

CLIENT SERVER COMPUTING

TEXT BOOK: “Client/Server Survival Guide”

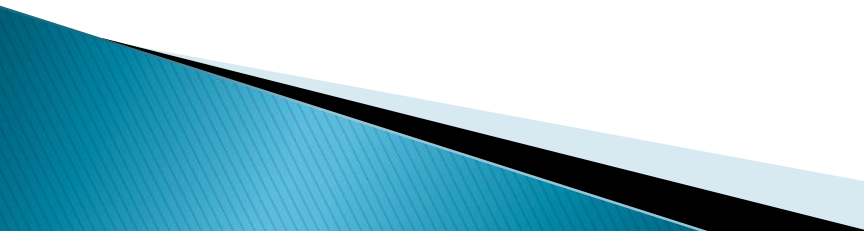
Authors: Robert Orfali, Dan Harkey & Jeri Edwards,
Wiley INDIA Publication, 3rd Edition, 2011.

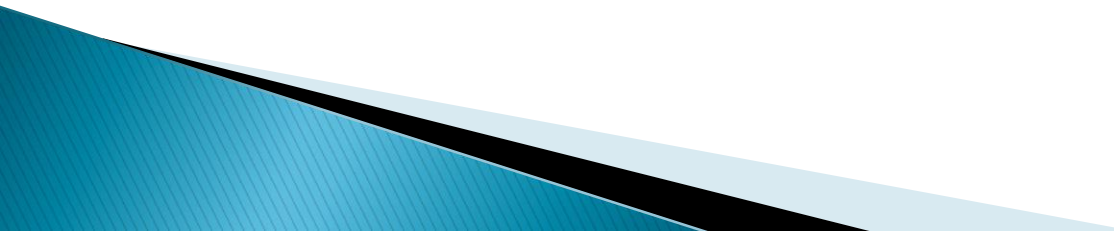
Prepared by : B.Loganathan

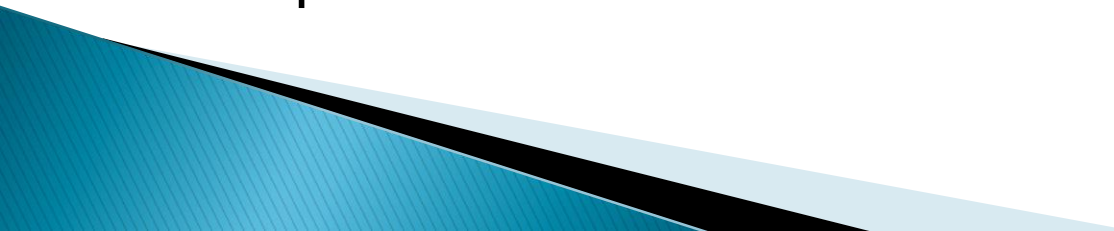
CLIENT SERVER COMPUTING

- ▶ **UNIT I: Client/Server computing–What is Client/Server–File servers, Database servers, Transaction servers, Groupware servers, Object application servers, Web application servers– FAT Servers or FAT Clients – 2-Tier versus 3-Tier - Client/Server–Client/Server building blocks.**
- ▶ **TEXT BOOK:“Client/Server Survival Guide”**
- ▶ Authors: Robert Orfali, Dan Harkey & Jeri Edwards,
- ▶ Publication : Wiley INDIA, Edition : 3rd, 201

What is Client/Server?

- ▶ Client and Servers are separate logical entities that work together over a network to accomplish a task.
 - ▶ All Client/Server systems have the following distinguishing characteristics:
 - ▶ **Service** : The server process is a provider of services. The client is a consumer of services. Client/Server provides a clean separation of function based on the idea of service.
 - ▶ **Shared Resources** : A server can service many clients at the same time and regulate their access to shared resources.
- 

- **Asymmetrical Protocols:** Client always initiate the dialog by requesting a service. Servers are passively awaiting requests from the clients. So the client becomes a server.
 - **Transparency of location :** The server is a process that can reside on the same machine as the client or on a different machine across the network. Client/server software usually masks the location of the server from the clients by redirecting the service calls when needed.
 - **Mix-and-match :** The client/server software is independent of hardware or operating system platforms. So we able to mix-and match client and server platforms.
- 

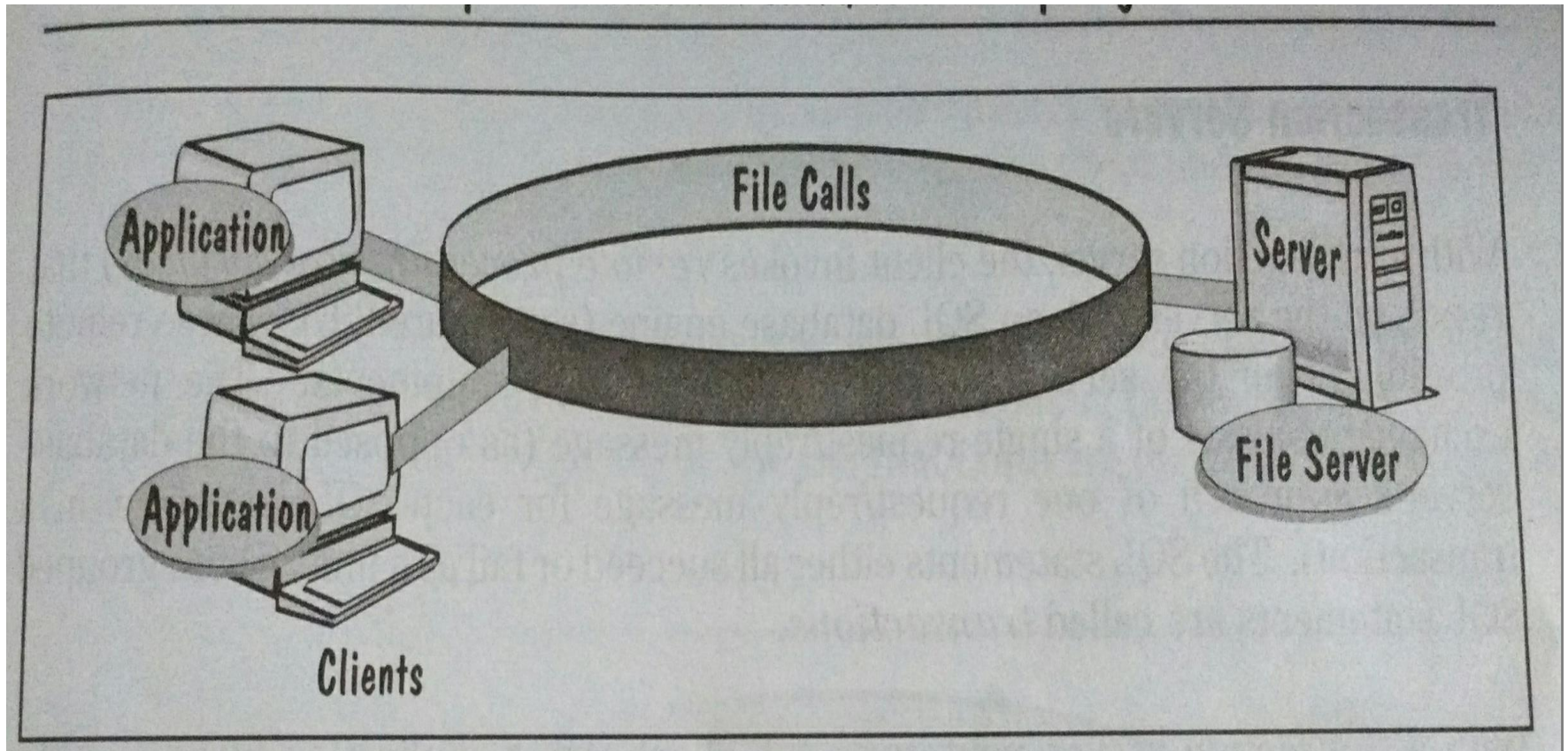
- ▶ **Message-based exchanges** : Clients and servers are loosely coupled systems that interact through a message-passing mechanism. The message is the delivery mechanism for the service request.
 - ▶ **Encapsulation of services** : The server is a specialist. A message tells a server what service is requested. Servers can be upgraded without affecting the client as long as the published message interface is not changed.
 - ▶ **Scalability** : Client/server systems can be scaled horizontally or vertically. Horizontal scaling means adding or removing client with only a slight performance impact. Vertical scaling mean either *migrating* to larger and faster server machine or distributing the processing *load* across multiple servers.
- 

- **Integrity** : The server code and server data is centrally managed which results in cheaper maintenance and the *guarding* of shared data integrity.

