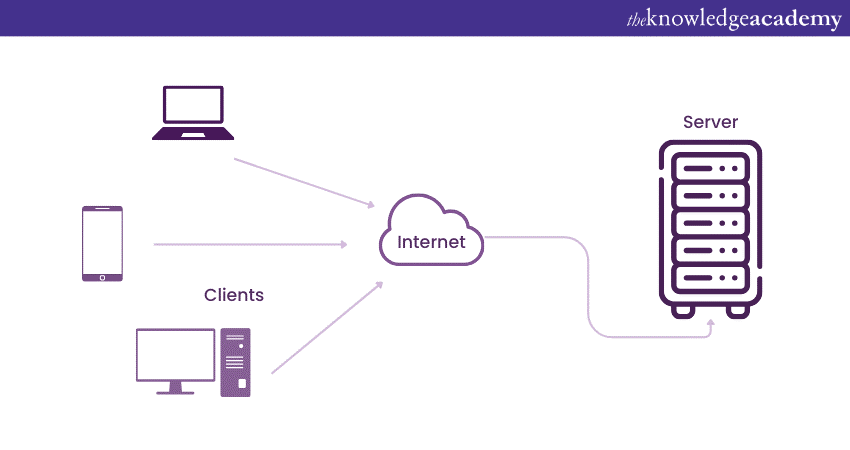
Client-Server Architecture PPT

The IT field is always changing and introducing new technologies that need to be integrated into the network. To stay ahead, IT professionals have to adapt to these changes quickly. To cope with this pressure, IT professionals use Client Server Architecture, which is a way of organising network resources between clients and servers.

A Client Server Architecture is a network-based computing structure where responsibilities and operations get distributed between clients and servers. Client-Server Architecture is widely used for network applications such as email, web, online banking, [e-commerce](https://www.theknowledgeacademy.com/blog/what-is-ecommerce/), etc. In this blog, we will explore Client-Server Architecture, components, types with examples, and how it works.

**What is Client Server Architecture?**



Client-Server Architecture is a network model that allows communication and data exchange between different applications over a single or multiple servers

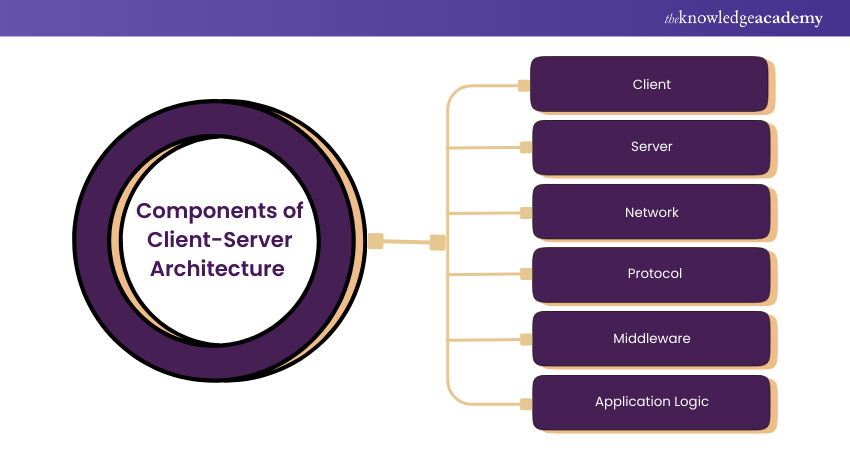
This model divides the system into two parts: the client side and the server side.

**a) Client:**An application that requests services from the [server](https://www.theknowledgeacademy.com/blog/what-is-a-server/), such as data retrieval, storage, calculations, or other functions.

**b) Server:** An application that processes client requests, sends responses, or performs specified actions. The server and client may reside on the same machine or different devices across the network. [Effective Communication](https://www.theknowledgeacademy.com/blog/what-is-effective-communication/) occurs via predefined protocols like HTTP, FTP, or SMTP.

Client-Server Architecture is widely used in applications such as email, web browsing, online banking, and e-commerce.

