Module 17

Client-Server Software Development

Lesson 41

Basic Ideas on Client-Server Software Development and Client-Server Architecture

Version 2 CSE IIT, Kharagpur

Specific Instructional Objectives

At the end of this lesson the student would be able to:

- Explain what is a client-server software.
- Explain the advantages of client-server software over centralized solutions.
- Explain the factors responsible for making client-server solutions feasible, affordable, and popular in recent times.
- Identify the advantages of client-server software development compared to monolithic ones.
- Identify the disadvantages of client-server software.
- Differentiate between host-slave computing and client-server computing. Give an example for each.
- Explain the two-tier client-server architecture.
- Explain the limitations of two-tier client-server architecture.
- Explain three-tier client-server architecture.
- Identify the functions of middleware.
- Identify the popular middleware standards.

Client-server software

A client is basically a consumer of services and server is a provider of services as shown in fig. 17.1. A client requests some services from the server and the server provides the required services to the client. Client and server are usually software components running on independent machines. Even a single machine can sometimes acts as a client and at other times a server depending on the situations. Thus, client and server are mere roles.

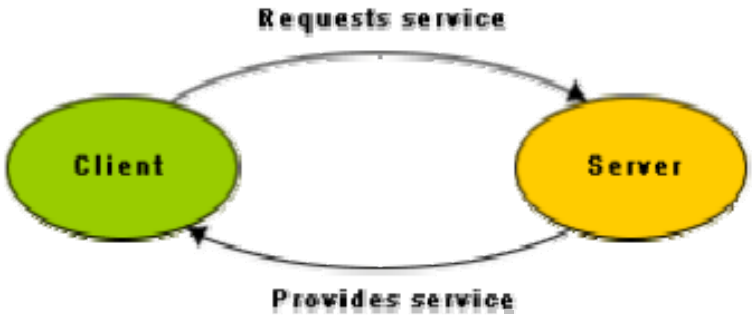

Fig. 17.1: Client-server model

Example:
A man was visiting his friend's town in his car. The man had a handheld computer (client). He knew his friend's name but he didn't know his friend's

address. So he sent a wireless message (request) to the nearest “address server” by his handheld computer to enquire his friend’s address. The message first came to the base station. The base station forwarded that message through landline to local area network where the server is located. After some processing, LAN sent back that friend’s address (service) to the man.

Advantages of client-server software

The client-server software architecture is a versatile, message-based and modular infrastructure that is intended to improve usability, flexibility, interoperability and scalability as compared to centralized, mainframe, time sharing computing.

Factors for feasibility and popularity of client-server solutions

Client-server concept is not a new concept. It already existed in the society for long time. A doctor is a client of a barber, who in turn is a client of the lawyer and so forth. Something can be a server in some context and a client in some other context. So client and server are mere roles as shown in fig. 17.2.

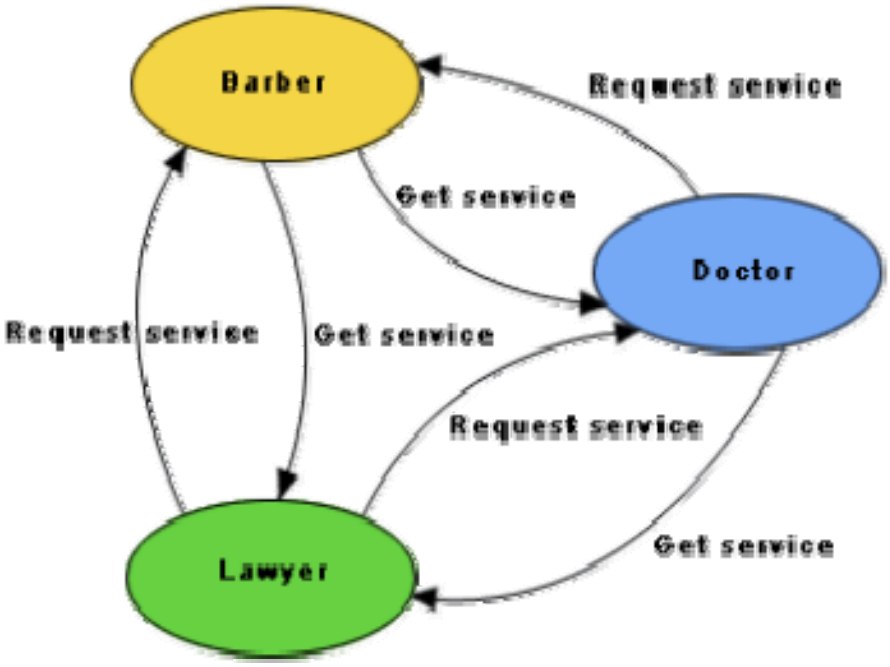

Fig. 17.2: Client and server as roles

There are many reasons for the popularity of client-server software development. Some reasons are:

- Computers have become small, decentralized and cheap
- Networking has become affordable, reliable, and efficient.
- Client-server systems divide up the work of computing among many separate machines. Thus client-server solutions are modular and loosely coupled. So they are easy to develop and maintain.