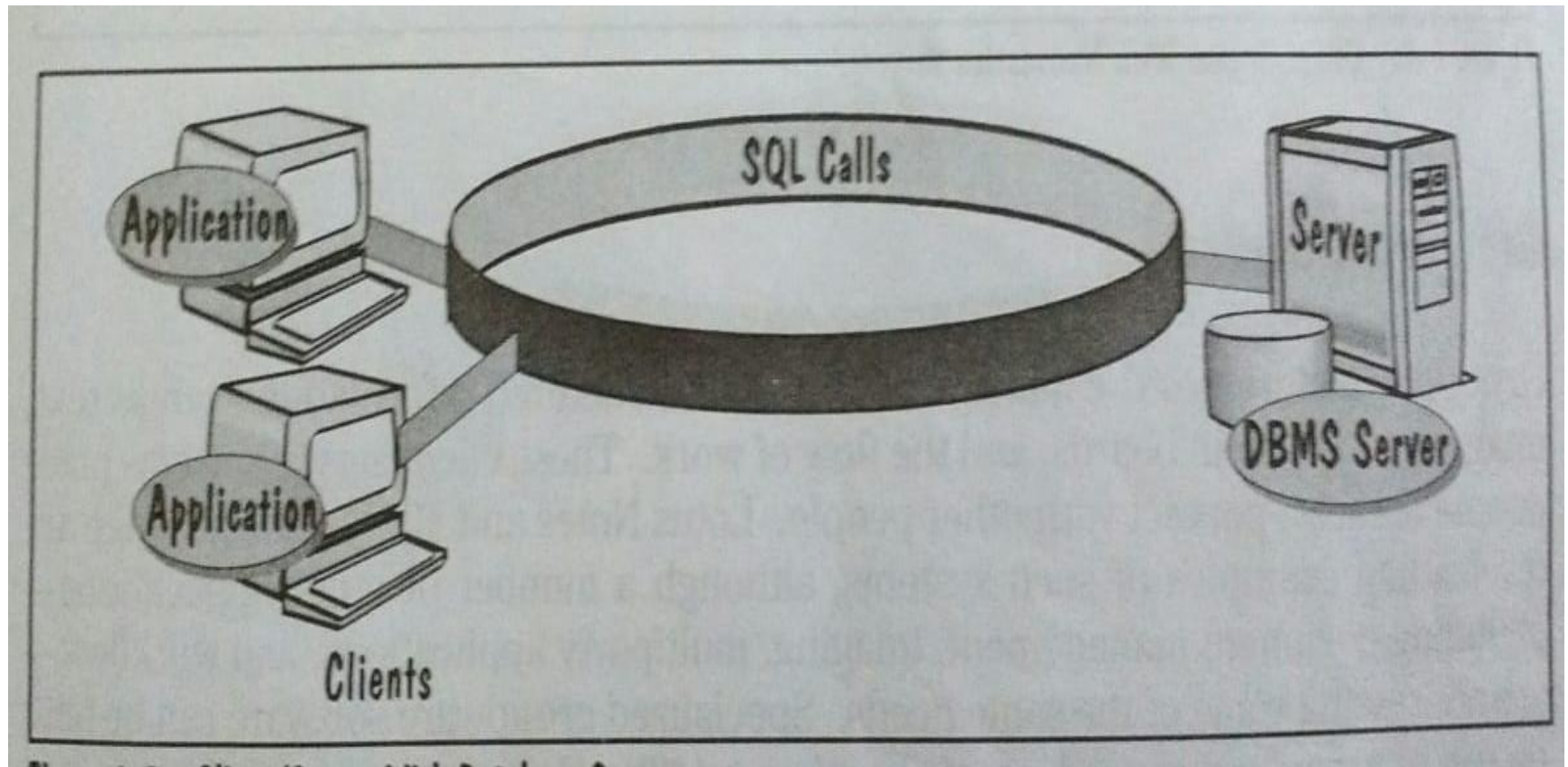
File Server

- The client passes requests for the file records over a network to the file server. This is a very primitive form of data service that necessitates many message exchanges over the network to find the requested data.
- File servers are useful for sharing files across a network. They are necessary for creating shared documents, images, engineering drawings and other large data objects.

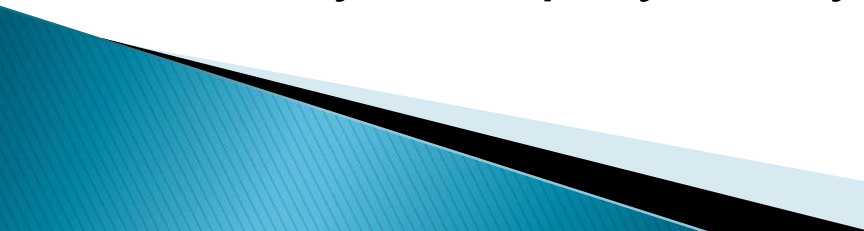
Client/Server with File Server



Client/Server with Database Server



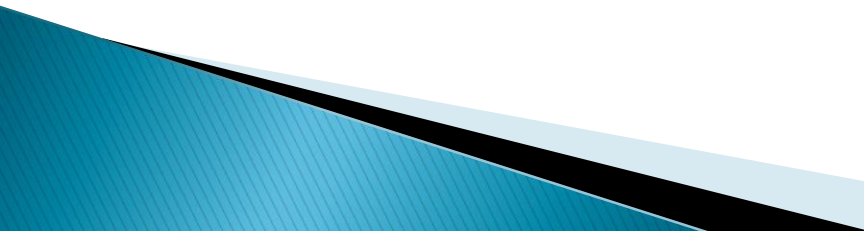
Database Servers

- The client passes SQL requests as messages to the database server. The results of each SQL command are returned over the network.
 - The server uses its own processing power to find the requested data instead of passing all the records back to a client (as in file server).
 - Database server provide the foundation for decision-support systems that require ad hoc queries and flexible reports.
 - They also play a key role in data warehousing.
- 

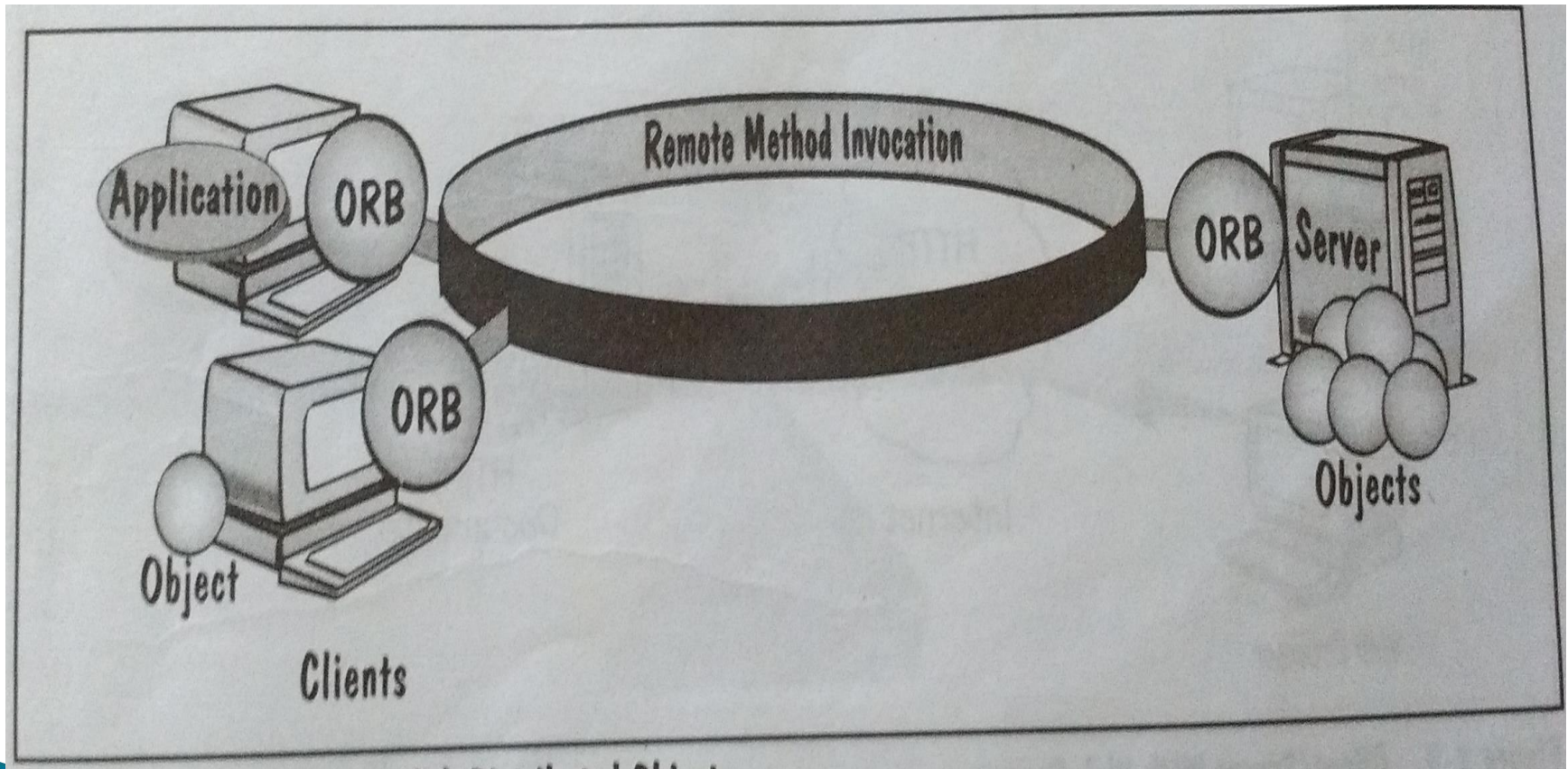
Transaction Servers

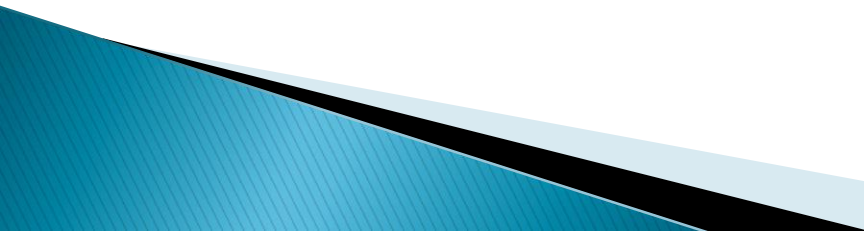
- In Transaction Server, the client invokes remote procedures/services that reside on the server with a SQL data base engine.
- These remote procedures on the server execute a group of SQL statements. The network exchange consist of a single request/reply message for each SQL statement. These grouped SQL statements are called transactions.
- The client component usually include a Graphical User Interface (GUI) and server component usually consists of SQL transactions against database. These applications are called OnLine Transaction Processing (OLTP).

