1

Introduction

1.1 WHAT IS CLIENT/SERVER COMPUTING?

According to MIS terminology, Client/Server computing is new technology that yields solutions to many data management problems faced by modern organizations. The term Client/Server is used to describe a computing model for the development of computerized systems. This model is based on distribution of functions between two types of independent and autonomous processes: Server and Client. A Client is any process that requests specific services from the server process. A Server is a process that provides requested services for the Client. Client and Server processes can reside in same computer or in different computers linked by a network.

When Client and Server processes reside on two or more independent computers on a network, the Server can provide services for more than one Client. In addition, a client can request services from several servers on the network without regard to the location or the physical characteristics of the computer in which the Server process resides. The network ties the server and client together, providing the medium through which the clients and the server communicate. The Fig. 1.1 given below shows a basic Client/Server computing model.

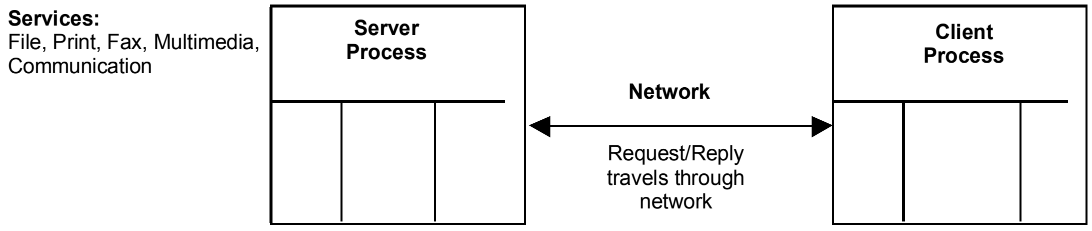

Services:
File, Print, Fax, Multimedia, Communication
Server Process
Network
Request/Reply travels through network
Client Process

Fig.1.1: Basic Client/Server Computing Model

From the Fig. 1.1 it is clear that services can be provided by variety of computers in the network. The key point to Client/Server power is where the request processing takes place. For example: Client/Server Database. In case of Client/Server database system, the functionality is split between the server system and multiple clients such that networking of computers allows some tasks to be executed on the client system.

1.1.1 A Server for Every Client

A file server can store any type of data, and so on simpler systems, may be the only server necessary. On larger and more complicated systems, the server responsibility may be distributed among several different types of servers. In this section, we have discussed the purpose of various available server:

File Server

All the files reside on the server machine. File Server provides clients access to records within files from the server machine. File Servers are useful for sharing files across a network among the different client process requesting the services. The server process is somewhat primitive because of tends to demand many message exchanges over the network to find the requested data.

The examples of File servers are:

- UNIX: Network File Services (NFS) created by Sun Micro systems.
- Microsoft Windows "Map Drive" e.g., Rivier College's "P-drive".
- Samba: An open Source/Free Software suite that provides seamless file and print services to SMB/CIFS clients (i.e., Microsoft Windows clients).

Print Server

This machine manages user access to the shared output devices, such as printers. These are the earliest type of servers. Print services can run on a file server or on one or more separate print server machines.

Application Server

This machine manages access to centralized application software; for example, a shared database. When the user requests information from the database, the application server processes the request and returns the result of the process to the user.

Mail Server

This machine manages the flow of electronic mail, messaging, and communication with mainframe systems on large-scale networks.

Fax Server

Provides the facility to send and receive the Faxes through a single network connection. The Fax server can be a workstation with an installed FAX board and special software or a specialized device dedicated and designed for Fax Services. This machine manages flow of fax information to and from the network. It is similar to the mail server.

Directory Services Server

It is found on large-scale systems with data that is distributed throughout multiple servers. This machine functions as an organization manager, keeping track of what is stored where, enabling fast and reliable access to data in various locations.

Web Server

This machine stores and retrieves Internet (and intranet) data for the enterprise. Some documents, data, etc., reside on web servers. Web application provides access to documents and other data. "Thin" clients typically use a web browser to request those documents. Such servers shares documents across intranets, or across the Internet (or extranets). The most commonly used protocol is HTTP (Hyper Text Transfer Protocol). Web application servers are now augmenting simple web servers. The examples of web application servers are Microsoft's Internet Information Server (IIS), Netscape's iPlanet IBM's WebSphere, BEA's WebLogic and Oracle Application Server.

Database Server

Data resides on server, in the form of a SQL database. Database server provides access to data to clients, in response to SQL requests. It shares the data residing in a database across a network. Database Server has more efficient protocol than File Server. The Database Server receives SQL requests and processes them and returning only the requested data; therefore the client doesn't have to deal with irrelevant data. However, the client does have to implement SQL application code. The example of database server is: Oracle9i database server.

Transaction Servers

The data and remote procedures reside on the server. The Server provides access to high-level functions, and implements efficient transaction processing. It shares data and high-level functions across a network. Transaction servers are often used to implement Online Transaction Processing (OLTP) in high-performance applications. A transaction server utilizes a more efficient protocol in comparison to a Database Server. The transaction Server receives high-level function request from the clients and it implements that function. Often it needs to return less information to the client than a Database Server. Examples of the Transaction servers mainly categorized as

- TP-Light with Database Stored Procedures like Oracle, Microsoft SQL Server etc.
- TP-Heavy with TP Monitors like BEA Tuxedo, IBM CICS/TX Series.

Groupware Servers

Liable to store semi-structured information like text, image, mail, bulletin boards, flow of work. Groupware Server provides services, which put people in contact with other people, that is because "groupware" is an ill-defined classification protocol differing from product to product. For Example: Lotus Notes/Domino and Microsoft Exchange.

Object Application Servers

Communicating distributed objects reside on the server. The object server primarily provides access to those objects from the designated client objects. The object Application Servers are responsible for sharing distributed objects across the network. Object Application Servers use the protocols that are usually some kind of Object Request Broker (ORB). Each distributed object can have one or more remote methods. ORB locates an instance of the object server class, invokes the requested method, and returns the results to the client object. Object Application Server provides an ORB and application servers to implement this. For example:

- Common Object Request Broker Architecture (CORBA): Iona's Orbix, Borland's Visibroker.
- Microsoft's Distributed Component Object Model (DCOM), aka COM+.
- Microsoft Transaction Server (MTS).

