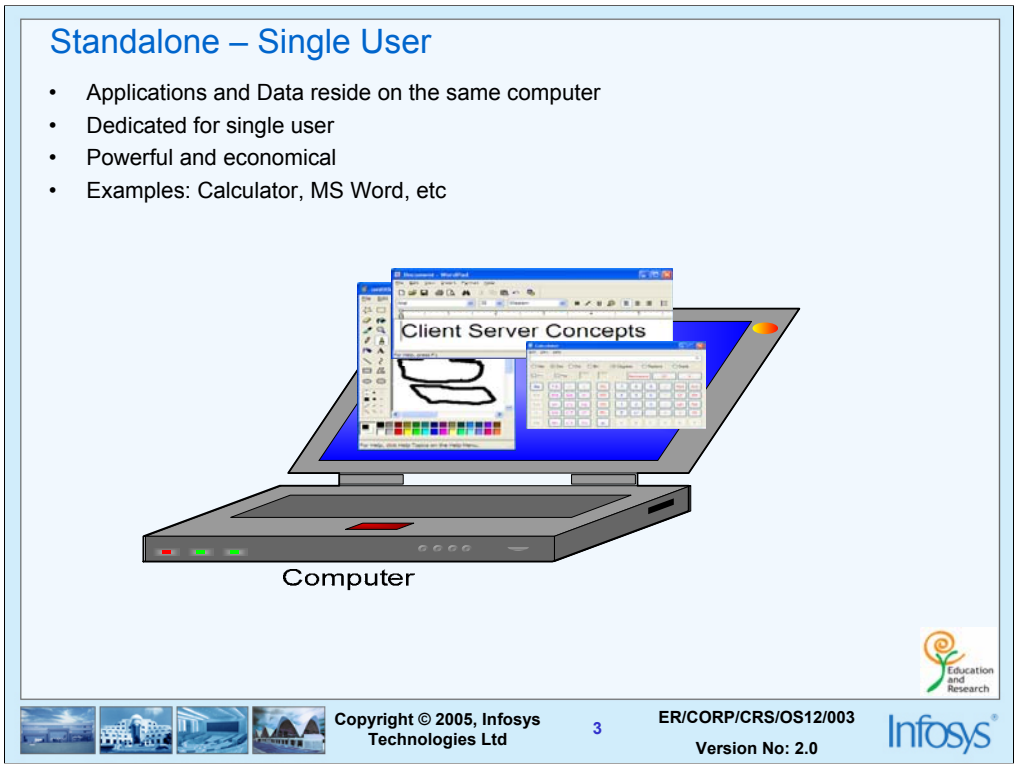
Infosys®
POWERED BY INTELLECT
DRIVEN BY VALUES

Client Server Concepts

Education and Research

Agenda

- Standalone (Single User)
- Host Centric (Multi User)
- Client/Server
- Network Operating System
- Middleware
- Application Layering
- Middleware: ODBC



Single user system computing is an approach that designs an application in such a way that all the applications and data reside on a single personal computer dedicated to the use of a single person.

- **Advantages**

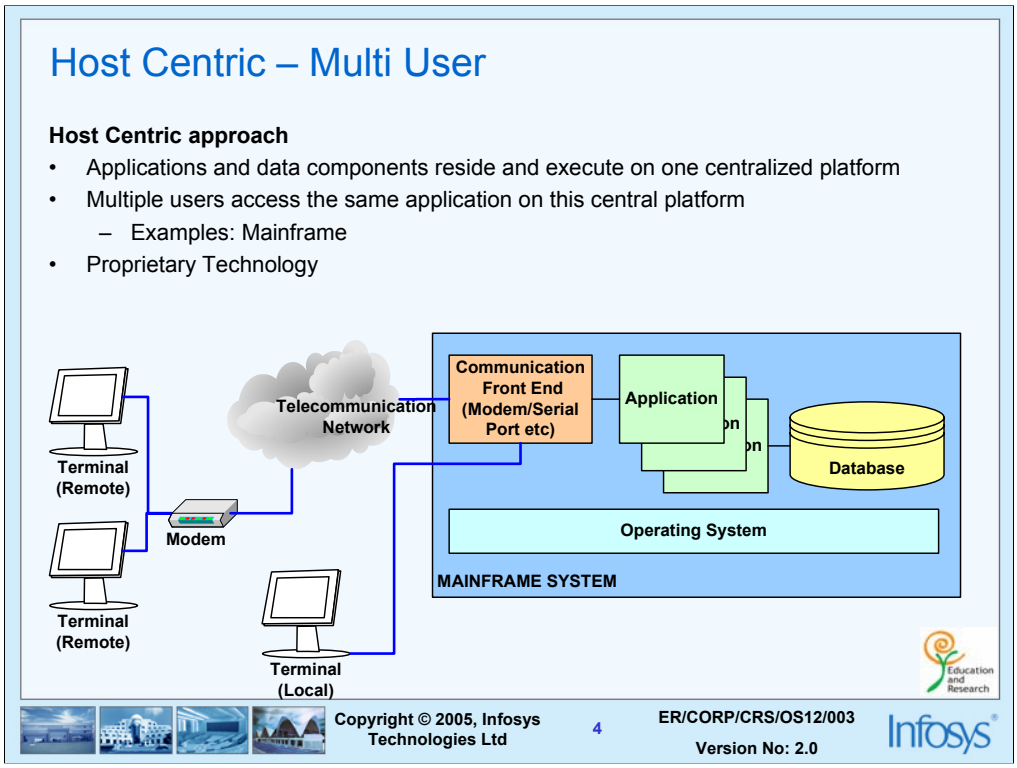
The advantages of using a stand-alone system are as follows.

1. A stand-alone system is considered to be a personal property. The user of the personal computer has a complete control of the working environment of the system. For example, if the user is not satisfied with MS-DOS operating system (OS), she can have Windows OS loaded in the system. If the person is not interested in the Windows OS then she can load Linux OS, and so on. Whatever the reason may be, the owner of the system has full authority to decide what she wants. In addition to the above-mentioned factors, third party applications are widely portable and usable in such systems. Thus, this serves as a good development environment.

2. Now a days, one can easily obtain powerful machines at a very cheap price.

Disadvantages

One of the major drawbacks in using stand-alone personal computers is that applications, data and resources, for example, cannot be *shared.*



Host based approach :

Host centric approach is an approach that design an application in such a way that both the functional and data components reside and execute on one centralized platform giving access to multiple users.

Advantages of Host Based approach

1. It is capable of providing simultaneous access to a very large number of users. The number of users accessing the system range from a few hundreds to few thousands.
2. These systems are stable, reliable and well supported.
3. Mainframe systems provide a one-stop-shopping which means that all the components right from hardware, software and network are made available by one vendor.

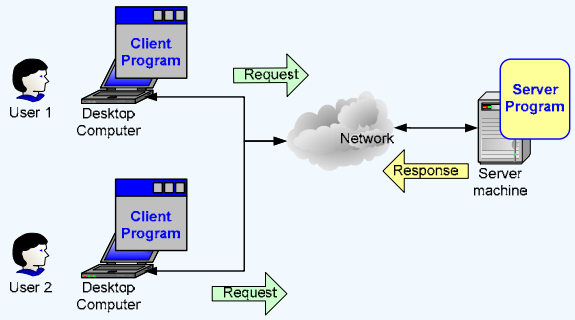
Limitation/Disadvantages of Host Based approach

1. Usually, mainframe technologies are proprietary and are incompatible across multiple vendors.
2. The cost associated with procuring and maintaining these systems are extremely high.

