5

Client/Server Application Components

5.1 INTRODUCTION

A Client/Server application stand at a new threshold brought on by the exponential increase of low cost bandwidth on Wide Area Networks, for example, the Internet and CompuServe; and shows a new generation of network enabled, multi-threaded desktop operating systems, for example, OS/2 Warp Connect and Windows 95. This new threshold marks the beginning of a transition from Ethernet Client/Server to intergalactic Client/Server that will result in the irrelevance of proximity. The center of gravity is shifting from single server, two tiers; LAN based departmental Client/Server to a post scarcity form of Client/Server where every machine on the global information highway can be both a client and a server. When it comes to intergalactic Client/Server applications, the imagination is at the controls. The promise of high bandwidth at very low cost has conjured visions of an information highway that turns into the world's largest shopping mall. The predominant vision is that of an electronic bazaar of planetary proportions replete with boutiques, department stores, bookstores, brokerage services, banks, and travel agencies. Like a Club Med, the mall will issue its own electronic currency to facilitate round the clock shopping and business to business transactions. Electronic agents of all kinds will be roaming around the network looking for bargains and conducting negotiations with other agents. Billions of electronic business transactions will be generated on a daily basis. Massive amounts of multimedia data will also be generated, moved, and stored on the network.

5.2 TECHNOLOGIES FOR CLIENT/SERVER APPLICATION

Some key technologies are needed at the Client/Server application level to make all this happen, including:

- **Rich transaction processing:** In addition to supporting the venerable flat transaction, the new environment requires nested transactions that can span across multiple servers, long-lived transactions that execute over long periods of time as they travel from server to server, and queued transactions that can be used in secure business-to-business dealings. Most nodes on the network should be able to participate in a secured transaction; super server nodes will handle the massive transaction loads.
- **Roaming agents:** The new environment will be populated with electronic agents of all types. Consumers will have personal agents that look after their interests; businesses will deploy agents to sell their wares on the network; and sniffer agents will be sitting on the network, at all times, collecting information to do system management or simply looking for trends. Agent technology includes cross-platform scripting engines, workflow, and Java-like mobile code environments that allow agents to live on any machine on the network.
- **Rich data management:** This includes active multimedia compound documents that you can move, store, view, and edit in-place anywhere on the network. Again, most nodes on the network should provide compound document technology — for example, OLE or OpenDoc – for doing mobile document management. Of course, this environment must also be able to support existing record-based structured data including SQL databases.
- **Intelligent self-managing entities:** With the introduction of new multi-threaded, high-volume, network-ready desktop operating systems; we anticipate a world where millions of machines can be both clients and servers. However, we can't afford to ship a system administrator with every $99 operating system. To avoid doing this, we need distributed software that knows how to manage and configure itself and protect itself against threats.
- **Intelligent middleware:** The distributed environment must provide the semblance of a single-system-image across potentially millions of hybrid Client/Server machines. The middleware must create this Houdini-sized illusion by making all servers on the global network appear to behave like a single computer system. Users and programs should be able to dynamically join and leave the network, and then discover each other. You should be able to use the same naming conventions to locate any resource on the network.

5.3 SERVICE OF A CLIENT/SERVER APPLICATION

In this section, the discussion is about most widely used five types of Client/Server applications. In no way is this meant to cover all Client/Server applications available today. The truth, there is no agreement within the computer industry as to what constitutes Client/Server and therefore, what one expect or vendor may claim to be Client/Server may not necessarily fit the definition of others.

In general, Client/Server is a system. It is not just hardware or software. It is not necessarily a program that comes in a box to be installed onto your computer's hard drive (although many software manufacturers are seeing the potential market for Client/Server products, and therefore are anxious to develop and sell such programs). Client/Server is a conglomeration of computer equipment, infrastructure, and software programs working together to accomplish computing tasks which enable their users to be more efficient and productive. Client/Server applications can be distinguished by the nature of the service or type of solutions they provide. Among them five common types of solutions are as given below.

- File sharing.
- Database centered systems.
- Groupware.
- Transactional processing.
- Distributed objects.

- **File sharing:** File sharing is Client/Server in its most primitive form. It is the earliest form of computing over a network. Some purists would deny that file sharing is Client/Server technology. In file sharing, a client computer simply sends a request for a file or records to a file server. The server, in turn, searches its database and fills the request. Usually, in a file sharing environment, the users of the information have little need for control over the data or rarely have to make modifications to the files or records. File sharing is ideal for organizations that have shared repositories of documents, images, large data objects, read-only files, etc.
- **Database centered systems:** The most common use of Client/Server technology is to provide access to a commonly shared database to users (clients) on a network. This differs from simple file sharing in that a database centered system not only allows clients to request data and data-related services, but it also enables them to modify the information on file in the database. In such systems, the database server not only houses the database itself; it helps to manage the data by providing secured access and access by multiple users at the same time. Database-centered systems utilize SQL, a simple computer language which enables data request and fulfillment messages to be understood by both clients and servers. Database-centered Client/Server applications generally fall into one of two categories:

  (*i*) Decision-Support Systems (DSS) or

  (*ii*) Online Transaction Processing (OLTP).

Both provide data on request but differ in the kinds of information needs they fulfill.

(*i*) Decision-support systems (DSS): Decision-Support Systems (DSS) are used when clients on the system frequently do analysis of the data or use the data to create reports and other documents. DSS provides a "snapshot" of data at a

particular point in time. Typically, DSS might be utilized for a library catalog, WWW pages, or patient records in a doctor's office.

(*ii*) Online transaction processing: Online Transaction Processing (OLTP) provides current, up-to-the-minute information reflecting changes and continuous updates. Users of an OLTP system typically require mission-critical applications that perform data access functions and other transactions with a one to two seconds response time.

Airline reservations systems, point-of-sale tracking systems (i.e., "cash registers" in large department stores or super markets), and a stockbroker's workstation are OLTP applications.

Structured Query Language (SQL): SQL, pronounced "sequel", stands for Structured Query Language. It is a simple set of commands which allows users to control sets of data. Originally developed by IBM, it is now the predominant database language of mainframes, minicomputers, and LAN servers. It tells the server what data the client is looking for, retrieves it, and then figures out how to get it back to the client.

SQL has become the industry standard for creating shared databases having received the "stamp of approval" from the ISO and ANSI. That's important since prior to this, there was no one commonly agreed upon way to create Client/Server database applications. With standardization, SQL has become more universal which makes it easier to set up Client/Server database systems in multi-platform/multi-NOS environments.

* **Groupware**

Groupware brings together five basic technologies multimedia document management, workflow, scheduling, conferencing, and electronic mail, in order to facilitate work activities. One author defines groupware as "software that supports creation, flow, and tracking of non-structured information in direct support of collaborative group activity." Groupware removes control over documents from the server and distributes it over a network, thus enabling collaboration on specific tasks and projects. The collaborative activity is virtually concurrent meaning that clients on the network, wherever they may be, can contribute, produce, and modify documents, and in the end, using the management and tracking features, synchronizes everything and produces a collective group product.