Groupware Servers

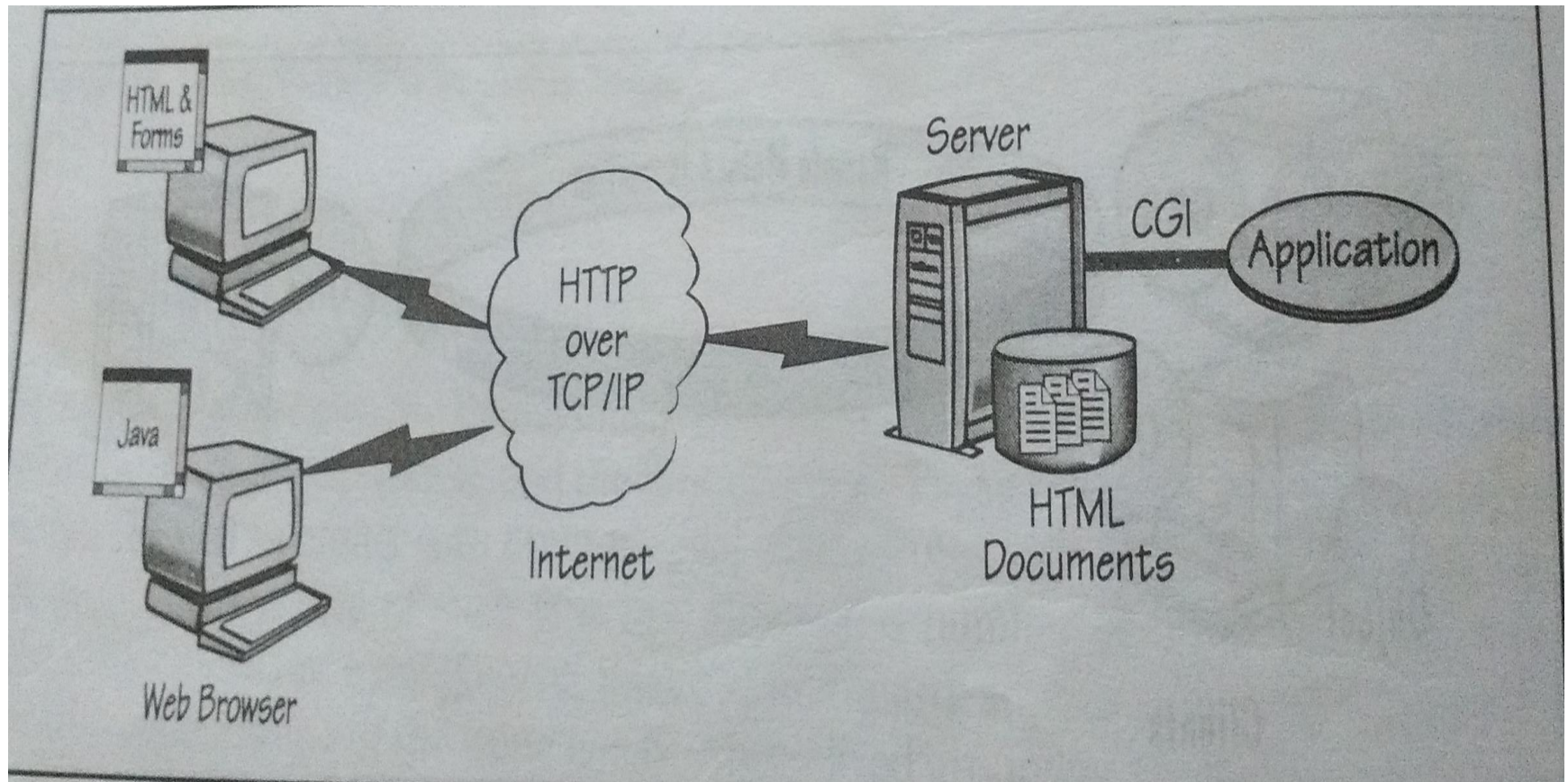
- ▶ Groupware address the management of semi-structured information such as text, image, mail, bulletin boards and flow of work. Examples are Lotus Notes, Microsoft Exchange and other applications including document management, multi party applications and work flow.
 - ▶ Applications are created by using scripting language form-based interfaces provided by the vendor. The communication middleware between the client and server is vendor-specific.
- 

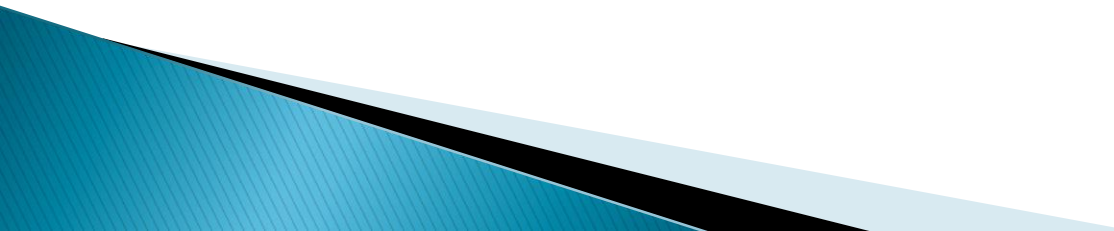
Object Application Servers



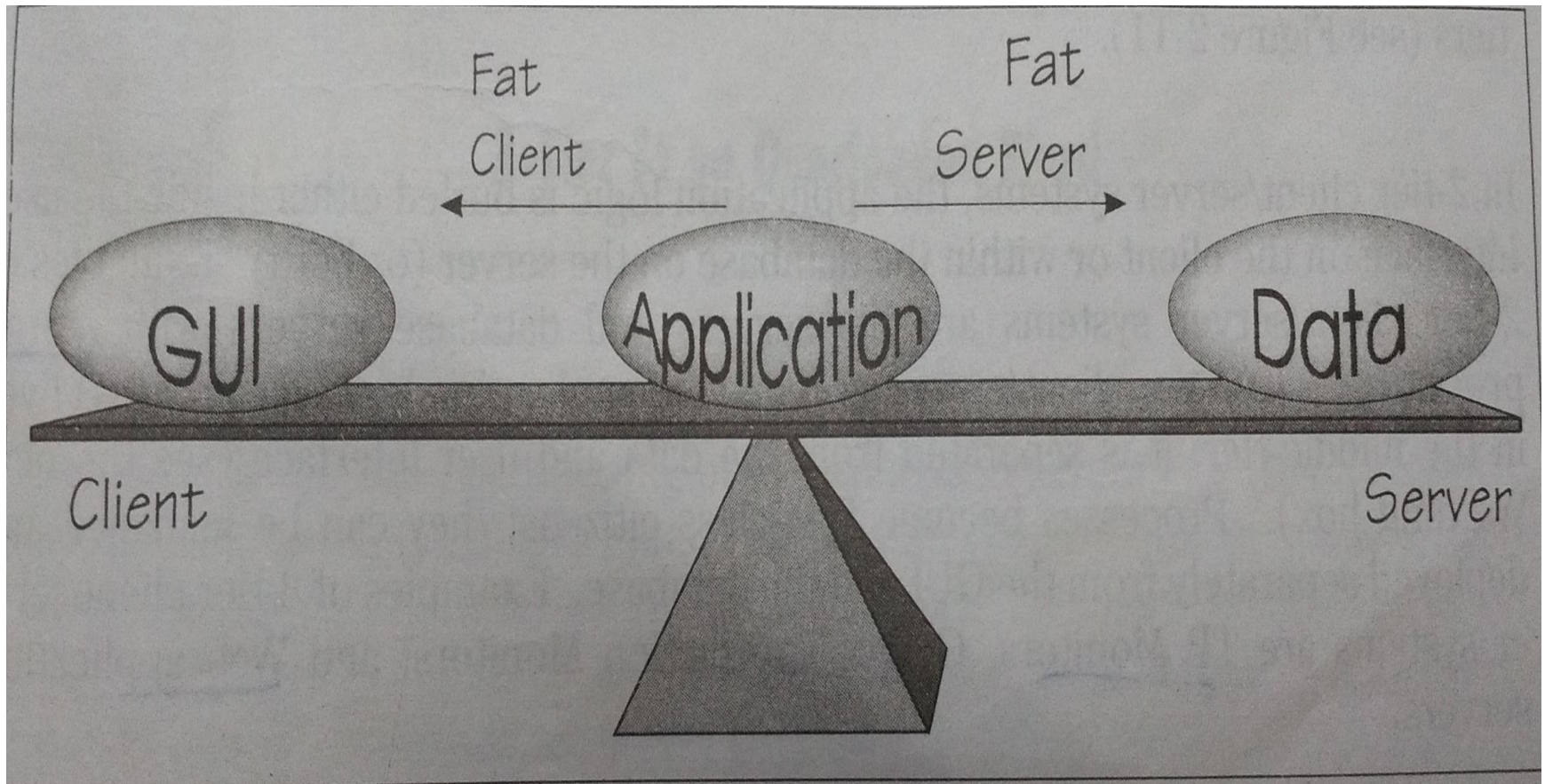
- ▶ In object server, the client/server applications are written as a set of communicating objects. Client objects communicate with server objects using an Object Request Broker(ORB).
 - ▶ The ORB locate an instance of that object server class, invokes the requested method and return the results to the client object.
 - ▶ The server object must provide support for concurrency and sharing.
 - ▶ Examples of commercial ORB's are Iona's Orbix, Inprise's VisiBroker, ICL's DAIS, JavaSoft's Java IDL, IBM's SOM.
- 