Client/Server Technology

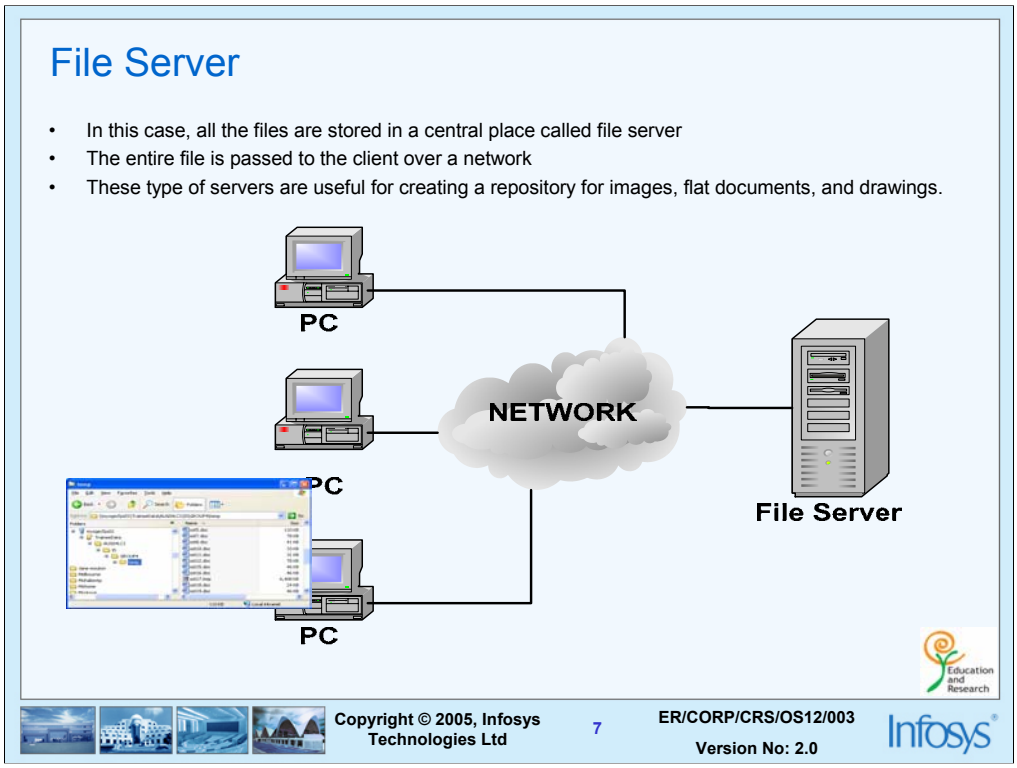
Client/Server Technology

- Client/Server technology involves the client and server, both having processing capabilities.
- The client request for a service and the server processes this request and sends a reply.



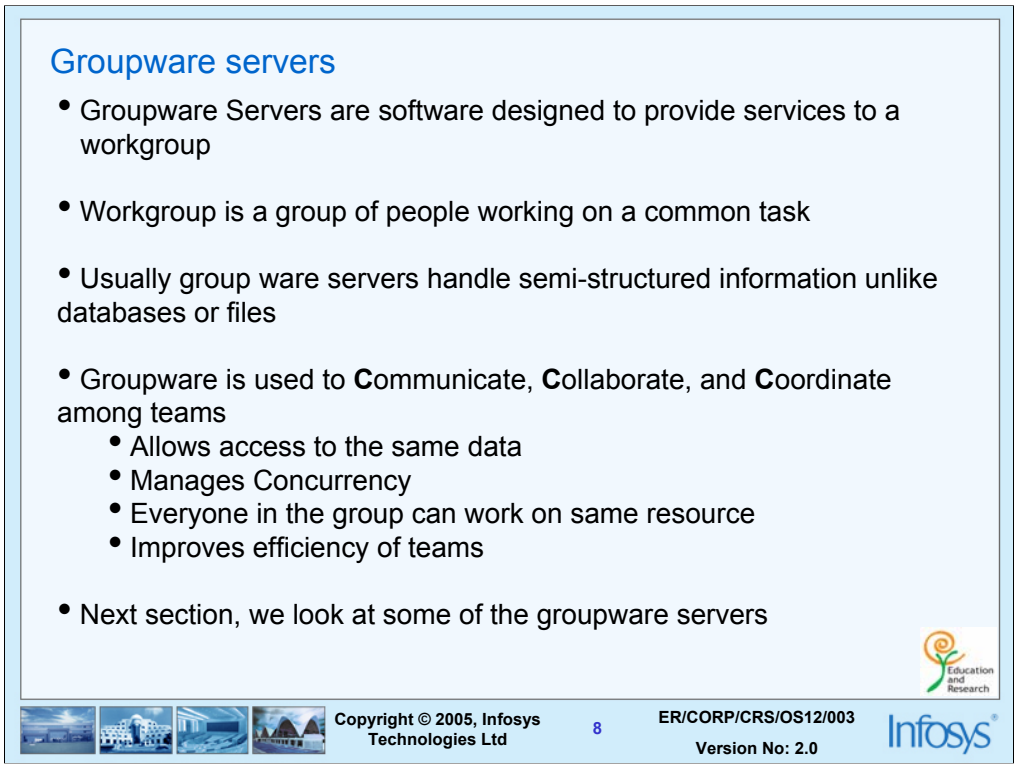
Types of services provided by different servers

- File Servers
- Groupware Servers
  - Mail Servers
  - Newsgroup Servers
  - Calendar Servers
  - Workflow Servers
- Database Servers
- Transaction Servers



File Server

In this case, all the files are stored in a central place called file server. The user sitting at the client-end places a request over a network for a file from the file server. The entire file is passed to the client over a network. File server also handles multiple requests. These type of servers are useful for creating a repository for images, flat documents, and drawings.



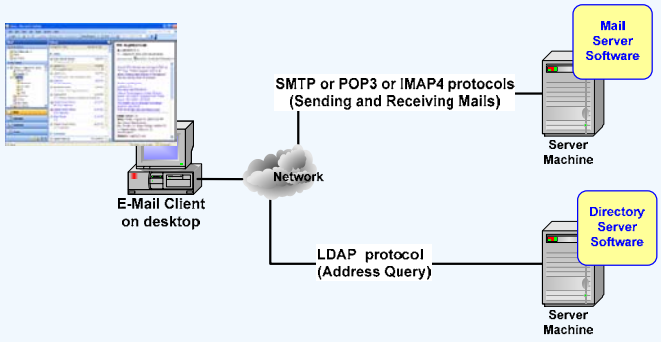
The function of the groupware sever is to do the management of semi-structured information such as text, mail, workflow, and bulletin boards. This kind of information is called as semi-structured because there is no specific structure as in case of relational database tables.

Relational database tables have normal forms and satisfy certain properties such as loss less decomposition and controlled redundancy.

Groupware servers place people in direct contact with other people to maintain workflow.

