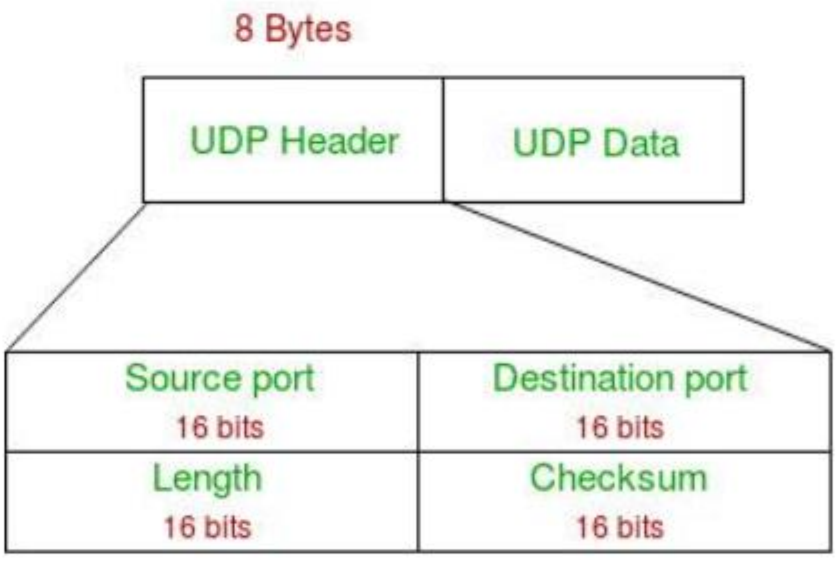
- User Datagram Protocol (UDP) is a Transport Layer protocol. UDP is a part of the Internet Protocol suite, referred to as UDP/IP suite. Unlike TCP, it is an unreliable and connectionless protocol. So, there is no need to establish a connection before data transfer. The UDP helps to establish low-latency and loss-tolerating connections over the network. The UDP enables process-to-process communication

UDP Header

- UDP header is an 8-byte fixed and simple header, while for TCP it may vary from 20 bytes to 60 bytes. The first 8 Bytes contain all necessary header information and the remaining part consists of data. UDP port number fields are each 16 bits long, therefore the range for port numbers is defined from 0 to 65535; port number 0 is reserved. Port numbers help to distinguish different user requests or processes.

Features

- UDP is used when acknowledgement of data does not hold any significance.
- UDP is good protocol for data flowing in one direction.
- UDP is not connection oriented.
- UDP does not provide congestion control mechanism.
- UDP does not guarantee ordered delivery of data.
- UDP is stateless.
- UDP is suitable protocol for streaming applications such as VoIP, multimedia streaming.



UDP Header

- **Source Port:** Source Port is a 2 Byte long field used to identify the port number of the source.
- **Destination Port:** It is a 2 Byte long field, used to identify the port of the destined packet.
- **Length:** Length is the length of UDP including the header and the data. It is a 16-bits field.
- **Checksum:** Checksum is 2 Bytes long field. It is the 16-bit one's complement of the one's complement sum of the UDP header, the pseudo-header of information from the IP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

UDP Operation (Services)

UDP uses concepts common to the transport layer. These concepts will be discussed here briefly, and then expanded in the next section on the TCP protocol.

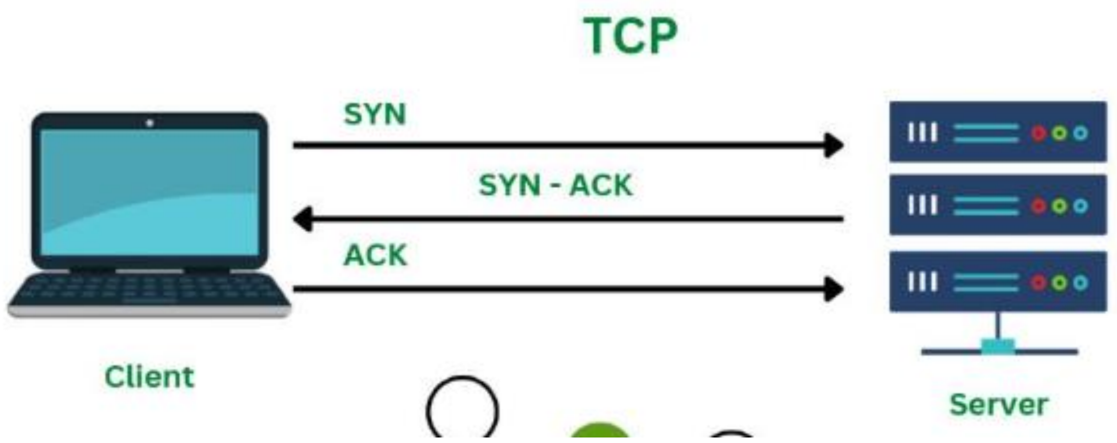
1. Connectionless Services
2. Flow and Error Control
3. Encapsulation and Decapsulation
4. Queuing

