**TCP Protocol**

* TCP stands for Transmission Control Protocol.
* TCP is a **connection-oriented**, **reliable** protocol used for transmitting data over a network. It ensures that data reaches the destination correctly and in order.
* It is present in transport layer.

**Working of TCP:**

* The data transmission via TCP has three phases:

1. ***Connection Establishment:***

* The sender TCP or client sends a packet to the receiver TCP or server requesting a connection.
* The server then sends an acknowledgement to the client.
* The client further acknowledges the server it completes the process of connection establishment.
* Since a connection is established before data transmission, TCP is a connection-oriented protocol and the connection establishment process is called three-way TCP connection handshake.

1. ***Data Transfer:***

* During data transfer TCP offers some key features which includes:

1. **Error Free Data Transfer:** It is provided by using the field checksum. The sender calculates and enters a value in this field. At the receiving end, the receiver performs the same process and calculates the checksum value. If it does not match with the value present in the checksum field, the TCP segment is discarded and no acknowledgement is sent to the sender, Since the sending side does not receive an ACK of the discarded packet, it is retransmitted. This time if the checksum value calculated at the receiver matches the one present in the TCP segment, it means the TCP segment is correct and is accepted. In this way TCP helps in error free data transfer.
2. **Ordered Data Transfer:** TCP adds a sequence number in the TCP segments. At the receiving end, the TCP module uses the sequence numbers to construct the application message in the correct order.
3. **Retransmission of lost segments:** For reliable data transfer, the receiver TCP sends an acknowledgement to the sender TCP for each TCP segment it receives, if ACK is not received, the TCP segment is retransmitted. Therefore, if a TCP segment is lost, the receiver will not send an ACK message to the sender. As a result, the sender TCP sends the lost segment again.
4. **Discarding duplicate segments:** The TCP client retransmits segments that it determines to be lost. However, the receiver TCP may receives segments that were considered to be lost. As a result, the receiving end will have two or more copies of the same segment. In such cases the unique sequence numbers in the TCP header of every segment helps to determine the duplicate segments, which are then discarded.
5. **Congestion Throttling and Flow control:** The goal to TCP is to send segments to the receiving end as fast as possible without losing them. When TCP first sends the segment it sets a timer. If the segments are acknowledged before the timer expires, TCP increases the transmission speed until the segments begin to become unacknowledged. Since the ACK for some segments is not received within the time, the sending TCP module retransmits the segments. When a significant number of segments have to be retransmitted, TCP slows down the data transfer rate. In this way, TCP handles congestion throttling or flow control.
6. ***Connection Termination:***

* When an endpoint wishes to stop its connection, it sends a finished message to the other endpoint. The other end point acknowledges the message. Both ends do this two-phase handshake process. Therefore, the connection termination follows a four-way handshake process.

**UDP Protocol**

* UDP stands for User Datagram Protocol.
* UDP is a connectionless, unreliable transport layer protocol used for transmitting data quickly without the overhead of connection establishment.
* It is present in the transport layer.
* The data transmission via UDP occurs in a single phase, as there is no connection setup or teardown.

**Working of UDP:**

***1. No Connection Establishment:***

* UDP is a connectionless protocol, meaning it does **not establish any connection** between the sender and receiver before sending data.
* The sender can directly start transmitting packets (datagrams) without performing a handshake.
* This makes UDP faster and more efficient for applications where speed is more important than reliability.
* Since there's no session management, it is suitable for **one-to-many** and **broadcast** communication scenarios.

***2. Data Transfer:***

* During data transfer, UDP offers a minimalistic but fast mechanism, with the following characteristics:

1. **No Error Correction**:

* UDP uses a checksum field to detect errors in the header and data.
* If an error is found, the UDP datagram is simply discarded.
* No acknowledgement is sent and no retransmission occurs, making it unreliable but fast.

1. **No Ordering of Data**:

* Each UDP packet is independent and may take a different path to the receiver.
* The receiving application must handle ordering, if necessary.
* UDP does not assign sequence numbers, so packets may arrive out of order or not at all.

1. **No Retransmission of Lost Data**:

* UDP does not track which packets were received or lost.
* If a packet is lost during transmission, it is not retransmitted.
* Applications using UDP must implement their own mechanism if reliability is needed.

1. **No Duplicate Detection:**

* Since UDP lacks sequence numbers and acknowledgments, it does not detect duplicate packets.
* If the receiver gets two identical packets, it will process both unless the application itself handles deduplication.

1. **No Flow Control or Congestion Handling**:

* UDP does not monitor network conditions or receiver status.
* It sends data as fast as the application generates it.
* There is no mechanism to slow down or pause data transmission if the network is congested or the receiver is overwhelmed.

***3. No Connection Termination:***

* Since UDP doesn’t create a connection, there is no formal termination process.
* Once data transmission is complete, the sender can simply stop sending packets.
* There is no exchange of FIN/ACK messages like in TCP.