Multimedia Document Managements (MMDM)

With groupware, clients can have access to documents and images as needed. Multimedia document management (MMDM) allows them to take those documents and modify them. The modifications can take place in real time with several clients making changes and modifications simultaneously, or they can be modified, stored on the server for review or future action by other clients. MMDM is, in essence, an electronic filing cabinet that holds documents in the form of text, images, graphics, voice clips, video, and other media.

Workflow

Workflow refers to technologies which automatically route events (work) from one program to the next in a Client/Server groupware environment. It determines what needs to be done to complete a task or project, then merges, transforms, and routes the work item through the collaborative process.

Workflow is especially applicable to routine tasks such as processing insurance claims or income tax return preparation.

Scheduling (or Calendaring)

Scheduling or calendaring is native to groupware technology. It electronically schedules things like meetings and creates shared calendars and "to do" lists for all users on client workstations. It can work with the workflow function to monitor progress on a project (i.e., monitoring the project completion timeline), schedule a meeting or conference among key persons if needed, and notify them by automatic e-mail.

Conferencing

Conferencing is another native groupware application. This allows users at different client workstations to hold "electronic meetings." These meetings can be either "real time" or "anytime." In real time conferencing clients are interacting simultaneously. With "anytime" conferencing, documents and messages are posted to bulletin boards and clients can add their "two cents" in the form of comments, messages, or modifications.

Electronic Mail (E-mail)

E-mail is an essential element of groupware technology. It facilitates communication among clients on a network. In a groupware environment, the e-mail can be integrated with the multimedia document management, workflow, scheduling/calendaring, and conferencing features.

- **Transactional Processing**

To create truly effective Client/Server solutions, the various components within the system (the application software, the network operating system, utilities, and other programs) need to work together in unison. If infrastructure and software which enable Client/Server computing are musicians in a symphony, transaction processing would be the conductor.

- **Distributed Objects**

A distributed object is a vague term used to describe the technologies which allow clients and servers from different technologies from different environments and platforms to work seamlessly together. Its goal is to provide users with single-image, easy-to-use, virtually transparent applications.

Distributed object technology is still in its infancy and has yet to fulfill its promise of making Client/Server into the flexible, robust, intelligent, and self-managing systems that most users want and expect. Distributed objects technology has great "potential" but at this point in time, it remains just that potential.

5.4 CATEGORIES OF CLIENT/SERVER APPLICATIONS

There are variety of ways to divide the processing between client and server. But the exact distribution of data and application programming depends on the nature of the database, the type of application supported, the availability of interoperable vendor equipment, and the usage patterns within an organization. Depending on the database applications various classes of Client/Server Application has been characterized.

(*i*) Host-based processing.
(*ii*) Server-based processing.
(*iii*) Client-based processing.
(*iv*) Cooperative processing.

(*i*) Host-based processing: Virtually all the processing is done on a central host, often user interface is via a dumb terminal. It is mainly mainframe environment, not true Client/Server computing. In such a processing's workstations have very limited role as shown in Fig. 5.1 given below:

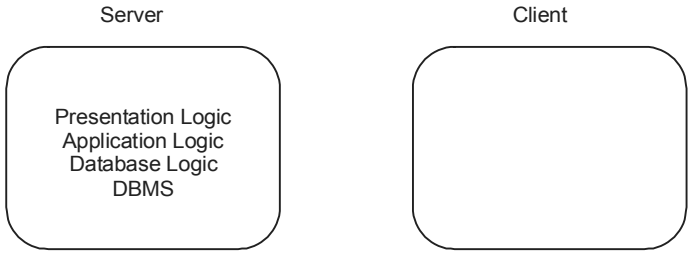

Fig.5.1: Host-base Processing

(*ii*) Server-based processing: All the processing is done on the server, and server is responsible for providing graphical user interface. A considerable fraction of the load is on the server, so this is also called *fat server* model shown in Fig. 5.2 given below:

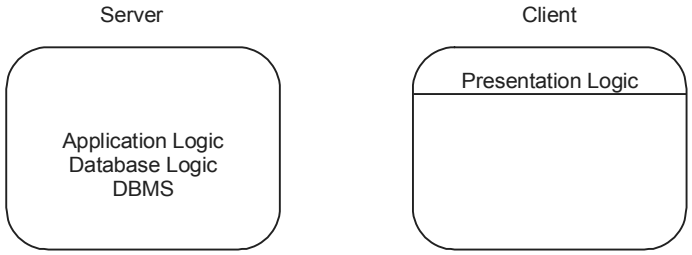

Fig.5.2: Server-base Processing

(*iii*) Client-based processing: Virtually all application processing may be done at the client, with the exception of data validation routines and other database logic functions that are best performed at the server. Some of the sophisticated database logic functions residing on the client side. This architecture is the most common Client/Server approach in current use. It enables the user to employ applications tailored to local need shown in Fig 5.3 given below:

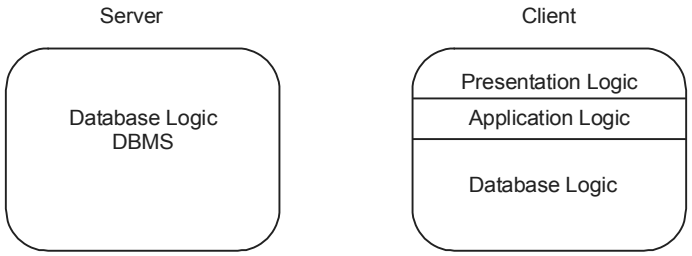

Fig. 5.3: Client-base Processing

(*iv*) Cooperative processing: Taking the advantage of the strengths of both client and server machine and of the distribution of data, the application processing is performed in an optimized fashion. Such a configuration of Client/Server approach is more complex to set up and maintain. But in the long run, this configuration may offer greater user productivity gain and greater network efficiency than others Client/Server approaches. A considerable fraction of the load is on the client, so this is also called *fat client* model. In case of some application development tools this model is very popular shown in Fig. 5.4 given below:

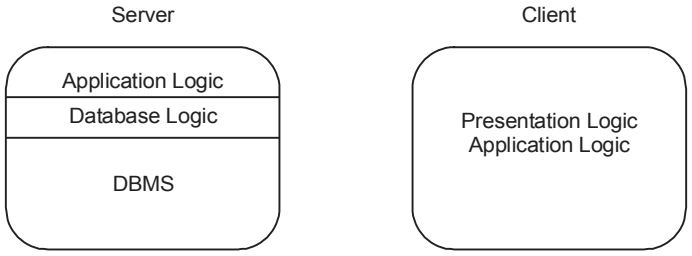

Fig. 5.4: Cooperative Processing

5.5 CLIENT SERVICES

Any workstation that is used by a single user is a client, it has been noticed during last decade the workstations are improving their performance surprisingly. Having same cost you can purchase CPU that can perform more than 50 times, main memory approximately

30 times and Hard Disk up to 40 times. These are considered as power factor of computer, hence, as a result, more sophisticated applications can be run from the workstations. To run various applications workstation uses the available operating systems like DOS, Windows (98, 2000, NT) and UNIX or Linux, Mac, OS/2. In case of, network environment (LAN, WAN) workstations also avails the services provided by the network operating systems. Client workstations request services from the attached server. Whether this server is in fact the same processor or a network processor, the application format of the request is the same. Network operating system translates or adds the specifics required by the targeted requester to the application request. Communication between all these running processes are better described by Inter Process Communication (IPC), these processes might be on the same computer, across the LAN, or WAN.

