Assignment 2

Date 2077/07/12
Page

1) Explain different types of architectural styles in distributed system.

=> Distributed systems are composed of various hardware and software (collectively called components) that communicate with each other only by transfer of messages.

The different types of architectural styles in distributed system are as follows:-

a) Layered architectures
b) Object-based architectures
c) Data-centered architectures
d) Event-based architectures

a) Layered architecture

In layered architecture, components are organized in a layer fashion where a component at layer L and is allowed to call components at the underly layer Li- but not the other way around, Control generally flow from layer to layer: request go down the hierarchy whereas the result flow upward. The advantages of using this approach is that the call always follow pre-defined path and that each layer can be easily replaced or modified without affecting the entire architecture.

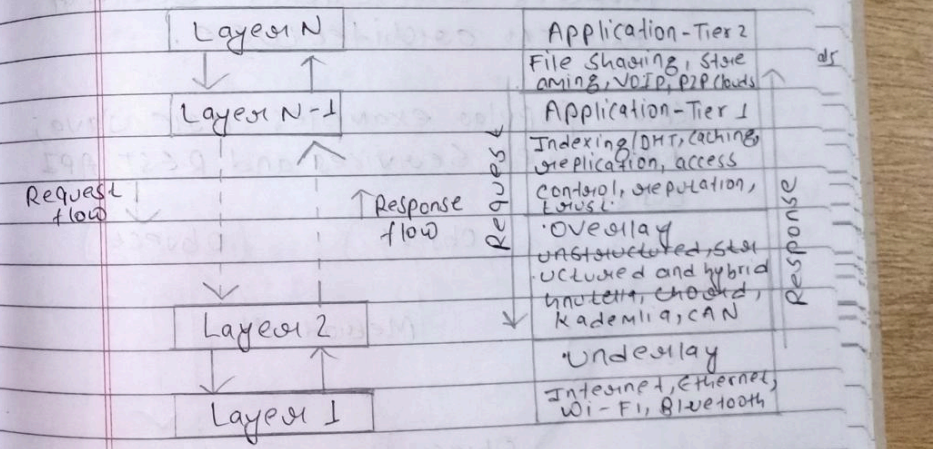

Fig: Showing Layered architecture

b) Object based architecture

This architecture does not have a sequential set of steps as in layered. Each components are referred as object, Where each objects can interact with other object through a given connector or interface. This architecture matches the client-server system architecture.

Some popular examples are java, RMI, Web services and REST API calls.

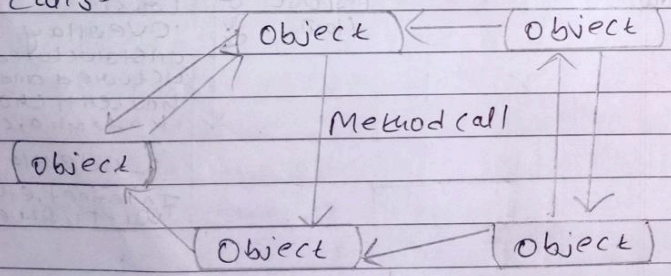

Fig: Showing Object-based architecture.

c) Data-centered architecture

It evolve around the idea that processes communicate through a common (passive or active) repository. It can be argued that for distributed system these architecture are as important as the layer architecture and object based architecture. This is more like producer consumer problems.

Forces: a wealth of networked applications have been developed that rely on a shared distributed file system in which virtually all communication takes place through files. Likewise, Web-based distributed systems are largely data centric: process communicate through use of shared Web-based data services.

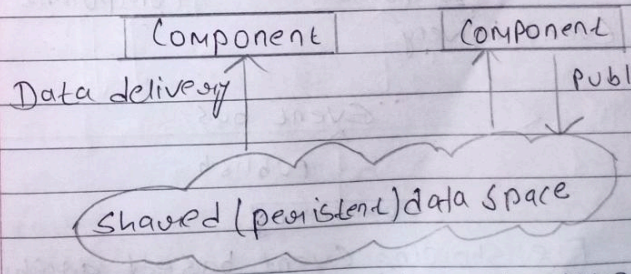

Fig: showing data centered architecture

d) Event-based architecture

It processes essentially communicate through the propagation of events which optionally carry data. When an event is generated, it will be sent through the bus and respective node can pull the event and use.