**Hypertext Transfer Protocol**

* HTTP or Hypertext Transfer Protocol is the application layer protocol that is implemented in the client program and the server program. An example of a client program is a Web Browser like Google Chrome and an example of a server program is a web server like Apache Web Server.
* When we open a web page on a web browser, we see that it contains some text, images, audio. These are called objects. Web pages are written in HTML language. The HTML file contains text and links to images. So, when we open a web page, initially the web browser requests the HTML file. It interprets the HTML file and then requests the linked images using their URLs. At last, the web page is displayed on the computer screen.
* The object or webpage is transferred using the application layer protocol-HTTP.
* HTTP uses TCP as its underlying transport protocol. Since TCP is a connection-oriented protocol the web browser first set up a connection with the web server. Once the connection is set up, the web browser and the web server access TCP through their sockets to share messages.
* The web browser sends a HTTP request message and the web server sends a HTTP response message.

**Working of HTTP:**

1. **Open Web Browser**: First, you open your web browser and type a website URL (e.g., www.example.com).
2. **DNS Lookup**: The browser asks a Domain Name Server (DNS) server to find out the IP address of the URL. DNS is basically a telephone book which contains the IP addresses of all websites.
3. **Send HTTP Request**: Once the browser has the website’s IP address, it sends an HTTP **request** to the server. The request asks the server for the resources needed to display the page (like text, images, and videos).
4. **Server Response**: The server processes your request and sends back an HTTP **response**. This response contains the requested resources (like HTML, CSS, JavaScript) needed to load the page.
5. **Displays the Web Page**: The browser receives the data from the server and displays the webpage on your screen.

**Hypertext Transfer Protocol Secure**

* Hypertext transfer protocol secure (HTTPS) is the secure version of HTTP, which is used to send data between a web browser and a website. HTTPS is encrypted in order to increase security of data transfer.
* It operates over port 443.
* HTTPS encrypts data using SSL (Secure Sockets Layer) or TLS (Transport Layer Security) protocols. This protocol secures communications by an asymmetric key cyptography.

1. Public Key: The web server public key is available to anybody who wants to interact with the web server. The web browser encrypts the data using the public key and information encrypted by the public key can only be decrypted by the private key.
2. Private Key: The private key is available only to the owner of the website. This key is present on the web server and is used to decrypt information encrypted by the public key.

**Working of HTTPS:**

1. **Open Secure Web Page**: Open your web browser and type a URL that starts with https:// (e.g., https://www.example.com).
2. **DNS Lookup**: The browser asks a Domain Name System (DNS) server to find out the IP address of the URL. DNS is basically a telephone book which contains the IP addresses of all websites.
3. **TLS Handshake (Secure Layer Initialization)**: Before sending any HTTP request, the browser and the server perform a TLS handshake to set up a secure, encrypted channel.
4. **Establish TCP Connection**: The browser establishes a TCP connection sing the IP address returned by DNS.
5. The browser receives the server’s digital certificate (SSL certificate) to verify its identity. If the certificate is valid (issued by a trusted Certificate Authority), the browser and server agree on encryption keys. They use asymmetric encryption initially to securely exchange a session key, which will be used for faster symmetric encryption during the session.
6. **Send Encrypted HTTP Request**:

* After the secure connection is established, the browser sends the HTTP request, but now it is encrypted using the TLS session key.

1. **Server Sends Encrypted HTTP Response**:

* The server processes the request and sends back the HTTP response, also encrypted.
* This response may contain the HTML file, images, CSS, JavaScript, etc.

1. **Displays the Secure Web Page**: The browser decrypts the data using the session key, processes the HTML, and renders the secure web page for the user.

**ICMP**

* ICMP or Internet Control Message Protocol is a network layer protocol.
* ICMP is designed to overcome the following two problems with the IP protocol:

1) No error reporting.

2) Lacks a mechanism for queries.

* ICMP is responsible for error reporting and query management.
* ICMP is used for error reporting if two devices connect over the internet and some error occurs. So, the router sends an ICMP error message to the source informing about the error.
* ICMP messages are not directly passed to the data link layer. Instead, the messages are first encapsulated inside IP datagrams before going to the lower layer.
* ICMP messages are of two types:

1)Error Reporting Messages: Reports problem that router or a host may encounter while processing of IP packet.

2)Query Messages: Fetch specific information from a router or a host.

* ICMP message consists of:

1. **Type:** Specifies the message type.
2. **Code:** Reason for message type.
3. **Checksum:** It is used for error checking.
4. **Rest of the header:** Rest of the header is specified for each message type.
5. **Data Section:** It carries the information for finding the original packet that has error in case of error reporting message and it carries extra information in case of query message.

* **Example: Destination Unreachable:** When a router or destination host cannot deliver an IP packet, it sends an ICMP Type 3 message back to the sender. The code indicates the exact reason.