Transmission Control Protocol

Transmission Control Protocol (TCP) is a connection-oriented protocol for communications that helps in the exchange of messages between different devices over a network. It is one of the main protocols of the TCP/IP suite. In OSI model, it operates at the transport layer(Layer 4). It lies between the Application and Network Layers which are used in providing reliable delivery services. The Internet Protocol (IP), which establishes the technique for sending data packets between computers, works with TCP.

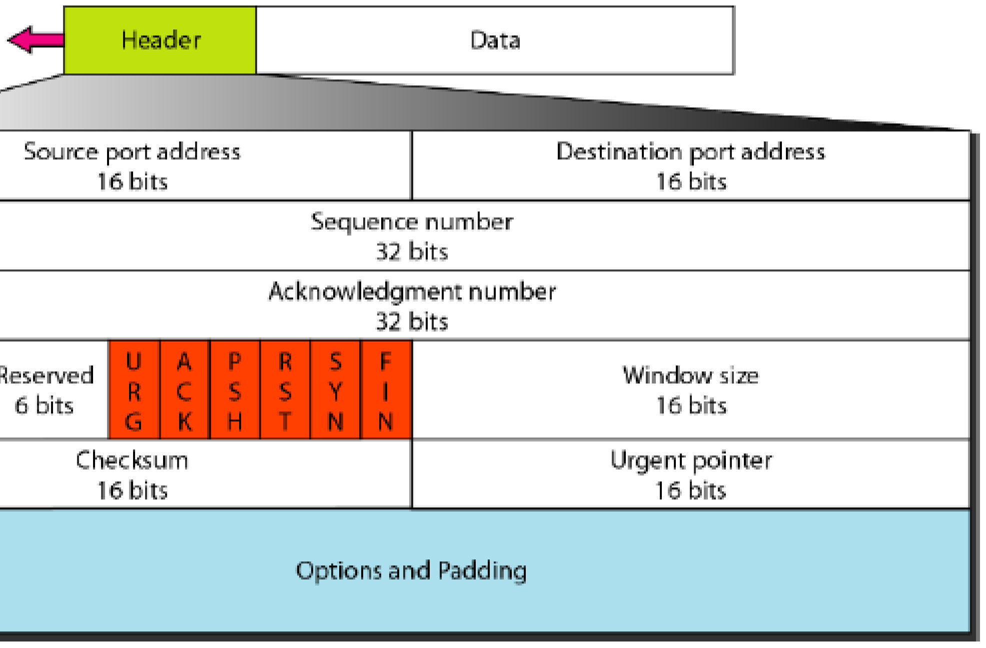
Features

- TCP is reliable protocol. That is, the receiver always sends either positive or negative acknowledgement about the data packet to the sender, so that the sender always has bright clue about whether the data packet is reached the destination or it needs to resend it.
- TCP ensures that the data reaches intended destination in the same order it was sent.
- TCP is connection oriented. TCP requires that connection between two remote points be established before sending actual data.
- TCP provides error-checking and recovery mechanism.
- TCP provides end-to-end communication.
- TCP provides flow control and quality of service.
- TCP operates in Client/Server point-to-nt mode.
- TCP provides full duplex server, i.e. it can perfopoirm roles of both receiver and sender.



TCP Header

The length of TCP header is minimum 20 bytes long and maximum 60 bytes.



- **Source Port (16-bits)** - It identifies source port of the application process on the sending device.
- **Destination Port (16-bits)** - It identifies destination port of the application process on the receiving device.
- **Sequence Number (32-bits)** - Sequence number of data bytes of a segment in a session.
- **Acknowledgement Number (32-bits)** - It indicates the next expected byte number from the sender.
- **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**- TCP control flags, which are special 1-bit indicators in the TCP header used to manage and control the connection state and data flow.
- **Urgent:** Indicates that some data is urgent.
- **ACK:** Indicates that the Acknowledgment Number field is valid.

- **PSH:** Requests that the receiver immediately deliver data to the application without waiting for the buffer to fill.
- **RST:** Resets the connection if there's an error or unexpected packet..
- **SYN:** Used to initiate a connection and synchronize sequence numbers.
- **FIN:** Indicates that the sender has finished sending data and wants to close the connection.
- **Window size:** This parameter provides the sender TCP's window size in bytes. Note that the length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes
- **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.
- **Urgent pointer**- This l6-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data.
- **Options-** There can be up to 40 bytes of optional information in the TCP header.