1.1.2 Client/Server: Fat or Thin

A Client or a Server is so named depending on the extent to which the processing is shared between the client and server. A thin client is one that conducts a minimum of processing on the client side while a fat client is one that carries a relatively larger proportion of processing load. The concept of Fat Clients or Fat Servers is given by one of the important criterion, that is, how much of an application is placed at the client end vs. the server end.

Fat Clients: This architecture places more application functionality *on the client machine(s).* They are used in traditional of Client/Server models. Their use can be a maintenance headache for Client/Server systems.

Fat Servers: This architecture places more application functionality *on the server machine(s).* Typically, the server provides more abstract, higher level services. The current trend is more towards fat servers in Client/Server Systems. In that case, the client is often found using a fast web browser. The biggest advantage of using the fat server is that it is easier to manage because only the software on the servers needs to be changed, whereas updating potentially thousands of client machines is a real headache.

1.1.3 Client/Server: Stateless or Stateful

A stateless server is a server that treats each request as an independent transaction that is unrelated to any previous request. The biggest advantage of stateless is that it simplifies the server design because it does not need to dynamically allocate storage to deal with conversations in progress or worry about freeing it if a client dies in mid-transaction. There is also one disadvantage that it may be necessary to include more information in each request and this extra information will need to be interpreted by the server each time. An example of a stateless server is a World Wide Web server. With the exception of cookies, these take in requests (URLs) which completely specify the required document and do not require any context or memory of previous requests contrast this with a traditional FTP server which conducts an interactive session with the user. A request to the server for a file can assume that the user has been authenticated and that the current directory and file transfer mode have been set. The Gopher protocol and Gopher+ are both designed to be stateless.

Stateful Server

Client data (state) information are maintained by server on status of ongoing interaction with clients and the server remembers what client requested previously and at last maintains the information as an incremental reply for each request.

The advantages of stateful server is that requests are more efficiently handled and are of smaller in size. Some disadvantages are their like state information becomes invalid when messages are unreliable. Another disadvantage is that if clients crash (or reboot) frequently, state information may exhaust server's memory. The best example of stateful server is remote file server.

Stateless vs Stateful Servers

There are some comparative analysis about stateless and stateful servers.

* A stateful server remembers client data (state) from one request to the next.
* A stateless server keeps no state information. Using a stateless file server, the client must specify complete file names in each request specify location for reading or writing and re-authenticate for each request.
* Using a stateful file server, the client can send less data with each request. A stateful server is simpler.

On the other hand, a stateless server is more robust and lost connections can't leave a file in an invalid state rebooting the server does not lose state information rebooting the client does not confuse a stateless server.

1.1.4 Servers and Mainframes

From a hardware perspective, a mainframe is not greatly different from a personal computer. The CPU inside a mainframe was, however, much faster than a personal computer. In fact, what a mainframe most closely resembled was a LAN. A mainframe was 'larger' in terms of:

* The raw speed expressed in instructions per second, or cycles.
* The amount of memory that could be addressed directly by a program.

Mainframes are the monstrous computer system that deals mainly the business functions and technically these giant machines will run MVS, IMS and VSAM operating systems. There is a common believe that a mainframe is 'database'. There are many reasons behind this belief:

* Many servers are either file or database servers running sophisticated database such as Sybase, Oracle and DB2.

* These servers connect to the mainframe primarily to access databases.
* Organisations use servers specifically to replace mainframe databases.
* Organisations keep applications on the mainframe usually for better database performance, integrity and functionality.

Mainframe users argue that in the long run, a mainframe is at least as good a server as a PC, and perhaps even better. And because the mainframe portrayed as a better server than a PC, the picture is clear: PC servers and mainframe servers compete at the back-end both are essentially databases.

There is some controversy as to whether servers will eventually replace mainframes. They may, but not in the near future. Mainframes still serve the purpose in managing the complex business rules of very large organizations and enterprises that are spread out over a very large area. But the increasing processing power of servers combined with their lower costs makes them the logical replacement to mainframe-based systems in the future.

In the meanwhile, Client/Server networks will often find it necessary to connect to mainframe-based systems. This is because some data can only be found in the mainframe environment, usually because the business rules for handling it are sufficiently complex or because the data itself is massive or sensitive enough that as a practical matter it remains stored there.

Connection to a mainframe requires some form of network – like access. Even if you are using a telephone and modem as your access hardware, you still require special software to make your workstation appear to the mainframe to be just another network terminal. Many vendors can provide the necessary software to handle this type of network extension.

A very natural question at this stage is: How do Client/Server Systems differ from Mainframe Systems?

The extent of the separation of data processing task is the key difference.

In mainframe systems all the processing takes place on the mainframe and usually dumb terminals are used to display the data screens. These terminals do not have autonomy.

On the other hand, the Client/Server environment provides a clear separation of server and client processes, both processes being autonomous. The relationship between client and server is many to many.

Various other factors, which can have, prime considerations to differentiate the mainframe and Client/Server systems:

- **Application development:** Mainframe systems are over structured, time-consuming and create application backlogs. On the other hand, PC-based Client/Server systems are flexible, have rapid application development and have better productivity tools.
- **Data manipulation:** Mainframe systems have very limited data manipulation capabilities whereas these techniques are very flexible in the case of Client/Server systems.

- **System management:** Mainframe systems are known to be integrated systems but in the case of Client/Server systems only few tools are available for system management.
- **Security:** Mainframe systems are highly centralized whether as Client/Server systems are relaxed or decentralized.
- **End user platform:** Mainframe systems comprise of dumb terminals, are character-based, single task oriented and of limited productivity. On the other hand, Client/Server systems are intelligent PC's with graphical user interface having multitasking OS with better productivity tools.

1.1.5 Client/Server Functions

The main operations of the client system are listed below:

- Managing the user interface.
- Accepts and checks the syntax of user inputs.
- Processes application logic.
- Generates database request and transmits to server.
- Passes response back to server.