Groupware servers - Email services

- E-Mail: Messages that are sent electronically via computer networks
- A mail server provides e-mail services to computers on a network
  - Mail can be accessed using E-mail clients
  - Mail service can also be used by business applications as well
  - Example: When you apply for leave, a mail is sent to the approving manager by the leave system
- A mail client uses POP3 or IMAP4 protocol to connect to a Mail server
- Mail Client uses LDAP protocol to access the e-mail address book



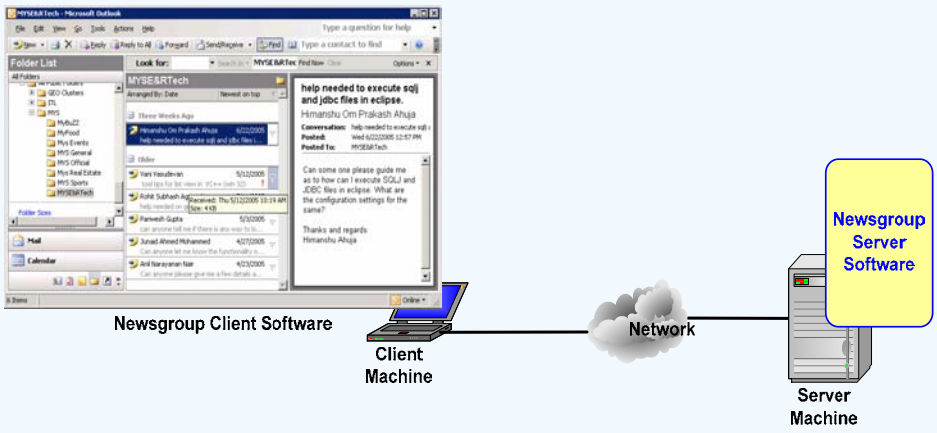
POP3 : Post Office Protocol version 3

IMAP : Internet Message Access Protocol version 4

LDAP : Lightweight Directory Access Protocol

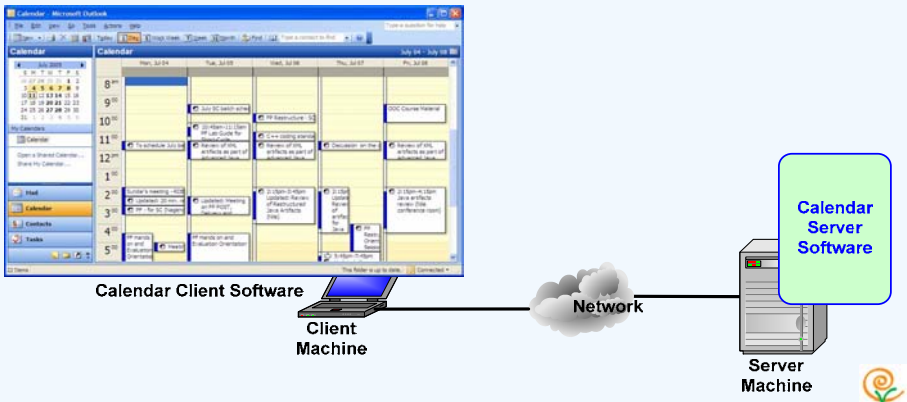
Groupware Servers - Newsgroup server

- Newsgroup server provide bulletin-board like services



Groupware servers - Calendar Services

- Calendar servers provide organization wise scheduling and Calendar services
  - Calendar servers can be used to setup meetings, reminders and scheduling events
- Team members can view others calendar as well
  - Applications can also use Calendar Services for automatic scheduling etc
  - Can be used to even block resources (people, conference rooms etc)



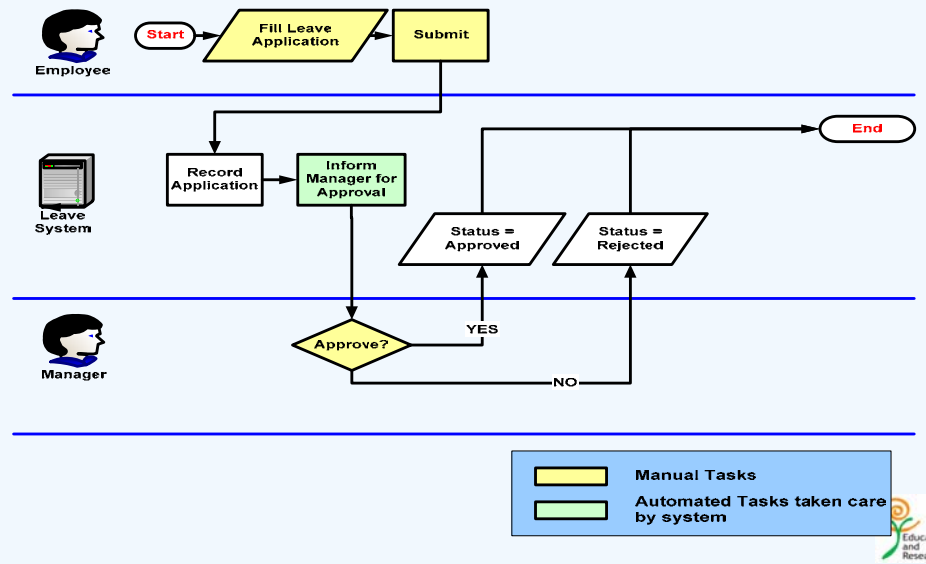
Groupware Servers – Workflow services

- Workflow: A workflow is a set of interdependent tasks that occur in a specific sequence.
- All these tasks together complete a business process
- Workflow services are predominantly used in business application
  - Many business transactions require workflow
  - In any application, Workflows can be part manual and part automated
- Example:
  - Deposit of a cheque in bank does not end with the deposit
  - After the initial step of deposit, the cheque goes in for clearing from the issuing bank
  - If cheque passes, amount is transferred to payee's account from the payer's account
  - To transfer amount from payer's account, there are several steps (Both physical and electronic)
  - If cheque bounces, then action taken is different.
  - All these tasks put together complete the deposit workflow

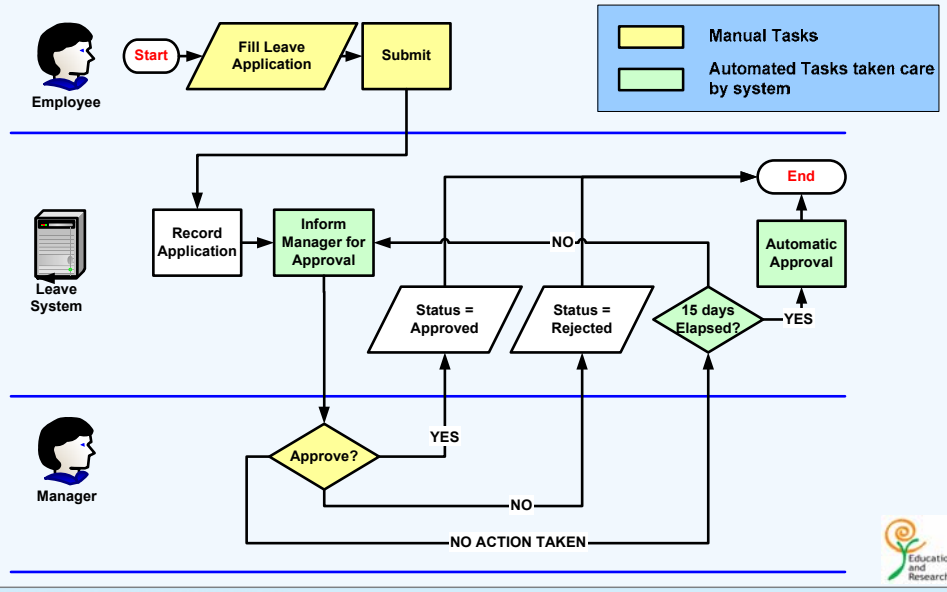
Groupware Service - Workflow

- Workflow servers offer workflow services
  - Can create a workflow which consists of tasks
  - Can provide decision making points in the workflow
  - Programmer can define the workflow paths
  - Business Applications can use these workflows

