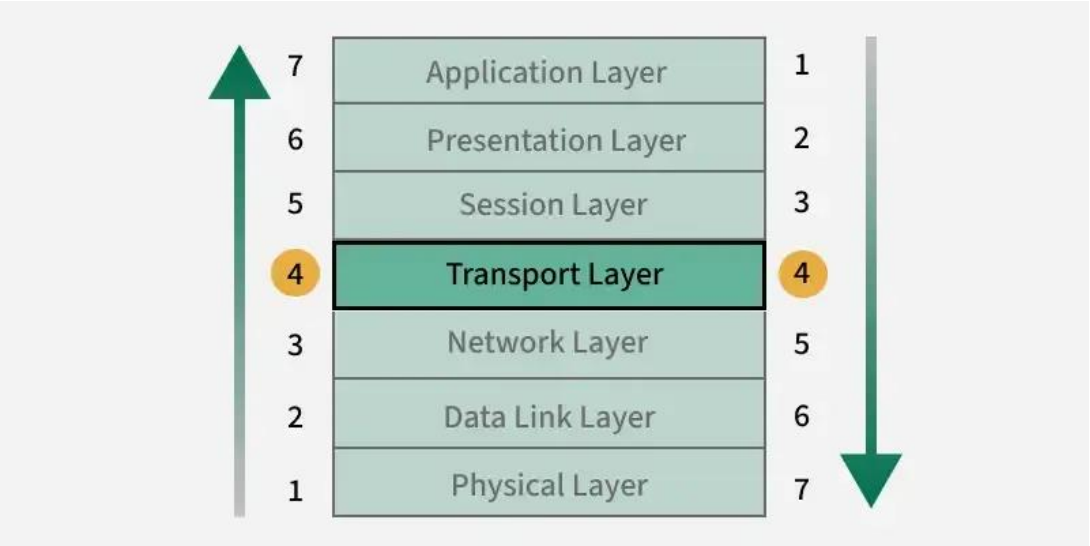
Transport Layer

The Transport Layer ensures end-to-end communication between applications on different hosts. It sits between the Network Layer (which delivers packets to the right machine) and the Session Layer (which manages communication sessions). Its main job is to deliver data reliably, efficiently, and in the correct order.



Transport Layer Services

The Transport Layer is responsible for providing end-to-end communication between devices in a network. It ensures reliable data transfer by performing key functions such as:

Segmentation and reassembly – breaking large messages into smaller segments and reassembling them at the destination.

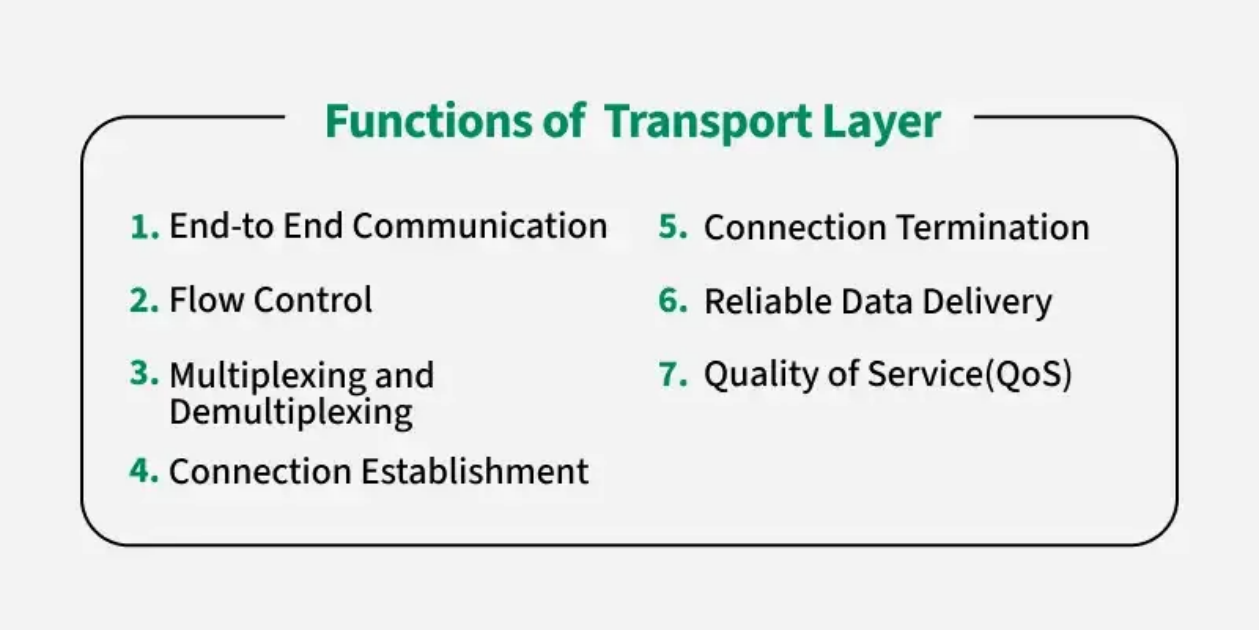
Flow control – preventing the sender from overwhelming the receiver by regulating the rate of data transmission.

