KTU NOTES

The learning companion.

KTU STUDY MATERIALS | SYLLABUS | LIVE NOTIFICATIONS | SOLVED QUESTION PAPERS

MODULE 3

CLIENT SERVICES

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.
- 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.

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.**

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.
- The communication between these processes can be seen as a method of co-operation between them. Processes can communicate with each other

through both:

- Shared Memory
- Message passing

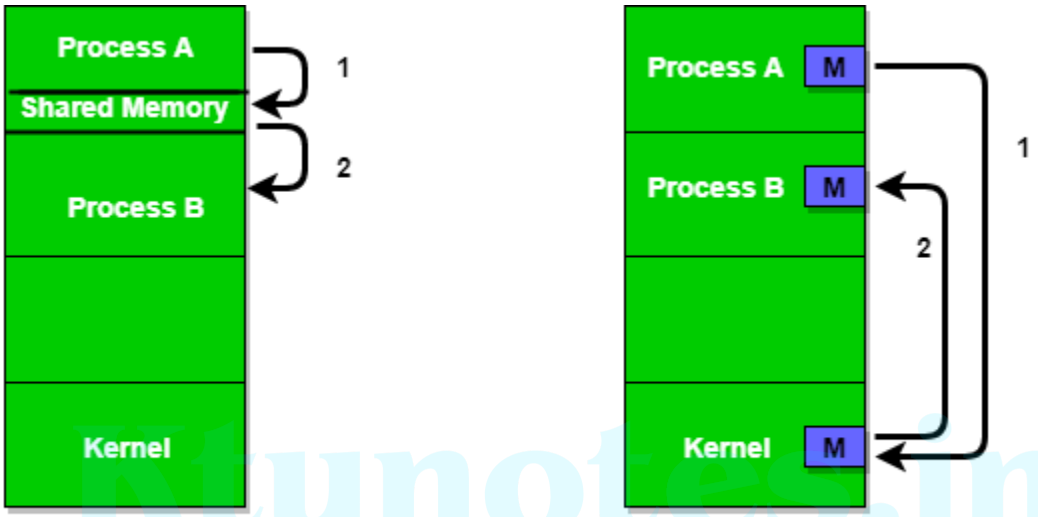


Figure 1 - Shared Memory and Message Passing

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. At least, there are two processes involved in an IPC.

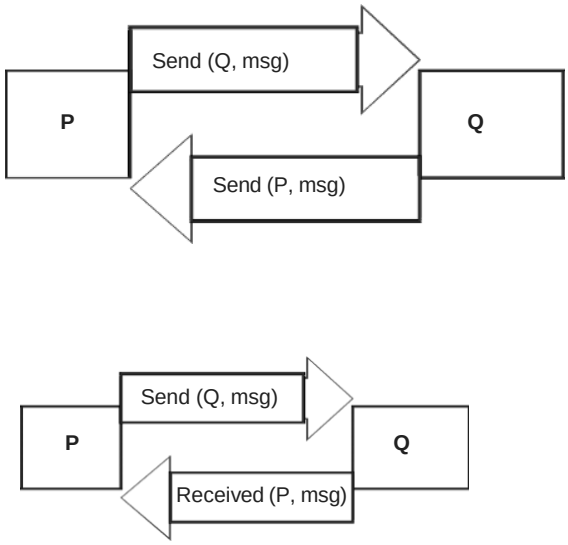


Fig.5.5: Message Passing

- 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.

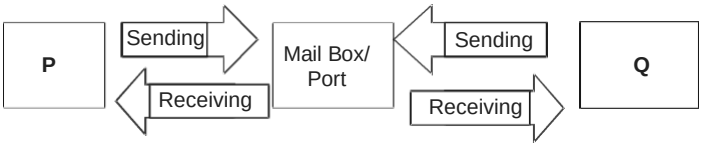


Fig. 5.6: Indirect Communication

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.

- Messages are directed and received from mailboxes (also referred to as ports)
  - Each mailbox has a unique id
  - Processes can communicate only if they share a mailbox

Communication between the processes takes place by executing calls to the **send** and **receive** primitive.These primitives, can be "blocking" and "non-blocking".