Groupware Servers - Workflow example - Leave System

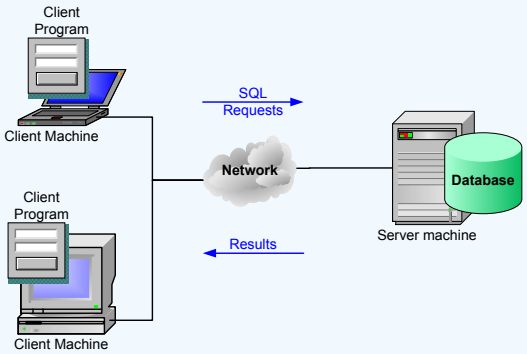


Groupware Servers - Workflow example - Leave System



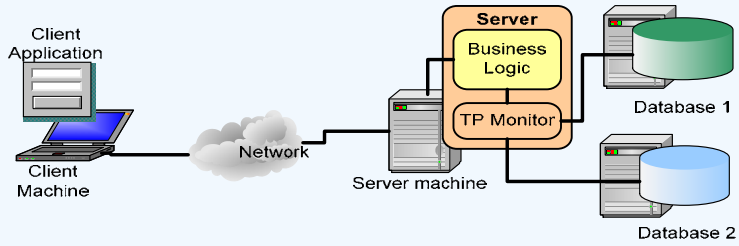
Database Server

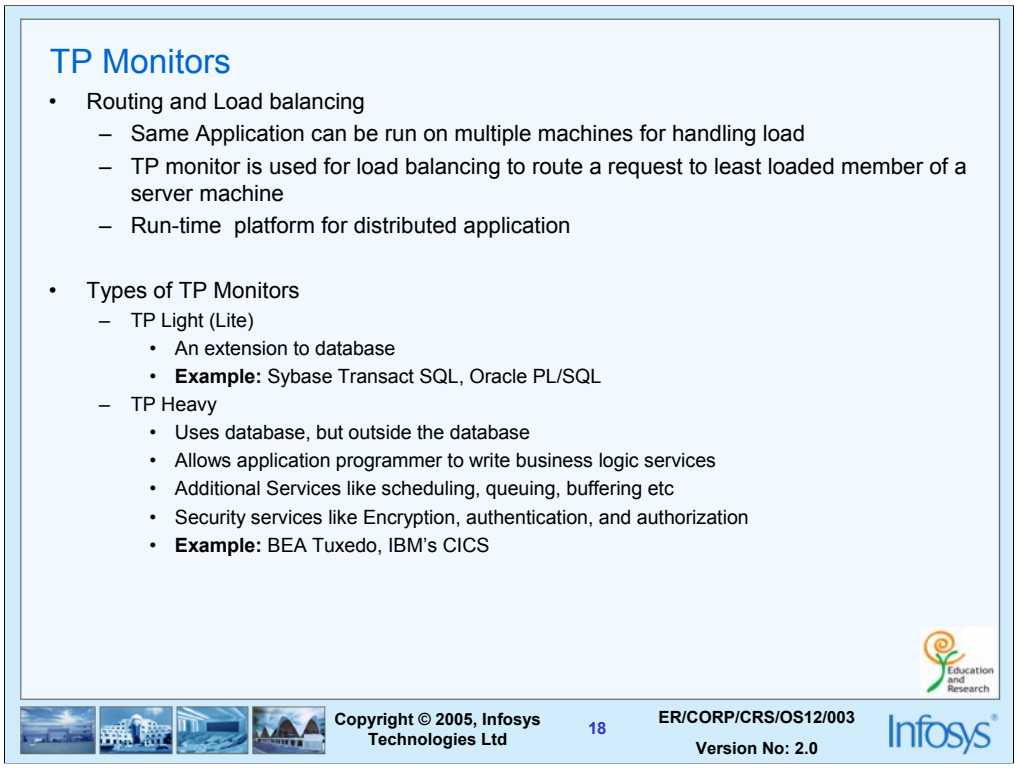
- With Database servers the client passes SQL requests as messages to the database servers .
- The code that processes the SQL request and the data reside on the same server.
- The server uses its own processing power to find the requested data. Play an important role in data warehousing.



Transaction Servers

- Client Invokes remote procedures/services that reside on the SQL database engine
- TP Monitor (Transaction Processing Monitor) provides Transaction Management
  - These remote services execute Group of SQL statements
  - The Network exchange consists of single request/reply message.
  - These SQL statements either all succeed or fail which is ensured by transaction server with TP monitors.
- Server side business logic typically consists of database transactions.





Transaction processing monitors

Transaction processing is supported by programs called *transaction processing monitors* (TP monitors). TP monitors perform the following three types of functions:

System runtime functions: TP monitors provide an execution environment that ensures the integrity, availability, and security of data; fast response time; and high transaction throughput.

System administration functions: TP monitors provide administrative support that lets users configure, monitor, and manage their transaction systems.

Application development functions: TP monitors provide functions for use in custom business applications, including functions to access data, to perform intercomputer communications, and to design and manage the user interface.

TXSeries provides two TP monitors, Customer Information Control System (CICS) and Encina. In CICS, the transaction processing monitor is implemented by developing one or more *CICS regions*, individual administrative units that support multiple concurrent application programs. In Encina, the transaction processing monitor is implemented by developing one or more Monitor application servers. The Monitor application servers run in a single administrative unit called a Monitor cell.

Both CICS regions and Encina Monitor application servers do the following:

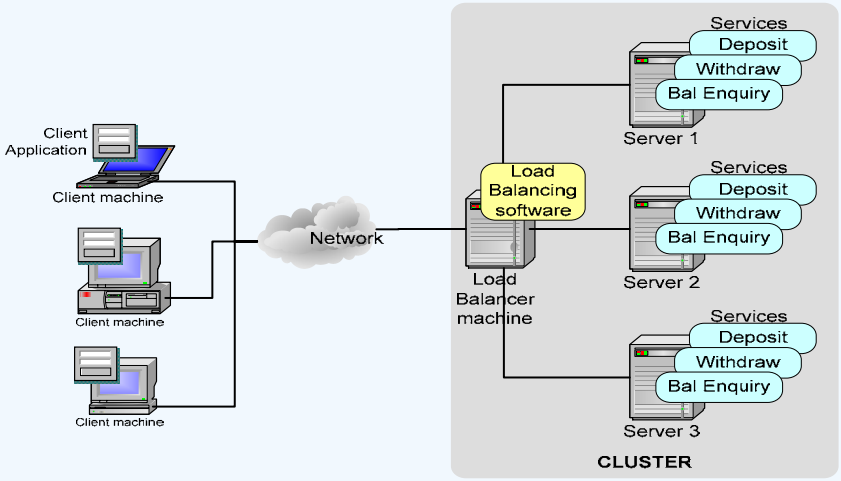
Perform work requested by one or more clients. For example, a user application running on one machine (the client machine) requests work to be done on another machine (the server machine). Typically, the region or application server accesses some data, applies some business logic to it, and then replies to the client. Such service is provided by running one or more programs on behalf of a transaction.