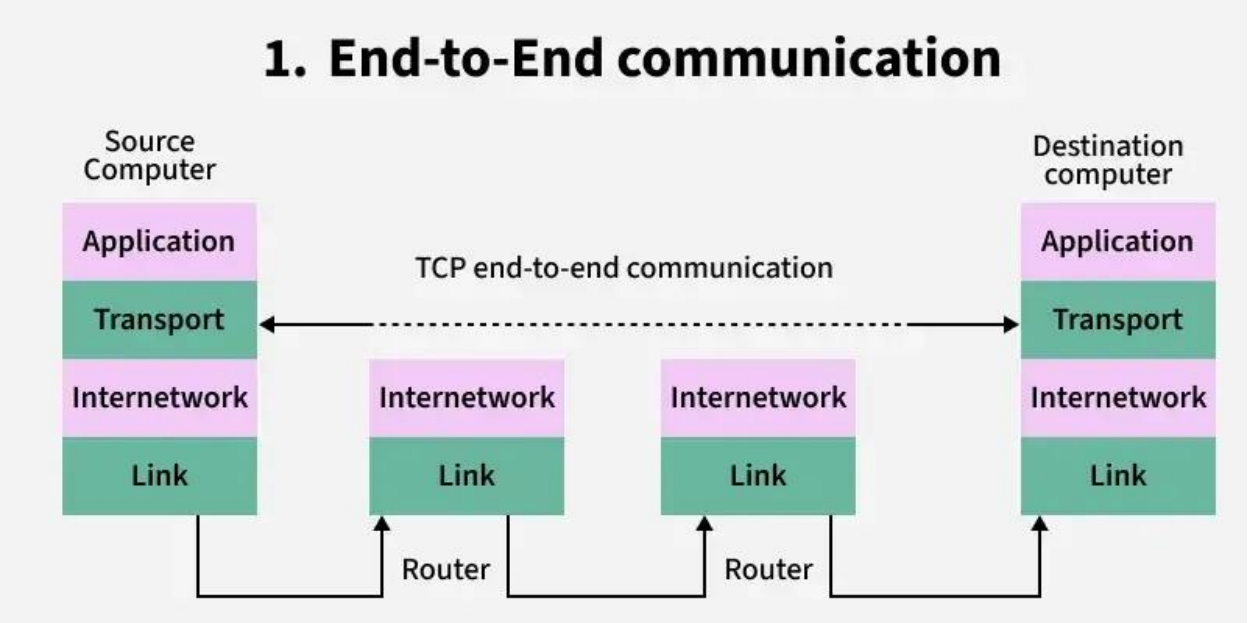
Error detection and correction – identifying errors in transmitted data using checksums and ensuring data integrity.

Retransmission of lost data – resending packets that are lost, damaged, or not acknowledged.

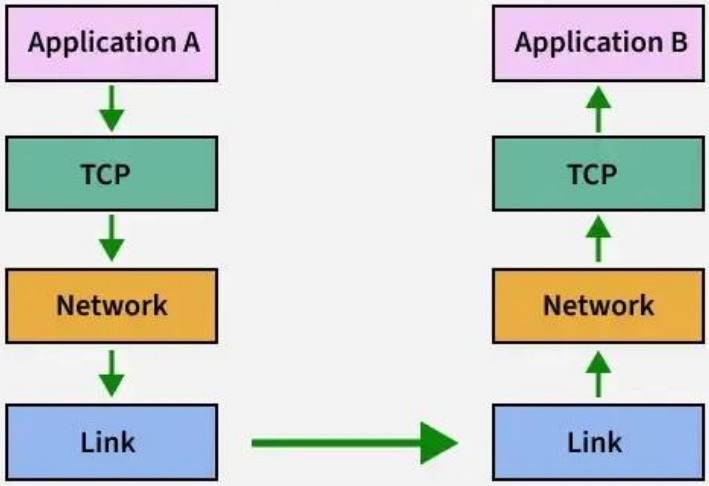
These features enable the transport layer to provide efficient, accurate, and reliable communication between applications running on different devices.