It's advantage is that the components can be easily add, remove or Modified. The heterogeneous components can communicate through any communication protocol. Some examples are: publisher-subscribe system, broadcast, point to point, enterprise service bus (ESB) etc.

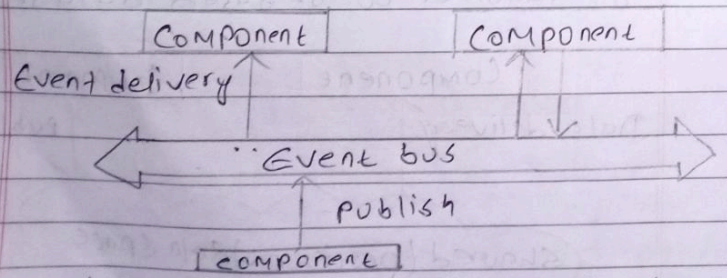

Fig: Showing Event based architecture

2) What is system architecture? Explain client server and peer to peer architecture.

=> A system architecture is the conceptual model that defines the structure, behaviour and more views of system

Client Server Architecture

It is a ~~[illegible]~~ computing model in which the server hosts, delivers and manages most of the resources and services to be consumed by client (which make request and server reply). This architecture has centralized security database. This database contains the security details, access details.

Advantages of client/server network

a) A client/server network contains the centralized system. Therefore, we can back up the data easily.

b) It has a dedicated server that improves the overall performance of whole system.
c) Security is better.
d) It also increases the speed of sharing resources as there is only single server administrator.

Disadvantages of client/server network
a) It is expensive as it require the server with large memory.
b) It require a dedicated network administrator to manage all resources.
c) If server computer is down then whole network will be affected.

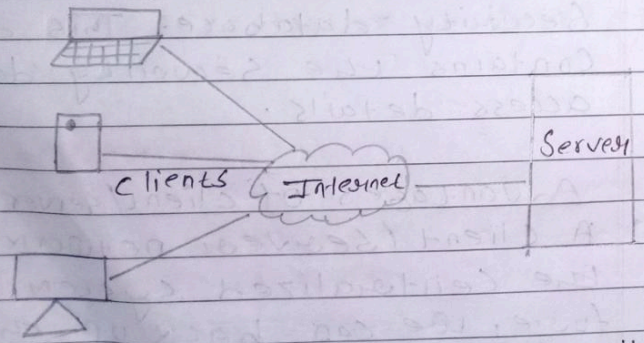

Fig: Client Server Architecture

Peer to peer network Architecture
It is a network in which all the computers are linked together with equal privelege and responsibilities for processing the data. It is useful for small environments, usually upto 10 computers. It has no dedicated server.

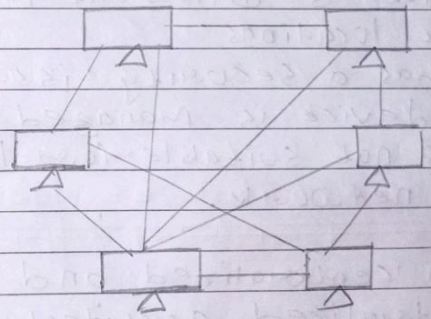

Fig: showing peer to peer network Architecture

Advantages
a) It has simple architecture and easy to install.
b) It is less costly.

c) It is suitable for small sized network.

d) It is easy to set up and maintain.

Disadvantages

a) It does not contain the centralized system. Hence, it cannot back up the data as data is different in different locations.

b) It has a security issue as the device is managed itself.

c) It is not suitable for large size network.

3) Explain centralized and decentralized architecture.

⇒

Centralized Architecture

A centralized architecture implies the availability of a single or a few entities that have control over the entire network. In the basic client-server model, processes in a distributed system are divided into two groups. A server is a process implementing a specific service and a client is a process that requests a service from server by sending it a request and subsequently waiting for server's reply.