The main operations of the server are listed below:

- Accepts and processes database requests from client.
- Checks authorization.
- Ensures that integrity constraints are not violated.
- Performs query/update processing and transmits responses to client.
- Maintains system catalogue.
- Provide concurrent database access.
- Provides recovery control.

1.1.6 Client/Server Topologies

A Client/Server topology refers to the physical layout of the Client/Server network in which all the clients and servers are connected to each other. This includes all the workstations (clients) and the servers. The possible Client/Server topological design and strategies used are as follows:

(*i*) Single client, single server

(*ii*) Multiple clients, single server

(*iii*) Multiple clients, multiple servers

(*i*) **Single client, single server:** This topology is shown in the Fig. 1.2 given below. In this topology, one client is directly connected to one server.

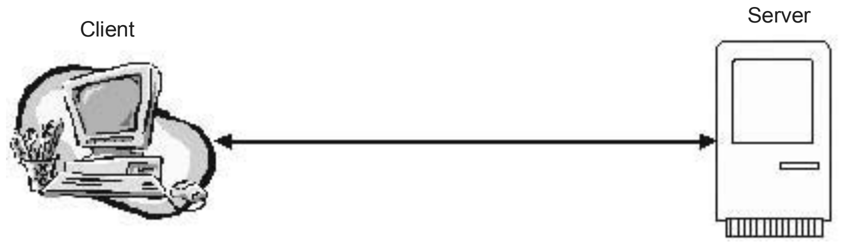


Fig.1.2: Single Client, Single Server

(*ii*) **Multiple clients, single server:** This topology is shown in the Fig. 1.3 given below. In this topology, several clients are directly connected to only one server.

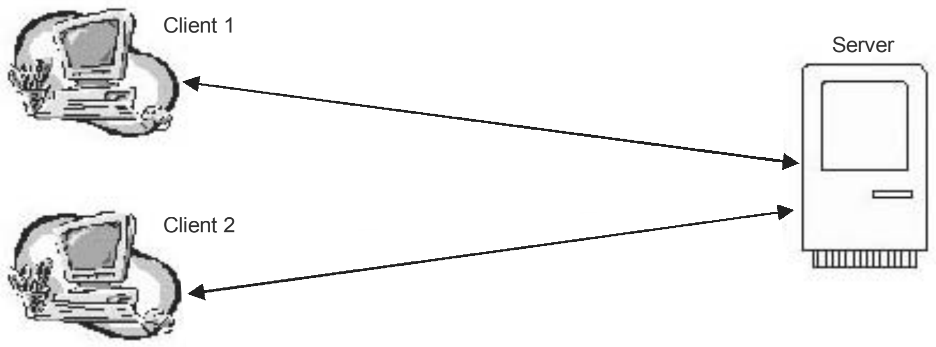


Fig.1.3: Multiple Clients, Single Server

(*iii*) **Multiple clients, multiple servers:** This topology is shown in the following Fig. 1.4 In this topology several clients are connected to several servers.

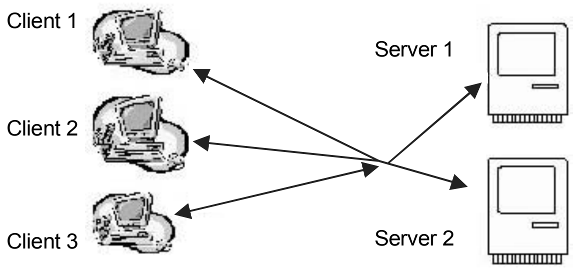


Fig.1.4: Multiple Clients, Multiple Servers

1.1.7 Integration with Distributed Computing

Distributed computing is the term used for implementation of technologies across heterogeneous environments. For operating systems, heterogeneous computing means the ability to communicate with other operating systems and protocols. Distributed computing is a complex architecture. It involves rearchitecture of applications, redevelopment of systems and increased efficiency in maintaining a network as a whole. Many distributed nodes work on behalf of one requesting client. This makes the system fault tolerant and decentralized, which is an obvious advantage over centralized systems. For the technology to become effective and revolutionary, developers of distributed applications have to do everything possible to minimize the complexity of development and maintenance and integrate their software with disparate platforms. Client/Server application designing necessitates the modularization of applications and their functions into discrete components. These components must be bounded only by encapsulated data and functions that may be moved between the systems. This design model gives Client/Server software more adaptability and flexibility.

1.1.8 Alternatives to Client/Server Systems

There are various client/server projects are running in industry by various companies. Before committing a project to Client/Server, some alternatives can be considered that includes:

- Movement of an existing mainframe application to a smaller hardware platforms, for examples IBM's ICCS transaction processing to an AS/400 or an OS/2 LAN Server.
- Replacement of mainframes computer terminals with PCs that is able to emulate terminals.
- Replacing an existing mainframe system with a packaged system that does the job better.
- Beautifying an existing mainframe application by adding a GUI front-end to it. There are programs available specifically to do this.

1.2 CLASSIFICATION OF CLIENT/SERVER SYSTEMS

Broadly, there are three types of Client/Server systems in existence.

(*i*) Two-tier
(*ii*) Three-tier
(*iii*) N-Tier

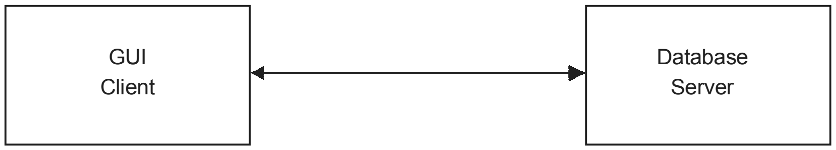
1.2.1 Two-tier Client/Server Model

The application processing is done separately for database queries and updates and for business logic processing and user interface presentation. Usually, the network binds the back-end of an application to the front-end, although both tiers can be present on the same hardware.