Advantages of client-server software development

There are many advantages of client-server software products as compared to monolithic ones. These advantages are:

- **Simplicity and modularity –** Client and server components are loosely coupled and therefore modular. These are easy to understand and develop.
- **Flexibility** – Both client and server software can be easily migrated across different machines in case some machine becomes unavailable or crashes. The client can access the service anywhere. Also, clients and servers can be added incrementally.
- **Extensibility** – More servers and clients can be effortlessly added.
- **Concurrency** – The processing is naturally divided across several machines. Clients and servers reside in different machines which can operate in parallel and thus processing becomes faster.
- **Cost Effectiveness** – Clients can be cheap desktop computers whereas severs can be sophisticated and expensive computers. To use a sophisticated software, one needs to own only a cheap client and invoke the server.
- **Specialization** – One can have different types of computers to run different types of servers. Thus, servers can be specialized to solve some specific problems.
- **Current trend** – Mobile computing implicitly uses client-server technique. Cell phones (handheld computers) are being provided with small processing power, keyboard, small memory, and LCD display. Cell phones cannot really compute much as they have very limited processing power and storage capacity but they can act as clients. The handhold computers only support the interface to place requests on some remote servers.

- **Application Service Providers (ASPs) –** There are many application software products which are very expensive. Thus it makes prohibitively costly to own those applications. The cost of those applications often runs into millions of dollars. For example, a Chemical Simulation Software named "*Aspen*" is very expensive but very powerful. For small industries it would not be practical to own that software. Application Service Providers can own ASPEN and let the small industries use it as client and charge them based on usage time. A client and simply logs in and ASP charges according to the time that the software is used.

- **Component-based development** – It is the enabler of the client-server technology. Component-based development is radically different from traditional software development. In component-based development, a developer essentially integrates pre-built components purchased off-the-shelf. This is akin to the way hardware developers integrate ICs on a Printed Circuit Board (PCB). Components might reside on different computers which act as servers and clients.

- **Fault-tolerance** – Client-server based systems are usually fault-tolerant. There can be many servers. If one server crashes then client requests can be switched to a redundant server.

There are many other advantages of client-server software. For example, we can locate a server near to the client. There might be several servers and the client requests can be routed to the nearest server. This would reduce the communication overhead.

Disadvantages of client-server software

There are several disadvantages of client-server software development. Those disadvantages are:

- **Security** – In a monolithic application, implementation of security is very easy. But in a client-server based development a lot of flexibility is provided and a client can connect from anywhere. This makes it easy for hackers to break into the system. Therefore, ensuring security in client-server system is very challenging.

- **Servers can be bottlenecks** – Servers can turn out to be bottlenecks because many clients might try to connect to a server at the same time. This problem arises due to the flexibility given that any client can connect anytime required.

- **Compatibility** – Clients and servers may not be compatible to each other. Since the client and server components may be manufactured by different

vendors, they may not be compatible with respect to data types, language, etc.

- **Inconsistency** – Replication of servers is a problem as it can make data inconsistent.

Host-slave computing vs. client-server computing

An example of a host-slave computing is a Railway-reservation system. The software is divided into two parts – one resides on the terminals of the booking clerks. The master at any time directs the slaves what to do. A slave can only make requests and master takes over and tells what to do.

On the other hand, in a client-server computing, different components are interfaced using an open protocol. In a master-slave they are proprietary. An example of a client-server system is a world wide web.

Two-tier client-server architecture

The simplest way to connect clients and servers is a two-tier architecture as shown in fig. 17.3. In a two-tier architecture, any client can get service from any server by initiating a request over the network. With two tier client-server architectures, the user interface is usually located in the user's desktop and the services are usually supported by a server that is a powerful machine that can service many clients. Processing is split between the user interface and the database management server. There are a number of software vendors who provide tools to simplify development of applications for the two-tier client-server architecture.