*) Two-tier architecture

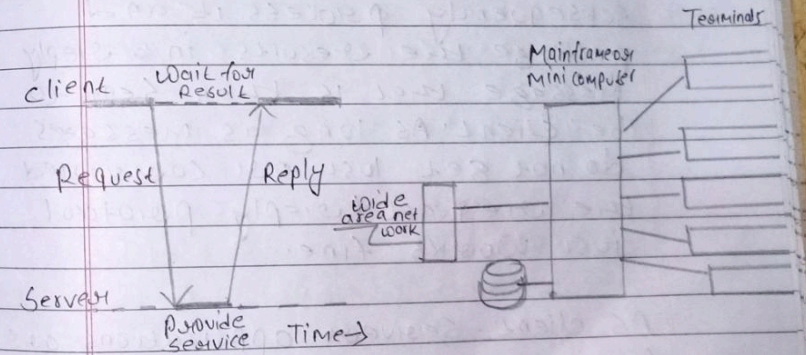

Fig: Showing 2-tier Architecture

Communication between a client and a server can be implemented by means of simple connectionless

protocol When the underlying network is fairly reliable. In these cases, when a client requests a service, it simply packages a message for the server; identifying the service it wants, along with necessary input data. The message is then sent to the server. Then later, in turn will always wait for incoming request subsequently process it and package the results in a reply message that is then sent to be client. As long as messages do not get lost or corrupted the request/reply protocol just works fine.

As client-server applications are targeted toward supporting user access to database, these are the essential architectural style:

a) The user-interface level (Display Mgmt)
b) The processing level (Contain applications)
c) The data level (Manage actual data)

Example explaining it

Consider a Internet search engine. Ignoring all animated banners, images and others; the user interface of a search engine is very simple. The back end is formed by huge database of web pages that has been prefetched and indexed. The core of search engine is a program that transforms the user's string of keywords into one or more database queries.

*) Three tier Architecture

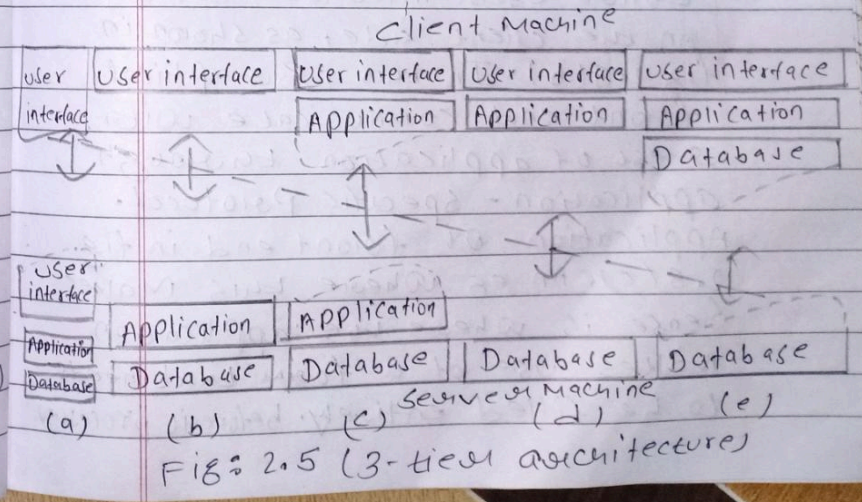

Fig: 2.5 (3-tier architecture)

In this organization, everything is handled by server while the client is essentially no more than a dumb terminal.

One possible organization is to have only the terminal-dependent part of user interface on the client machine, as shown in Fig. 2.5(a) and gives the applications remote control over presentation of their data.

An alternative is to place the entire user interface software on the client side, as shown in fig 2.5(b). In this, graphical front end communicate with rest of application through application-specific protocol. Application of front end in fig 2.5(c). An eg where this makes sense is where the application makes use of a form that needs to be filled entirely before it processed.

Another example of fig 2.5(c) is that of a word processor in which the basic editing functions execute on the client side where they operate on locally cached or in memory data.

