Q.1> Explain the goals of D.S. in detail.

Four important goals that should be met to make building a distributed system worth the effort. A distributed system should make resources easily accessible; it should reasonably hide the fact that the resources are distributed across a network; it should be open; and it should be Scalable.

1. <u>Making Resources Available</u>

The main goal of D.S. is to make it easy for the users to access remote resources & to share them in a Controlled & efficient way.

Connecting users & resources also makes it easies to collab & exchange info.

2. <u>Distribution Transparency</u>

An important goal of a DS is to hide the fact that its processes & resources are physically distributed across multiple Computers. A DS that is able to present itself to users & applications as if it were only a single Computer system is said to be transparent.

There are many types of transparancies: Access, location, Migration, Replication, Relocation, Concurrency, Failure & Persistance.

3. <u>Openness</u>

An open DS is a system that offers services according to std. rules that describe the syntax & semantics of those services. In DS, services are generally specified through interfaces often described in an Interface Definition Language (IDL).

Scalability

It Can be measured along 3 dimensions:-

1> Scalable with respect to size of users & resources.

2> Scalable with respect to geography of users & resources.

3> Scalable with respect to the Admin Nodes.

Q.2> Differentiate between cluster Computing Systems & grid C.S.

Characteristic	Cluster C.S.	Grid C.S
1. population	Commodity Computers	Commodity & high end Computers
2. Ownership	Single	Multiple
3. Discovery	Membership Services	Centralised Index & decentralised info
4. User Management	Centralized	Decentralized
5. Resource Management	Centralized	Distributed
6. Allocation	Centralized	Decentralized
7. Inter-operability	VIA & proprietary	No Standards being developed
8. Single System Image	Yes	No
9. Scalability	100s	1000s
10. Capacity	guaranteed	Varies but high
11. Throughput	Medium	High.
12. Speed (Latency, Bandwidth)	Low, High	High, Low

Q.3> What are the applications of Distributed Information Systems

There are 2 important applications of Distributed Info. Systems. They are:-

1> <u>Transaction Processing Systems</u>

Programming using transactions require special primitives that must either be supplied by underlying D.S. or ~~they~~ by the runtime System. There are primitives like BEGIN_TRANSACTION which marks the start of a transaction, and try to commit, ABORT_TRANSACTION kills the transaction & restores old values, etc.

This all-or-nothing property of transactions is one of the 4 characteristic properties of a TPS. They are:-

1> Atomicity
2> Consistency
3> Isolation
4> Durability.

2> <u>Enterprise Application Integration</u>

Application Components should be able to Communicate directly with each other & not merely by means of request/reply behaviour.

Several types of Communication middleware exist with RPC, an application Component can effectively send a request to another application Component by doing a local procedure call, which results in request being packaged as a message & sent to the Callee.

A remote method Invocation (RMI) can also be used which is essentially same as RPC except that it operates on objects instead of Applications.

Scanned by CamScanner

Q.4) What are the requirements of distributed pervasive system? Explain any one application.

The 3 requirements of pervasive ~~app~~ systems are :-

1. Embrace contextual changes
2. Encourage ad hoc composition.
3. Recognize sharing as the default.

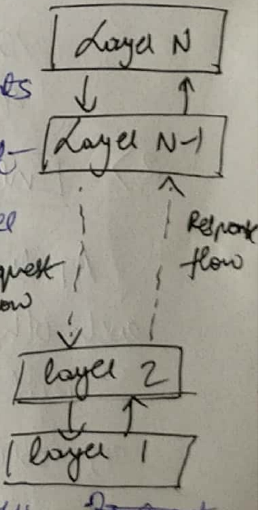
Home systems is a popular application of D.P.S which may perhaps also be least constrained. They are systems built around home networks. These systems generally consist of one or more personal computers, but more importantly integate typical consumer electronics such as TVs, audio & video equipment, gaming devices, phones & other personal wearables into a single system

Q.5) Describe different architectural styles of D.S.

The different architectural styles of D.S. are :-

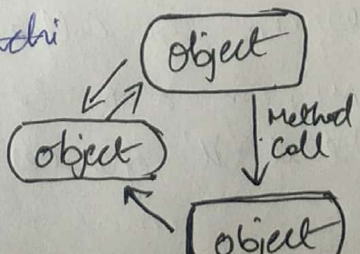
1. Layered architecture

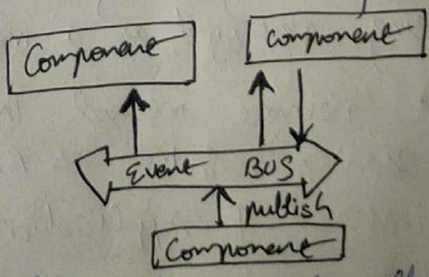
The basic idea for layered style is simple components are organised in a layered fashion where a component at layer L is allowed to call components at underlying layer but not the other way around.



2. Object-based architecture

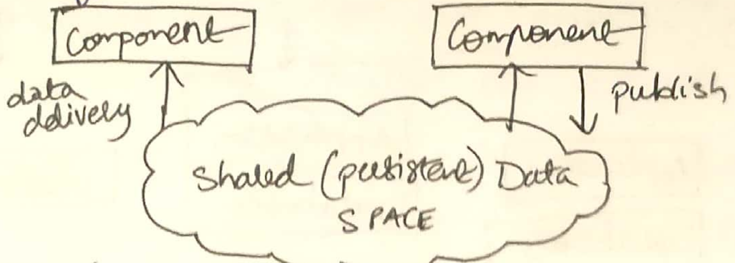
Data centred architectures evolve around the idea that processes communicate through a common repository. ~~Event~~ -~~based archi~~





3. In Event based architectures, processes communicate through the propagation of events, which optionally also carry data in a publish / subscribe type events.