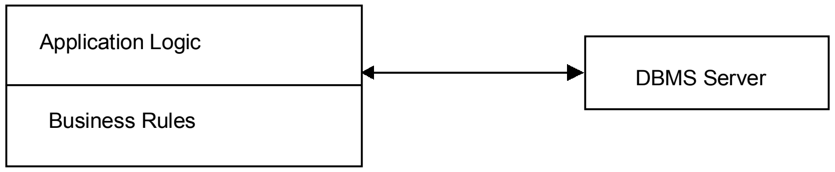
Sometimes, the application logic (the real business logic) is located in both the client program and in the database itself. Quiet often, the business logic is merged into the presentation logic on the client side. As a result, code maintenance and reusability become difficult to achieve on the client side. On the database side, logic is often developed using stored procedures.

In the two-tier architecture, if the Client/Server application has a number of business rules needed to be processed, then those rules can reside at either the Client or at the Server. The Fig. 1.5 below clarifies this situation.

(*a*) Centralized Two-tier Representation:



(*b*) Business Rules Residing on the Client:



(*c*) Business Rules Residing on the Server:

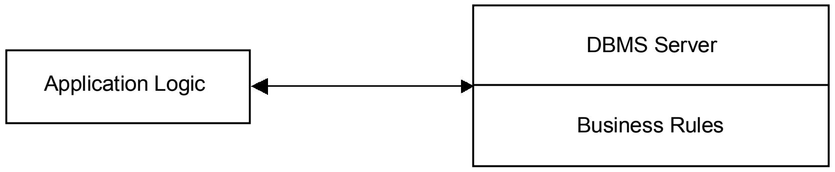


Fig.1.5: The Two-tier Approach Illustrated

The architecture of any client/server environment is by definition at least a two-tier system, the client being the first tier and the server being the second.

The Client requests services directly from server i.e. client communicates directly with the server without the help of another server or server process. The Fig. 1.6 (at the end of this section) illustrates a two-tier Client/Server model.

In a typical two-tier implementation, SQL statements are issued by the application and then handed on by the driver to the database for execution. The results are then sent back via the same mechanism, but in the reverse direction. It is the responsibility of the driver (ODBC) to present the SQL statement to the database in a form that the database understands.

There are several advantages of two-tier systems:

- Availability of well-integrated PC-based tools like, Power Builder, MS Access, 4 GL tools provided by the RDBMS manufacturer, remote SQL, ODBC.
- Tools are relatively inexpensive.
- Least complicated to implement.
- PC-based tools show Rapid Application Development (RAD) i.e., the application can be developed in a comparatively short time.
- The 2-tier Client/Server provides much more attractive graphical user interface (GUI) applications than was possible with earlier technology.

- Architecture maintains a persistent connection between the client and database, thereby eliminating overhead associated with the opening and closing of connections.
- Faster than three-tier implementation.
- Offers a great deal of flexibility and simplicity in management.

Conversely, a two-tier architecture has some disadvantages:

- As the application development is done on client side, maintenance cost of application, as well as client side tools etc. is expensive. That is why in 2-tier architecture the client is called 'fat client'.
- Increased network load: Since actual processing of data takes on the remote client, the data has to be transported over the network. This leads to the increased network stress.
- Applications are loaded on individual PC i.e. each application is bound to an individual PC. For this reason, the application logic cannot be reused.
- Due to dynamic business scenario, business processes/logic have to be changed. These changed processes have to be implemented in all individual PCs. Not only that, the programs have to undergo quality control to check whether all the programs generate the same result or not.
- Software distribution procedure is complicated in 2-tier Client/Server model. As all the application logic is executed on the PCs, all these machine have to be updated in case of a new release. The procedure is complicated, expensive, prone to errors and time consuming.
- PCs are considered to be weak in terms of security i.e., they are relatively easy to crack.
- Most currently available drivers require that native libraries be loaded on a client machine.
- Load configurations must be maintained for native code if required by the driver.
- Problem areas are encountered upon implementing this architecture on the Internet.

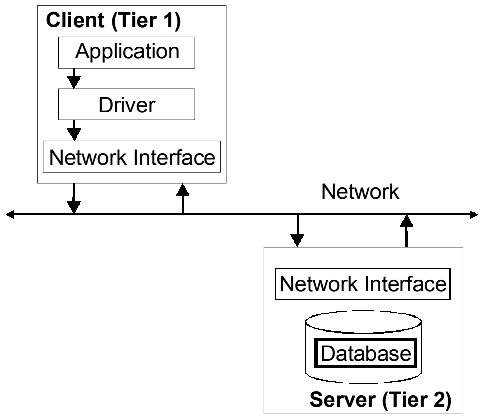


Fig. 1.6: Two-tier Client/Server Model

1.2.2 Three-tier Client/Server Model

Reusability is hard to achieve if pieces of business logic must be distributed across systems and several databases are involved. To avoid embedding the application's logic at both the database side and the client side, a third software tier is inserted in between. In the three-tier architecture, most of the business logic is located in the middle tier (here business logic is encapsulated as a component in a separate tier). In this structure, when the business activity or business rules change, only the middle tier must be modified.

In three-tier architecture application responsibilities are divided into three logical categories (in other words, the business system must provide three types of main services).

- **Presentation (GUI) or user services:** Include maintaining the graphical user interface and generating what users see on the monitor. Presentation Logic dealing with:
  - Screen formatting
  - Windows management
  - Input editing
  - What-if analysis
- **Application services or business rules:** These include executing applications and controlling program flow. Business logic dealing with:
  - Domain and range validation
  - Data dependency validation
  - Request/response architecture of Inter Process Communication level
- **Database services or data server:** Which refers to the management of underlying databases. Server logic deals with:
  - Data access
  - Data management
  - Data security
  - SQL parsing

Based on these three components, the three-tier architecture of Client/Server system is shown in fig. 1.8 below. In three-tier model, a third server is employed to handle requests from the client and then pass them off to the database server. The third server acts as proxy for all client requests. Or, in other words we can say:

"In three-tier client/server system the client request are handled by intermediate servers which coordinate the execution of the client request with subordinate servers."

All client requests for the database are routed through the proxy server, thereby creating a more secure environment for your database.

In two-tier environment, we can say that the client uses a driver to translate the client's request into a database native library call. In a three-tier environment, the driver translates the request into a "network" protocol and then makes a request via the proxy server. Figure 1.8 represents the three-tier Client/Server model.