In many client-server environments, the organizations shown in fig 2.5(d) and (e) are particularly popular. These organizations are used where the file system or database. Essentially, most of the application is running on client machine but all operations on files or database entries go to server. Fig 2.5(e) represents the situation where the client's local disk contains part of data. For eg, when browsing the web, a client can gradually build a huge cache on local disk of most recent inspected web pages.

Decentralized Architecture

In modern architecture (horizontal distribution) a client or server may be physically split up into logically equivalent parts, but each part is operating on its own share of complete data set thus balancing the load.

A class of modern system architectures that support horizontal distribution known as peer to peer systems.

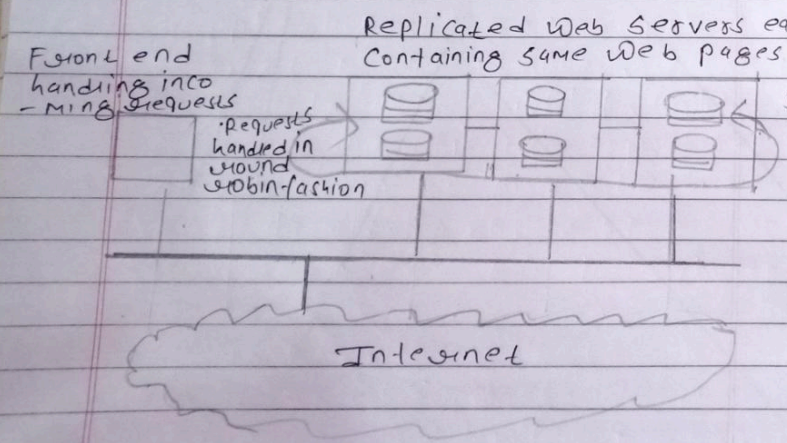

Fig: Peer to peer network

The process that constitute a peer to peer system are all equal. This means that functions that need to be carried out are represented by every process that constitutes the distributed system. Most of the interaction between process is symmetric.

It has two types

(i) Structured P2P system
(ii) unstructured P2P system

(i) Structured P2P system

- The node has to follow pre-defined structure.
- Every structured network suffers from poor scalability.
- The hash function value is used to insert an object in hash table and to retrieve it.
- A common approach that can be used to tackle the coordination between nodes, is to use DHT.

(ii) Unstructured P2P System

- There is no specific structure.
- The scalability is very high.
- There is no path for node to node transmission.
- These system rely on randomized algorithms for constructing an overlay network.

4) Explain Middleware organization.

⇒ The term Middleware suggests that it is software positioned between operating system and application. The concepts of object model ideally reflect the characteristics of distributed systems. An object encapsulates state and behaviour. The interface hides the details. An object therefore become a unit of distribution.

Tasks of Middleware.

a) Object Model Support
Middleware should offer mechanisms to support the concept incorporated in object model.

b) Operational interaction
Middleware should allow the operational interaction between two objects.

c) Remote interaction
Middleware should allow the interaction between two objects located in different address spaces.

d) Distribution transparency
From the standpoint of program, interaction between objects is identical for both local and remote interactions.

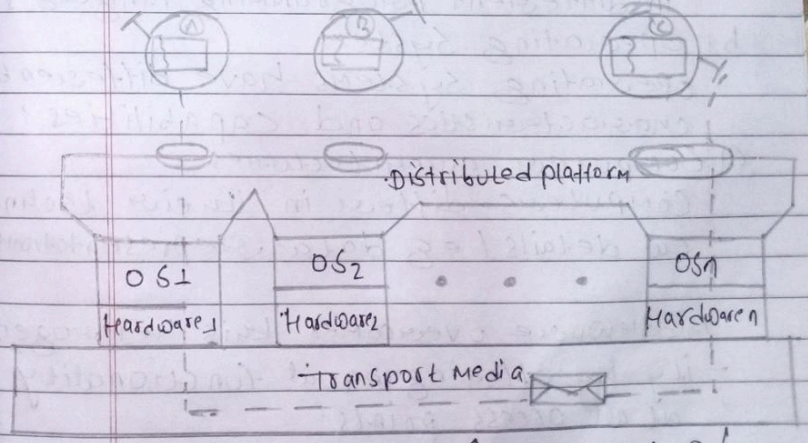

Fig: 2.7 Middleware for support of object based application

Middleware is located between application and operating system (in fig 2.7). Because, an object model serves as underlying paradigm, the application is represented as a set of interacting objects. Each object is explicitly allocated to a hardware platform. The heterogeneity hide by middleware ~~offers~~ exists at different places.