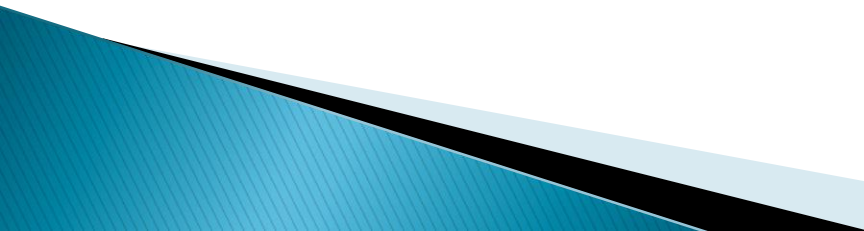
Web Application Server

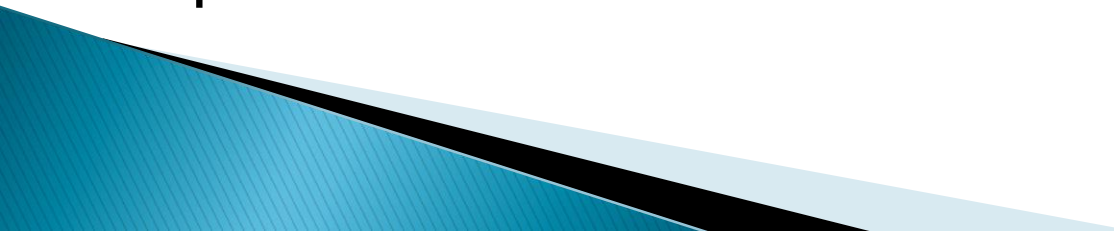


- ▶ The World Wide Web(WWW) is the first intergalactic client/server application. It consists of thin, portable, universal client that talk to super fat servers.
 - ▶ Web application servers are a new class of internet software. They are standard HTTP servers with server-side component frameworks. Functionally, they are very similar to object servers.
 - ▶ Examples are Microsoft's MTS Web Server, CORBA/Java's Enterprise JavaBeans, Netscape/Kiva's Application Server, IBM's Websphere.
- 

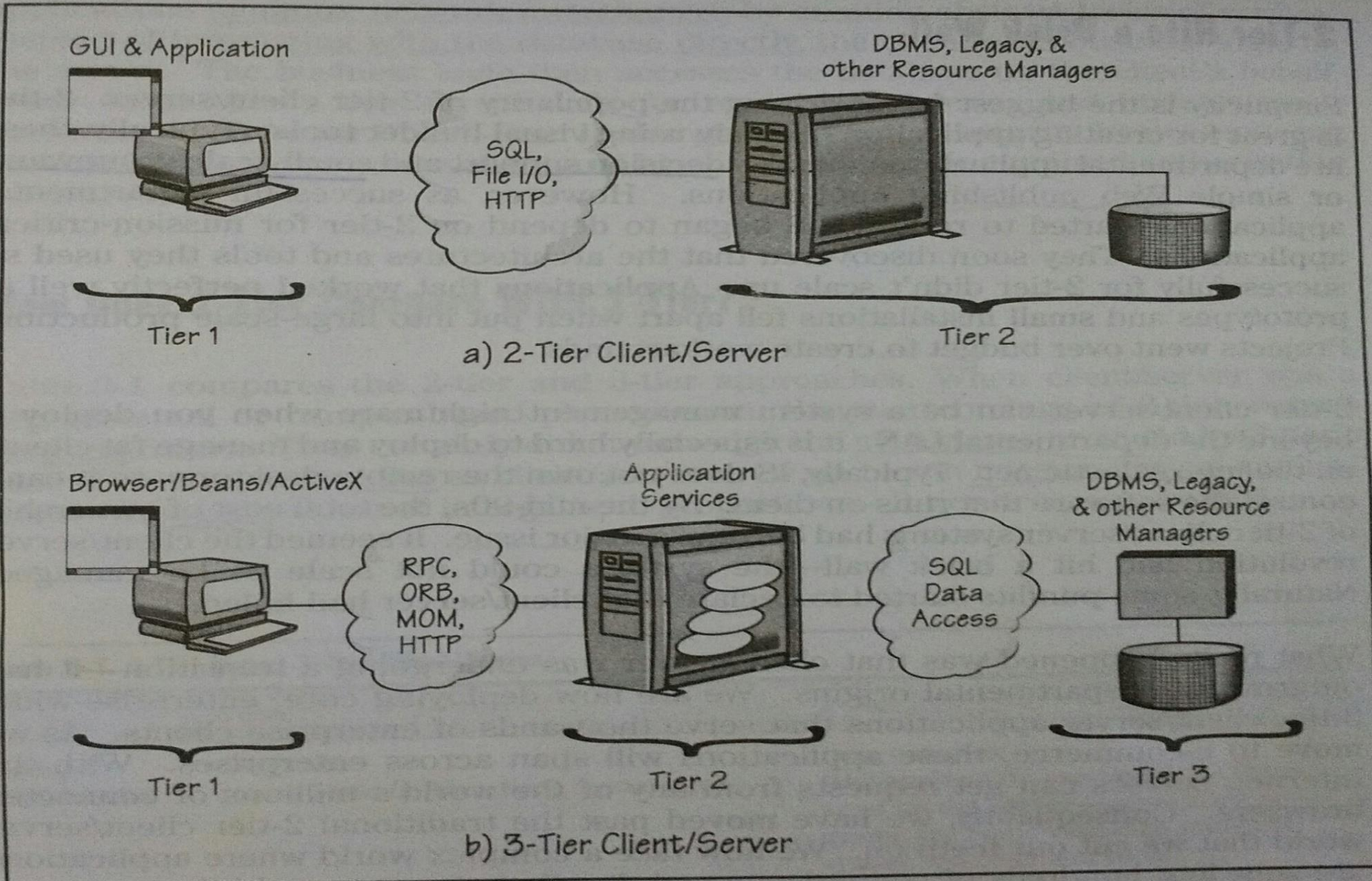
Fat Servers Or Fat Clients

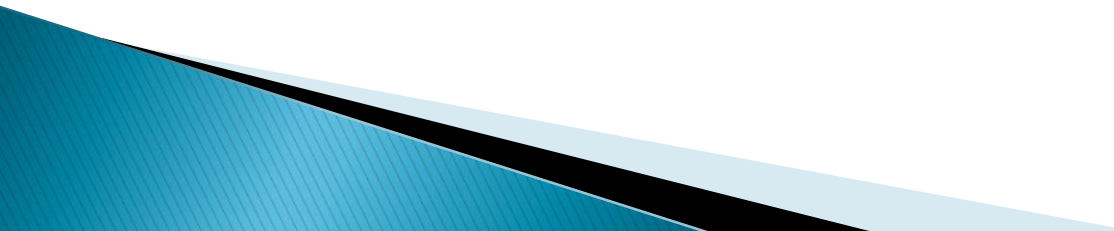


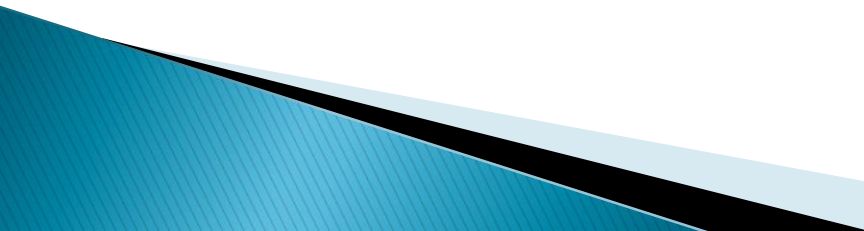
- ▶ Client/server application can be differentiated by *how* the distributed application is split between the client and the server.
 - ▶ The fat client model does more functions on the Client. The fat server model places more function on the server.
 - ▶ Database and File servers are examples of fat client.
 - ▶ Groupware, Transaction and Web servers are examples of fat servers.
- 