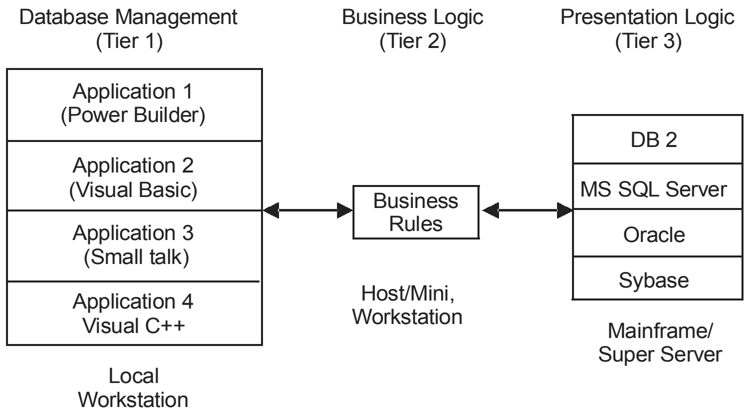


Fig.1.7: Three-tier Architecture

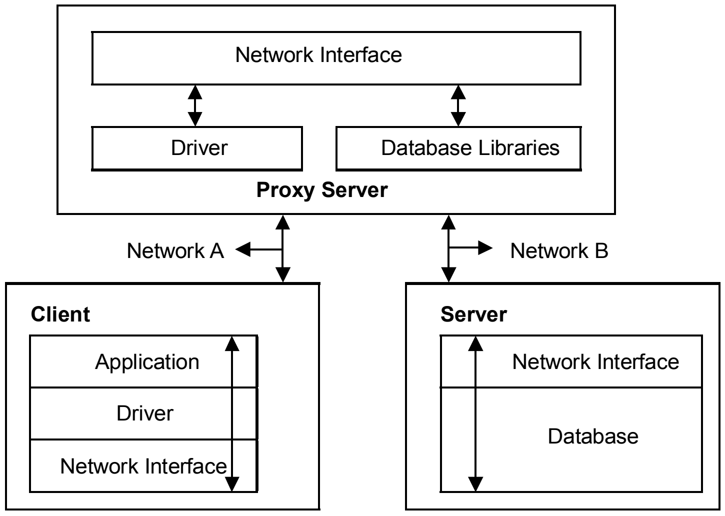


Fig.1.8: Three-tier Client/Server Model

The proxy server makes the database request on behalf of the client and passes the results back after they have been serviced by the database. This approach eliminates the need for DBMS to be located on the same server. There are a couple of drawbacks to this model. One is that it requires that a small server process (listener) be set up on the middle server. Secondly, it requires all your client requests be transmitted into a "network" protocol.

First-tier (client-tier): The main responsibility of this tier is to receive user events and to control the user interface and presentation of data. As most of the software is removed from the client, the client is called "Thin Client". Mainly browser and presentation code resides on this tier.

Second-tier (application-server-tier): The complex application logic is loaded here and available to the client tier on request from client. This level forms the central key towards solving the 2-tier problem. This tier can protect direct access of data. Object oriented analysis aims in this tier to record and abstract business processing in business projects. This way it is possible to map this tier directly from the case tools that support object oriented analysis.

Three-tier (database-server-tier): This tier is responsible for data storage. This server mostly operates on a relational database.

The boundaries between tiers are logical. One can run 3-tiers in one and the same machine. The important fact is that the system is neatly structured and well-planned definitions of the software boundaries exist between the different tiers. Some of the advantages of using three-tier model include:

- Application maintenance is centralized with the transfer of the business logic for many end users into a single application server. This eliminates the concern of software distribution that are problematic in the traditional two-tier Client/Server model.
- Clear separation of user-interface-control and data presentation from application-logic. Through this separation more clients are able to have access to a wide variety of server applications. The two main advantages for client-applications are clear: quicker development through the reuse of pre-built business-logic components and a shorter test phase, because the server-components have already been tested.
- Many users are able to access a wide variety of server applications, as all application logic are loaded in the applications server.
- As a rule servers are "trusted" systems. Their authorization is simpler than that of thousands of "untrusted" client-PCs. Data protection and security is simpler to obtain. Therefore, it makes sense to run critical business processes that work with security sensitive data, on the server.
- Redefinition of the storage strategy won't influence the clients. RDBMS' offer a certain independence from storage details for the clients. However, cases like changing table attributes make it necessary to adapt the client's application. In the future, even radical changes, like switching from an RDBMS to an OODBMS, won't influence the client. In well-designed systems, the client still accesses data over a stable and well-designed interface, which encapsulates all the storage details.
- Load balancing is easier with the separation of the core business logic from the database server.
- Dynamic load balancing: if bottlenecks in terms of performance occur, the server process can be moved to other servers at runtime.
- Business objects and data storage should be brought as close together as possible. Ideally, they should be together physically on the same server. This way network load for complex access can be reduced.
- The need for less expensive hardware because the client is 'thin'.
- Change management is easier and faster to execute. This is because a component/ program logic/business logic is implemented on the server rather than furnishing numerous PCs with new program versions.

- The added modularity makes it easier to modify or replace one tier without affecting the other tier.
- Clients do not need to have native libraries loaded locally.
- Drivers can be managed centrally.
- Your database server does not have to be directly visible to the Internet.

An additional advantage is that the three-tier architecture maps quite naturally to the Web environment, with a Web browser acting as the 'thin' client, and a Web server acting as the application server. The three-tier architecture can be easily extended to N-tier, with additional tiers added to provide more flexibility and scalability. For example, the middle tier of the three-tier architecture could be split into two, with one tier for the Web server and another for the application server. Some disadvantages are:

- The client does not maintain a persistent database connection.
- A separate proxy server may be required.
- The network protocol used by the driver may be proprietary.
- You may see increased network traffic if a separate proxy server is used.

