Chapter: 1

Introduction

Contents:

- Network Programming Features and Scope
- Network Programming Language, Tools & Platforms
- Client and Server Application
- Client Server Model and Software Design

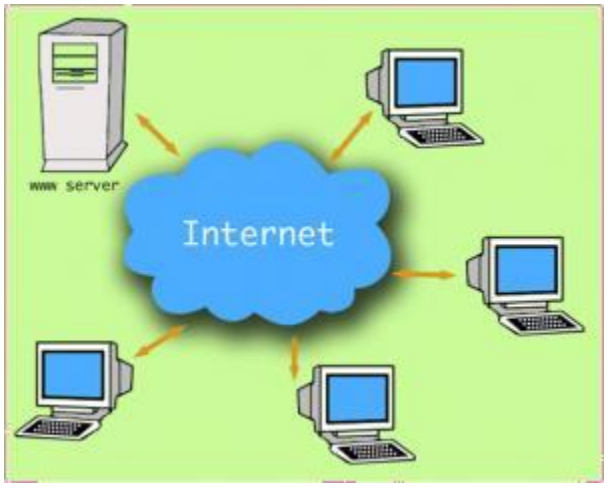
Definition:

Network: A network is a group of two or more computers or other electronic devices that are

interconnected for the purpose of exchanging data and sharing resources.

Group of Device:

- Connect
- Communicate



Network Programming:

Network programming is a type of programming that involves writing code to communicate with other devices over a network.

Network Programming Feature and Scope:

some key features of network programming:

- Socket Programming: Sockets are a fundamental concept in network programming. Sockets are endpoints for sending and receiving data across a network. Network programming often involves using sockets to establish connections between devices.
- Client-Server Architecture: Network programming often involves creating client-server applications, where one device acts as the server, providing resources or services, and the other devices act as clients, accessing those resources or services.
- Protocols: Network programming involves using network protocols to ensure that devices can communicate with each other. Protocols specify the rules and procedures for transmitting and receiving data over a network.

- Multi-threading: Network programming often requires the use of multiple threads to handle simultaneous connections and requests. Multi-threading allows an application to handle multiple tasks at the same time.
- Security: Network programming involves implementing security measures to protect against unauthorized access and data breaches. This may include encryption, authentication, and access controls.
- Asynchronous I/O: Network programming often requires performing I/O operations without blocking the execution of the program. Asynchronous I/O allows an application to continue processing while waiting for I/O operations to complete.
- Remote Procedure Calls (RPC): Network programming can involve using remote procedure calls to allow programs on different devices to communicate with each other. RPC allows a program

to call a function on a remote device as if it were a local function call.

The **scope** of network programming is vast and covers a range of topics related to the communication between two or more computing devices over a network.

Network programming is essential for building a wide range of **networked applications and systems**, and it plays a critical role in the modern digital world.

Network Programming Language Tools and Platform:

- Python (open-source programming language)
- Java (general-purpose, object-oriented programming language)

- Perl (general-purpose programming language)
- Bash (command-line-interface tool)

In addition to these programming languages, several **platforms** like Apache Kafka, RabbitMQ, and Redis are also commonly used for network programming.

Apache Kafka:

- Open source distributed streaming
- Operate as a distributed system
- Stream processing platform
- High throughput
- Low latency platform for handling real-time feeds.

RabbitMQ:

- RabbitMQ is designed to handle messaging between applications, particularly in distributed systems.
- High-throughput messaging systems.

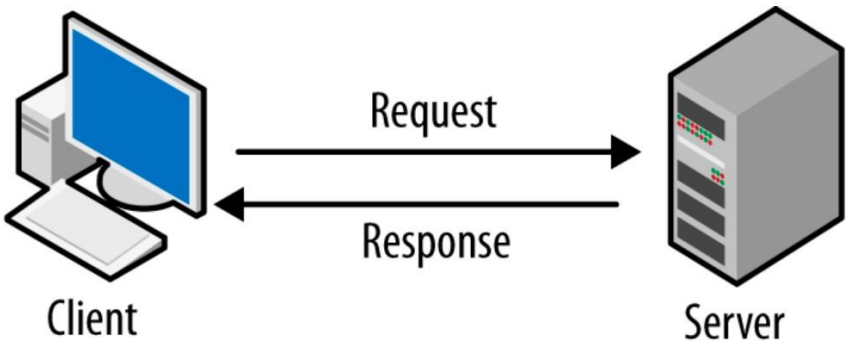
Redis:

- Redis is an open-source, in-memory data structure store that can be used as a database, cache, and message broker.
- It is designed to be fast, efficient, and reliable, making it an excellent choice for applications that require real-time data processing.

