How Does Client-Server Architecture Work?

Client-server architecture is a common computing model where multiple client devices

(like computers, phones, tablets) interact with a central server to access services, data, or

resources. It mainly works on a request-response cycle.

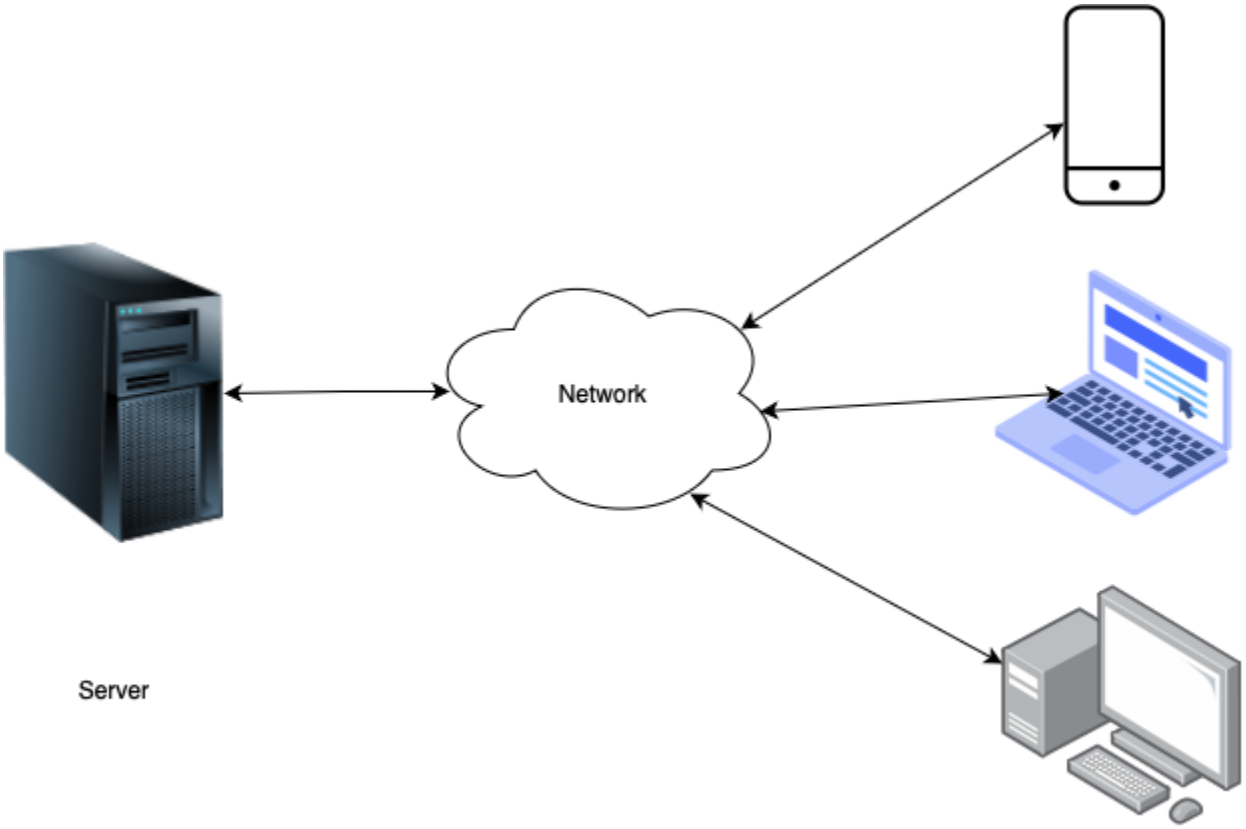

fig:1.1 **Client-Server Architecture**

How It Works:

a)Client initiates the request :The process starts when a client (usually a web browser or an app) sends a request to the server. This is done using standard communication protocols like HTTP, HTTPS, FTP, or TCP/IP.Example: When a user types a website URL in the browser, it sends a request to the serve that hosts that website.

b) Server processes the request: Once the server receives the request, it processes it. This may involve fetching data from a database, performing some calculations, or preparing the content that the user asked for. Example: A web server might fetch a web page or some user data from its database and prepare it to be sent back to the browser.

c) Server sends the response: After processing, the server sends the response back to the

client.Example: The server might return an HTML page, an image, or some data in JSON format.

d) Client displays the response: Finally, the client receives the response and displays it to

the user in a readable format.

Example: A browser will take the HTML page sent by the server and render it visually as a

web page.

In simple term, the client makes a request, the server does the work, and then sends the

results back and this cycle happens.

Factors to Consider When Designing Software

These are the some of the factors to be considered:

1) User Requirements :The first step is understanding what the users actually need. The

software should solve their problems or make their tasks easier. If user needs are not

clearly understood, the software may not be useful.

2)Simplicity and Usability :A simple and clean design makes the software easier to use. A

user-friendly interface helps people interact with the software smoothly.

3) Performance :The software should be able to perform tasks quickly and handle the

expected number of users or data.

4) Security: Proper Security is very necessary, especially if the software handles sensitive data.

5) Scalability: It should be designed in a way that allows it to grow.

6) Reliability: The software should be stable and not crash frequently. It should consistently perform its tasks without errors.

Why do we really need network programming tools and platforms? Explain some of

them.

They are important for the following reasons:

a)Easy Communications: Help computers, servers, and applications communicate with

each other across a network.

b) Security: Protect sensitive data from hackers and unauthorized access, especially in

online transactions.

c) Debugging and Troubleshooting: Help find, trace, and fix network-related issues like

slow speed or disconnection.

d) Faster Performance: Ensure data is transferred quickly and efficiently, improving overall

system speed.

e) Cloud and Web Applications: Allow applications to work online, such as cloud storage

platforms or e-commerce websites.

Some Important Tools and Platforms:

a)SSH – Secure Shell Access: Provides secure remote access to another computer over a network,And Commonly used by developers and system administrators to manage servers securely.

b) Wireshark – Network Traffic Monitor: Helps monitor and analyze network traffic in real-time. It is useful for diagnosing slow internet connections or detecting security threats.

c) cURL – Command-Line Data Transfer:Used to send and receive data from URLs via command line.Supports various protocols like HTTP, FTP, and more — ideal for testing APIs or downloading files.

d)Telnet – Remote Access Protocol: It allows users to connect to remote systems using a command-line interface. Mainly used for testing connectivity to specific ports, though less secure than SSH.