- ▶ Fat client are more traditional form of client/server. The bulk of application runs on the client side. In both file and database server models, the client know how the data are organized and stored on the server side.
 - ▶ Fat servers are easier to manage and deploy on the network because most of the applications runs on the server side. Fat server are used to minimize network interchanges by creating more abstract level of service.
 - ▶ The client in the server model provides the GUI and interact with the server through remote procedure calls.
- 

2-Tier Vs 3-Tier



- ▶ High-brow client/server pundits prefer to use the terms 2-tier, 3-tier and N-tier client/server architectures instead of fat clients and fat servers.
 - ▶ In 2-tier client server system, the application logic is buried either inside the user interface on the client or within the database on the server.
 - ▶ Examples of 2-tier client server systems are file servers and database servers.
 - ▶ In 3-tier client server systems, the application logic lives in the middle-tier and it is separated from the data and user interface.
 - ▶ Examples of 3-tier client server systems are TP Monitor, Object Transaction Monitors and Web server.
- 

- ▶ 2-tier is great for creating applications quickly using visual builder tools. Example are small scale groupware, web publishing applications.
 - ▶ 3-tier is a new growth area for client/server computing because it meets the requirement of large scale-internet and intranet applications.
 - ▶ 3-tier client/server systems are more scalable, robust, flexible and integrate data from multiple sources.
 - ▶ 3-tier substitutes a few server calls for many SQL queries and updates, so it perform much better than 2-tier.
- 

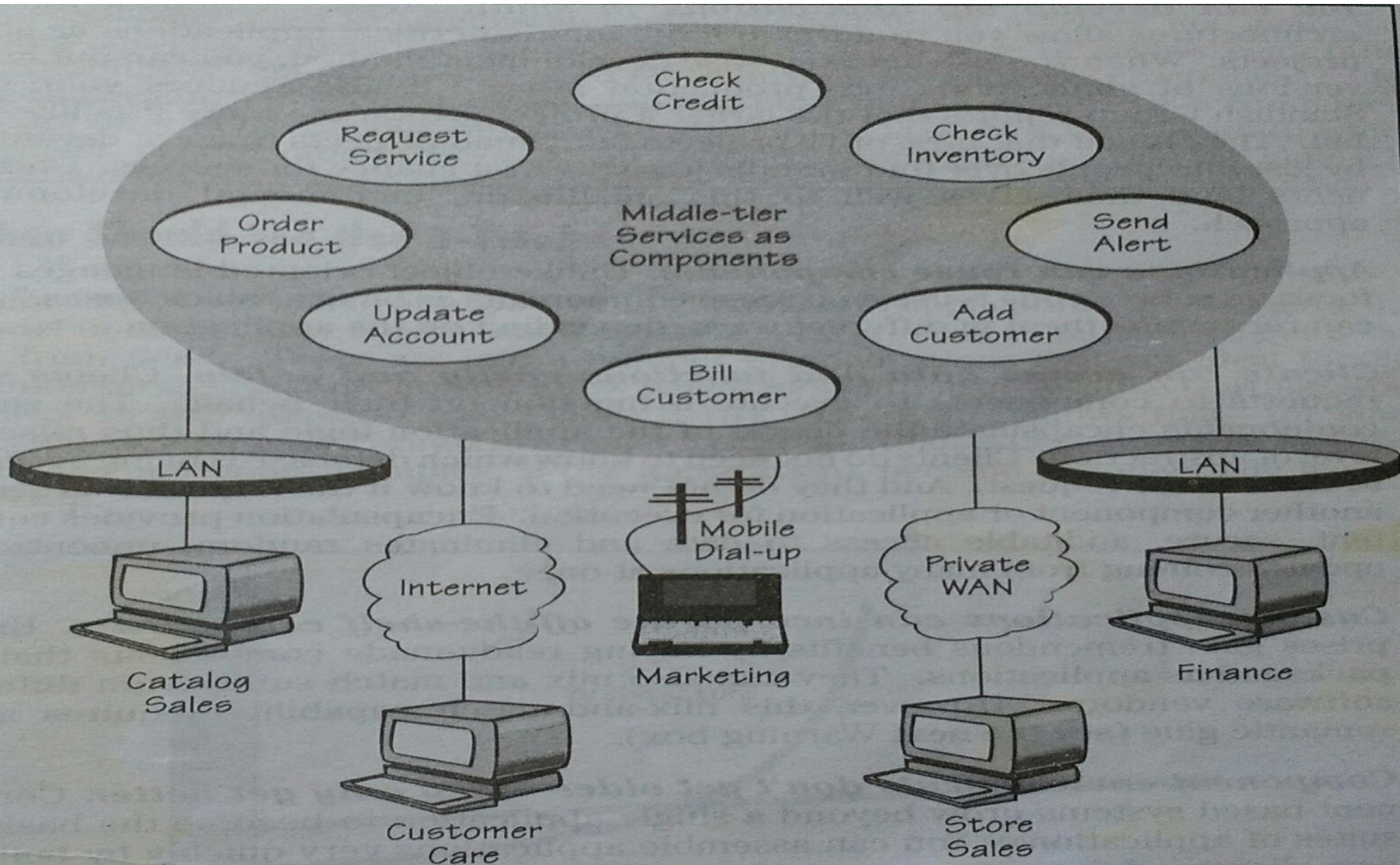
Comparison of 2-Tier And 3-Tier

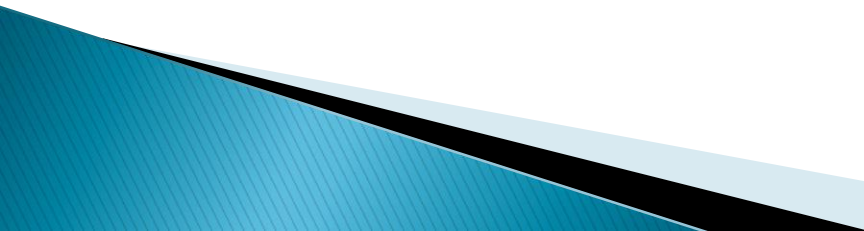
	2-Tier	3-Tier
System administration	Complex (more logic on the client to manage)	Less complex (the application can be centrally managed on the server—application programs are made visible to standard system management tools)
Security	Low (data-level security)	High (fine-tuned at the service, method, or object type level)
Encapsulation of data	Low (data tables are exposed)	High (the client invokes services or methods)
Performance	Poor (many SQL statements are sent over the network; selected data must be downloaded for analysis on the client)	Good (only service requests and responses are sent between the client and server)
Scale	Poor (limited management of client communications links)	Excellent (concentrates incoming sessions; can distribute loads across multiple servers)
Application reuse	Poor (monolithic application on client)	Excellent (can reuse services and objects)

Table 2-1. 2-Tier Versus 3-Tier Client/Server. (Continued)

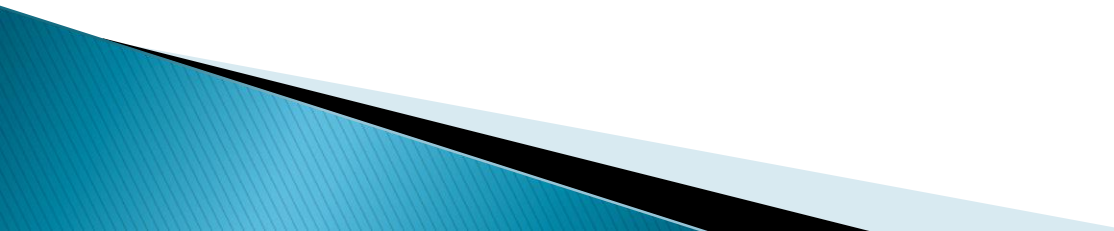
	2-Tier	3-Tier
Ease of development	High	Getting better (standard tools can be used to create the clients, and tools are emerging that you can use to develop both the client and server sides of the application)
Server-to-server infrastructure	No	Yes (via server-side middleware)
Legacy application integration	No	Yes (via gateways encapsulated by services or objects)
Internet support	Poor (Internet bandwidth limitations make it harder to download fat clients and exacerbate the already noted limitations)	Excellent (thin clients are easier to download as applets or beans; remote service invocations distribute the application load to the server)
Heterogeneous database support	No	Yes (3-tier applications can use multiple databases within the same business transaction)
Rich communication choices	No (only synchronous, connection-oriented RPC-like calls)	Yes (supports RPC-like calls, but can also support connectionless messaging, queued delivery, publish-and-subscribe, and broadcast)
Hardware architecture flexibility	Limited (you have a client and a server)	Excellent (all three tiers may reside on different computers, or the second and third tier may both reside on the same computer with component-based environments, you can distribute the second tier across multiple servers as well)
Availability	Poor	Excellent (can restart the middle tier component on other servers)

Components of N-Tier Architecture



- ▶ The middle-tier is implemented as a collection of components that are used in a variety of client initiated business transactions(not monolithic).
 - ▶ Clients frequently combine several middle-tier components within the single business transaction. A component can call other components to help it implement a request.
 - ▶ Component based application offer significant advantages over monolithic applications.
- 