Some of the main services that client performs (role of client) are listed below:

- Responsible for managing the user interface.
- Provides presentation services.
- Accepts and checks the syntax of user inputs. User input and final output, if any, are presented at the client workstation.
- Acts as a consumer of services provided by one or more server processors.
- Processes application logic.
- The role of the client process can be further extended at the client by adding logic that is not implemented in the host server application. Local editing, automatic data entry, help capabilities, and other logic processes can be added in front of the existing host server application.
- Generates database request and transmits to server.
- Passes response back to server.

But in client server model one thing is very obvious that the services are provided by combination of resources using both the client workstation processor and the server processor. For an example, let us take very common example of client server application, a database server provides data in response to an SQL request issued by the workstation application. Local processing by the workstation might calculate the invoice amount and format the response to the workstation screen. Now, it is important to understand that a workstation can operate as a client in some instances while acting as a server in other instances. For example, in a LAN Manager environment, a workstation might act as a client for one user while simultaneously acting as a print server for many users. In other words we can say "the client workstation can be both client and server when all information and logic pertinent to a request is resident and operates within the client workstation."

Apart from these services discussed above some of the other important services that are directly or indirectly attached with the client services are given below:

(*a*) Inter process communication.

(*b*) Remote services.

(*c*) Window services.

(*d*) Dynamic data exchange.
(*e*) Object linking and embedding.
(*f*) Common object request broker architecture (CORBA).
(*g*) Print/Fax services.
(*h*) Database services.

5.5.1 Inter Process Communication

The communication between two processes take place via buffer. The alternative way of communication is the process of the interprocess communication. The simple mechanism of this is synchronizing their action and without sharing the same address space. This play an important role in the distribute processing environment.

While signals, pipes and names pipes are ways by which processes can communicate. The more redefined method of inter process communication are message queues, semaphores and shared memory. There are four types of mechanisms, involved for such a communications:-

(*i*) Message passing.
(*ii*) Direct communication.
(*iii*) Indirect communication.
(*iv*) Remote procedures call.

(*i*) Message passing: This mechanism allows process to communicate without restoring the shared data, for example in micro kernel, message is passed to communicate in which services acts as an ordinary user where these services act outside the kernel. At least, there are two processes involved in an IPC. See the Fig. 5.5 given below:

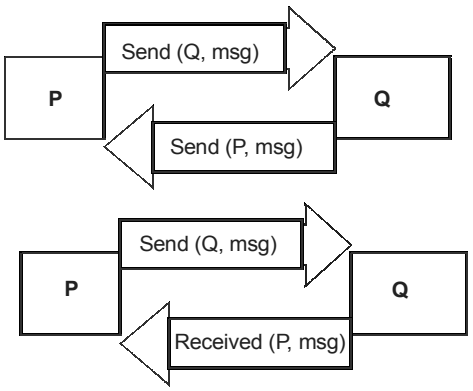

Fig.5.5: *Message Passing*

- Sending process for sending the message.
- Receiving process for receiving the message.

Messages sent by the processes are of two types, fixed and variable. For the communication to be taking place, a link is to be set in between the two processes.

(*ii*) Direct communication: In this mechanism of communication processes have to specify the name of sender and recipient process name. This type of communication has the following features:

- A link is established in between the sender and receiver along with full known information of their names and addresses.
- One link must be established in between the processes.
- There is symmetry in between the communication of the processes.

(*iii*) Indirect communication: In indirect communication, messages are sending to the mail box and then they are retrieved from mailbox, see the Fig 5.6 given below:

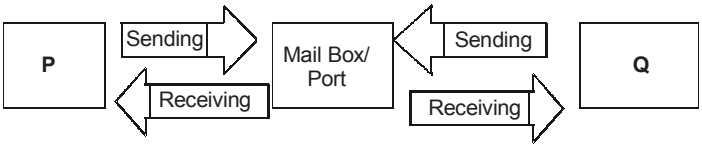

Fig. 5.6: Indirect Communication

The role of the mailbox is quite similar to the role of the postman. The indirect communication can also communicate with other processes via one or more mailbox. The following features are associated with indirect communication:

- A link is established between a pair of process, if they share a mailbox.
- A link is established between more than one processes.
- Different number of links can be established in between the two communicating processes.

Communication between the processes takes place by executing calls to the send and receive primitive. Now there is several different ways to implement these primitives, they can be "blocking" and "non-blocking". The different possible combinations are:

- *Blocking send*: Sending the process is blocked until the message is received.
- *Non-blocking send*: In it process sends the message and then it resumes the operation.
- *Blocking receive:* Receiver is blocked until the message is available.
- *Non-blocking receive:* The receiver receives either a valid message or a null.

(*iv*) Remote procedures call: RPC is a powerful technique for constructing distributed, client-server based applications. The essence of the technology is to allow programs on different machines to interact using simple procedure call or return semantics, just as if the two programs were on the same machine. It is based on extending the notion of conventional or local procedure calling, so that the called procedure need not exist in the same address space as the calling procedure. The two processes may be on the same system, or they may be on different systems with a network connecting them. That is, the procedure call is used for access to remote services.

In client-server based applications a binding is formed when two applications have made a logical connection and are prepared to exchange commands and data. This client server binding specifies how the relationship between a remote procedure and the calling program will be established. By using RPC, programmers of distributed applications avoid the details of the interface with the network. The transport independence of RPC isolates the application from the physical and logical elements of the data communications mechanism and allows the application to use a variety of transports.

How RPC Works: An RPC mechanism is analogous to a function call. Like a function call, when an RPC is made, the calling arguments are passed to the remote procedure and the caller waits for a response to be returned from the remote procedure. Figure 5.7 illustrates the general architecture of remote procedure call mechanism that takes place during an RPC call between two networked systems. The client makes a procedure call that sends a request to the server and waits. The thread is blocked from processing until either a reply is received, or it times out. When the request arrives, the server calls a dispatch routine that performs the requested service, and sends the reply to the client. After the RPC call is completed, the client program continues. RPC specifically supports network applications.