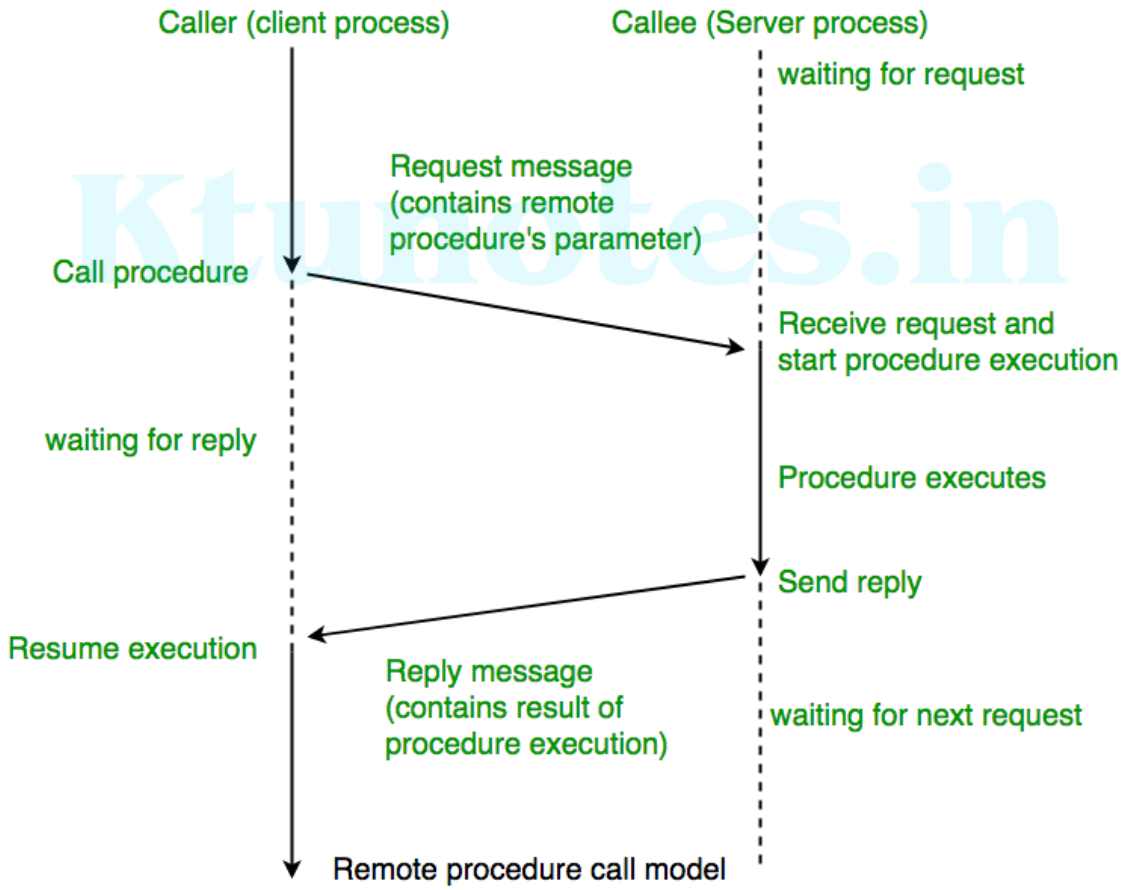
Properties of communication link

- Link established only if processes share a common mailbox
- A link may be associated with many processes
- Each pair of processes may share several communication links

- Link may be unidirectional or bi-directional

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.
- In this 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.

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.