The primary functions of the Transport Layer are:



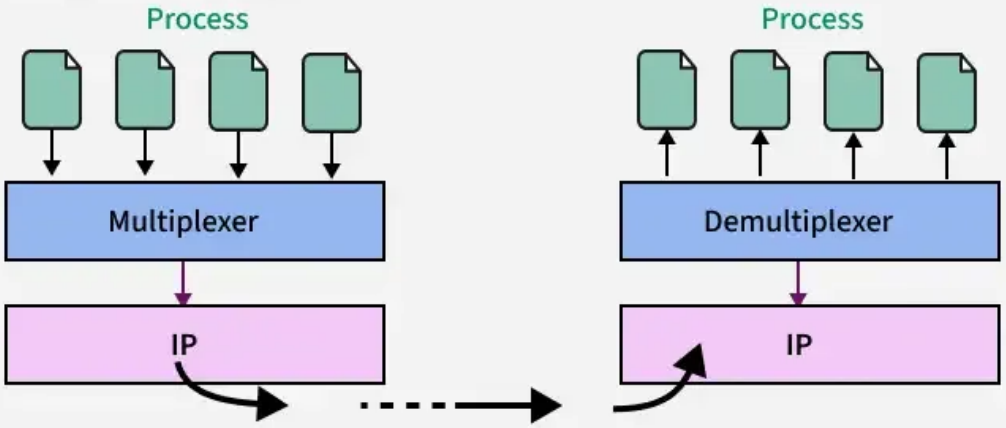


2. Flow Control



3. Multiplexing and Demultiplexing

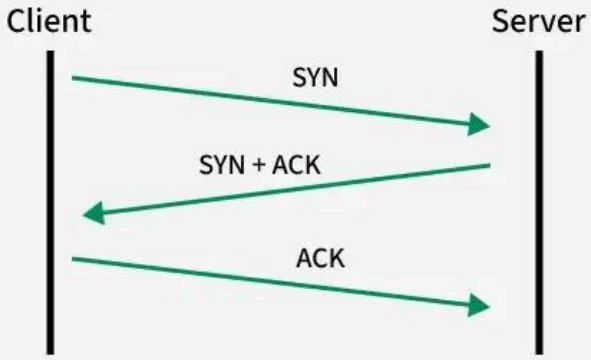
It manages multiple data streams on the same network connection by assigning unique ports.



4. Connection Establishment

TCP 3 Way Handshake

5. Connection Termination



6. Reliable Data Delivery

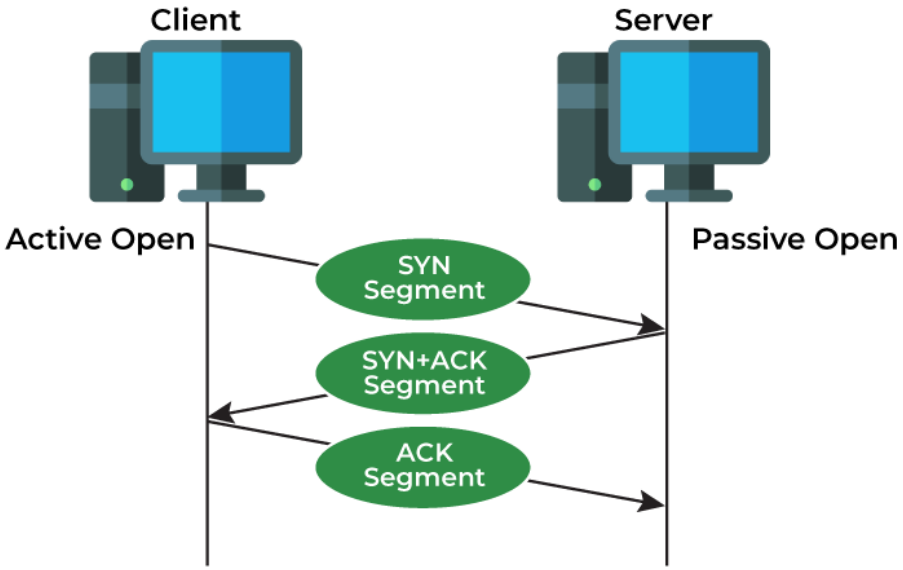
It ensures all data is delivered correctly, in sequence, and without duplication (in protocols like TCP).