▶ **Advantages are:**

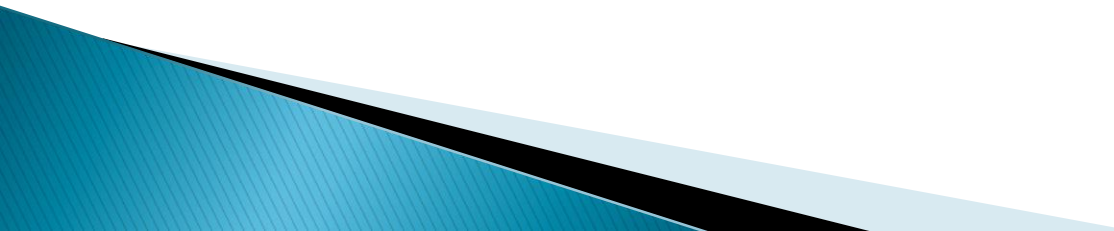
- ▶ i) We can develop big applications in small steps.
 - ▶ ii) Applications can reuse components.
 - ▶ iii) Client can access and functions easily and safely.
 - ▶ iv) Custom applications can incorporate off-the-shelf components.
 - ▶ v) Component environment do not get older and they only get better.
- 

Client Server Building Block

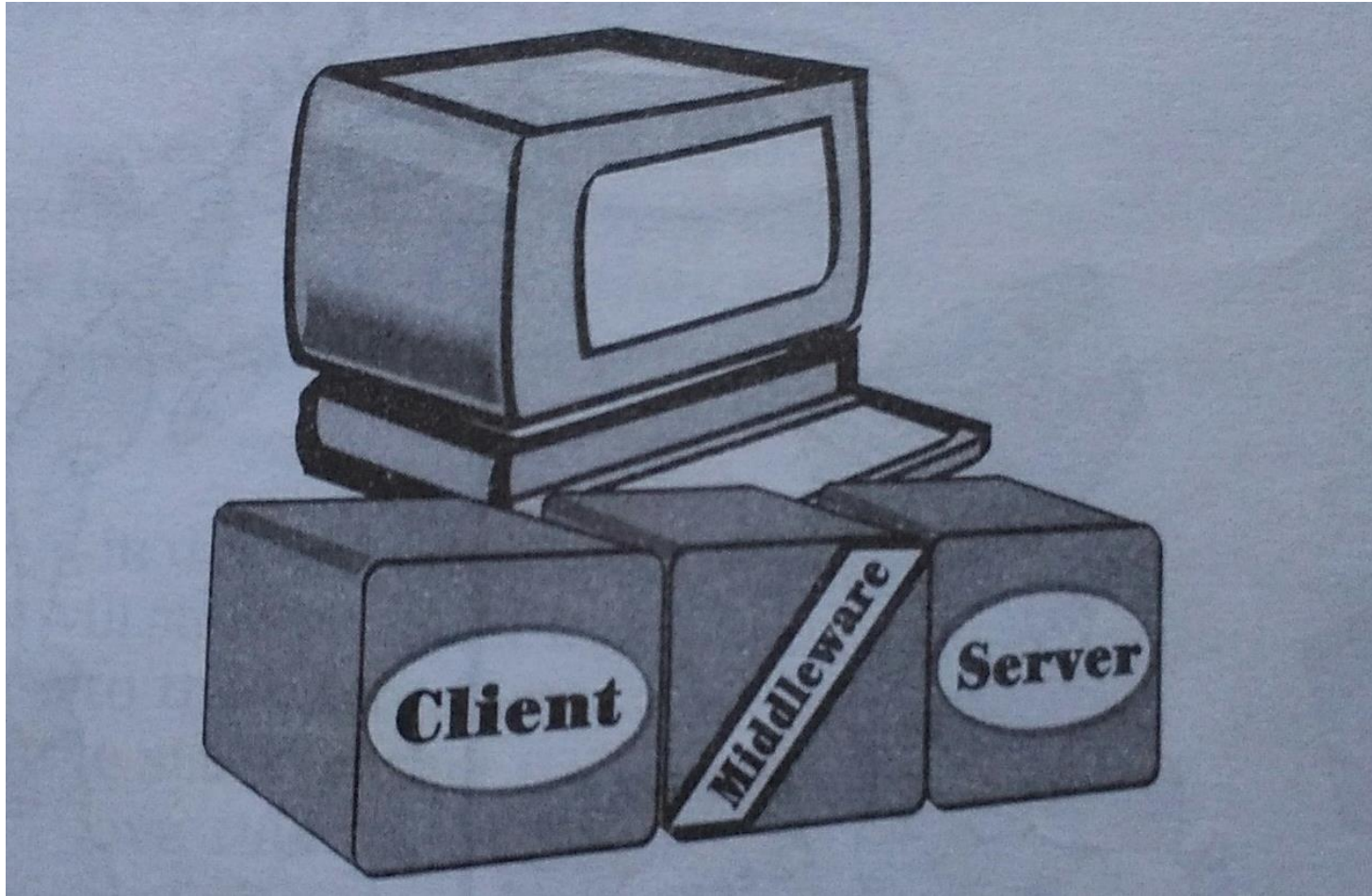
- ▶ ***Three Basic Building Block*** of Client/Server :

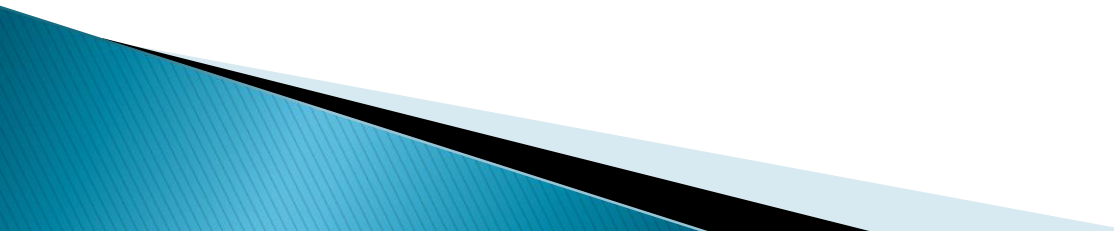


- ▶ This model is really is a game of putting together things with building blocks: a client, a server and the slash(/) that ties the client to the server.
- ▶ Building blocks are used in four situations:
- ▶ i) **Client/server for tiny shops and nomadic tribes** : It is a building block implementation that run the client, middleware software and most of the business services on the same machine. It is used for one-person shops, home offices and mobile users.
- ▶ ii) **Client/Server for small shops and departments** : It is classic Ethernet client/single-server building-block implementation. It is used in small shops, departments and branch offices.

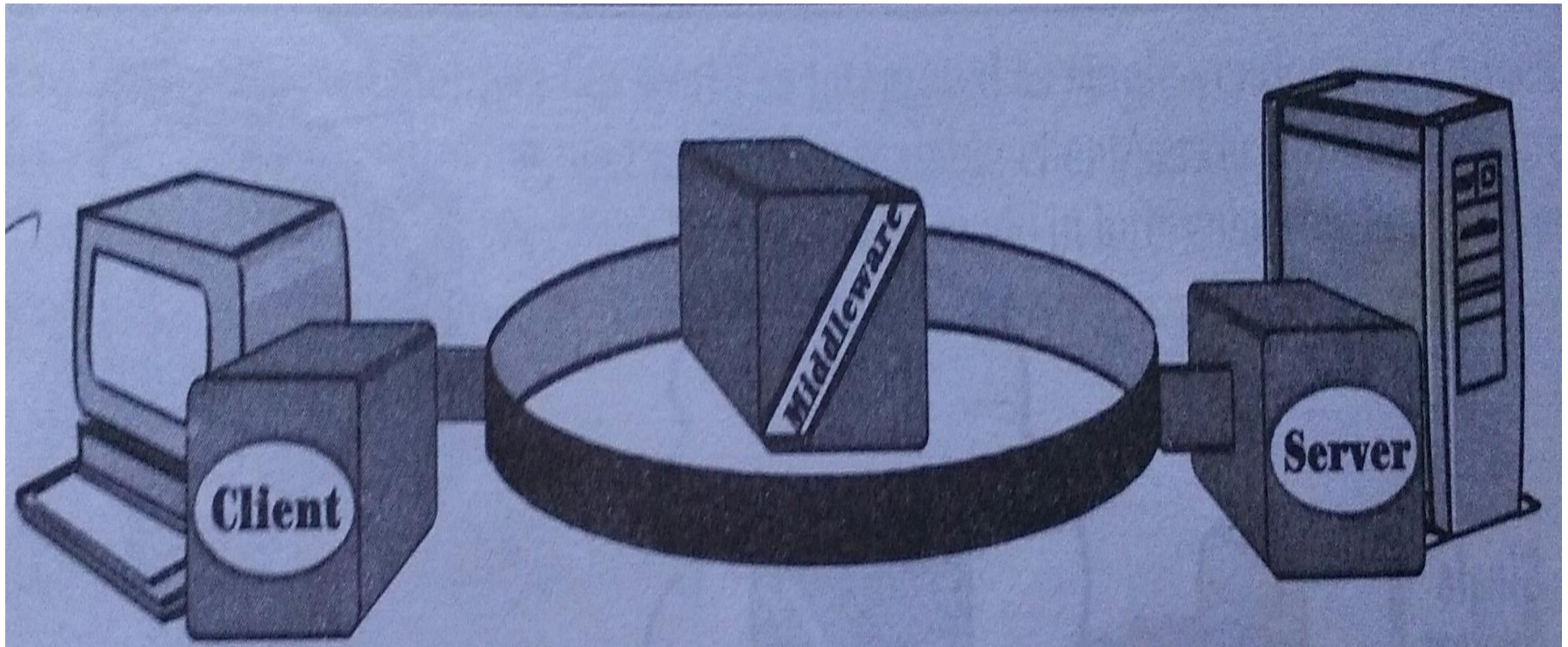
- ▶ iii) **Client/server for intergalactic enterprises** : It is the multi-server building-block implementation of client/server. The servers present a single-system image to the client. This is used to meet the initial needs of the intergalactic client/server computing.
 - ▶ iv) **Client/server for post-scarcity world** : It transform every machine in the world into both a client and a server. Personal agents on every machine will handle all the negotiations with the peer agents anywhere in the universe.
- 