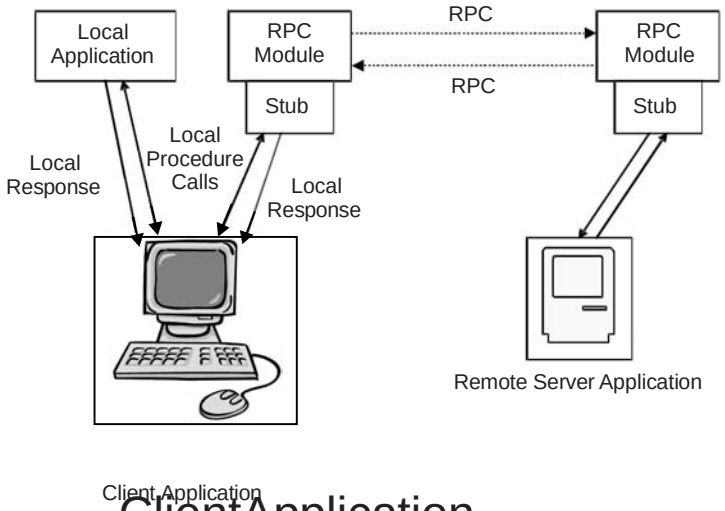
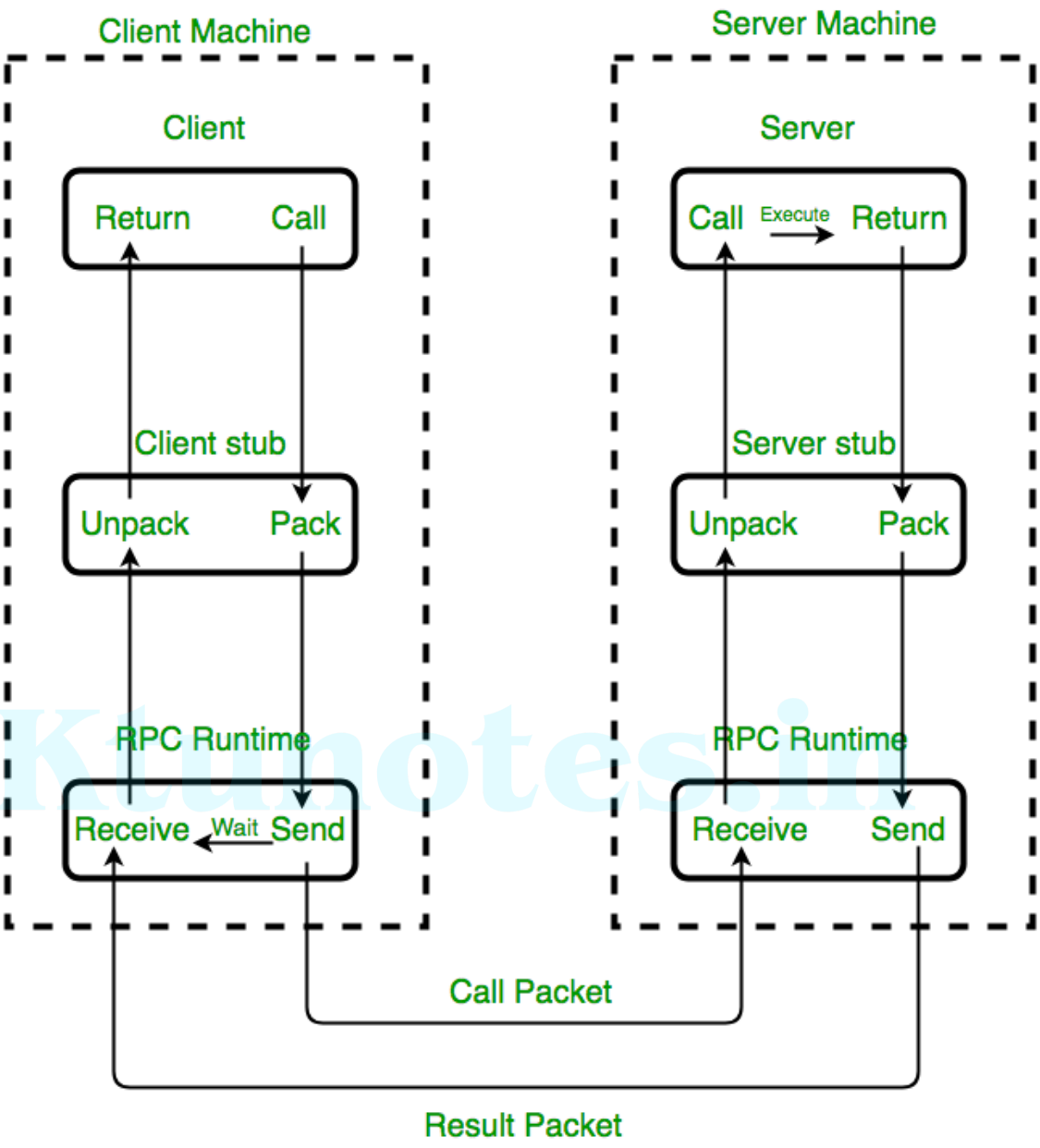


Fig. 5.7: RPC Mechanism

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. 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.



Implementation of RPC mechanism

During an RPC, the following steps take place:

1. The client calls the client stub. The call is a local procedure call with parameters pushed onto the stack in the normal way.
2. The client stub packs the procedure parameters into a message and makes a system call to send the message.

The packing of the procedure parameters is called marshalling.

3. The client's local OS sends the message from the client machine to the remote server machine.
4. The server OS passes the incoming packets to the server stub.
5. The server stub unpacks the parameters -- called unmarshalling -- from the message.
6. When the server procedure is finished, it returns to the server stub, which marshals the return values into a message. The server stub then hands the message to the transport layer.
7. The transport layer sends the resulting message back to the client transport layer, which hands the message back to the client stub.
8. The client stub unmarshalls the return parameters, and execution returns to the caller.