1.2.2.1 Transaction Processing Monitors

It is the extension of the two-tier Client/Server architecture that splits the functionality of the 'fat' client into two. In the three-tier Client/Server architecture, the 'thin' client handles the user interface only whereas the middle layer handles the application logic. The third layer is still a database server. This three-tier architecture has proved popular in more environments, such as the Internet and company Intranets where a Web browser can be used as a client. It is also an important architecture for TPM.

A Transaction Processing Monitor is a program that controls data transfer between client and server in order to provide consistent environment, particularly for online transaction processing (OLTP).

Complex applications are often built on top of several resource managers (such as DBMS, operating system, user interface, and messaging software). A Transaction Processing Monitor or TP Monitor is a middleware component that provides access to the services of a number of resource managers and provides a uniform interface for programmers who are developing transactional software. Figure 1.9 illustrates how a TP Monitor forms the middle tier of three-tier architecture. The advantages associated with TP Monitors are as given below:

Transaction Routing: TP monitor can increase scalability by directing transactions to specific DBMS's.

Managing Distributed Transaction: The TP Monitor can manage transactions that require access to data held in multiple, possibly heterogeneous, DBMSs. For example, a transaction may require to update data item held in an oracle DBMS at site 1, an Informix DBMS at site 2, and an IMS DBMS at site 3. TP Monitors normally control transactions using the X/Open Distributed transaction processing (DTP) standards. A DBMS that support this standard can function as a resource manager under the control of a TP Monitor acting as a transaction manager.

Load balancing: The TP Monitor can balance client requests across multiple DBMS's on one or more computers by directing client services calls to the least loaded server. In addition, it can dynamically bring in additional DBMSs as required to provide the necessary performance.

Funneling: In an environment with a large number of users, it may sometimes be difficult for all users to be logged on simultaneously to the DBMS. Instead of each user connecting to the DBMS, the TP Monitor can establish connections with DBMS's as and when required, and can funnel user requests through these connections. This allows a large number of users to access the available DBMSs with a potentially smaller number of connections, which in turn would mean less resource usages.

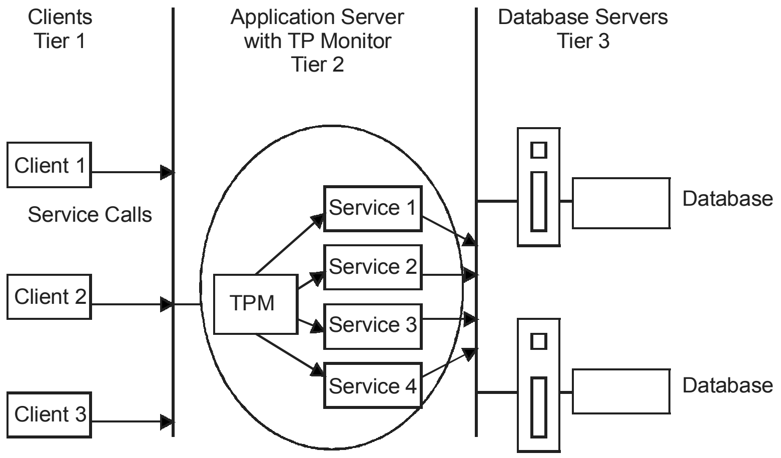


Fig.1.9: Middleware Component of TPM

Increased reliability: The TP Monitor acts as transaction manager, performing the necessary action to maintain the consistency of database, with the DBMS acting as a resource manager. If the DBMS fails, the TP Monitor may be able to resubmit the transaction to another DBMS or can hold the transaction until the DBMS becomes available again.

A TP Monitor is typically used in environments with a very heavy volume of transaction, where the TP Monitor can be used to offload processes from the DBMS server. Prominent examples of TP Monitors include CICS and Encina from IBM (which are primarily used on IBM AIX or Windows NT and bundled now in the IBM TXSeries) and Tuxido from BEA system.

1.2.2.2 Three-tier with Message Server

Messaging is another way to implement the three-tier architecture. Messages are prioritized and processed asynchronously. Messages consist of headers that contain priority information, and the address and identification number. The message server connects to the relational DBMS and other data sources. The difference between the TP monitor technology and the message server is that the message server architecture focuses on intelligent messages, whereas the TP Monitor environment has the intelligence in the monitor, and treats transactions as dumb data packets. Messaging systems are good solutions for wireless infrastructures.

1.2.2.3 Three-tier with an Application Server

The three-tier application server architecture allocates the main body of an application to run on a shared host rather than in the user system interface client environment. The application server does not drive the GUIs; rather it shares business logic, computations, and a data retrieval engine. Advantages are that with less software on the client there is less security to worry about, applications are more scalable, and support and installation costs are less on a single server than maintaining each on a desktop client. The application server design should be used when security, scalability, and cost are major considerations.

1.2.2.4 Three-tier with an ORB Architecture

Currently, work is going on in the industry towards developing standards to improve interoperability and determine what the common Object Request Broker (ORB) will be. Developing client/server systems using technologies that support distributed objects holds great promise, as these technologies support interoperability across languages and platforms, as well as enhancing maintainability and adaptability of the system. There are two-prominent distributed object technologies at present:

- Common Object Request Broker Architecture (CORBA).
- COM/DCOM (Component Object Model/Distributed Component Object Model).

Standards are being developed to improve interoperability between CORBA and COM/DCOM. The Object Management Group (OMG) has developed a mapping between CORBA and COM/DCOM that is supported by several products.

Distributed/collaborative enterprise architecture: The distributed/collaborative enterprise architecture emerged in 1993. This software architecture is based on Object Request Broker (ORB) technology, but goes further than the Common Object Request Broker Architecture (CORBA) by using shared, reusable business models (not just objects) on an enterprise-wide scale. The benefit of this architectural approach is that standardized business object models and distributed object computing are combined to give an organization flexibility to improve effectiveness organizationally, operationally, and technologically. An enterprise is defined here as a system comprised of multiple business systems or subsystems. Distributed/collaborative enterprise architectures are limited by a lack of commercially-available object orientation analysis and design method tools that focus on applications.

1.2.2.5 Three-tier Architecture and Internet

With the rapid development of Internet and web technology, Client/Server applications running over Internets and Intranets are becoming a new type of distributed computing. A typical web application uses the following 3-tier architecture.

