# Introduction to Network Protocols

In computer networking, a **network** is a group of devices (computers, servers, etc.) connected through various communication channels (wired or wireless) to allow resource sharing, data exchange, and communication. Networks can vary in size, from small home networks to large networks like the Internet.

The devices within a network communicate with each other by following certain rules, known as **network protocols**. These protocols are standardized methods for reliable and efficient data exchange.

## Why Network Protocols?

- Protocols define the format of data packets so that devices can process transmitted data correctly.
- They specify how devices can exchange messages: Sequence of messages, types of messages (e.g., request, response), acknowledgements, etc.
- They provide a way to address and route data packets: how devices are identified, how addresses are assigned, and how routing decisions are made to send data to the desired destinations.
- They include features to detect and correct errors during data transmission: checksums, error detection codes, error correction algorithms, etc.
- They implement flow control mechanisms to manage the rate of data transmission.
- The goal of several protocols is to ensure reliable data transmission. For this, they guarantee the successful delivery of data packets using mechanisms like acknowledgements, retransmissions, error recovery, etc.
- Some protocols include security features to protect data confidentiality and integrity.
- Protocols are designed to scale well with growth in the size of

networks.

## The OSI model of Networking

The OSI (Open Systems Interconnection) model is a structured framework to understand the system of computer networking. It divides the whole system into seven different layers.

- Each layer has specific responsibilities.
- Each layer interacts with adjacent layers to facilitate communication.

**Seven layers of the OSI model (from top to bottom)**

1. **Application Layer** (Top layer): Provides services and protocols that enable software applications to access network resources and exchange data. Examples of protocols: HTTP (Hypertext Transfer Protocol), SMTP (Simple Mail Transfer Protocol), and FTP (File Transfer Protocol).
2. **Presentation Layer:** Responsible for data formatting, encryption, and compression.
3. **Session Layer:** Establish, manage, and terminate communication sessions between applications.
4. **Transport Layer:** Ensure reliable and efficient data transfer between end systems. It segments data received from the upper layers into smaller units (data packets) and handles end-to-end error recovery, flow control, and congestion control. Examples of protocols: TCP (Transmission Control Protocol) and UDP (User Datagram Protocol).
5. **Network Layer:** Determine the best path for data packets to reach their destination across different networks, using routing protocols and IP addressing.
6. **Data Link Layer:** Provide reliable and error-free transmission of data over the physical layer. It is responsible for establishing and terminating connections between network nodes, performing error detection and correction, and handling the flow of data frames between adjacent nodes.
7. **Physical Layer** (Lowest Layer): Deals with the physical transmission

of data. It defines the electrical, mechanical, and physical aspects of the network like cables, connectors, and signalling methods.

## Types of Network Protocols

There are various types of network protocols that serve different purposes and operate at different layers of the OSI model. Each protocol has its own set of rules, formats, and functions.

Here are some commonly used network protocols:

- Internet Protocol (IP)
- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)
- Hypertext Transfer Protocol (HTTP)
- Hypertext Transfer Protocol Secure (HTTPS)
- File Transfer Protocol (FTP)
- Simple Mail Transfer Protocol (SMTP)
- Domain Name System (DNS)
- Remote Procedure Call (RPC)
- Dynamic Host Configuration Protocol (DHCP)

Let's move forward to discuss some of them in detail.

## Internet Protocol (IP)

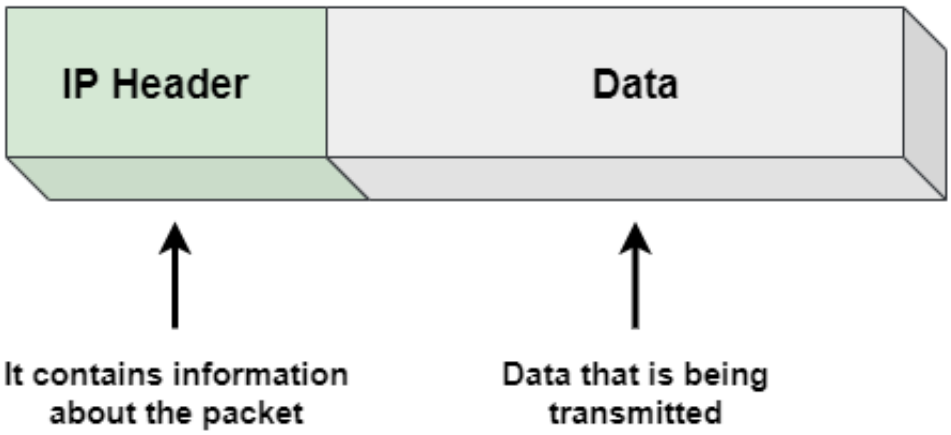
Internet Protocol (IP) is a network layer protocol. It allows data packets to be routed and addressed in order to pass through networks and reach their destination. When one machine wants to send data to another machine, it sends it in the form of IP packets.