7. Quality of Service (QoS) (Optional)

It ensures priority and performance for specific types of data, like video calls or file transfers.

TCP and UDP

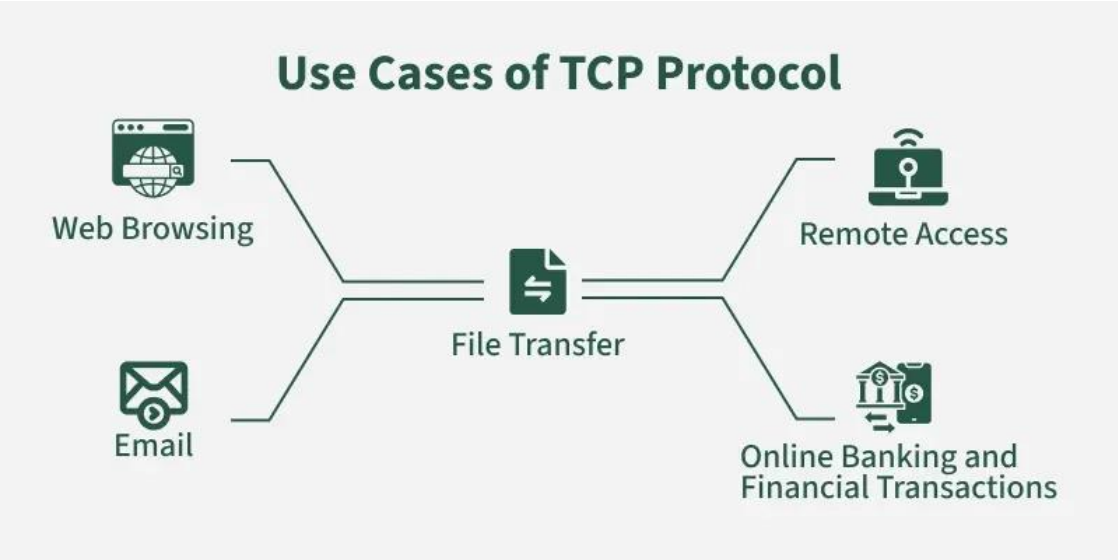
TCP is a layer 4 core communication protocol within the internet protocol which ensures reliable, ordered, and error-checked delivery of data between devices. When two devices establish a TCP connection, they perform a three-way handshake to confirm each other's presence and agree on parameters for data exchange.



TCP breaks information into packets, sends them, and then ensures all packets arrive in the correct order. If any packets are lost or damaged during transmission, TCP automatically requests them to be re-sent. This approach makes TCP ideal for applications where data accuracy is more important than speed, such as browsing web pages, sending emails, and downloading files because users receive complete and correctly sequenced information.

Use Cases of TCP Protocol

TCP (Transmission Control Protocol) is one of the main parts of the internet which provides reliable and ordered data delivery. Here are some of its key use cases:



1. Web Browsing:

When you type a URL into your browser, your computer uses TCP to establish a connection with the web server.

TCP ensures that the HTML, CSS, and JavaScript files that make up the webpage are delivered accurately and in the correct order.

2. Email:

Protocols like SMTP (Simple Mail Transfer Protocol) and IMAP (Internet Message Access Protocol) rely on TCP for sending and receiving emails.

TCP guarantees that your emails are delivered completely and in the correct sequence.

3. File Transfer:

Protocols like FTP (File Transfer Protocol) and SFTP (Secure File Transfer Protocol) utilize TCP for transferring files between computers.

TCP's reliability ensures that files are transferred accurately and without data corruption.

4. Remote Access:

Protocols like Telnet and SSH (Secure Shell) use TCP for remote access to other computers.

TCP ensures that commands and data are transmitted reliably, allowing you to interact with remote systems securely.