- The user interface runs on the desktop as client.
- The client is connected (through one or more immediate server links) to a web server, which may be a storehouse for downloadable applets (Software components).
- This web server is, in turn, is supported by a database server which keeps track of information specific to the client interest and history.

These web applications rely on Internet standards (HTTP, HTML, XML etc.) as well as distributed objects programming languages.

1.2.3 N-tier Client/Server Model

N-tier computing obliges developer to design components according to a business schema that represents entities, relationship, activities roles, and rules, thereby enabling them to distribute functionality across logical and physical tiers, allowing better utilization of hardware and platform resources, as well as sharing of those resources and the components that they support to serve several large applications at the same time.

Another aspect of splitting tiers is that application developers and administrators are able to identify bottlenecks and throw hardware at them to enable load-balancing and fail-over of certain nodes. The splitting may be between application logic components, security logic, and presentation logic, computational-intensive and I/O-intensive components and so on. The most common approach used when designing N-tier system is the three-tier architecture. Three-tier and N-tier notations are similar, although N-tier architecture provides finer-grained layers. Architectures often decide to layout much more than three -tiers to deploy services (An infrastructure that supports three-tier is often made of several machines and services whose functionalities aren't part of the three-tier design).

Figure 1.10 shown below depicts the N-tier architecture.

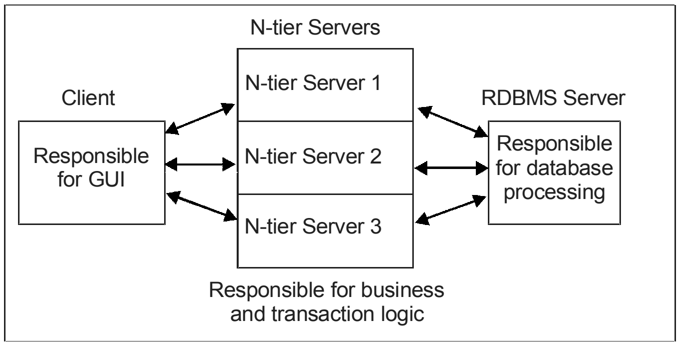


Fig.1.10: N-tier Architecture

N-tier computing provides many advantages over traditional two-tier or single-tier design, which includes the following:

- Overall performance has been improved.
- The business logic is centralized.
- Enhanced security level is attained.

An alternative to N-tier computing includes *fat server/fat client.* A *fat server* locates business logic within the RDBMS on the server. The client issues remote procedure calls to the server to execute the process. Fat servers are the best suited for structured and consistent business logic, such as online transaction processing (OLTP). Modern RDBMS products support fat servers through stored procedures, column rules, triggers, and other methods.

A *fat client* embeds business logic in the application at the client level. Although a fat client is more flexible than a fat server, it increases network traffic. The fat client approach is used when business logic is loosely structured or when it is too complicated to implement at the middle-tier level. Additionally, fat client development tools, such as 4GL languages, sometimes offer more robust programming features than do middle-tier programming tools. Decision support and ad-hoc systems are often fat client based.

1.3 CLIENTS/SERVER—ADVANTAGES AND DISADVANTAGES

1.3.1 Advantages

There are various advantages associated with Client/Server computing model.

(*i*) **Performance and reduced workload:** Processing is distributed among the client and server unlike the traditional PC database, the speed of DBMS is not tied to the speed of the workstation as the bulk of the database processing is done at the back-end. The workstation only has to be capable of running the front-end software, which extends the usable lifetime of older PC's. This also has the effect of reducing the load on the network that connects the workstation; instead of sending the entire database file back and forth on the wire, the network traffic is reduced to queries to and responses from the database server. Some database servers can even store and run procedures and queries on the server itself, reducing the traffic even more.

(*ii*) **Workstation independence:** Users are not limited to one type of system or platform. In an ORACLE-based Client/Server system the workstations can be IBM – compatible PCs, Macintoshes, UNIX workstations, or any combinations of the three. In addition, they can run any of a number of operating systems such as MS-DOS, Windows, IBM's OS/2, Apple's System 7 etc. That is, application independence is achieved as the workstations don't all need to use the same DBMS application software. Users can continue to use familiar software to access the database, and developers can design front-ends tailored to the workstation on which the software will run, or to the needs of the users running them.

(*iii*) **System interoperability:** Client/Server computing not only allows one component to be changed, it also makes it is possible for different type of components systems (client, network or server) to work together.

(*iv*) **Scalability:** The modular nature of the Client/Server system may be replaced without adversely affecting the rest of the system. For example, it is possible to upgrade the server to a more powerful machine with no visible changes to the end user. This ability to change component system makes Client/Server systems especially receptive to new technologies in both hardware and software.

(*v*) **Data integrity:** Client/Server system preserves the data integrity, DBMS can provide number of services that protect data like, encrypted file storage, real time backup (while the database is being accessed), disk mirroring (where the data is automatically written to duplicate database on another partition of same hard disk drive), disk duplexing (where the data is automatically written to a duplicate database on a different hard disk drive), transaction processing that keeps the track changes made to the database and corrects problems in case the server crashes. (Transaction processing is a method by which the DBMS keeps a running log of all the modifications made to the database over a period of time).

(*vi*) **Data accessibility (enhanced data sharing):** Since the server component holds most of data in a centralized location, multiple users can access and work on the data simultaneously.

(*vii*) **System administration (centralized management):** Client/Server environment is very manageable. Since data is centralized, data management can be centralized. Some of the system administration functions are security, data integrity and back up recovery.

(*viii*) **Integrated services:** In Client/Server model all information that the client is entitled to use is available at the desktop, through desktop interface, there is no need to change into a terminal mode or to logon into another processor to access information. The desktop tools – e-mail, spread sheet, presentation graphics, and word processing are available and can be used to deal with the information provided by application and database server's resident on the network. Desktop user can use their desktop tools in conjunction with information made available from the corporate systems to produce new and useful information using the facilities DDE/OLE, Object-oriented design.