**Components of Client-Server Architecture**



Client-Server Architecture depends on three main components that need to work together for it to function. These components are:

**a) Client:**A client device or software that requests services from a server. Clients are consumer-facing and often include web browsers, mobile applications, or desktop applications that people can interact with. They communicate with the server to retrieve data, make transactions, or perform other tasks by delegating that responsibility to the server.

**b) Server:**A server is a computer or program that offers services or solutions to clients over a network. Servers handle processing of client requests, which includes tasks like file storage, database access, and application hosting, along with backend activities like computations, [data management](https://www.theknowledgeacademy.com/blog/data-management/), and business logic, significantly reducing what clients need to handle.

**c) Network:**This serves as the channel through which clients and servers are connected for data transfer between them. Networks range from local area networks (LAN) within a single building to wide area networks (WAN) and the internet, which can span countries. It acts as the intermediary, facilitating the interchange of requests and responses between the clients and servers, which influences the speed and reliability of these interactions.

**d) Protocol:**Protocols are rules that define how data is exchanged between clients and servers, ensuring communication is orderly, secure, and understandable. Common protocols include HTTP or HTTPS for web services, FTP for file transfers, and SMTP for email. They help bridge communication between different systems, independent of their technology stack.

**e) Middleware:** Middleware acts as a bridge between client-side and server-side code, enabling them to communicate. It performs tasks such as authentication, load balancing, data translation, and message queuing, simplifying interactions within the Client-Server model by enhancing transaction speed, scalability, and integration.

**f) Application Logic:** Application logic is the code and processes that determine how a server responds to client requests, involving business rules, [Big Data Processing](https://www.theknowledgeacademy.com/blog/big-data-processing/), and workflows on the server side. It ensures the server correctly interprets client requests, performs necessary calculations or data manipulations, and delivers appropriate responses.

**Characteristics of Client-Server Architecture**

Client-Server architecture has distinct characteristics:

a) The client and server machines can differ greatly in hardware and software requirements and may come from different vendors.

b) The network can scale horizontally by adding more client machines or vertically by upgrading to more powerful servers or a multi-server configuration.

c) A single server can offer multiple services simultaneously, though each service must run its own server program.

d) Both client and server applications interact directly with a transport layer protocol, establishing communication to send and receive information.

e) Both client and server computers require a full stack of protocols, where the transport protocol utilises lower-layer protocols to send and receive individual messages.

**How Does Client-Server Architecture Work?**

The basic steps of how Client-Server Architecture works are:

1) In the first step, the client sends a request to the server using the network medium. The request can be a query, a command, or a message.

2) In the second step, the server receives the request and processes it according to its logic and data. The server may access its own resources or other servers to fulfil the request.

3) In the third step, the server sends a response back to the client using the network medium. The response can be [Data](https://www.theknowledgeacademy.com/blog/what-is-data/), an acknowledgement, or an error message.

4) Lastly, the client receives the response and displays it to the user or performs further actions based on it.

**What are Some Examples of Client-Server Architecture?**

You might be surprised at how often you encounter Client-Server Architecture in everyday life. Here are three common examples:

**1) Email Servers:**Email has evolved into the primary communication method for businesses due to its speed and convenience. Various server components work together to deliver emails between users across different mail servers.

**2) File Servers:** When saving documents on services like [Google Docs](https://www.theknowledgeacademy.com/blog/what-is-google-docs-definition-and-working/) or [Microsoft Office](https://www.theknowledgeacademy.com/blog/what-is-microsoft-office/), you're interacting with file servers. These servers store data centrally and allow multiple clients to access it.

**3) Web Servers:** These high-powered servers host websites, which web clients access through DNS or an IP address. Here's a simplified process:

a) A user enters a URL in the browser.

b) The browser requests the IP address from the  Domain Name System (DNS) .

c) The DNS server provides the IP address to the browser.