Pros and cons of RPC

Here are some of the advantages RPC provides for developers and application managers:

- Helps clients communicate with servers via the traditional use of procedure calls in high-level languages.
- Can be used in a distributed environment, as well as the local environment.
- Supports process-oriented and thread-oriented models.
- Hides the internal message-passing mechanism from the user.
- Requires only minimal effort to rewrite and redevelop the code.
- Provides abstraction, i.e., the message-passing nature of network communication is hidden from the user.
- Omits many of the protocol layers to improve performance.

On the other hand, some of the disadvantages of RPC include the following:

- The client and server use different execution environments for their respective routines, and the use of resources (e.g., files) is also more complex. Consequently, RPC systems aren't always suited for transferring large amounts of data.
- RPC is highly vulnerable to failure because it involves a communication system, another machine and another process.

- There is no uniform standard for RPC; it can be implemented in a variety of ways.
- RPC is only interaction-based, and as such, it doesn't offer any flexibility when it comes to hardware architecture.

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.
- Some important services are remote login, remote command execution, remote backup services, remote tape drive access and remote boot services, and remote data access .
- 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 consistent GUI applications.
- Remote command execution is when a process on a host create 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.
- In certain cases to run some application, some of the workstation clients 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.

- 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 .

- 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.

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 the case of client workstation may have several windows open on-screen at a time.

- It helps to provide notification messages of events that happened in 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.

- 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.

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 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 user can use to establish DDE links. This is true with MS Excel, Word, Powerpoint, 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 user issue a command in the client application.

- **Object Linking and Embedding (OLE)**
- Object Linking and Embedding two services collectively called as a single one, carried out with simple edit menu procedures.

Object linking and embedding (OLE) is a Microsoft technology that facilitates the sharing of application data and objects written in different formats from multiple sources.

Linking establishes a connection between two objects, and embedding facilitates application data insertion.

OLE object meaning is graphic,spreadsheet,msword, etc. that can be embedded into a document called the "**container application**."

If the object allowed to be edited, the application associated with it is called "**server application**".

- OLE is a software package that accesses data created from another application 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.
    - Linking adds a link in a document that points to source data stored somewhere else. Linked objects are stored in the document as a path to the original linked data, usually a separate file from the container document.

    - Embedding, on the other hand, adds one document directly to the other. Embedded objects are stored with the document that contains them.
-
- 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.
- 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).

CORBA=COMMON OBJECT REQUEST BROKER ARCHITECTURE

CORBA an object oriented architecture that provides a mechanism that allows various clients to share/call the object (applications) over a mixed networks.

CORBA is a standard developed by object management group(OMG) to provide interconnectivity among distributed objects.

CORBA is one of the important middleware solution which enables the exchange of information, independent of hardware platforms, programming languages and operating systems.

It is a design specification for ORB (object Request Broker) model, where an ORB provides a mechanism required for distributed objects to communicate with one another whether locally or on remote devices written in different languages.

The CORBA Interface Definition language allows the development of language and interface to distributed objects.

Using CORBA, application components can communicate with one another no matter where they are located or who has designed them.

It provides location transparency.

It can be termed as a software bus.

Data communication from client to server is accomplished through a well defined obeject oriented interface.

Through this object oriented technology developers can use various OOP features.
Objects separates client with server
Objects are described with interfaces – operations (methods) – attributes (properties)

CORBA is based on the distributed object computing model, which combines the concepts of distributed computing (client and server) and object-oriented computing (based on objects and operations).

In object-oriented computing, objects are the entities that make up the application, and operations are the tasks that a server can perform on those objects.
For example, a banking application could have objects for customer accounts, and operations for depositing, withdrawing, and viewing the balance in the accounts.

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.
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.