Stream Control Transmission Protocol (SCTP)

Stream Control Transmission Protocol (SCTP) is a network protocol that is connection-oriented and used for transmitting multiple streams of data simultaneously between any two endpoints that have established a connection in a computer network. SCTP is a transport layer of Internet Protocol (IP).

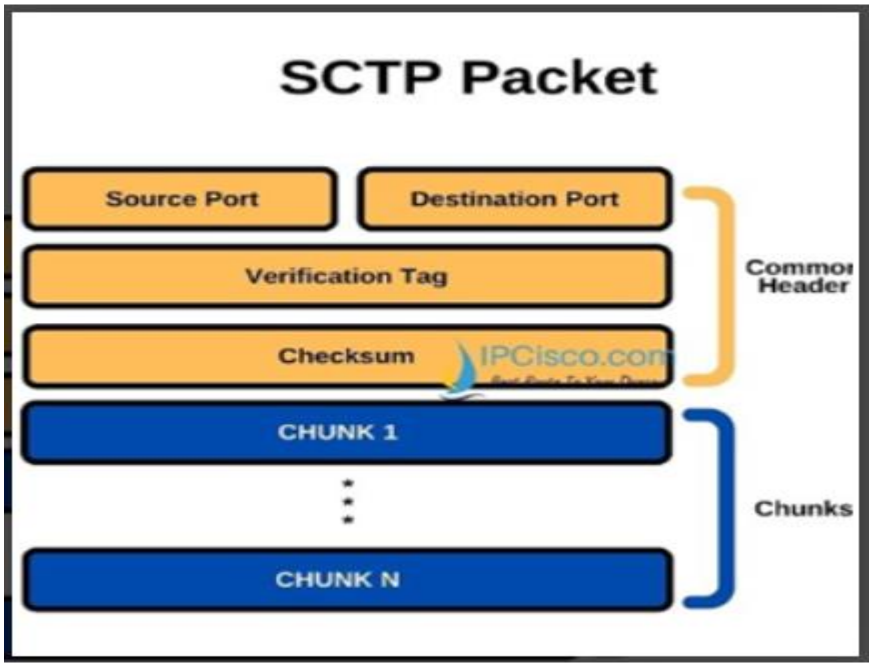
Features

1. Message-Oriented:
    - Unlike TCP (which is byte-oriented), SCTP transmits data in distinct messages.
    - Each message is delivered completely and in order, preventing data mixing.
2. Multi-Streaming:
    - A single SCTP connection (called an association) can have multiple streams.
    - Each stream delivers messages independently
      Example:
      If one stream is delayed, other streams can still continue transmission without waiting.

3. Multi-Homing:
    - An endpoint can have multiple IP addresses.
    - If one path fails, SCTP automatically switches to another — providing fault tolerance and path redundancy.
4. **Reliable Transmission:**
    - Like TCP, SCTP provides error detection, acknowledgment, and retransmission for reliability.
5. **Four-Way Handshake:**
    - SCTP uses a 4-step handshake to establish a connection, protecting against SYN flooding attacks (unlike TCP's 3-way handshake).
6. **Congestion and Flow Control:**
    - SCTP uses algorithms similar to TCP to avoid network congestion.

SCTP Packet

- SCTP protocol packet consist of two main parts Header and Payload. The Header is common but Payload have variable chunks.
- The Common SCTP header is 12 byte long and made of the 4 parts
- Port Number (Source): shows the sending port
- Port Number (Destination): shows the receiving port
- Verification tag: a 32 bit random value which differentiate the packets from the previous connection
- Checksum: a CRC32 algorithm for detection of error.



Advantages of SCTP

- SCTP provides a full duplex connection. It can send and receive the data simultaneously.
- SCTP protocol possesses the properties of both TCP and UDP protocol.
- SCTP protocol does not depend on the IP layer.
- SCTP is a secure protocol.

Disadvantages of SCTP

- More complex implementation than TCP/UDP
- Not as widely supported (some firewalls/routers block SCTP)
- Overhead due to additional features

Client server model

A client-server model is a networking computing system design that illustrates a relationship between two or more computers, where the client computers request and receive services or resources from a powerful centralized server computer. It describes a specific way devices access the information you store in servers. It also allows multiple clients to open applications or retrieve files from an individual server, which helps maintain consistency across all devices. Many companies across various industries use servers to store and access information, offering more processing power and providing more extensive storage space.