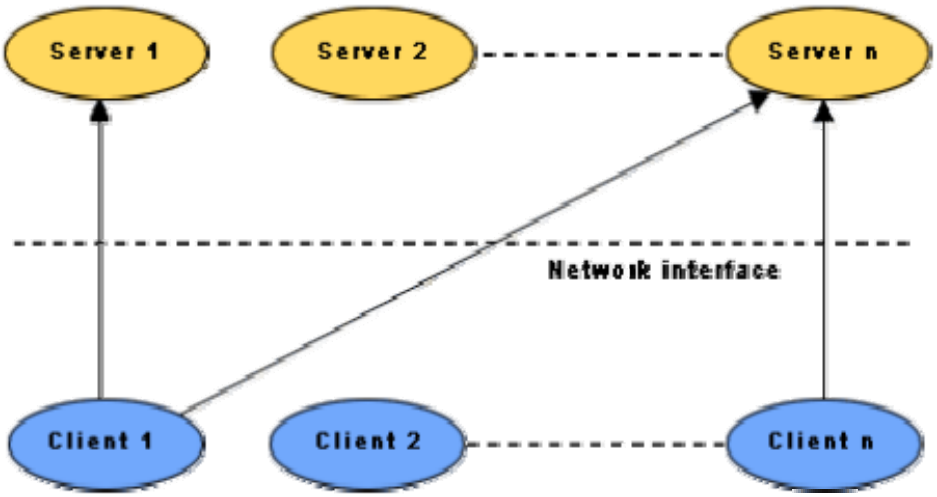

Fig. 17.3: Two-tier client-server architecture

Limitations of two-tier client-server architecture

A two tier architecture for client-server applications is an ideal solution but is not practical. The problem is that client and server components are manufactured by different vendors and the different vendors come up with different sets of interfaces and different implementation standards. That's the reason why clients and servers can often not talk to each other. A two tier architecture can work only in an open environment. In an open environment all components have standard interfaces. However, till date an open environment is still far from becoming practical.

Three-tier client-server architecture

The three-tier architecture overcomes the important limitations of the two-tier architecture. In the three-tier architecture, a middleware was added between the user system interface client environment and the server environment as shown in fig. 17.4. The middleware keeps track of all server locations. It also translates client's requests into server understandable form. For example, if the middleware provides queuing, the client can deliver its request to the middleware and disengage because the middleware will access the data and return the answer to the client.

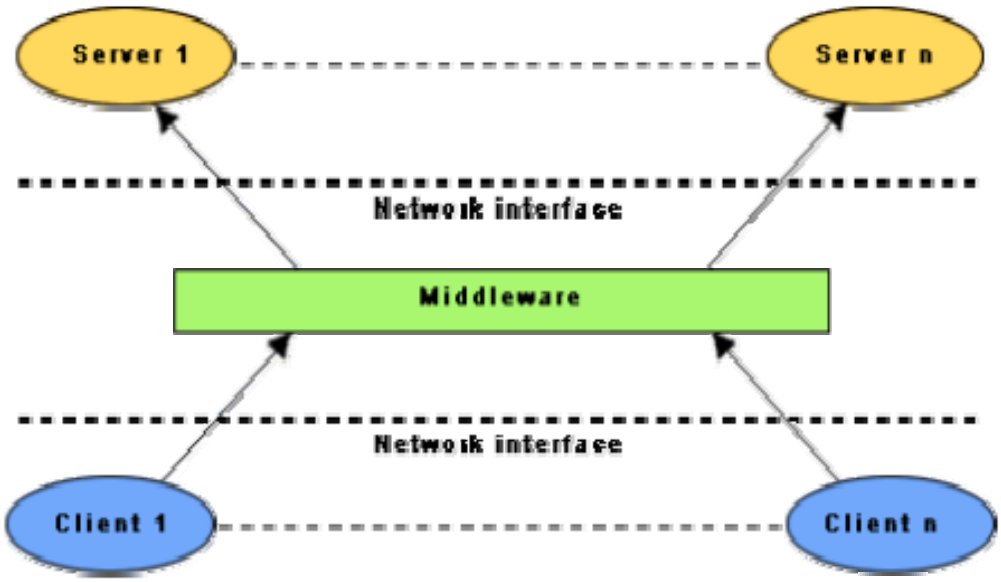

Fig. 17.4: Three-tier client-server architecture

Functions of middleware

The middleware performs many activities such as:

- It knows the addresses of servers. So, based on client requests, it can locate the servers.

- It can translate between client and server formats of data and vice versa.

Popular middleware standards

Two popular middleware standards are:

- CORBA (Common Object Request Broker Architecture)

- COM/DCOM

CORBA is being promoted by Object Management Group (OMG), a consortium of a large number of computer industries such as IBM, HP, Digital etc. Actually OMG is not a standards body, they only try to promote de facto standards. They don't have any authority to make or enforce standards. They just try to popularize good solutions with the hope that if they become highly popular they would automatically become standard.

COM/DCOM is being promoted by Microsoft alone.