These platforms provide **easy-to-use APIs** and infrastructure for building distributed and scalable applications.

Client and Server Application:

- A client-server application is a distributed computing architecture that consists of two main components: a client and a server.
- The client is a program or application that requests services or resources from the server, while the server is a program or application that provides those services or resources to the client.
- A client-server application is **a program that runs on the client-side while accessing the information over a remote server**.



Client Server Model and Software Design:

- The client-server model is a **software architecture** in which a client sends requests to a server, which responds by sending data or performing some operation.
- In this model, the client and server communicate over a network using a standardized protocol such as **TCP/IP**. The client sends a request to the server, which processes the request and sends a response back to the client.
- In software design, the client-server model is often used to create distributed systems where different parts of the system are located on different machines.
- In the client-server architecture, when the client computer sends a request for data to the server through the internet, the server accepts the requested process and deliver the data packets requested back to the client. Clients do not share any of their resources.

Examples of Client-Server Model are Email, World Wide Web, etc.

Advantages of Client-Server model:

- Centralized system with all data in a single place.
- Cost efficient requires less maintenance cost and Data recovery is possible.
- The capacity of the Client and Servers can be changed separately.

Disadvantages of Client-Server model:

- Clients are prone to viruses, Trojans and worms if present in the Server or uploaded into the Server.
- In case of server failure entire network will be failed.

- Scalaility
- Phishing or capturing login credentials or other useful information of the user are common and MITM(Man in the Middle) attacks are common.

2 Tier Architecture:

In 2-tier architecture, the client communicates directly with the server.

The server provides the necessary services or resources to the client, which then uses them locally.

In this model, the client performs the presentation layer and the application layer, while the server performs the data layer.

For example, in a simple client-server system like a file sharing system, the client application running on a user's computer directly accesses the file server to retrieve or upload files.

3 Tier Architecture:

In 3-tier architecture, the client communicates with an application server that serves as an intermediary between the client and the database server.

This model separates **the presentation layer, application layer, and data layer** into three separate tiers.

The first tier is the presentation tier, which includes the user interface and the client application.

The second tier is the application tier, which includes the business logic and processing of data.

The third tier is the data tier, which includes the database server where data is stored.

For example, in a web-based e-commerce application, the client interacts with the presentation tier (e.g., the user interface on a web page), which sends requests to the application server.

The application server processes these requests and interacts with the database server to retrieve or update data.

The application server then sends the updated information back to the presentation tier, which displays it to the user.

Design Consideration of Client Server Application Architecture:

- Client and Server machines need different amounts of hardware and software resources.

- Client and server machines may belong to different vendors.
- Horizontal scalability (increase of the client machine) and vertical scalability (migration to a more powerful server or to a multiserver solution)
- A client or server application interacts directly with a transport layer protocol to establish communication and to send or receive information.
- The transport protocol then uses lower layer protocols to send or receive individual message. Thus, a computer needs a complete stack of protocols to run either a client or a server.
- A single server-class computer can offer multiple services at the same time, a separate server program is needed for each service.

Few Protocols and Description:

IPv4:

Internet Protocol version 4. IPv4 uses 32-bit addresses and provides packet delivery service for TCP, UDP, SCTP, ICMP (Internet Control Message Protocol), and IGMP. (Internet Group Management Protocol).

ICMP is a protocol used for error reporting and diagnostics in IP networks, while IGMP is a protocol used for managing multicast groups in IP networks.

IPv6:

Internet Protocol version 6. IPv6 uses 128-bit addresses.

TCP:

Transmission Control Protocol. TCP is a **connection-oriented** protocol that provides a **reliable**, **full-duplex** byte stream to its users

UDP:

User Datagram Protocol. UDP is a **connectionless** protocol, and UDP sockets are an example of datagram sockets.

SCTP:

Stream Control Transmission Protocol. SCTP is a **connection-oriented** protocol that provides a reliable full-duplex association.

SCTP provides a reliable, message-oriented transport layer protocol that is well-suited for a wide range of applications that require message-oriented communication over IP networks.

ICMP:

Internet Control Message Protocol. ICMP handles error and control information between routers and hosts.