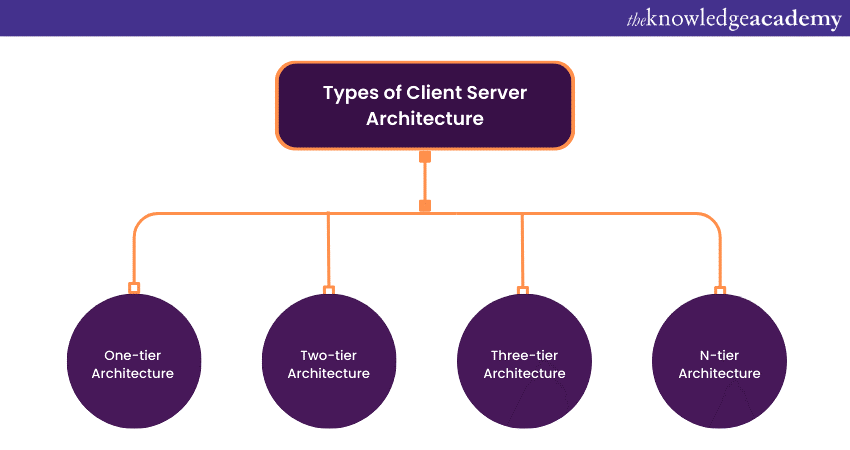
d) The browser sends an HTTPS or HTTP request to the web server.

e) The server sends back the requested files.

f) The user retrieves the files, and the process continues as needed.

These examples highlight how Client-Server Architecture forms the foundation of many digital services we rely on daily.

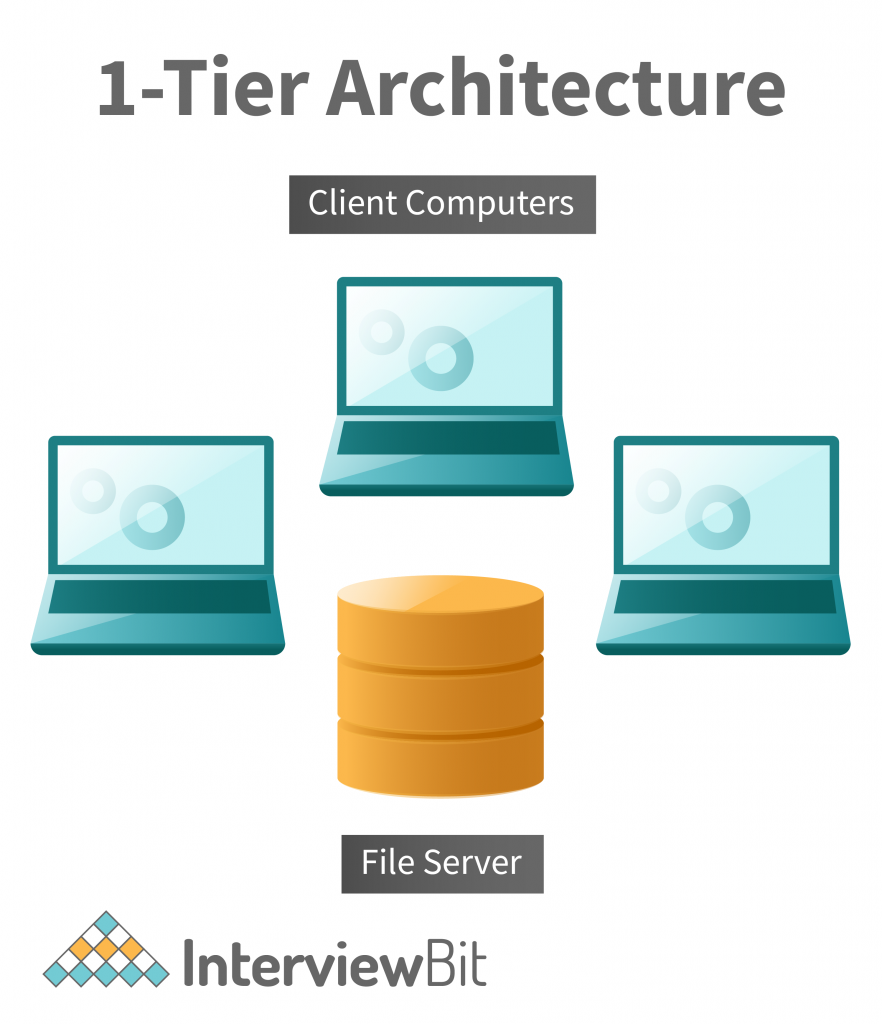
**Types of Client-Server Architecture**



There are different types of Client-Server Architecture, depending on how many tiers or layers are involved in the communication process. Some of the common types are:

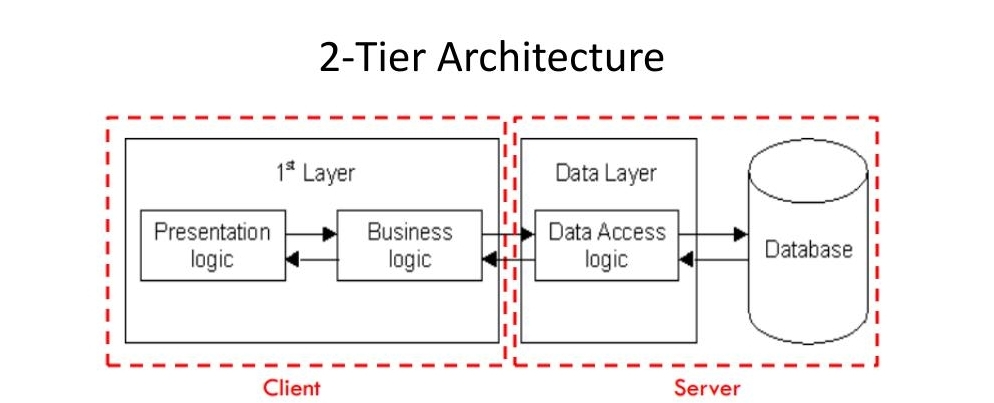
**1) One-tier Architecture**

A self-contained application on a single platform. In one-tier architecture, the client, server, and database are all on the same machine. The client handles user interaction and business logic, the server provides services like data storage and processing, and the database manages data. While simple and popular for small apps, this architecture is rarely used in production because it doesn't meet most system requirements.



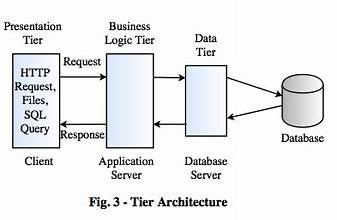
**2) Two-tier Architecture**

This basic Client-Server Architecture involves direct communication between the client and server without an intermediate layer. The client manages the [User Interface (UI)](https://www.theknowledgeacademy.com/blog/what-is-ui-design/) and business logic, while the server handles data storage and processing. An example is a web browser requesting pages from a web server, which responds with HTML files. It's easy to implement but has drawbacks like low scalability, high network traffic, and security risks.

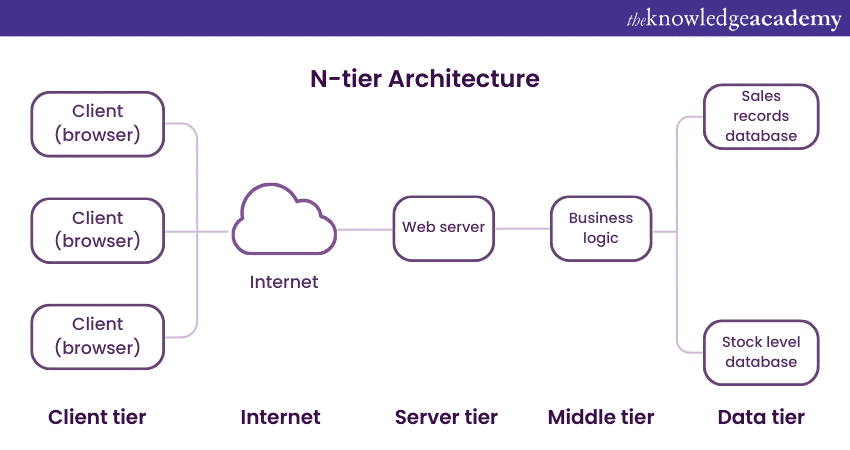


**Three-tier Architecture**

A more complex client-server setup with an intermediate layer (usually middleware or an application server) that handles business logic, acting as a bridge between the client and server. The client deals with the user interface (UI), while the server manages data storage. An example is an online banking system where the client is a web browser, the middleware checks transactions, and the server stores account data. This architecture improves scalability, performance, and security but increases complexity and cost.