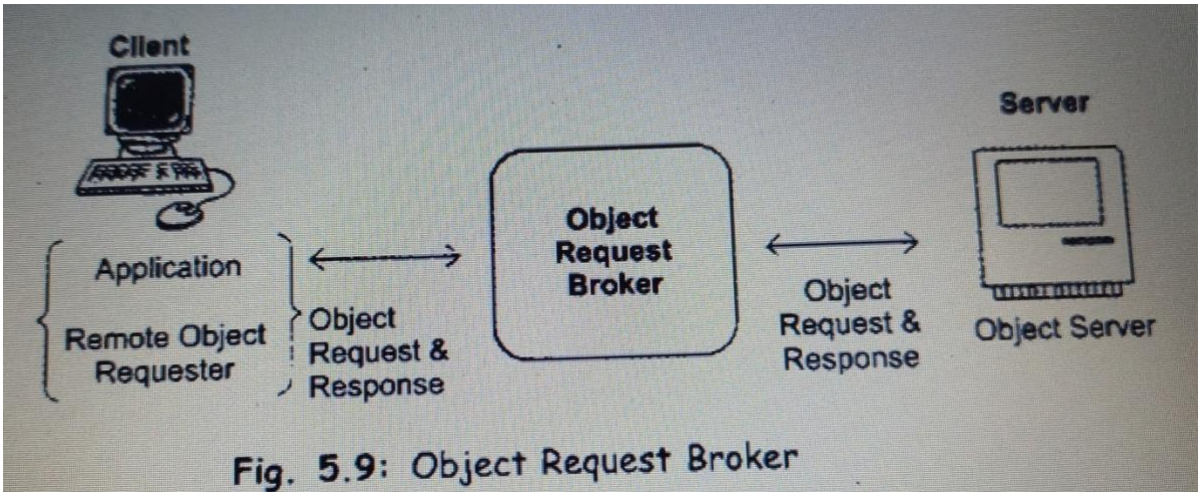
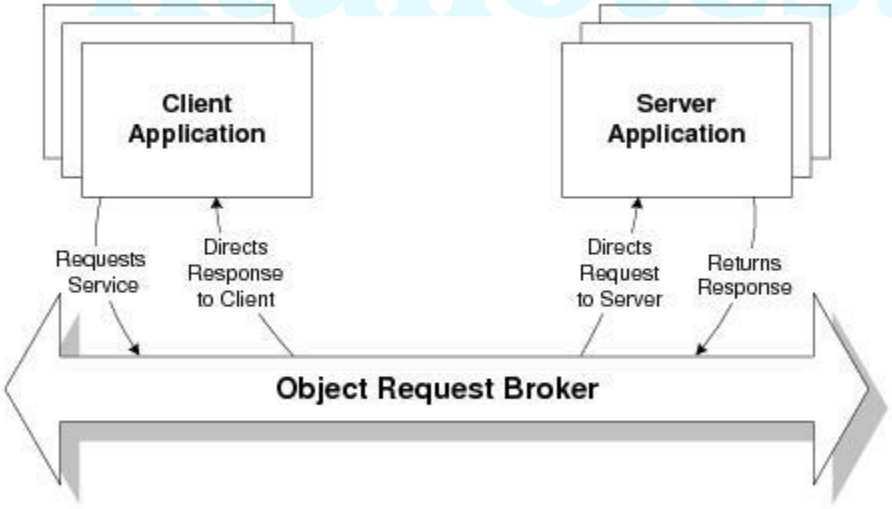


Fig. 5.9: Object Request Broker

In CORBA, a component can act as both a client and a server. A component is considered as a server if it contains CORBA objects whose services are accessible from some other CORBA object. Likewise, a component is considered as a client if it access services from some other CORBA objects.



Print/Fax Services

Client generates print/fax requests to the printer/fax machine without knowing whether they are free or busy.

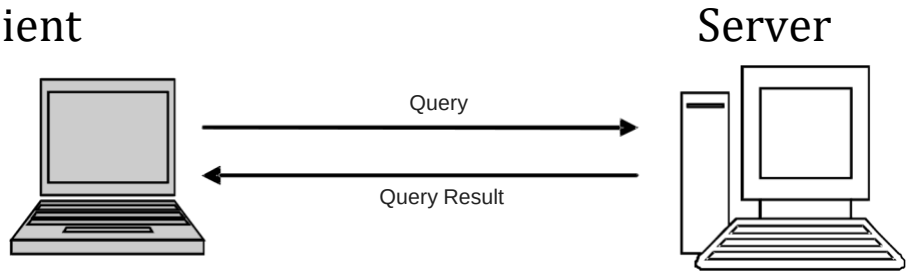
Network operating system helps the client to generate the requests. These requests are redirected by the NOS redirector software and managed by the print/fax server queue manager.

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.

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.



Execution of SQL

- Hence, most of the database requests are made using the SQL syntax. 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. Client applications 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. Some application programs (spreadsheet and statistical–analysis packages) uses the client server interface directly to access data from back-end server.

SERVER SERVICES

Refer the notes in the group