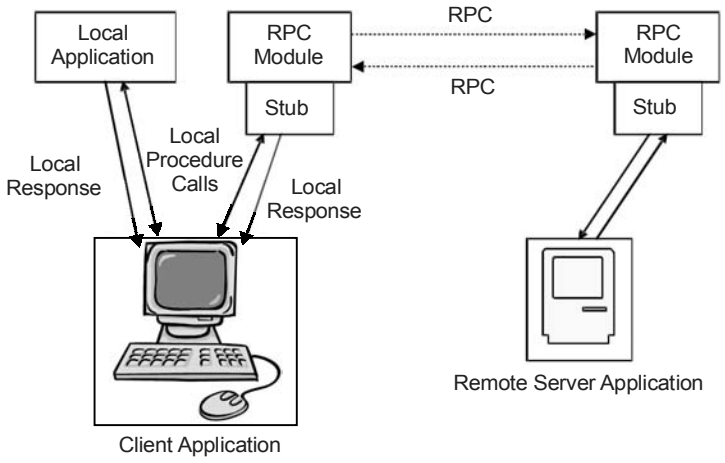

Fig. 5.7: RPC Mechanism

A remote procedure is uniquely identified by the triple: (program number, version number, procedure number), the program number identifies a group of related remote procedures, each of which has a unique procedure number. A program may consist of one or more versions. Each version consists of a collection of procedures which are available to be called remotely. Version numbers enable multiple versions of an RPC protocol to be available simultaneously. Each version contains a number of procedures that can be called remotely. Each procedure has a procedure number. Procedure may or may not be transparent to the user that the intention is to invoke a remote procedure on the same machine.

A stub procedure is added in the callers users address space (or dynamically linked to it at call time). This stub procedure creates a message that identifies the procedure being called and includes the parameters. Stub procedure provides a perfectly local procedure call abstraction by concealing from PROM programs the interface to the underlying RPC system.

RPC provides method for communication between processes residing over a distributed system. Procedure call is used for access to remote services. Basic concepts about this technique is that allowing programs residing on different machines to interact using simple procedures in a similar way like two programs running on the same machine. That is programmers feels an isolation from the network intricacies and got easy access to network functions by using RPC systems services.

RPCs, are APIs, layered on top of a network IPC mechanism, allows users to communicate users directly with each others. They allows individual processing components of an application to run other nodes in the network. Distributed file systems, system management, security and application programming depend on the capabilities of the underlying RPC mechanisms. Server access control and use of a directory service are common needs that can be met with RPCs. RPCs also manage the network interface and handle security and directory services. Tools of RPC comprises of:

- A language and a compiler to produce portable source code.
- Some run time facilities to make the system architecture and network protocols transparent to the application procedure.

The mechanism of RPC can be considered as a refinement of reliable, blocking message passing. Figure 5.8 given below, illustrates the general architecture understanding.

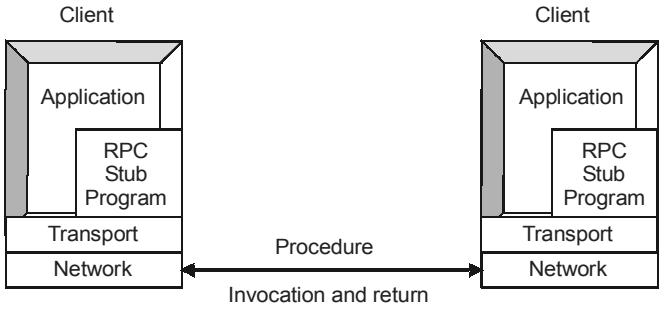

Fig. 5.8: General Architecture

Here, the structure of RPC allows a client to invoke a procedure on the remote host locally, which is done with the help of "stub" which is provided by the client. Thus, when the client invokes the remote procedure RPC calls the appropriate stub, passes the parameters to it, which are then provided, to remote procedure. This stub locates the port on the server and **marshalling** involves packaging the parameter into a form, which may be transmitted over network. The stub then transmits a message to server using message passing. Now the message sent by the host is received at the client side with the help of similar type of stub.

Limitation of RPC: There are number of limitations associated with RPC given below.

1. RPC requires synchronous connections. If an application uses an RPC to link to a server that is busy that time then application will have to wait for the data rather than switching to other task.
2. Local procedure call fails under the circumstances where RPC can be duplicated under and executed more than one, which is due to unreliable communication.
3. The communication in between the client and server is done with help of the standard procedure calls; therefore some binding must take place during the link load and execution, such that the process is replaced by the address. The RPC binds the same thing to the client and server. A general problem that exists is that there is no shared memory in between them so how they can come to know about the address of the other system.
4. The binding information may be predetermined in the form of the port address, at the compile time, a RPC call, that has a fix port number is associated with it. Once a program is compiled, it then cannot change its port number.
5. Binding can be done dynamically by rendezvous mechanism. Typically an operating system provides rendezvous demon requesting the port address of RPC, it needed to execute. The port address is then returned and the RPC call may be sent to the port until the process terminates.

5.5.2 Remote Services

In client server model applications can be invoked directly from the client to execute remotely on a server. The workstation is responsible to provide various remote services. Among them some services like remote login, remote command execution, remote backup services, remote tape drive access and remote boot services, and remote data access are important. Software available with Network Operating System is responsible to run on the client workstation to initiate all these remote services. Client server technology supports full-powered workstations with the capability for GUI applications consistent with the desktop implementation. Remote command execution is when a process on a host cause a program to be executed on another host, usually the invoking process wants to pass data to the remote program, and capture its output also. From a client workstation backup services may be invoked remotely. Some of the business functions such as downloading data from a host or checking a list of stock prices might also be invoked locally to run remotely. To run the application, some of the workstation clients (like X-terminals) do not have the local storage facility. In such scenario, client provides appropriate software's that are burned into E-PROM (Erasable Programmable Read-Only Memory) to start the initial program load (IPL)that is known as Boot Process. If E-PROM is inbuilt with X-terminals to hold the Basic Input/Output System services. Then partial operating system will be able to load the remote software's that provides the remaining services and applications functions to the client workstation. This is known as remote boot service provided by client workstation and X-terminals.

Remote data access is one of the ISO multi-site transaction processing and communication protocol used for heterogeneous data access. Using RDA technology, any client running an application will be able to access more than one database residing at the different servers.

5.5.3 Window Services

In client server application, operating system at the client workstation provides some windows services, these services are capable of to move, view, activate, hide, or size a particular window. This is very helpful in implementation of several applications because a client workstation may have several windows open on-screen at a time. And also, all these applications interact with message services provided to notify the user of events that occur on a server. Application programs running on workstations have been written with no windowing sensitivity. These application programs are written under virtual screen assumptions, that virtual screens are generally dissimilar to the actual available physical screens. Now with the help of interface software client application places data into virtual screen, and then the windows services handles manipulation and placement of application windows. Thus, pursuing that way application development has been enhanced tremendously due to developers less involvement in managing or building the windowing services. The client user is fully in grip of his desktop and can give priority to the most important tasks at hand simply by positioning the window of interest to the workstation.