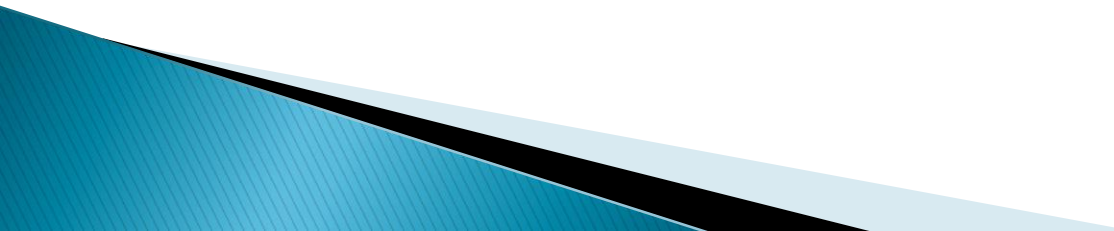
Client/Server For Tiny Shops And Nomadic Tribes



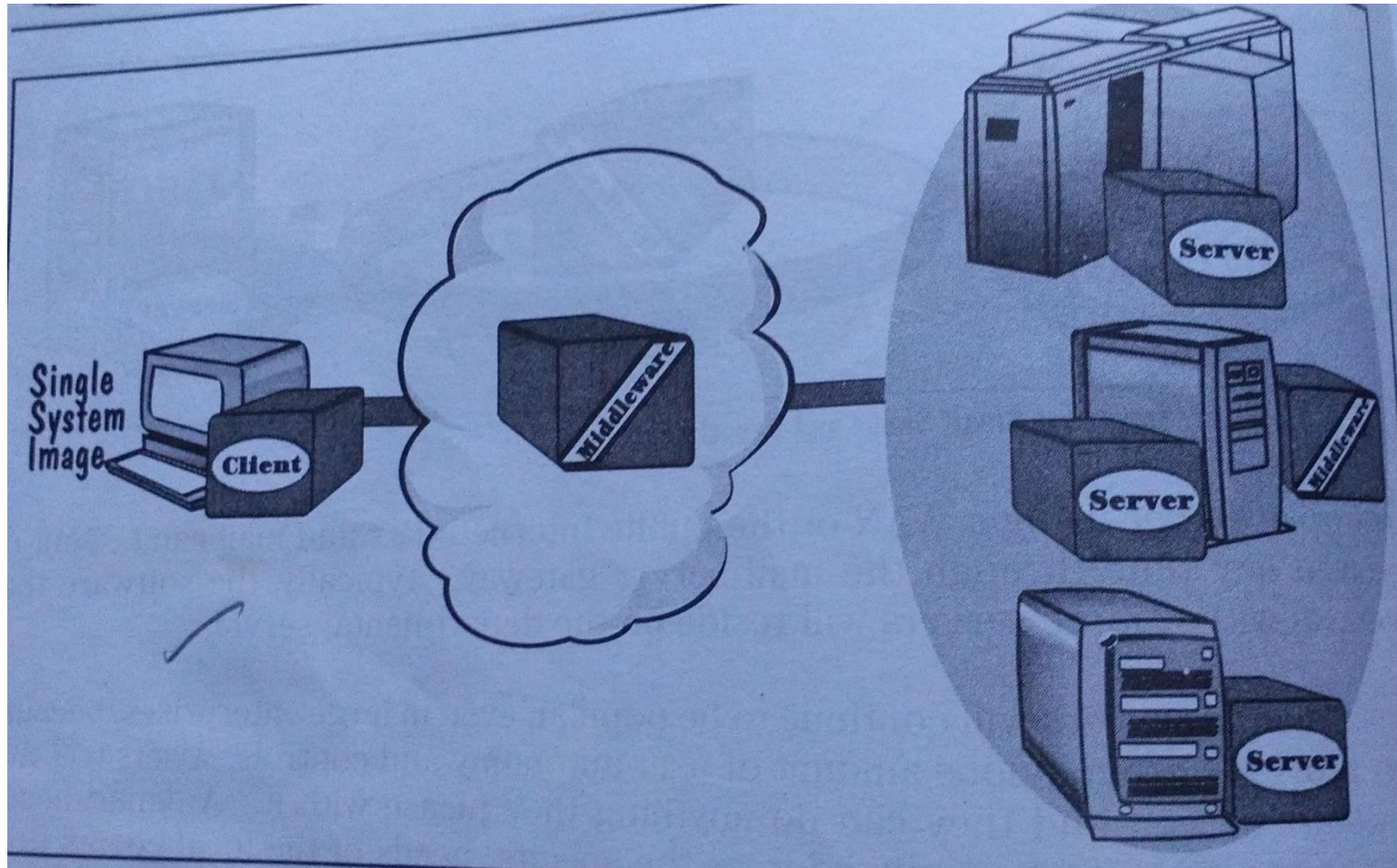
- ▶ Client/server is easy to run the client and server portion of an application on the same machine.
 - ▶ For example a client server application for a dentist's office can be sold in a single-user-package.
 - ▶ The business-critical client/server application run on one machine and does some occasional communications with outside servers to exchange data, refresh a database, send or receive mail.
- 

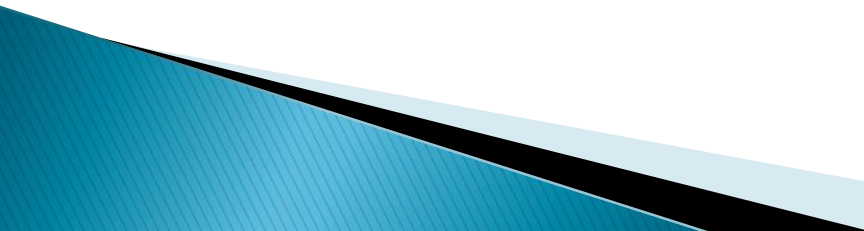
Client/Server For Small Shops And Departments



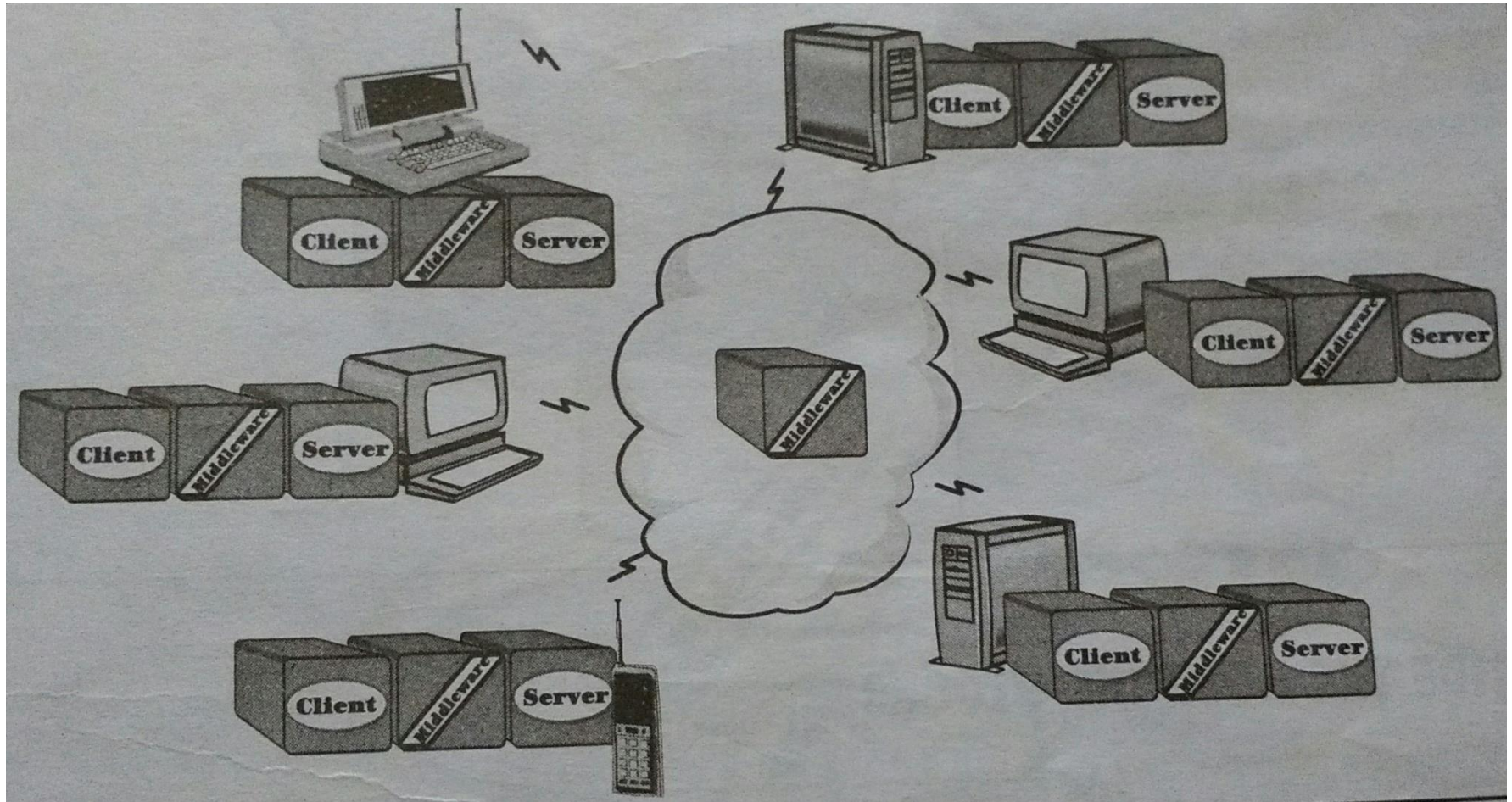
- ▶ This client/server architecture is particularly well suited for the LAN based single establishments. It consists of multiple clients talking to a local server.
 - ▶ This model is used in small businesses. For example, a multi user dentist office, departments of large corporations and branch offices of a bank.
 - ▶ The single-server nature of the model tends to keep the middleware simple. This model work very well in small businesses and departments that depend on single-server or on very loosely-coupled multi-server arrangements.
- 