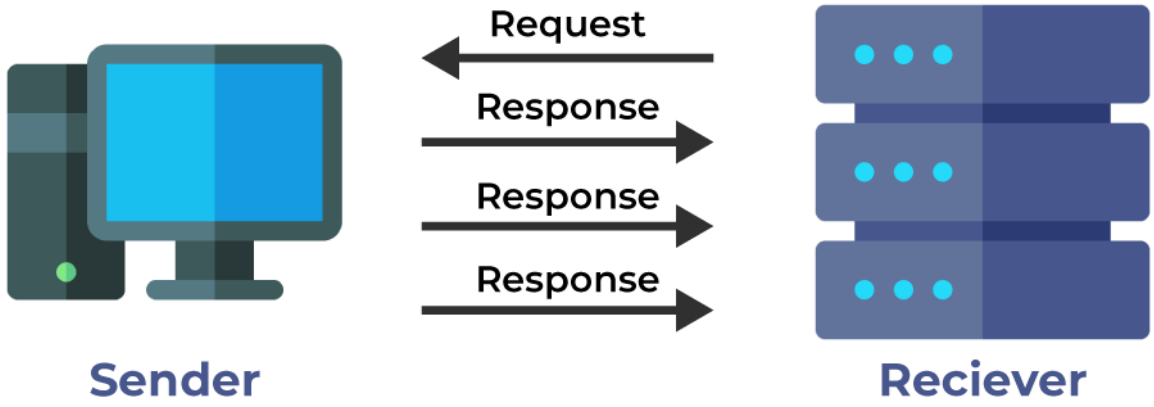
5. Online Banking and Financial Transactions:

TCP's reliability and security are crucial for online banking and financial transactions.

It ensures that sensitive data is transmitted securely and accurately, preventing data loss or corruption.

What is User Datagram Protocol (UDP)?

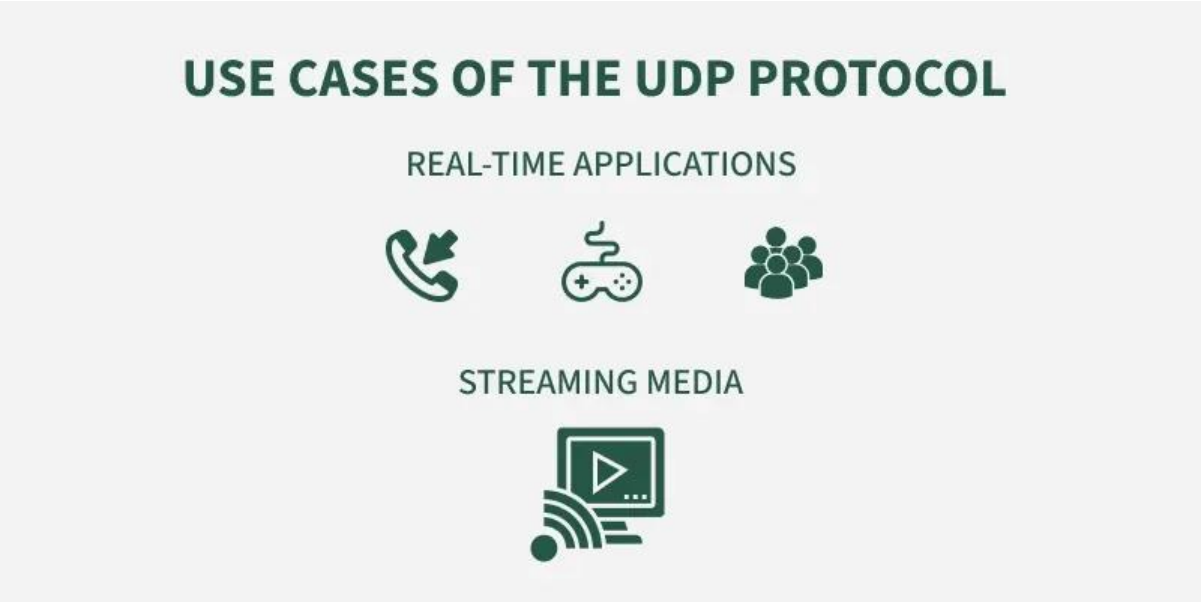
User Datagram Protocol (UDP) is a layer 4 communication protocol used in the internet’s network layer, transport layer, and session layer. Unlike TCP it sends data as independent packets called datagrams without first establishing a dedicated connection. This means UDP does not guarantee delivery, order, or error correction, it simply sends data and hopes it arrives.



Because it skips these checks, UDP has very low overhead and latency which makes it ideal for applications where speed is more important than perfect reliability. Examples include live video streaming, online gaming, and voice calls, where a few missed packets are often less noticeable than the delay that comes from waiting for perfect delivery.

Use Cases of the UDP Protocol

UDP (User Datagram Protocol) is a connectionless protocol that prioritizes speed and efficiency over reliability. Here are some key use cases of the UDP:



1. Real-time Applications:

Voice over IP (VoIP): Services like Skype, Zoom, and Google Meet often utilize UDP for real-time voice and video communication. While some packet loss is acceptable, minimizing latency is crucial for a smooth conversation.