An IP packet is the fundamental unit of data that is transmitted over the internet or any other network that uses the Internet Protocol (IP). The IP packet has two main parts: IP header and data (payload). Here size of the packet includes the size of both IP header and data section.

- IP header section contains information about the packet like source

and destination IP addresses, total size of the packet, version of the IP, and other control information.
- Payload contains the actual data being transmitted i.e. web pages, images, or any other type of content.

Other important information in the IP header are: 1) Time-to-live (TTL) field, which determines the number of hops a packet can make before it's discarded 2) Protocol field, which specifies the protocol being used for the data section.




When a device sends a packet, it first determines the IP address of the destination device. The packet is then sent to the first hop on the network i.e. a router. The router examines the destination IP address in the header and determines the next hop on the network to which the packet should be forwarded. This process continues until the packet reaches its final destination.

Here routing algorithms are used to determine the best path for the packets to take through the network. We will cover various routing algorithms in a separate blog.

**What is an IP address?**

- IP addresses are like digital addresses that identify and locate devices on a network. They are unique and assigned to each device or domain that connects to the Internet.
- IP addresses are written as four numbers separated by periods, such

as 192.168.1.1. These numbers represent the numerical value of the address.
- There are two main types of IP addresses: IPv4 and IPv6.

**IPv4 vs. IPv6**

IPv4 was the first version of IP introduced in 1983. It uses a 32-bit address scheme i.e. it can accommodate a maximum of 4,294,967,296 unique IP addresses. Despite its limitations, IPv4 still carries a significant chunk of internet traffic today, around 94%. IPv4 also provides several benefits like encrypted data transmission and cost-effective data routing.

With the increasing demand for IP addresses, IPv6 was developed in the 1990s to overcome the limitations of IPv4. It uses 128-bit address space, capable of accommodating 3.4 x 10^38 unique IP addresses. It is also known as IPng (Internet Protocol next generation).

IPv6 offers improved routing, packet processing and security features compared to IP4. But IPv6 is not compatible with IPv4, and upgrading can present a challenge.

## Transmission Control Protocol (TCP)

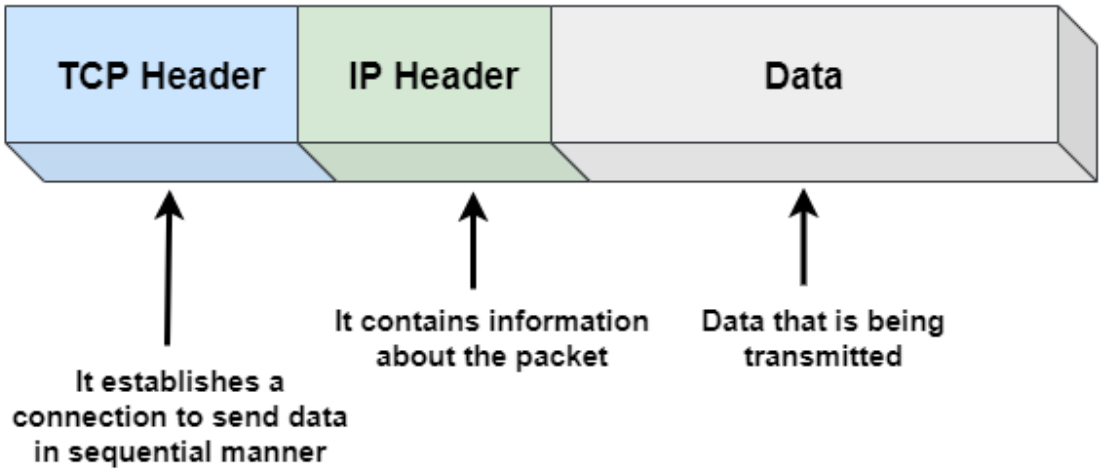
When sending large files like images or emails, a single IP packet may not be sufficient. In these cases, multiple packets are used to transmit the data. Unfortunately, this increases the risk of packets not reaching their destination, getting lost, or arriving in the wrong order. That's where the transport layer protocol comes into the picture.