The NOS provides some software's on the client workstation which is able to manage the creation of pop-up windows that display alerts generated from remote servers. Print complete, E-mail receipt, Fax available, and application termination are examples of alerts that might generate a pop-up window to notify the client user.

5.5.4 Dynamic Data Exchange (DDE)

DDE is usually described as a conversation between two applications, a client application and a server application. As we know that the client program is on that requests (receives) the information, and the server is the one that response (supplies) it. DDE is a feature of some operating systems (like Windows 98, OS/2) presentation manager that enable users to pass data between applications to application. For an example, if an application wants to connect a Microsoft Excel spreadsheet with Microsoft Word for windows report in such a way that changes to the spreadsheet are reflected automatically in the report, in that case Microsoft Word for windows is the client and Microsoft Excel is the server. A DDE conversation always concerns a particular topic and a particular item. The topic and item spell out the nature of the information that the client is requesting from the server. For an example, if the Word for Windows document is to receive data automatically from a range named IBM in a Microsoft Excel worksheet, named STOCKS.XLS then STOCKS.XLS is the topic and IBM is the item. With most of the programs, a simplest way to set up a DDE link is to copy a block of data from the server application to the clipboard, activate the client application, move the insertion point to the location in the receiving document where

you want the information to go, and then use a Paste Link command. With most server programs, some times it requires to save data in a disk file before to paste it into a client programs. Using Paste Link is the easiest way to establish a DDE link, but it's not the only way. Some programs that act as DDE clients have commands that allow you to set up a DDE connection without first putting the source data on the clipboard. Many DDE supporting applications also have macro language that you can use to establish DDE links. This is true with MS Excel, Word, Powerpoint, dynacomm, and many other advanced windows applications. A DDE link may be automatic or manual. An automatic link is refreshed whenever the source data changes, provided both the client and server applications are running. A manual link is refreshed only when you issue a command in the client application.

5.5.5 Object Linking and Embedding (OLE)

Object Linking and Embedding two services collectively called as a single one, carried out with simple edit menu procedures. OLE is a software package accesses data created from another through the use of a *viewer* or *launcher*. These viewers and launchers must be custom built for every application. With the viewer, users can see data from one software package while they are running another package. Launchers invoke the software package that created the data and thus provide the full functionality of the launched software. To link with OLE copy data from OLE supporting program to the Clipboard. Then use the paste link command in another OLE supporting program. To embed, follow the same procedure but use Paste instead of Paste Link. Both programs must support OLE, the program that supplies the data must support OLE as a server application, and the one that receives the data must support as a client application. Some program may support OLE in one mode or the other, (it means either as a server or as client only). For an example, Paintbrush can act only as a server. Write and Card file can act only as OLE clients. Some other programs are also available which support OLE in both modes. Most of the Windows applications support OLE, and also the Microsoft has released its OLE 2.0 Software Development Kit (SDK). The toolkit greatly simplifies OLE integration into third-party, developed applications. Organizations wanting to create a consistent desktop are beginning to use the OLE SDK as part of custom applications. OLE 2.0 extends OLE capabilities to enable a group of data to be defined as an object and saved into a database. This object can then be dragged and dropped into other applications and edited without the need to switch back to the application which created it. This provides a more seamless interface for the user. In OLE 2.0, the active window menu and toolbar change to that of 1-2-3. The user deals only with the object, with no need to be aware of the multiple software being loaded. Generally, the OLE is known as an extension to DDE that enables objects to be created with the object components software aware *(*a reference to the object or one of its components automatically launches the appropriate software to manipulate the data). Both the techniques (OLE and DDE) require the user to be aware of the difference between

data sources. DDE and OLE provide a substantial advantage; any DDE-or OLE-enabled application can use any software that supports these data interchange APIs. An e-mail application will be able to attach any number of components into the mail object without the need to provide custom viewers or launchers. But here, linking with OLE offers one big advantage over linking with DDE; that is the ability to launch the server program directly forms the client document. Client need not to remember from where the data came from. And if the server renames or relocates a document it requires repairing or abandoning any links in client documents. Most OLE client's applications include commands to assist such an application with this.

5.5.6 Common Object Request Broker Architecture (CORBA)

CORBA an object oriented architecture that provides an mechanism that allows various clients to share/call the object (applications) over a mixed networks. More specifically CORBA is a process of moving objects over network proving cross platform for data transfer. A client that needs a service sends a request to an object request broker (which acts as a directory) of all the remote services available on the network, illustrated in Fig. 5.9 given below:

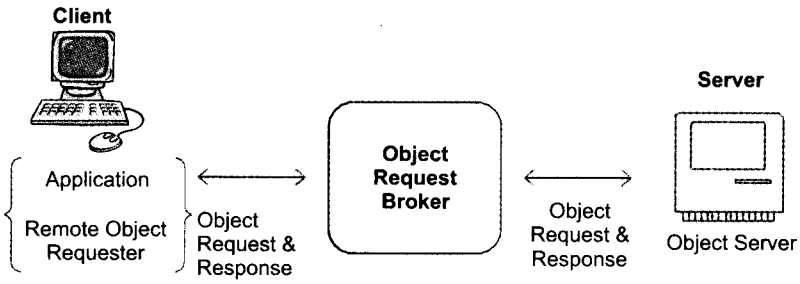

Fig. 5.9: Object Request Broker

The broker calls the appropriate object and passes along any relevant data. Then the remote object services the request and replies to the broker, which returns the response to the client. The object communication may rely on an underlying message or RPC structure or be developed directly on top of object-oriented capability in the operating system. The architecture of CORBA application is similar to the client server architecture by maintaining the notion of client and servers. In CORBA, a component can act as both a client and server. A component is considered as a server if it contains CORBA objects whose services are accessible form some other CORBA object. Likewise, a component is considered as a client if it access services from some other CORBA objects. Here a component can simultaneously provides and use various services and so any component can be considered as a client or as a server depending on the nature of the application running currently. When considering a single remote method invocation, however the role of client and server can be temporarily reversed because a CORBA object can participate in multiple interactions simultaneously.

Furthermore, any component that provides an implementation for an object is considered as a server, at least where that object is concerned. If a component creates an object and provides other components with visibility to that object (i.e., allows other components to obtain references to that object), that components acts as server for that object; any requests made on that object by other components will be processed by the components that creates the object. Thus, a CORBA server means the components execute methods for a particular object on behalf of other components (Clients). An application component can provide services to other application components while accessing services from other components. In that scenario, the components is acting as a client of one component and as a server to the other components i.e., two components can simultaneously act as client and server to each other, illustrated in Fig. 5.10, shown below. CORBA concepts and its role in Client/Server architecture is discussed in more detail in Chapter 9.