**N-tier Architecture**

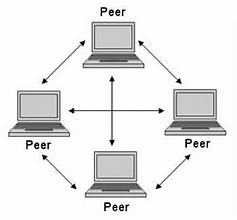


A more flexible Client-Server Architecture with more than three tiers, allowing greater scalability, flexibility, and modularity. Each tier can be distributed across different machines or networks and updated independently. An example is an e-commerce system with a web browser displaying the product catalog, a web server handling HTTP requests, an application server processing business logic, and a database storing product information. While suitable for complex systems, it requires more resources and management.

In addition to these types, there are also some variations or subtypes of Client-Server Architecture, such as:

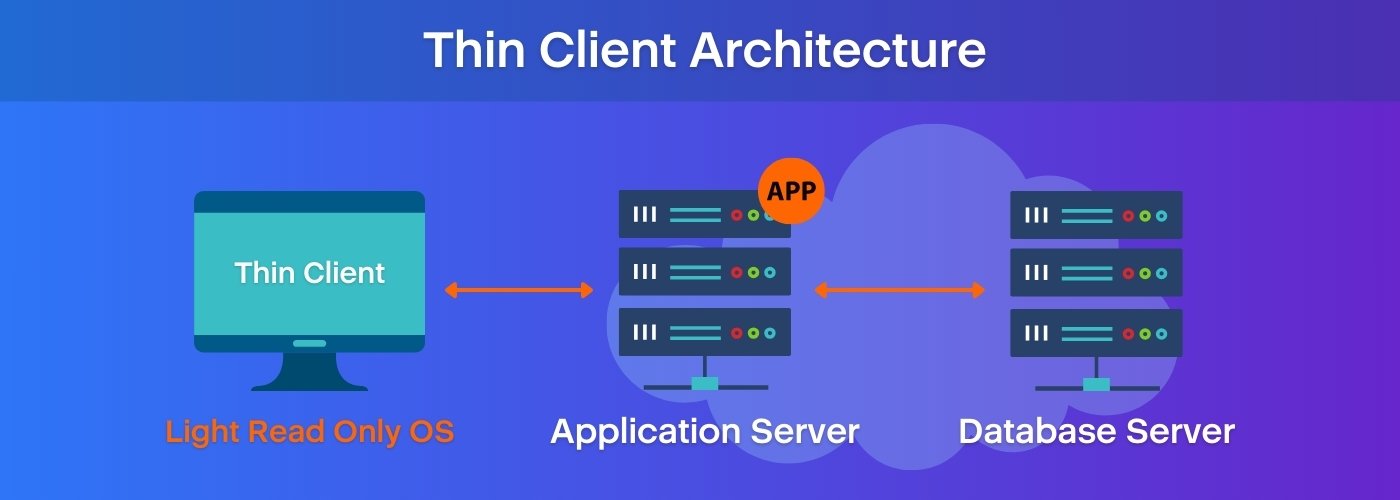
**Peer-to-peer Architecture**

This Client-Server Architecture allows each node to act as both a client and a server, with no central authority or hierarchy. Nodes can directly request or provide services to each other. An example is BitTorrent, a file-sharing system where nodes can download or upload files from each other without a central server, enhancing reliability, availability, and [resilience](https://www.theknowledgeacademy.com/blog/what-is-resilience/). However, it presents challenges like security, coordination, and quality control.