Transmission Control Protocol (TCP) is a transport layer protocol, which is built on top of the Internet Protocol (IP). It is used in conjunction with the IP to provide a complete suite of communication protocols for transmitting data.

- TCP breaks data into small packets and numbers them in sequence to ensure that they are received in the correct order at the receiving end. Lost data packets can also be retransmitted using TCP. Overall,

IP is responsible for delivering the packets, while TCP helps to put them back in the correct order.

- When a machine wants to communicate with another machine using TCP, it establishes a connection with the destination machine through a sequenced acknowledgement (a handshake process). Once the connection is established, the two machines can communicate freely. In simple words, TCP is a connection-oriented protocol, where connection must be established between applications before data transfer can occur.
- It also provides error-checking mechanisms to ensure that the data is received without any errors.




Here is how TCP works:

1. **Connection establishment:** The sender and receiver establish a connection using a three-way handshake before transmitting data. The sender sends an SYN (synchronize) packet to the receiver, the receiver responds with an SYN-ACK (synchronise-acknowledgement) packet, and the sender sends an ACK (acknowledgement) packet to complete the connection.
2. **Packet creation:** Now sender divides data into TCP segments, with each segment containing a sequence number to ensure the data is reassembled correctly on the receiver. The segments also contain other information like source port number, destination port number,

and checksums for error detection.
3. **Packet transmission:** Now TCP segments are sent over the network to the destination IP address and port number. TCP uses flow control mechanisms to ensure that the sender does not overwhelm the receiver with too much data at once.
4. **Packet delivery:** Now destination receives the TCP segments and reassembles them into the original data. If a segment is lost or corrupted, the receiver sends a request for retransmission to the sender.
5. **Connection termination:** Once data transmission is complete, the sender and receiver initiate a connection termination using a four-way handshake. The sender sends a FIN (finish) packet to the receiver, the receiver sends an ACK packet to acknowledge the FIN, and then sends its own FIN packet. The sender responds with an ACK packet to acknowledge the receiver's FIN packet.

**Use cases of TCP in real life applications**

TCP is used by a wide range of applications. Here are some examples:

- Email applications like Outlook or Gmail use the Simple Mail Transfer Protocol (SMTP) over TCP to send and receive email messages.
- When you transfer files using protocols like FTP or SFTP, these protocols use TCP to ensure the reliable transmission of the data.
- Remote desktop software uses TCP to transmit desktop screen data and user inputs between the client and server.

## User Datagram Protocol (UDP)

UDP is a connectionless transport layer protocol. Unlike TCP, UDP does not provide any reliability, flow control, or error recovery functions. As a result, UDP is useful in situations where reliability mechanisms of TCP are not necessary and faster transmission is desired. In other words, UDP is useful in applications where speed and low latency are more important than reliability. Here dropped packets can be tolerated without significant impact on the application performance.

Here is how UDP works:

1. **Packet creation:** Sender encapsulates the data into a UDP datagram, which includes source and destination ports, as well as the length of the datagram.
2. **Packet transmission:** UDP datagram is then sent over the network to the destination IP address and port number. UDP does not establish a connection before transmitting data, so the data is simply sent to the destination without any handshaking or connection setup.
3. **Packet delivery:** Destination receives UDP datagram and processes the data within the datagram. Since UDP is an unreliable protocol, it does not provide any mechanism for detecting lost or corrupted packets. So the receiving application must implement its own error detection and recovery mechanisms if required.
4. **Packet acknowledgement:** If the receiving application requires acknowledgement of receipt, it can send a response packet back to the sender, indicating that the data was received successfully.

**Use cases of UDP in real-life applications**

There are some situations where UDP is preferred over TCP.