a) programming languages
Different object can be developed in different programming languages.

b) operating system
operating systems have different characteristics and capabilities.

c) Computer architectures
Computers differ in their technical details (eg data representations).

Middleware overcomes this heterogen-ity by offering equal functionality at all access points.

5) Explain about interceptor and general approach to adaptive software.

=> Interceptor is a software construct that will break the usual flow of control and allow other code to be executed. To make matters concrete, consider interception as supported in many object based distributed systems.

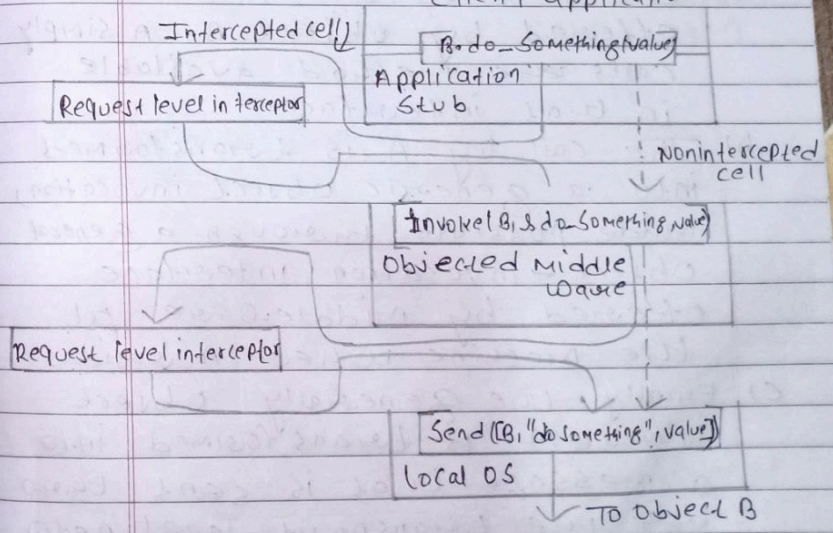

Fig 2.15 Using interceptors to handle remote object invocations

The basic idea is simple: an object A can call a method that belongs to an object B, while the latter resides on a different machine than A. Such a remote-object invocation is carried as a three-step approach:

a) Object A is offered a local interface that is exactly the same as the interface offered by object B. A simply calls the method available in that interface.

b) The call by A is transformed into a generic object invocation, made possible through a general object-invocation interface offered by middleware at the machine where A resides.

c) Finally, the generally object invocation is transformed into a message that is sent through the transport-level network interface as offered by A's local operating system.

General approach to adaptive software

Actually interceptors offers to adapt the middleware. The strong influences from environment (i.e resulting from Mobility, a strong variance in quality of service of networks, failing hardware and so on) have brought many designers of Middleware to consider the construction of adaptive software.

There are three basic techniques to come to software adaption. They are described below:

a) Separation of concerns
b) Computational reflection
c) Component-based design

a) Separating concerns

It relates to the traditional way of modularizing systems: Separate the parts that imple

ment functionality from those that take of other things such as reliability, performance etc. One can argue that developing ~~software~~ middleware for distributed applications is largely about handling extra functionalities independent from applications.

b) Computational reflection
It refers to the ability of a program to inspect itself and if necessary adapt it's behaviour. Reflection has been built into programming languages, including Java and offers a powerful facility for runtime modifications.

c) Component-based design
It design supports adaptation through composition. A system may either be configured statically at design time or dynamically at runtime. The latter requires support for late binding, a technique that has been successfully applied in programming language environments, but also for operating systems where modules can be loaded and unloaded at will.