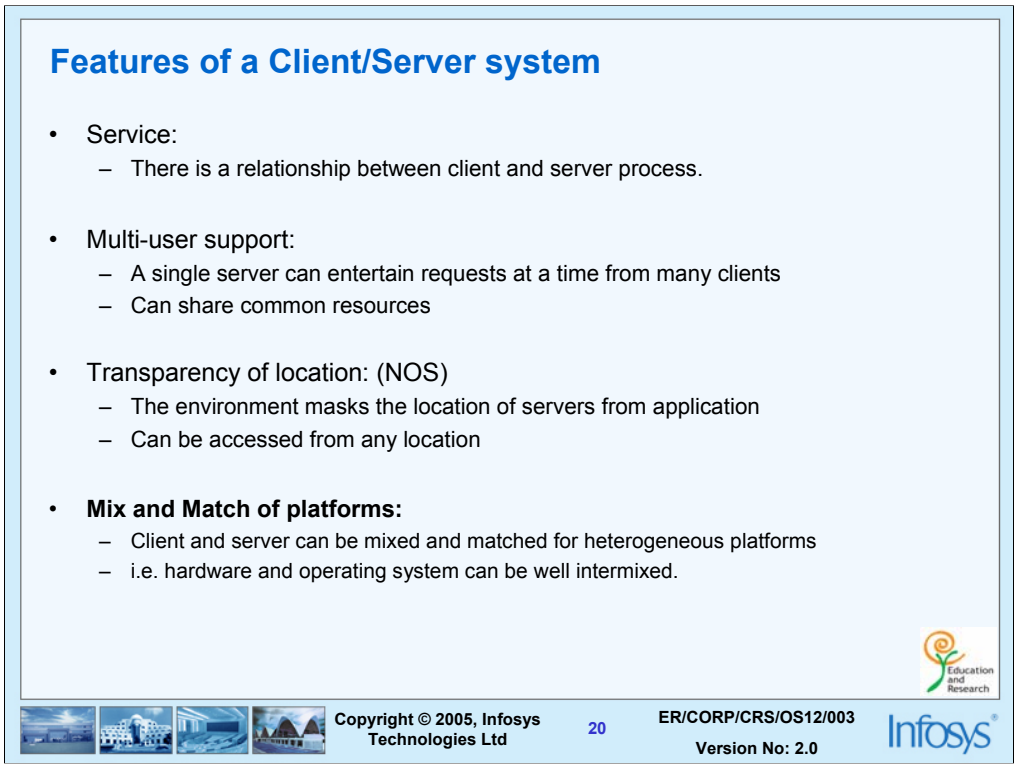
Maintain and use a pool of multithreaded processes, each of which provides a complete environment for running a transaction. In CICS, such processes are called *application servers*. In Encina, Monitor application servers run individual instances of server code called *processing agents* (PAs).

Coordinate all the facilities needed by its application servers. For example, they coordinate the security of the application servers, obtain data and storage that they need, and log their transactions. Among other advantages, multiple CICS regions and Monitor application servers can be used to provide a distributed transaction processing environment for greater throughput and management of workload.

Subcontract many services to other servers better able to do the work but provide extra services needed for integrated transaction processing. For example, they can use Structured File Server (SFS) files or DB2(R) databases to store and manage user data. They also provide services to locate and interface with the resource managers, record ongoing changes to data, and coordinate the update of data across multiple resource managers.

TP Monitor


Client Application
Client machine
Client machine
Client machine
Network
Load Balancing software
Load Balancer machine
Services
Deposit
Withdraw
Bal Enquiry
Server 1
Services
Deposit
Withdraw
Bal Enquiry
Server 2
Services
Deposit
Withdraw
Bal Enquiry
Server 3
CLUSTER



features of a client/server system

A typical client/server system has a event driven graphical user interface on the front-end and a relational database management system serving data from the back-end. Apart from this, a client/server system has some distinctive features.

Service: There is a relationship between client and server process. Typically, the client initiates the request and the server responds back to the request by providing the service. This means there is a clear de-marcation between client and server based on the notion of service.

Resource sharing: A single server can entertain many requests at a time from the clients and thus can regulate access to shared resources.

Transparency of location: In a client/server environment, it doesn't matter which client you login. The environment masks the location of servers from application.

Mix and Match: Client and server can be mixed and matched for heterogeneous platforms i.e. hardware and operating system can be well intermixed.

Loosely coupled system: Client/server systems belong to the class of loosely coupled systems in the sense that they interact through message passing for requests and replies. In contrast, tightly coupled systems interact with shared memory.

Features of a Client/Server system

- Scalability:
  - Scalability of a server or an application is the ability to provide service without degradation in performance as load increases
- Client Server applications provide scope for scalability of application

Network Operating System (NOS)

- Many of the network based services like File services, Mail Services etc. are analogous to services provided in a system by the OS
- A NOS is an extension to OS that provides transparent access to resources on a network.
  - Examples
    - Novell Netware
    - UNIX and its variants
    - Windows NT, Windows 2000 Adv Server, Windows 2003 Adv Server

Each system has its own operating system. Then, what is the need for another operating system (NOS)? Yes, there is a need. The main function of the NOS is to provide a transparent environment in the network context.

Here goes the explanation of all NOS services:

• Location transparency: A user need not be aware of the location or physical address of a resource. This can be achieved through aliases. For example, if InfERno is the name of the server machine then it suffices to use an alias at the clients side stating that InfERno is an alias name for 204.4.56.12 provided there is a name server.

• Logon transparency: This means a user should be able to login to more than one systems through a single password which works on all the servers and for all the services on the network which the user has access to.

• Replication transparency : A user need not be aware of multiple copies of a resource. Sometimes a resource is replicated due to technical reasons. For example, FTP servers are replicated according to country area-wise or sometimes these are replicated to keep the system near 100% uptime. This is achieved through mirror images of the sites.

• Local versus remote access transparency : There should be no distinction between a service at a local and remote site.

• Directory services : It provides a lookup service which translates an object name nto a physical network address. For example, in Unix, a computer name InfERno gets translated to 204.4.56.12.

•Time services : This service supplies time with a tolerance so that time can be matched across systems. This is also sometimes integrated with other facilities such as RPC and security to issue time stamp.

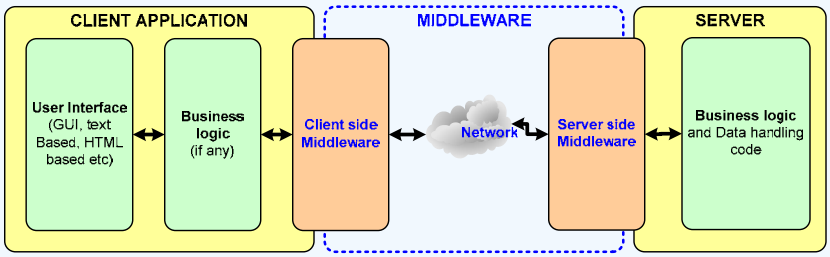
•Failure handling : It provides smooth and fault free operations by detecting and isolating the faults in the requests and responses without the knowledge of the user.