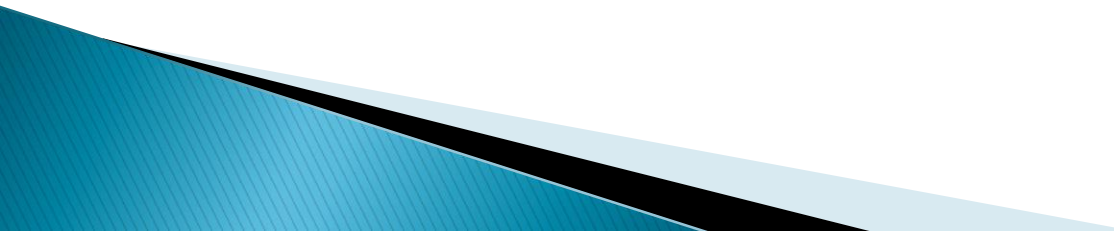
Client/Server For Intergalactic Enterprises



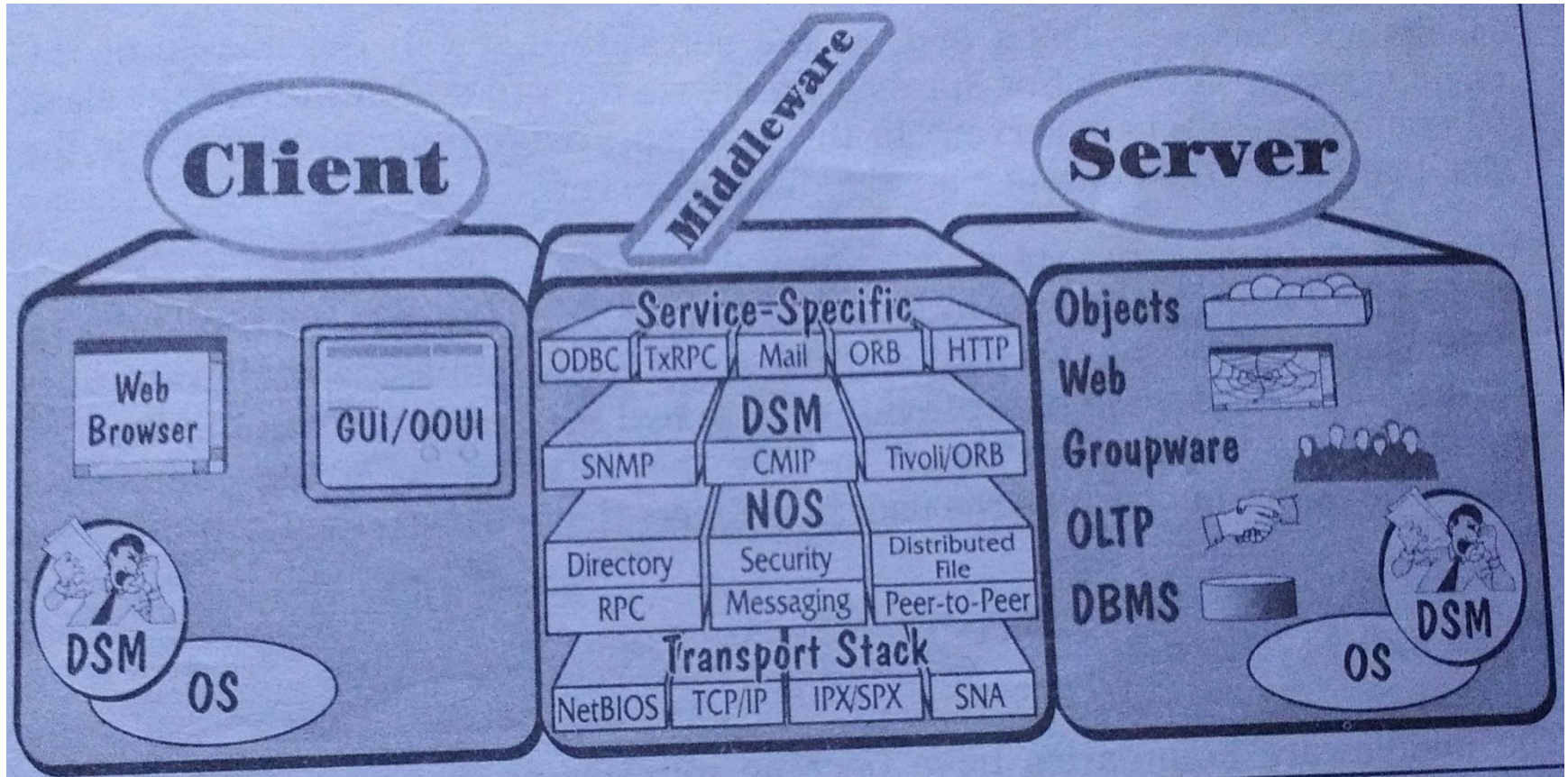
- ▶ The client/server enterprise model addresses the needs of establishments with a mix of heterogeneous servers. This is an area that is getting a lot of industry attention as solutions move from a few large computers to multiple-servers that live on the internet, intranets and corporate backbone networks.
 - ▶ One of the great things about this client/server model is that it is upwardly scalable. When more processing power is needed for various intergalactic functions, more servers can be added or the existing server machine can be traded up for the latest generation of super server machine.
- 

Client/Server For Post-Scarcity World



- ▶ In this model, every machine is both a client and a full function server. We call this plentiful environment is the “*post-scarcity world*”.
 - ▶ We imagine the typical post-scarcity machine as a 2000 cellular notebook powered by a top-of-the-line processor and loaded with 10 MB RAM and 100 GB or more of disk space.
 - ▶ Every machine is a full-functional server, it will run at a minimum, a file server, database server, workflow agent, object transaction monitor and web server.
- 

Client/Server Software Infrastructure



The Client/Server Software Infrastructure:

- ▶ **i)The client building block** : It runs the client side of the application. It runs on an Operating System (OS) that provides a Graphical User Interface (GUI) or an Object Oriented User Interface (OOUI) and that can access distributed services.
- ▶ The client also runs a component of the Distributed System Management (DSM) elements. This could be anything from a simple agent to the entire front-end of the DSM application on a managing station.
- ▶ **ii)The server building block** : It runs the server side of the application. The server application typically runs on top of shrink-wrapped server software package.

- ▶ The five contending server platforms for creating next generation of client/server applications are SQL database servers, OLTP monitors, Groupware servers, Object servers and Web servers.
- ▶ The server side depends on the operating system to interface with the middleware building block that brings the request for service. The server also runs a Distributed System Management (DSM) component.
- ▶ **iii)The middleware building block** : It runs on both the client and server sides of an application. This building block consists of three categories : Transport Stacks, Network Operating Systems (NOS) and Service-Specific Middleware.

- ▶ **Middleware is the nervous system of the client/server infrastructure. Like the other two building blocks, the middleware also has a DSM software component.**
 - ▶ **The Distributed System Management (DSM) application runs on every node in a client/server network. The managing work station collect information from all its agent on the network and display it graphically.**
 - ▶ **The managing work station also instruct its agent to perform actions on its behalf (network within a network).**
- 