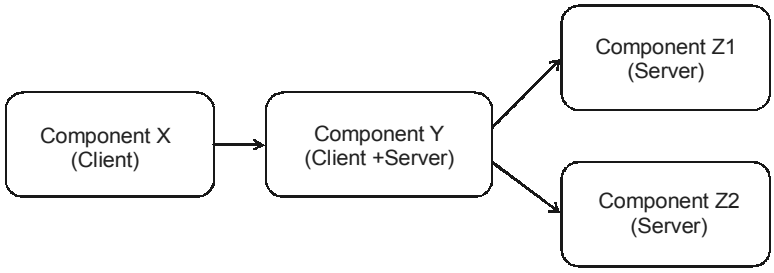

Fig. 5.10: *Acting as a Client and a Server*

5.5.7 Print/Fax Services

Client generates print/fax requests to the printer/fax machine without knowing whether they are free or busy. In that task network operating system helps the client to generate the requests. These requests are redirected by the NOS redirector software and managed by the print/fax server queue manager. The users at the client workstation can view the status of the print/fax queues at any time. And also some of the print/fax servers acknowledge the client workstation when the print/fax request is completed.

5.5.8 Database Services

Client/Server model provides integration of data and services allow clients to be isolated from inherent complexities such as communication protocols. The simplicity of client server architecture allows clients to make request that are routed to the database server (These requests are made in the form of transactions). In other words, client application submit database request to the server using SQL statements. Once received the server processes the SQL statement and the request are returned to the client application. That is illustrated in Fig. 5.11.

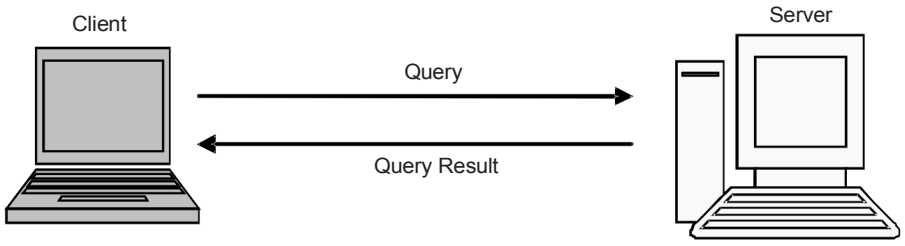

Fig. 5.11: Execution of SQL

Hence, most of the database requests are made using the SQL syntax. Now, the SQL's have became the industry standard language supported by many vendors. Because the language uses a standard form, the same application may be run on multiple platforms. Client application can concentrate on requesting input from users, requesting desired data from server, and then analyzing and presenting this data using the display capabilities of the client workstation. Furthermore, client applications can be designed with no dependence on the physical location of the data. If the data is moved or distributed to the other database servers, the application continues to function with little or no modification. Client applications can be optimized for the presentation of data and server can be optimized for the processing and storage of data. Application development tools are used to construct the user interfaces (interface of clients with server and also interface of front-end user to the back-end server); they provide graphical tools that can be used to construct interfaces without any programming. Examples of such tools are Visual Basic, Borland Delphi, Magic, and Power Builder. Some application programs (spreadsheet and statistical–analysis packages) uses the client server interface directly to access data from back-end server.

5.6 SERVER SERVICES

The server is responsible for controlling and providing shared access to available server resources. Remote workgroups have needed to share these resources when they are connected with server station through a well-managed network. The applications on a server must be isolated from each other so that an error in one application cannot damage another application. Furthermore, the server is responsible for managing the server-requester interface so that an individual client request response is synchronized and directed back only to the client requester. This implies both security when authorizing access to a service and integrity of the response to the request. For a Client/Server applications servers performs well when they are configured with an operating system that supports shared memory, application isolation, and preemptive multitasking (an operating system with preemptive multitasking enables a higher priority task to preempt or take control of the processor from a currently executing, lower priority task). These preemptive multitasking ensures that no single task can take overall the resources of the server and prevent other

tasks from providing service. There must be a means of defining the relative priority of the tasks on the server. These are specific requirements to the Client/Server implementation.

One of the prime server characteristic is that it must support for multiple simultaneous client request for service. So that, the server must provide shared memory services and multitasking support. In that respect, the following server platform provides best processors for client server implementation are IBM System/370, DEC VAX , Intel and RISC (Reduced Instruction Set Computers like Sun SPARC, IBM/Motorola PowerPC, HP PA RISC, SGI MIPS, and DEC Alpha). Some of the main operations that server perform are listed below:

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

Apart from these services discussed above some of the other important services that are directly or indirectly attached with the server services in a network operating system environment are given below:

(*i*) Application services.
(*ii*) File services.
(*iii*) Database services.
(*iv*) Print/fax/image services.
(*v*) Communications services.
(*vi*) Security systems services.
(*vii*) Network management services.
(*viii*) Server operating system services.

(*i*) Application services: Application servers provide business services to support the operation of the client workstation. In the Client/Server model these services can be provided for entire partial business functions that are invoked by IPC (Inter Process Communication) or RPCs request for service. A collection of application servers may work in concert to provide an entire business function. For an example, in a inventory control system the stock information may be managed by one application server, sales information are maintained by another application server, and purchase information are maintained by a third application server. On larger and more complicated systems, server responsibility may be distributed among several different types of servers. All these servers are running at different operating systems on various hardware platforms and may use different database servers. The

client application invokes these services without consideration of the technology or geographic location of the various servers.

(*ii*) **File services:** A file server can store any type of data, and so on simpler systems, may be the only server necessary. Space for storage is allocated, and free space is managed by the file server. Catalog functions are also provided by the file server to support file naming and directory structure.

File services are responsible to handle access to the virtual directories and files located on the client workstation and to the server's permanent storage. Redirection software provides these services which are installed as part of client workstation operating system. Finally, all clients' workstation requests are mapped into the virtual pool of resources and redirected as necessary to the appropriate local or remote server. The file services provide this support at the remote server processor. File server manages databases, software's, shared data, and backups that are stored on tape, disk, and optical storage devices. In order to minimize the installation and maintenance effort of software, software should be loaded directly from the server for execution on the client workstations. New versions of any application software can be updated on the server and made immediately available to all users.

(*iii*) **Database services:** Early database servers were actually file servers with a different interface. Products such as dBASE, Clipper, FoxPro, and Paradox execute the database engine primarily on the client machine and use the file services provided by the file server for record access and free space management. These are new and more powerful implementations of the original flat-file models with extracted indexes for direct record access. Currency control is managed by the application program, which issues lock requests and lock checks, and by the database server, which creates a lock table that is interrogated whenever a record access lock check is generated. Because access is at the record level, all records satisfying the primary key must be returned to the client workstation for filtering. There are no facilities to execute procedural code at the server, to execute joins, or to filter rows prior to returning them to the workstation. This lack of capability dramatically increases the likelihood of records being locked when several clients are accessing the same database and increases network traffic when many unnecessary rows are returned to the workstation only to be rejected.