(*ix*) **Sharing resources among diverse platforms:** Client/Server model provides opportunities to achieve open system computing. Applications can be created and implemented without much conversance with hardware and software. Thus, users may obtain client services and transparent access to the services provided by database, communications, and application servers. There are two ways for Client/Server application operation:

- They can provide data entry, storage, and reporting by using a distributed set of clients and servers.
- The existence of a mainframe host is totally masked from the workstation developer by the use of standard interface such as SQL.

(*x*) **Masked physical data access:** SQL is used for data access from database stored anywhere in the network, from the local PC, local server or WAN server, support with the developer and user using the same data request. The only noticeable difference may be performance degradation if the network bandwidth is inadequate. Data may be accessed from CD-ROM, HDD, Magnetic disk, and optical disk with same SQL statements. Logical tables can be accessed without any knowledge of the ordering of column. Several tables may be joined to create a new logical table for application program manipulation without regard to its physical storage format.

(*xi*) **Location independence of data processing:** Users log into an application from the desktop with no concern for the location or technology of the processors involved. In the current user centered word, the desktop provides the point of access to the workgroup and enterprise services without regard to the platform of application execution. Standard services such as login, security, navigation, help, and error recovery are provided consistently amongst all applications. Developers today are provided with considerable independence. Data is accessed through SQL without regard to the hardware or OS location providing the data. The developer of business logic deals with a standard process logic syntax without considering the physical platform.

(*xii*) **Reduced operating cost:** Computer hardware and software costs are on a continually downward spiral, which means that computing value is ever increasing. Client/Server computing offers a way to cash in on this bonanza by replacing expensive large systems with less expensive smaller ones networked together.

(*xiii*) **Reduced hardware cost:** Hardware costs may be reduced, as it is only the server that requires storage and processing power sufficient to store and manage the application.

(*xiv*) **Communication costs are reduced:** Applications carry out part of the operations on the client and send only request for database access across the network, resulting in less data being sent across the network.

1.3.2 Disadvantages

There are various disadvantages associated with the Client/Server computing model.

(*i*) **Maintenance cost:** Major disadvantages of Client/Server computing is the increased cost of administrative and support personnel to maintain the database server. In the case of a small network, the network administrator can usually handle the duties of maintaining the database server, controlling the user access to it, and supporting the front-end applications. However, the number of database server users rises, or as the database itself grows in size, it usually becomes necessary to hire a database administrator just to run the DBMS and support the front-ends.

(*ii*) **Training cost:** Training can also add to the start-up costs as the DBMS may run on an operating system that the support personnel are unfamiliar with.

(*iii*) **Hardware cost:** There is also an increase in hardware costs. While many of the Client/Server database run under the common operating systems (Netware, OS/2 and Unix) and most of the vendors claim that the DBMS can run on the same hardware side by side with the file server software. It usually makes sense from the performance and data integrity aspects to have the database server running on its own dedicated machine. This usually means purchasing a high-powered platform with a large amount of RAM and hard disk space.

(*iv*) **Software cost:** The overall cost of the software is usually higher than that of traditional PC based multi-user DBMS.

(*v*) **Complexity:** With so many different parts comprising the entire Client/Server, i.e., the more are the pieces, which comprise the system the more things that can go wrong or fail. It is also harder to pinpoint problems when the worst does occur and the system crashes. It can take longer to get everything set up and working in the first place. This is compounded by the general lack of experience and expertise of potential support personnel and programmers, due to the relative newness of the technology.

Making a change to the structure of database also has a ripple effect throughout the different front-ends. It becomes a longer and more complex process to make the necessary changes to the different front-end applications, and it is also harder to keep all of them in synchronization without seriously disrupting the user's access to the database.

1.4 MISCONCEPTIONS ABOUT CLIENT/SERVER COMPUTING

Client/Server technology can be stated to be an "architecture in which a system's functionality and its processing are divided between the client PC (Front-end) and database server (back-end)." This statement restricts the functionality of Client/Server software to mere retrieval and maintenance of data and creates many misconceptions regarding this technology, such as:

(*i*) Graphical user interface is supposed to be a necessity for presentation of application logic. As in the case of X-Windows graphical user interface, the implementation comprises both client and server components that may run on the same and different physical computers. An X-Windows uses Client/Server as architecture. This however, does not imply that Client/Server must use GUI. Client/Server logic remains independent of its presentation to the user.

(*ii*) Client/Server software is not always database centric. Client/Server computing does not require a database, although in today's computing environment Client/ Server is synonymous with databases. RDBMS packages are the most popular Client/Server applications. A major disadvantage of Client/Server technology is that it can be data centric and the developers can exploit these capabilities.

(*iii*) Client/Server technology does not provide code reuse. Tools that help create Client/Server applications may provide this benefit. Client/Server application build on component-based modeling enhanced code reusability.

(*iv*) Client/Server designing is not event-driven. Client/Server technology merges very well with event-driven systems but the latter are not a requirement for Client/Server. Client/Server application designing involves architecture of software that has innate features of portability with respect to communication pattern, and a well-build non-monolithic code with division of functionality and processing into different components.

EXERCISE 1

1. Client server is modular infrastructure, this is intended to improve Usability, Flexibility, Interoperability and Scalability. Explain each with an example, in each case how it helps to improve the functionality of client server architecture.
2. Explain the following.
   (*a*) Computing in client server architecture over Mainframe architecture has certain advantages and disadvantages. Describe atleast two advantages and disadvantages for each architecture.
   (*b*) Client/Server architecture could be explained as 2-tier architecture. Explain.
   (*c*) Explain the working of three-tier architecture with an application server.
   (*d*) How does the client server interaction work? Explain with a sketch.
   (*e*) Describe the server function and client responsibility in this architecture.
3. Differentiate between Stateful and Stateless servers.
4. Describe three-level schema architecture. Why do we need mapping between schema levels?
5. Differentiate between Transaction server and Data server system with example.
6. In client server architecture, what do you mean by Availability, Reliability, Serviceability and Security? Explain with examples.
7. How client/server computing environment is different from mainframe based computing environment?
8. In the online transaction processing environment, discuss how transaction processing monitor controls data transfer between client and server machines.