Online Gaming: Many online games rely on UDP for fast, low-latency communication between players and game servers. This ensures responsiveness and prevents gameplay delays.

Video Conferencing: Similar to VoIP, UDP is used for real-time video conferencing applications where timely delivery of video and audio streams is essential.

2. Streaming Media:

Live Streaming: Services like Twitch, YouTube Live, and Netflix use UDP for streaming audio and video content. While some packet loss is acceptable, UDP's speed and efficiency are crucial for delivering a smooth streaming experience.

3. Network Management Protocols:

DNS (Domain Name System): UDP is commonly used for DNS lookups, where quick responses are essential for resolving domain names into IP addresses.

SNMP (Simple Network Management Protocol): This protocol is used for monitoring and managing network devices. UDP's speed and efficiency make it suitable for collecting performance data from network devices.

DHCP (Dynamic Host Configuration Protocol): UDP is used for dynamically assigning IP addresses to devices on a network.

4. Broadcast and Multicast:

Broadcast Applications: UDP is well-suited for broadcast applications where a single message needs to be sent to multiple recipients simultaneously, such as network discovery protocols.

Multicast Applications: UDP is used for multicast applications where a message needs to be sent to a specific group of recipients efficiently.

What is TCP vs UDP?

Session Multiplexing:

A single host with a single IP address is able to communicate with multiple servers. While using TCP, first a connection must be established between the server and the receiver and the connection is closed when the transfer is completed. TCP also maintains reliability while the transfer is taking place.

UDP on the other hand sends no acknowledgement of receiving the packets. Therefore it provides no reliability.

Segmentation:

Information sent is first broken into smaller packets for transmission.

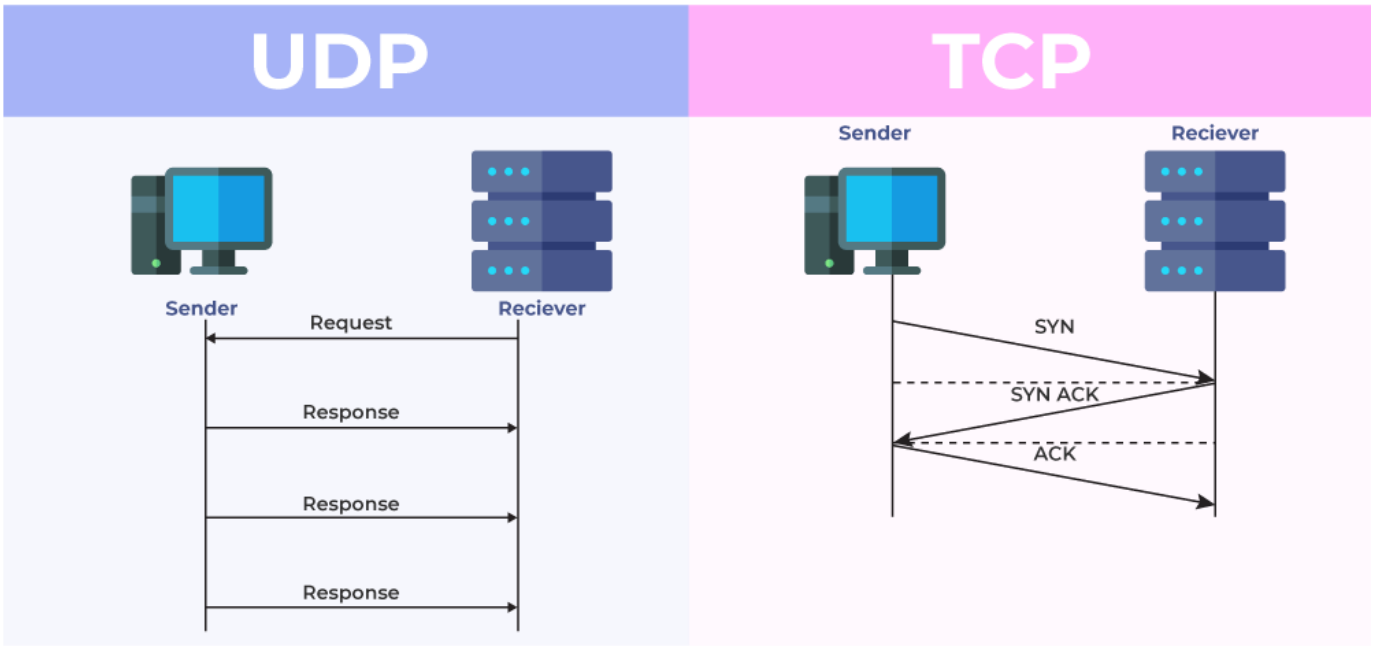
Maximum Transmission Unit or MTU of a Fast Ethernet is 1500 bytes whereas the theoretical value of TCP is 65495 bytes. Therefore data has to be broken into smaller packets before being sent to the lower layers. MSS or Maximum Segment Size should be set small enough to avoid fragmentation. TCP supports MSS and Path MTU discovery with which the sender and the receiver can automatically determine the maximum transmission capability.