The lack of server execution logic prevents these products from providing automatic partial update backout and recovery after an application, system, or hardware failure. For this reason, systems that operate in this environment require an experienced system support programmer to assist in the recovery after a failure. When the applications are very straight forward and require only a single row to be updated in each interaction, this recovery issue does not arise. However, many Client/Server applications are required to update more than a single row as part of one logical unit of work.

Client/Server database engines such as Sybase, IBM's Database Manager, Ingres, Oracle, and Informix provide support at the server to execute SQL requests issued from the client workstation. The file services are still used for space allocation and basic directory services, but all other services are provided directly by the database server. Relational database management systems are the current technology for data management. Figure 4.1 charts the evolution of database technology from the first computers in the late 1950s to the object-oriented database technologies that are becoming prevalent in the mid-1990s.The following DBMS features must be included in the database engine:

- Performance optimization tools.
- Dynamic transaction backout.
- Roll back from, roll forward to last backup.
- Audit file recovery.
- Automatic error detection and recovery.
- File reclamation and repair tools.
- Support for mirrored databases.
- Capability to split database between physical disk drives.
- Remote distributed database management features.
- Maintenance of accurate and duplicate audit files on any LAN node.

In the Client/Server implementation, database processing should offload to the server. Therefore, the database engine should accept SQL requests from the client and execute them totally on the server, returning only the answer set to the client requestor. The database engine should provide support for stored procedures or triggers that run on the server.

The Client/Server model implies that there will be multiple concurrent user access. The database engine must be able to manage this access without requiring every developer to write well-behaved applications. The following features must be part of the database engine:

- Locking mechanisms to guarantee data integrity.
- Deadlock detection and prevention.
- Multithreaded application processing
- User access to multiple databases on multiple servers.

(*iv*) Print/fax/image services: High-quality printers, workstation-generated faxes, and plotters are natural candidates for support from a shared server. The server can accept input from many clients, queue it according to the priority of the request and handle it when the device is available. Many organizations realize substantial savings by enabling users to generate fax output from their workstations and queue it at a fax server for transmission when the communication costs are lower. Incoming faxes can be queued at the server and transmitted to the appropriate client either on receipt or on request. In concert with workflow management techniques, images can be captured and distributed to the appropriate client

workstation from the image server. In the Client/Server model, work queues are maintained at the server by a supervisor in concert with default algorithms that determine how to distribute the queued work.

Incoming paper mail can be converted to image form in the mail room and sent to the appropriate client through the LAN rather than through interoffice mail. Centralized capture and distribution enable images to be centrally indexed. This index can be maintained by the database services for all authorized users to query. In this way, images are captured once and are available for distribution immediately to all authorized users. Well-defined standards for electronic document management will allow this technology to become fully integrated into the desktop work environment. There are dramatic opportunities for cost savings and improvements in efficiency, if this technology is properly implemented and used. Chapter 9 discusses in more detail the issues of electronic document management.

(*v*) Communications services: Client/server applications require LAN and WAN communication services. Basic LAN services are integral to the NOS. WAN services are provided by various communications server products. Chapter 5 provides a complete discussion of connectivity issues in the Client/Server model.

(*vi*) Security systems services: Client/server applications require similar security services to those provided by host environments. Every user should be required to log in with a user ID and password. If passwords might become visible to unauthorized users, the security server should insist that passwords be changed regularly. The enterprise on the desk implies that a single logon ID and logon sequence is used to gain the authority once to access all information and process for the user has a need and right of access. Because data may be stored in a less physically secure area, the option should exist to store data in an encrypted form. A combination of the user ID and password should be required to decrypt the data.

New options, such as floppy less workstation with integrated Data Encryption Standard (DES) coprocessors, are available from vendors such as Beaver Computer Company. These products automatically encrypt or decrypt data written or read to disk or a communication line. The encryption and decryption are done using the DES algorithm and the user password. This ensures that no unauthorized user can access stored data or communications data. This type of security is particularly useful for laptop computers participating in Client/Server applications, because laptops do not operate in surroundings with the same physical security of an office. To be able to access the system from a laptop without properly utilizing an ID number and password would be courting disaster.

5.7 CLIENT/SERVER APPLICATION: CONNECTIVITY

The communication middleware software provides the means through which clients and servers communicate to perform specific actions. It also provides specialized services to

the client process that insulate the front-end applications programmer from the internal working of the database server and network protocols. In the past, applications programmers had to write code that would directly interface with specific database language (generally a version of SQL) and the specific network protocol used by the database server. For example, when writing a front-end application to access an IBM OS/2 database manager database, the programmer had to write SQL and Net BIOS (Network Protocol) command in the application. The Net BIOS command would allow the client process to establish a session with the database server, send specific control information, send the request, and so on. If the same application is to be use with a different database and network, the application's routines must be rewritten for the new database and network protocols. Clearly such a condition is undesirable, and this is where middleware comes in handy. Here definition of middleware is based on the intended goals and main functions of this new software category. In chapter three communication middleware is also discussed, further role and mechanism of middleware is discussed in this section.

5.7.1 Role and Mechanism of Middleware

Role of middleware component can be exactly understand by the way in which Client/ Server computing being used , we know that there are number of approaches are there like host-based processing, server based processing, cooperative processing and client based processing. And all these depend on application functionality being used in Client/Server architecture. A Middleware component resides on both Client/Server machine enabling an application or user at a client to access a variety of services provided by server.

In other words, we can say middleware provides basis for logical view of Client/Server architecture. Morever, middleware enables the realization of the promises of distributed Client/ Server computing concepts. See the Fig. 5.12 (*a*) and (*b*) that depicts the role and logical view of middleware in Client/Server architecture. The entire system of the architecture represents a view of a set of applications and resources available to clients. Any client need not be concerned with the location of data or indeed the location of the application.