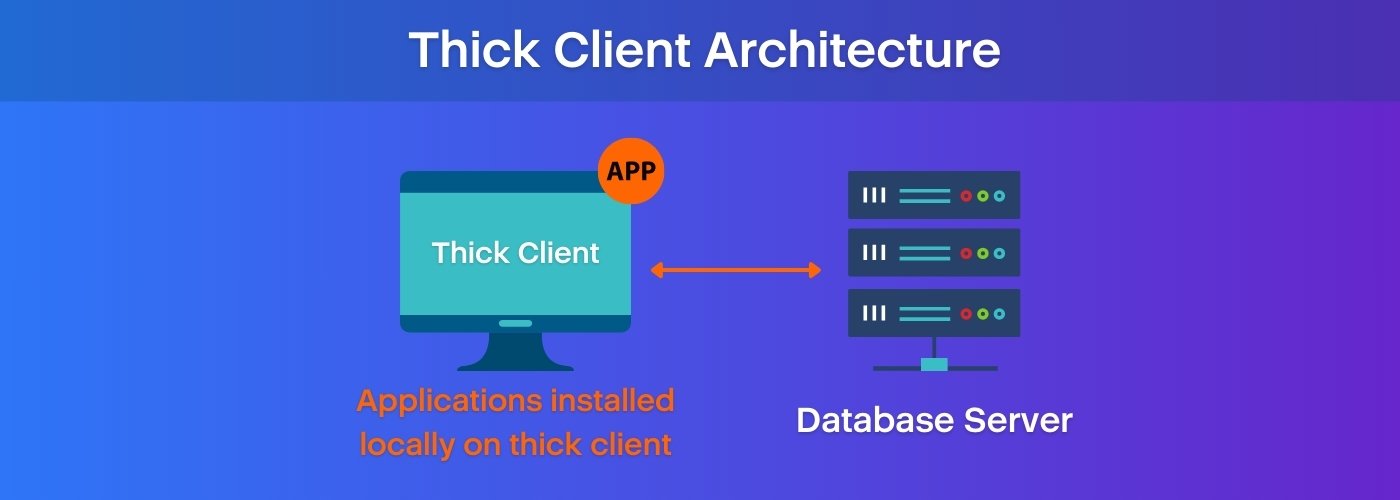
**Thin-client Architecture**

In this architecture, the client has minimal functionality, relying on the server for processing and data storage. The client handles only the user interface (UI) and sends inputs to the server, which performs all computations and returns the results. An example is Google Docs, where the client is a web browser accessing the document editor while the server stores and updates data in the cloud. This architecture reduces client-side hardware costs, maintenance, and security risks but increases network dependency, bandwidth usage, and server load.



**Fat-client Architecture**

In a fat-client setup, the client is highly functional, performing most processing and data storage locally, and only occasionally communicates with the server for synchronisation or backup. An example is an offline[mobile app](https://www.theknowledgeacademy.com/blog/what-is-a-mobile-app/) like Evernote, where the client runs the app and stores notes locally, syncing with the cloud when connected. This architecture boosts performance, responsiveness, and availability on the client side but increases hardware requirements, software complexity, and risks of data inconsistency.



**Advantages and disadvantages of Client-Server Architecture**

Client-Server Architecture has some advantages and disadvantages, depending on the requirements and constraints of the system.

**Advantages**

Some of the advantages are:

1) It is centralised, which means that all data and services are stored and managed in a single place. This makes it easier to maintain, update, and secure the system.

2) It is cost-efficient, as it requires less hardware and software resources for the client side. The client only needs a network connection and an application or web browser to access the server.

3) It has high performance and low latency, as the server can handle many requests from many clients simultaneously and efficiently.

**Disadvantages**

Some of the disadvantages are:

1) It has limited scalability, as it depends on the capacity and availability of the server. If the server is overloaded or fails, the system may not function properly or at all.

2) It has high network dependency, as it relies on the network connection between the client and the server. If the network is slow or disrupted, the system may experience delays or errors.

3) It has complex architecture, as it involves multiple components and layers that need to be designed, implemented, and coordinated. The system may also face challenges such as security, synchronisation, and compatibility.

**Conclusion**

Client Server Architecture is a powerful and flexible way of managing network resources and applications. It allows IT professionals to handle the challenges of a dynamic and competitive environment, by distributing the workload and responsibilities between clients and servers. Client-Server Architecture also enables the network's scalability, security, reliability, and performance. It is a vital concept for any IT professional to understand and master.

--Rachakonda Sushmitha