- TCP is a reliable protocol but this reliability comes at the cost of latency. In applications like online gaming, Video streaming (YouTube or Netflix) or video conferencing (Skype or Zoom), low latency is crucial for a good user experience. So in such cases, UDP can be a better choice. Here dropped packets can be tolerated in favor of speed.
- Low-power devices like IoT sensors have limited processing power and battery life, and the overhead of establishing a TCP connection can be too high. So in such cases, one can use lightweight protocols like Constrained Application Protocol (CoAP), which rely on UDP.
- DNS (Domain Name System) primarily uses UDP for communication between DNS servers and clients. But if a DNS query or response packet is too large to fit within a single UDP datagram, then DNS can use TCP to transmit the packet in multiple parts. In such cases, DNS

uses TCP as a fallback protocol.

## HyperText Transfer Protocol (HTTP)

Hypertext Transfer Protocol (HTTP) is an application layer protocol. It is built on top of Transmission Control Protocol (TCP) and provides a higher level of abstraction that allow developers to focus on their applications rather than the details of TCP and IP packets. Some common HTTP methods are PUT, GET, POST, and DELETE.

- HTTP protocol operates on a request-response model i.e. one machine sends a request for information, and the other machine sends back a response with the requested information. This is how text, images, and other files are shared all across the World Wide Web.
- HTTP is a stateless protocol i.e. the client and server only maintain awareness of each other during the active connection. Once the connection is terminated, both parties forget about each other's existence.

HTTP has several benefits that make it a popular choice for transmitting information over the web. It requires low memory and CPU usage due to its few concurrent connections. Additionally, it has the ability to report errors without closing connections, which can reduce network congestion. But HTTP is not equipped with encryption capabilities, so it may not be the most secure option for transmitting sensitive information.

### Hypertext Transfer Protocol Secure (HTTPS)

Hypertext Transfer Protocol Secure (HTTPS) is a secured version of Hypertext Transfer Protocol (HTTP). Unlike HTTP, which transfers data in a plain text format, HTTPS transfers data in an encrypted format and protects it from interpretation or modification by hackers during the transfer of packets.

It is often used when sensitive information like passwords and financial

transactions is transmitted. But HTTPS may be slower than HTTP due to the overhead of the encryption and decryption process.

## File Transfer Protocol (FTP)

File Transfer Protocol (FTP) is used for transferring files between hosts, both local and remote. It runs on top of Transmission Control Protocol (TCP) and creates two TCP connections: control connection and data connection. Control connection is used to transfer control information like passwords and commands to retrieve and store files, while data connection is used to transfer the actual file. Both connections run in parallel during the entire file transfer process.

- FTP has several advantages: ability to share large files at the same time, resume file sharing if it is interrupted, recover lost data and schedule file transfers.
- FTP has some disadvantages as well. It lacks security, as data, usernames, and passwords are transferred in plain text, which makes them vulnerable to malicious actors. FTP also lacks encryption capabilities, which makes it non-compliant with industry standards.

## Remote Procedure Call (RPC)

Remote Procedure Call (RPC) allows a program on one device to request a service from a program on another device, without the need to understand details of the network. It is used for interprocess communication in client-server based applications.

- RPC works on a client-server model, where requesting program is the client and service providing program is the server. It uses either Transmission Control Protocol (TCP) or the User Datagram Protocol (UDP) to carry messages between communicating programs.
- There are several advantages to using RPC. It can improve performance by omitting many protocol layers and minimize code rewriting or redevelopment efforts. However, RPC has not yet been proven to work effectively over wide-area networks and does not

support other transport protocols besides TCP/IP.

## Challenges with Network Protocols

- Handling large amounts of data, traffic, and devices can affect protocol performance.
- Some network protocols designed for specialized applications can add significant overhead. Due to this, they can consume network resources and impact performance.
- Achieving fault tolerance and reliable data transfer in different network conditions can be challenging.
- Hackers may exploit protocol weaknesses to gain unauthorized access or disrupt network operations.
- Sometimes, different systems may implement protocols differently or support different versions of protocols. This can lead to compatibility issues.
- With advancements in technology, network protocols need to evolve.

Thanks to Chiranjeev and Navtosh for their contribution in creating the first version of this content. If you have any queries or feedback, please write us at contact@enjoyalgorithms.com. Enjoy learning, Enjoy system design!