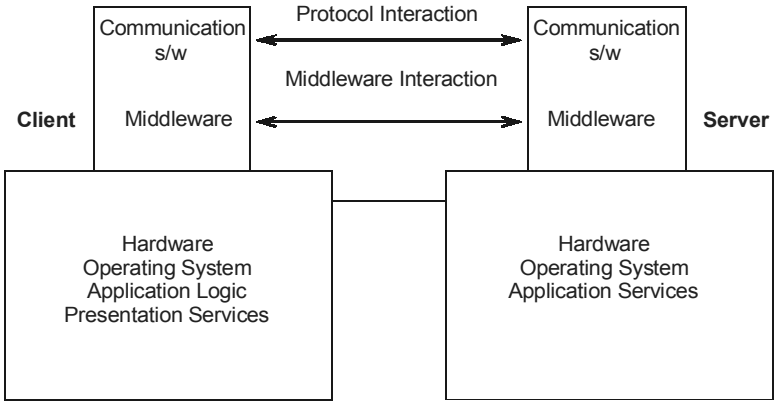

Fig. 5.12(a): Middleware Role in Client/Server Architecture

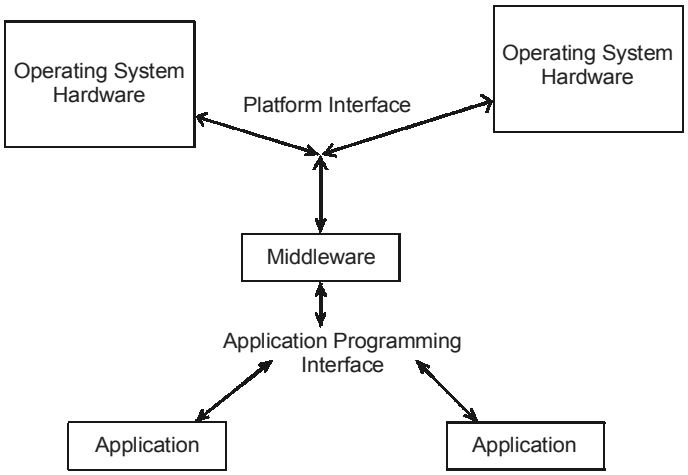

Fig. 5.12(*b*): Middleware Role in Client/Server Architecture

All applications operate over a uniform application programming interface. The middleware is responsible for routing client requests to the appropriate server. Also middleware used to overcome operating system as well as network incompatibility. This middleware running on each network component ensures that all network users have transparent access to applications and resources of any networks.

5.8 CLIENT/SERVER APPLICATION: LAYERED ARCHITECTURE

An essential factor in the success of a client/server environment is the way in which the user interacts with the system as a whole. Thus the design of the user interface to the client machine is critical. In most client/server systems, there is heavy emphasis on providing a graphical user interface that is easy to use, easy to learn, yet powerful and flexible. Section follow covers the design issues of client/server architecture. And also designing issues associated with layered application architecture with their interfaces in the three-layered application architecture.

5.8.1 Design Approach

In client/server architecture as a design approach, the functional components of an application are partitioned in a manner that allows them to be spread and executed across different computing platforms, and share access to one or more common repositories. Client/server architecture is therefore a design approach that distributes the functional processing of an application across two or more different processing platforms. The phrase 'client/server' reflects the role played by an application's functions as they interact with one another. One or more of these functions is to provide a service, typically in the form of a database server that is commonly used by other functions across the application(s). In this regard, it is important to discuss the concept of 'layered application architecture' .

Application design architecture plays a crucial role in aiding development of robust applications. Figure. 5.13 given below shows the three layers of any fairly large business

application. The most detailed layer is the database layer, the centre of an application. The next higher layer is the business rule layer. Finally, the highest level of abstraction is the document layer. This is the layer visible to users of the application.

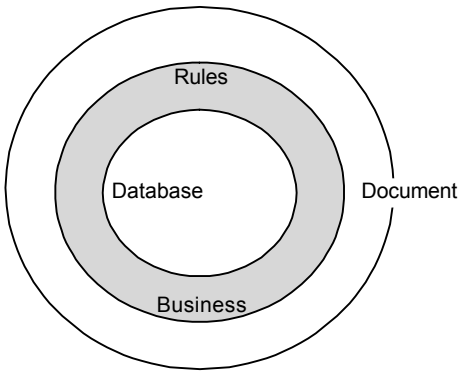

Fig. 5.13: Three-layered Application Architecture

Designing a client/server application offers challenges not normally faced by developers of mainframe-oriented multiuser applications. To realize full benefits of client/server architecture, developers need to incorporate a greater potential for functionality and data distribution in the fundamental design of their applications.

A client/server application operates across multiple platforms, i.e. a server platform for the database, and a client platform for the application. At this minimum level of utilization, the design of client/server does not differ much from its mainframe counterpart, and the physical issues faced by developers on both the platforms are primarily those of packaging. But to take full advantage of client/server paradigm, developers need to address issues of functional distribution for their applications, not just across client and server platform but also across all nodes of the network. Issues surrounding functional distribution constitute the single biggest difference between physical designs of multiuser and client/server application.

5.8.2 Interface in Three Layers

The key to use a three-layered application architecture is to understand the interfaces used in all its three layers. Figure 5.14 illustrates these interfaces. They are:

- Graphical user interface.
- Process request interface.
- Transaction and query manager interface.

An interface enables a component in one layer to communicate with a component in another layer; it also enables a component to interact with another component in the same layer. In Fig. 5.14 communication between components in the same layer is indicated by a semicircular arrow. Moreover, we can say that this describes a collection of mutually cooperating components that make request to each other, both within and across layers, with its components working together thus, and processing various requests—wherever they comes from; a business application comes to life.

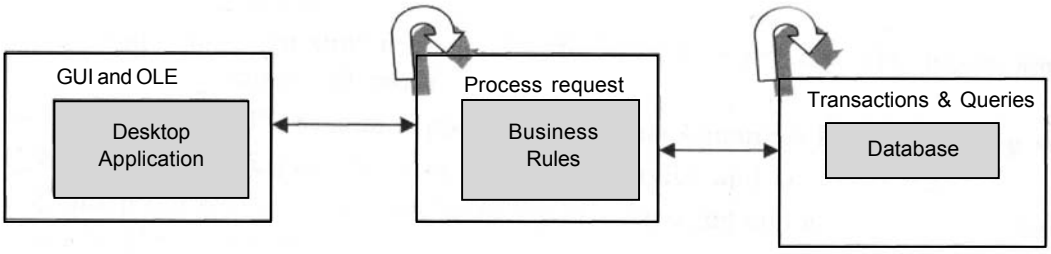

Fig. 5.14: Interface in a Three-layered Application Architecture

Cooperating components in a layered application design provide the following:

- A framework for building highly flexible applications that can be changed easily to meet the changing needs of business.
- A high level of software reuse.
- Easier development of large, complex applications that can sustain high throughput levels in both decision support and transaction environments.
- Easier development of distributed applications that support centrally and self-managed teams.

EXERCISE 5

1. In Client/Server computing, explain the following with example in detail
   (*a*) Dynamic Data Exchange
   (*b*) RPC, Remote Procedure Call
   (*c*) Remote Boot Service
   (*d*) Diskless Computer
   (*e*) Object-linking and embedding
2. Explain the role of client in Client/Server computing and also explain the various services provide by client.
3. Explain the server functionality, in detail, for Client/Server computing.
4. What is Interring Process Communication (IPC) and what are services provided by IPC? Also explain various protocol used for IPC.
5. What was the primary motivation behind the development of the RPC facility? How does a RPC facility make the job of distributed applications programmers simpler?
6. Explain basic Interprocess Communication Mechanism. Explain Port Management and Message Passing in IPC.
7. What are the main similarities and differences between the RPC model and the ordinary procedure call model?
8. Why do most RPC system support call by value semantics for parameter passing?
9. Explain the difference between the terms service and server.
10. Explain the role of server in Client/Server computing and also explain the various services provided by server.