The client-server model in the transport layer, particularly with protocols like TCP and UDP, describes how applications establish and manage communication over a network.

The functions of client-server networks can include:

- Interacting with temporary and local storage
- Sending data requests to a server
- Interfacing between servers and users
- Completing database operations
- Connecting with other servers
- Processing user requests
- Writing files on servers
- Accessing server files
- Querying a database
- Creating interactive web applications

Components of client-server networks

The components of client-server networks can include:

Client:

A client is an application that initiates communication by requesting a service from a server.

It actively seeks to connect to a server's known address and port number.

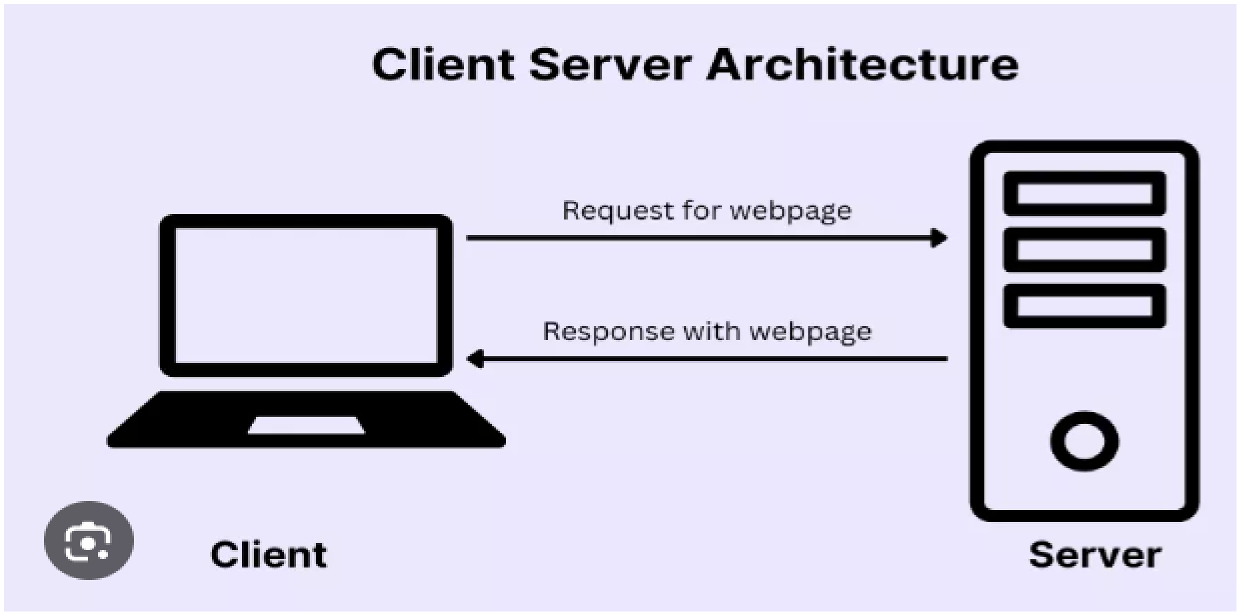
Examples include a web browser requesting a webpage or an email client fetching emails.

Server:

A server is an application that passively waits for incoming connection requests from clients.

It listens on a specific port number, awaiting requests for the services it provides.

Examples include a web server hosting websites or an email server storing and sending emails.



How it Works (TCP Example):

Server Setup:

- The server application starts and creates a "socket," which is an endpoint for communication.
- It then binds this socket to a specific local IP address and port number.

The server enters a "listening" state, waiting for client connection requests.

Client Connection:

- The client application also creates a socket.
- It then attempts to establish a connection to the server by sending a connection request (e.g., a TCP SYN packet) to the server's IP address and port number.

Connection Establishment (Three-Way Handshake):

- If the server is listening, it responds to the client's request (e.g., with a SYN-ACK packet).
- The client then acknowledges this response (e.g., with an ACK packet), completing the three-way handshake and establishing a reliable, connection-oriented session.

Data Exchange:

- Once the connection is established, the client can send requests to the server, and the server can send responses back to the client.
- TCP ensures reliable and ordered delivery of data segments.

Connection Termination:

- When the communication is complete, either the client or the server can initiate the termination of the connection (e.g., using a four-way handshake in TCP).

UDP (Connectionless):

- With UDP, the client-server model is simpler as there's no explicit connection establishment or termination.
- The client sends datagrams to the server's known address and port, and the server sends responses back to the client's address and port.
- UDP does not guarantee reliable delivery or order, making it suitable for applications where speed is prioritized over absolute reliability (e.g., streaming media, online gaming).