Features of NOS

- Location transparency: Can access services anywhere
  - XNet (Extranet) in Infosys can be used to log into Sparsh from anywhere in the world through a secure network
- Logon transparency
  - Same User Id can be used across multiple applications
  - User Authentication done using a common and central system
    - Examples: Win NT Domain, Kerberos in UNIX world
- Replication transparency: Automatic backup and recovery mechanisms
  - One server failure will not result in entire application going down
  - Since difference services are running on different physical servers, failure in one will not impact on all operations of applications
    - Example: If Mail server goes down, only mail service is affected
  - Mirror servers are used to avoid failure in critical applications
- Resource Sharing: Sharing of resources like files, folders, printers etc
- Directory Services: Single point for accessing all the resources in a network
  - Example: Network Neighbourhood, LDAP Protocol used to access directory services

Middleware

- Middleware is software glue between Client and Server, which helps the communication between the client and the server

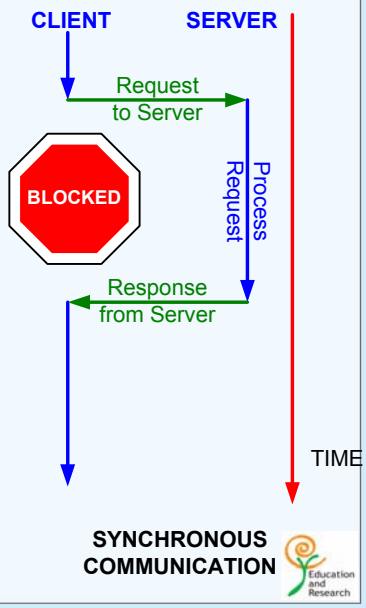


Types of Middleware

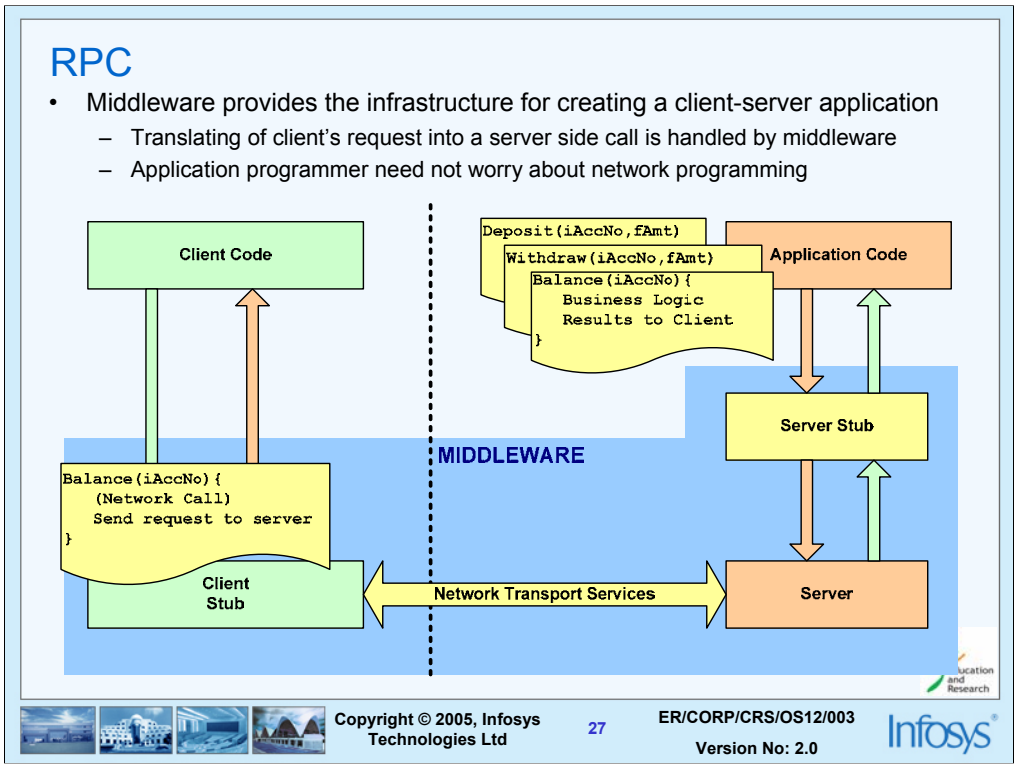
- Peer to Peer interfaces (P2P)
  - Indicates End to End communication in the same protocol
  - Since the protocol is symmetrical sometimes it is called program to program.
  - In this type of network each workstation has equivalent capabilities and responsibilities.
  - Example: File sharing systems (Napster etc), Instant Messengers
- Remote Procedure Call (RPC)
  - Middleware that allows a computer program running on one host to cause code to be executed on another host remotely
  - Programmer doesn't have to explicitly code for achieving this. Middleware provides libraries or functions to achieve this
- Message Oriented Middleware (MOM)
  - Uses a queue like structure to pass messages between two hosts
  - Asynchronous
  - connectionless, asynchronous transactional message store-and-forward capability.

Remote Procedure Call (RPC)

- Client process invokes a remotely located procedure in a server process
  - Server process may or may not reside on the same physical machine as the client
- When a client invokes a remote procedure
  - The remote procedure executes as if it is a local function call
  - Sends the response back to the client process
  - Involves network communication for data transfer
- RPC uses synchronous communication between client and server
  - Client is blocked until server responds (Client is typically in a wait state till the responds)
  - Proceeds further only after the server responds.



SYNCHRONOUS COMMUNICATION



Let us look at the sequence of steps that take place in a remote procedure call.

The client issues a call to the local procedure. This is also called as client stub.

The stub ***packs*** the message by determining the network address of the server and binds to it.

The runtime libraries associated with the stubs do this binding.

1. The network messages are then sent to the remote site through network transport services.
2. The messages are sent over the communication media and are received by the other end.
3. The server network system informs the server stub that a request has arrived for it.
4. The server stub gets the network message and *unpacks* it, executes into a local procedure call format and executes the call at the server end.
5. The server, after having executed the procedure, develops a response that is sent to the server stub.
6. The server stub translates the response into a message on the network.
7. The server network system sends the response back to the client network system.
8. The client network system sends the response message to the client stub.
9. The client stub receives the response message, translates the message into call response and sends back to the client.

Here are some of the issues in RPC.

Who writes stubs?

Stubs can be generated automatically or by a programmer. For a programmer-generated stub, the RPC provides a set of functions, which can be used to construct a stub. Such a mechanism is used in Sun RPC.

How are the parameters passed across procedures?