UDP doesn't support this, therefore it depends on the higher layer protocols for data segmentation.

Flow Control:

If sender sends data faster than what receiver can process then the receiver will drop the data and then request for a retransmission, leading to wastage of time and resources. TCP provides end-to-end flow control which is realized using a sliding window. The sliding window sends an acknowledgement from receiver's end regarding the data that the receiver can receive at a time.

UDP doesn't implement flow control and depends on the higher layer protocols for the same.



Connection Oriented:

TCP is connection oriented, i.e., it creates a connection for the transmission to take place, and once the transfer is over that connection is terminated.

UDP on the other hand is connectionless just like IP (Internet Protocol).

Reliability:

TCP sends an acknowledgement when it receives a packet. It requests a retransmission in case a packet is lost.

UDP relies on the higher layer protocols for the same.

Headers:

The size of TCP header is 20-bytes (16-bits for source port, 16-bits for the destination port, 32-bits for seq number, 32-bits for ack number, 4-bits header length)

The size of the UDP header is 8-bytes (16-bits for source port, 16-bits for destination port, 16-bits for length, 16-bits for checksum), it's significantly smaller than the TCP header.

Both UDP and TCP header is comprised of 16-bit Source port(these are used for identifying the port number of the source) fields and 16-bits destination port (these are used for specifying the offered application) fields.

How to Choose Between TCP and UDP?

On the Basis of Reliability vs. Speed:

TCP provides reliable, ordered, and error-checked delivery of data which makes it ideal for applications where accuracy matters more than speed (e.g., web pages, emails, file transfers).

UDP offers faster, connectionless transmission without guaranteeing delivery or order, suitable for real-time applications like video streaming, online gaming, or voice calls, where speed and low latency are more important than perfect accuracy.

On the Basis of Connection Overhead:

TCP establishes and maintains a connection between sender and receiver, adding overhead but ensuring stable data flow and retransmissions if packets are lost.

UDP does not establish a dedicated connection, reducing latency and overhead, but leaving error-handling to the application if needed.

On the Basis of Use Case :

TCP is best when demanding correctness such as web browsing, file downloads, and secure data transfers.

UDP is preferred when you need minimal delay and can tolerate some data loss such as live broadcasts, online gaming, and VoIP.

Simple Network Management Protocol (SNMP)

Simple Network Management Protocol (SNMP) is a widely used application-layer protocol for managing and monitoring network devices such as routers, switches, servers, printers, firewalls, and load balancers. It operates in the application layer of the TCP/IP model and allows administrators to manage performance, identify faults, and plan for network growth.

What is Simple Network Management Protocol (SNMP)?

SNMP is an Internet Standard protocol used for managing and monitoring network-connected devices. It works over UDP ports 161/162 and supports monitoring, fault detection, and remote configuration.

Architecture of SNMP

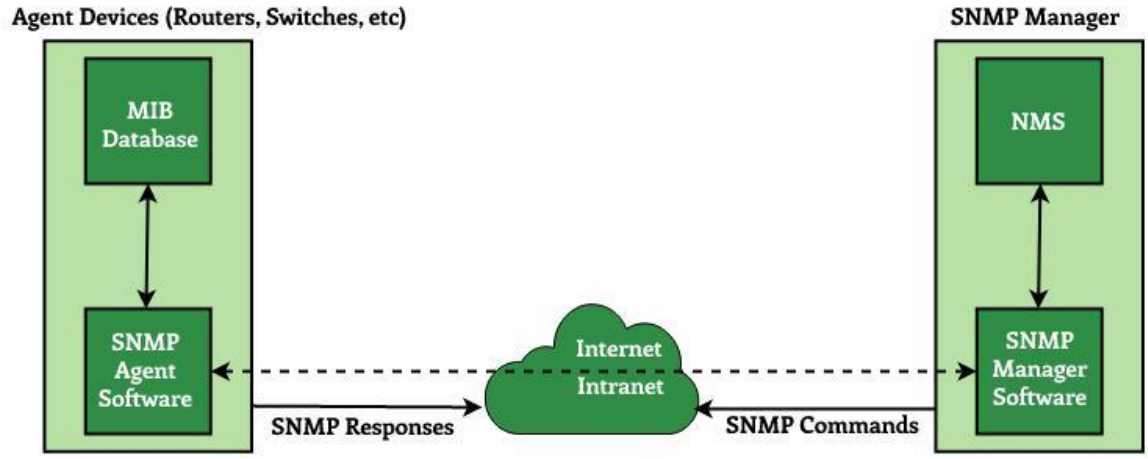
There are mainly three main components in SNMP architecture:

SNMP Manager: It is a centralized system used to monitor the network. It is also known as a Network Management Station (NMS). A router that runs the SNMP server program is called an agent, while a host that runs the SNMP client program is called a manager.

SNMP agent: It is a software management software module installed on a managed device. The manager accesses the values stored in the database, whereas the agent maintains the information in the database. To ascertain if the router is congested or not, for instance, a manager can examine the relevant variables that a router stores, such as the quantity of packets received and transmitted.

Management Information Base:MIB consists of information on resources that are to be managed. This information is organized hierarchically. It consists of objects instances which are essentially variables. A MIB, or collection of all the objects under management by the manager, is unique to each agent. System, interface, address translation, IP, UDP , and EGP , ICMP , TCP are the eight categories that make up MIB. The MIB object is home to these groups.

SNMP Architecture



SNMP Messages

GetRequest : It is simply used to retrieve data from SNMP agents. In response to this, the SNMP agent responds with the requested value through a response message.

GetNextRequest : To get the value of a variable, the manager sends the agent the GetNextRequest message. The values of the entries in a table are retrieved using this kind of communication. The manager won't be able to access the values if it doesn't know the entries' indices. The GetNextRequest message is used to define an object in certain circumstances.

SetRequest : It is used by the SNMP manager to set the value of an object instance on the SNMP agent.

Response : When sent in response to the Set message, it will contain the newly set value as confirmation that the value has been set.

Trap : These are the message sent by the agent without being requested by the manager. It is sent when a fault has occurred.

InformRequest : It was added to SNMPv2c and is used to determine if the manager has received the trap message or not. It is the same as a trap but adds an acknowledgement that the trap doesn't provide.

SNMP Security Levels

noAuthNoPriv: This (no authentication, no privacy) security level uses a community string for authentication and no encryption for privacy.

authNopriv: This security level (authentication , no privacy) uses HMAC with Md5 for authentication and no encryption is used for privacy.

authPriv: This security level (authentication, privacy) uses HMAC with MD5 or SHA for authentication and encryption uses the DES-56 algorithm.

Versions of SNMP

SNMPv1: It uses community strings for authentication and uses UDP only. SNMPv1 is the first version of the protocol. It is described in RFCs 1155 and 1157 and is simple to set up.

SNMPv2c: It uses community strings for authentication. It uses UDP but can be configured to use TCP. Improved MIB structure elements, transport mappings, and protocol packet types are all included in this updated version. However, it also makes use of the current "community-based" SNMPv1 administrative structure, which is why the version is called SNMPv2c. RFC 1901, RFC 1905, and RFC 1906 all describe it.

SNMPv3: It uses Hash-based MAC with MD5 or SHA for authentication and DES-56 for privacy. This version uses TCP. Therefore, the conclusion is the higher the version of SNMP, the more secure it will be. NMPv3 provides the remote configuration of SNMP entities. This is the most secure version to date because it also includes authentication and encryption, which may be used alone or in combination. RFC 1905, RFC 1906, RFC 2571, RFC 2572, RFC 2574, and RFC 2575.6 are the RFCs for SNMPv3.

Characteristics of SNMP

SNMP is used to monitor network.

It detects any network faults.

It can also be used to configure remote devices.

It allows a standardized way of collecting information about all kinds of devices from various manufacturers among the networking industry.

Advantages of SNMP

It is easy to implement.

Agents are widely implemented.

Agent level overhead is minimal.

It is robust and extensible.

Polling approach is good forLAN based managed object.

It offers the best direct manager agent interface.

Limitation of SNMP

It does not scale well.

There is no object orietned data view.

It has no standard control definition.

It has many implementation specific (private MIB) extensions.

It has high communication overhead due to polling