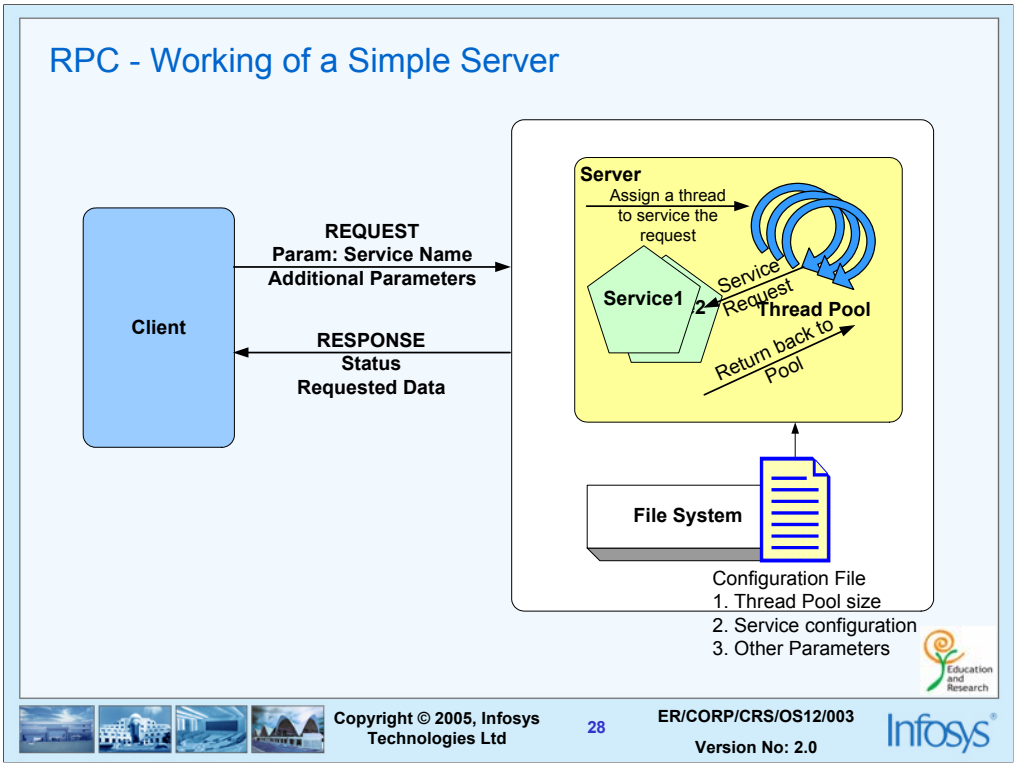
Parameter passing has to be done by value.

A parameter passed by address makes no sense because that address on a different machine has no meaning. For this reason it is not allowed. **How and when is the binding done?**

Binding refers to establishing a connection between a client and server. Binding requires two major items a) locate the host b) locate a server on the host. In some sense, where should the directory of hosts and servers be located. This can be solved by either using a central or a distributed directory.

How is the data format handled in cases of different machines?

A common approach used in RPCs is an intermediate data format called eXternal Data Representation (XDR).



Overview of working of a Server

The Setup: Consider the case of a simple Client Server. A Server is usually a process which is running in the background as a daemon process (Not Interactive). A server usually has a configuration file which it uses to read the parameters on startup.

Typical configuration parameters can be the Number of threads in the Thread-Pool, The mapping of services to physical executables or libraries etc. A server usually offers one or more services to clients.

Most servers create a Thread Pool on start-up. The server by default does not do any processing unless a client sends in a request. On receiving a request, the server assigns one of the threads in the pool to process the task and continues to wait for further requests. The thread executes the code for the requested services and also sends the response back to the client. It then returns back to the thread pool.

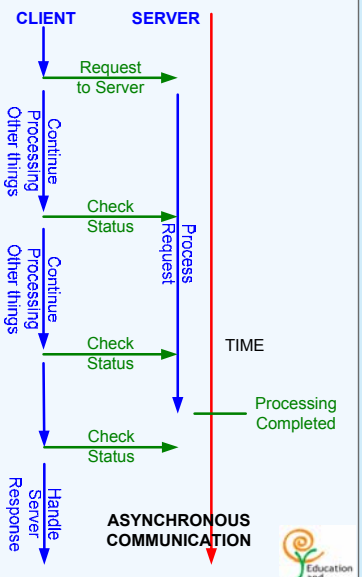
Having a thread pool with ‘n’ number of threads in the above scenario allows the server to service at least ‘n’ concurrent requests. Increasing the number of threads in the thread pool will increase the number of concurrent requests the server can handle. However, every physical hardware machine has a finite limit on the number of threads that can be spawned beyond which, performance gets degraded.

Tuning of Server: The performance of a server can be improved or kept optimal for various loads by modifying the parameters in the configuration file carefully. This process is also known as “Tuning of the Server”. The concepts explained here will help better understand the load, scalability and Load Balancing aspect of server side applications, later in this session.

Message Oriented Middleware (MOM)

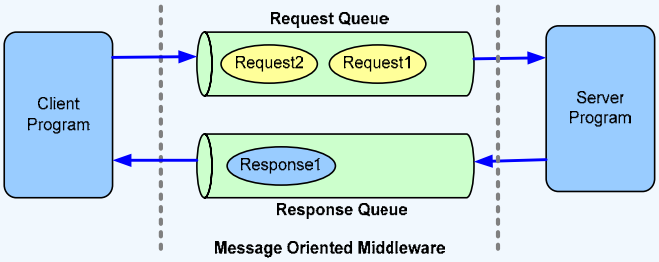
- A client/server infrastructure that is asynchronous in nature
  - Client does not wait in a blocked state till the server responds
  - Client can do other tasks while the server processes the request
- Interoperability: Allows the application to be distributed over multiple heterogeneous platforms.

  Examples:
  - A mainframe talking to a non-mainframe applications
  - IBM MQ Series is one such Message Oriented Middleware, JMS in J2EE Applications
- Reduces complexity of developing applications that span multiple operating systems
- Masks the application developer from the details of the various operating system and network interfaces



MOM…

- Client initiates communication by putting messages on to the Message queue
- No interaction with server process (de-coupled), thus achieves asynchronous communication
  - Client can continue with other work after putting a message in queue
  - Message queues also provide temporary storage when the destination program is busy or not connected
- Server uses another queue to send responses back



Comparing MOM and RPC

Feature	MOM	RPC
Client/Server time Relation Ship	Asynchronous	Synchronous
Sequencing	Not required	Servers come up first
Style	Queued	Call return
Load Balancing	Not Applicable because it is de-coupled	Required because of synchronous communication
Performance	Slow	Fast
Metaphor	**Post Office Like**	**Telephone**

Application Layering

Splitting an application in a Client/Server Architecture

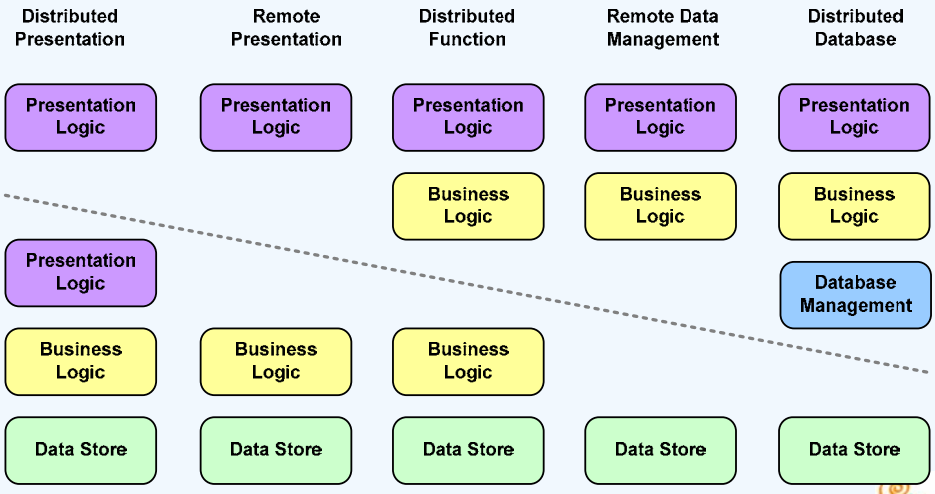
An application can be split from two perspectives

1. From the perspective of the different functionalities
   - Presentation Logic
   - Business Logic
   - Data Source

2. From the perspective of the concentration of functionalities
   - Fat Client
   - Fat Server

Its all about how you split the client/server applications into functional units that you can assign either to client or to one or more servers.

Application Partitioning - Gartner's Group Model



Gartner Group Model

- **Distributed Presentation Logic**: A part of the presentation layer goes to the client. The other part of presentation layer, business and data management layer goes to the server.
- **Remote Presentation Logic:** Only the presentation layer goes to the client. Business and data management layer goes to the server.
- **Distributed Business Logic**: The presentation layer and a part of the business layer goes to the client. The other part of business layer and data management goes to the server.
- **Remote Data Management**: Both the presentation and business layer goes to the client. The data management layer goes to the server.
- **Distributed Database :** Presentation layer, business layer and a part of the data management layer goes to the client. The other part of the data management goes to the server.

Fat Client and Fat Server

- **Fat Client:** Most of the application's code resides on client side
- **Fat Server:** Most of the application's code resides on the server side
- What is Thin Client and Thin Server?

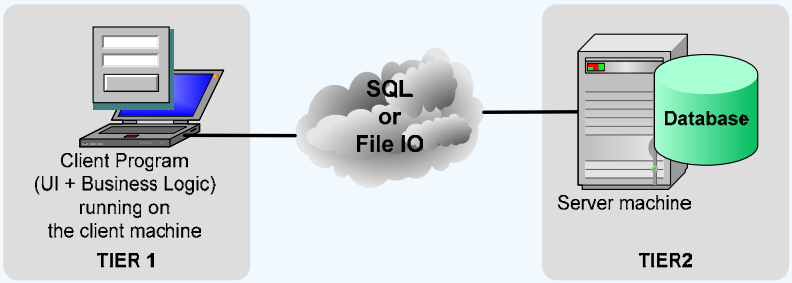
Thick/Fat refers to extent of functionality

2 Tier Application

•Two-tier applications remain the most common client/server architecture.

•The entire application is decomposed into two sets of services.

•The client combines UI services + business services and the other data services.



Architecture can be layered as

2 Tier Application:

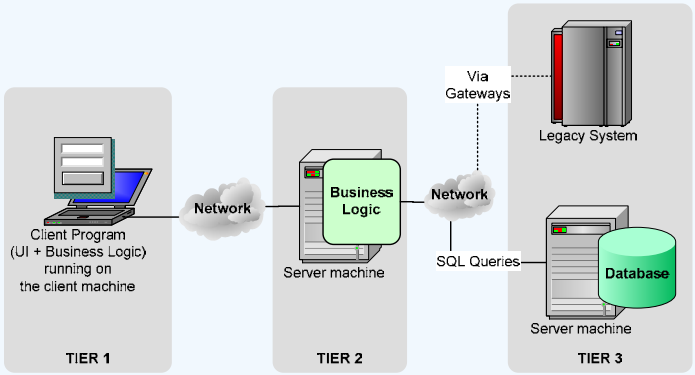
UI and business logic reside on Client and data reside on database server

3 Tier Application:

UI resides in Client business logic resides in business logic server and data in database server

3 Tier Application

- It decomposes an application into three sets of services: UI, business, and data.
- Business logic is moved to an application server
- Shared data to a database server



2 Tier Vs 3 Tier

	2Tier	3 Tier
System Administration	Complex	Less Complex
Security	Low	High
Encapsulation of Data	Low	High
Performance	Good	Good
Scalability	Poor	Excellent
Application Reuse	Poor	Excellent
Legacy Application Integration	Difficult to implement	Yes (Via Gateways)
Hardware Architecture Flexibility	Limited	Excellent

2Tier Vs 3 Tier
System Administration
2Complex: More logic on the client to manage
3Less Complex: The application can be centrally managed on the server

Security:
2Low : data level security
3High: fine tuned at the service method

Encapsulation of Data
2Low : tables are exposed
3High:Client invokes methods or services

Performance:
2 poor: Many SQL statements are sent over the network ;
3:good :only service requests and responses are sent between the client and server

Scalability:
2 Poor: Limited
3 Excellent : Can distribute load across multiple servers

Application Reuse:
2Poor: monolithic application on client
3Excellent: Can reuse services and object

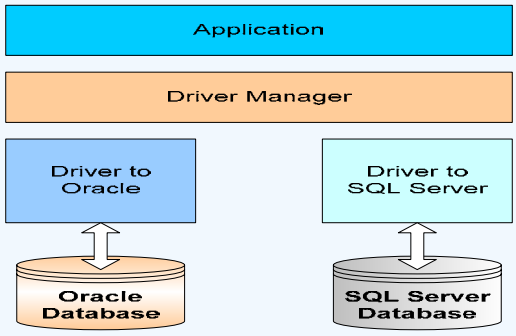
Middleware: Open Database Connectivity (ODBC)

- ODBC is an API by Microsoft that allows applications to access a database by making SQL calls
- It is a call level interface where a single application can access remote databases under disparate DBMS such as Informix, Oracle, Sybase
- ODBC relies on data servers to convert the ODBC calls into different database formats
  - That means Oracle drivers are needed to access Oracle databases
  - Sybase drivers are needed to access Sybase databases and so on

ODBC

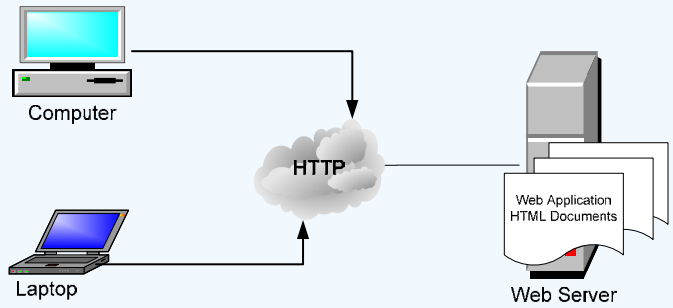
ODBC has four components

1. Application : Calls ODBC Functions ,Retrieves ,Processes and Reports the results
2. Driver Manager :Loads drivers, uses odbc.ini to map data source name to a specific DLL
3. Driver : Establishes a connections to datasource translates requests and responses by handling standard errors.
   - Different Databases have different drivers
4. Data source



Web Technologies

- World wide web is also based on Client-server technology
  - One of the most popular and dominant client server technologies today
- (Covered in detail in Introduction to Web Technologies course)



Summary

- Standalone (Single User)
  - Early computing systems, standalone single-user applications
  - Examples: Calculator, text editor etc
- Host Centric (Multi User)
  - Mainframe systems, UNIX systems with terminals
  - Reporting applications, banking etc
- Client/Server
  - Client and Servers both have processing capabilities
  - Different types of services provided by servers
- Network Operating System
- Middleware
  - Peer to Peer
  - RPC
  - MOM
- Application Layering
- Middleware: ODBC

Thank You!