4) Event based architectures can be combined with data centered ~~ar~~ architectures, yielding what is also known as shared data spaces.



Q.6) Explain centralised architecture.

In basic client-server model, processes in a D.S. are divided into 2 groups. A server that implements a specific service & client that requests a service from a sender. Such communication can be implemented by ~~a~~ means of simple connectionless protocol when the network is reliable.

Application Layering

3 levels of distinction exist :-

1. User-Interface level, that contains necessary interfaces such as display management, etc.

2. Processing level, implements UI, a back end for accessing databases & control logic.

3. Data-level, contains programs that maintain actual data.

Multi-tiered Architectures

The simplest organization is to have 2 machines: client & server.

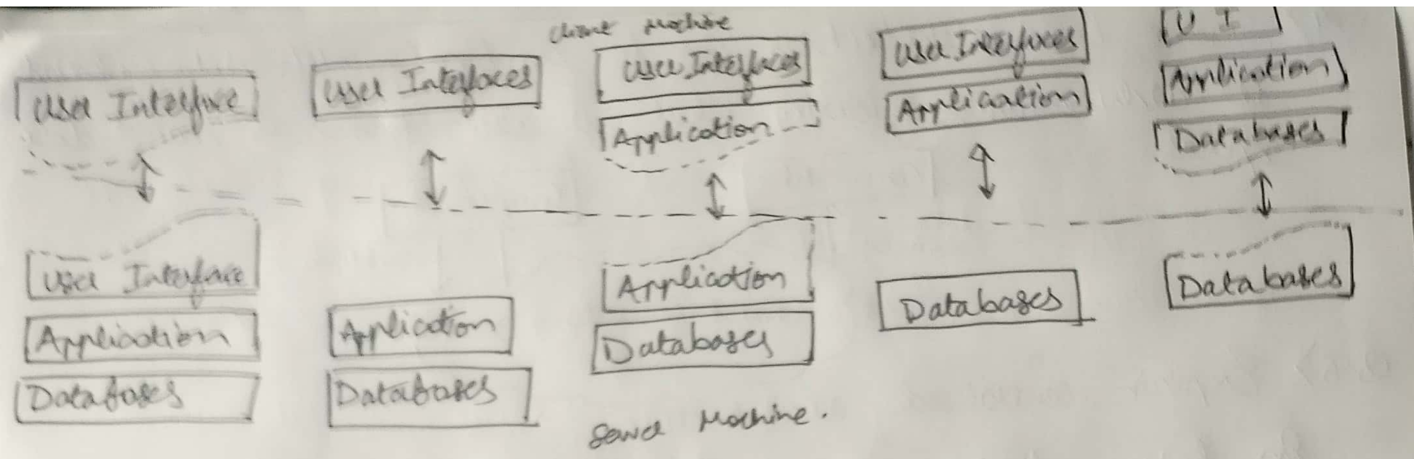
1. A client machine containing only user interface

2. A server machine containing processing & data level part

In this organization everything is handled by server while client only presents the response of server in a GUI

other organisation is to have only terminal dependent part of UI & give apps ~~the~~ remote control over presentation of their data.

figure next page shows other organisation of multitiered architectures

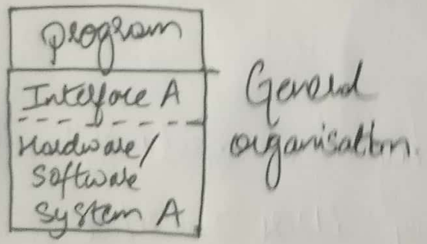


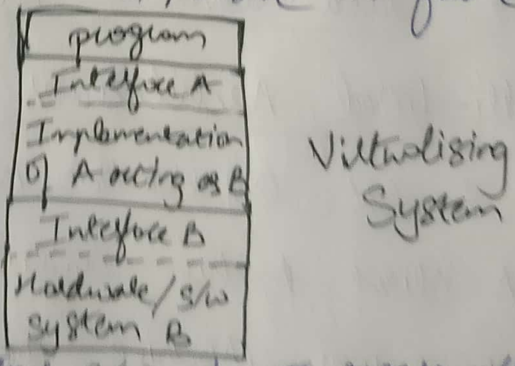
Q.7) Write short notes on Interceptors.

Conceptually an Interceptor is nothing but a software construct that will ~~be~~ break the usual flow of control & allow other application specific code to be executed. In many cases having only limited interception facilities will improve management of the software & distributed system as a whole.

Q.8) Explain the role of virtualisation in DS.

Virtualisation deals with extending an existing interface so as to mimic the behaviour of another system. One of the most important reasons for introducing virtualisation was to allow legacy software to run on expensive mainframe hardware.





First, while h/w & low level systems s/w change reasonable fast, s/w at higher levels of abstraction are more stable. Virtualisation can help here by porting legacy interfaces to new platforms & opening up the latter for large classes of existing programs.
The diversity of platforms & machines can be reduced by essentially letting each application run on its own Virtual Machine including related libraries & operating systems which run on common platforms.

the client ... the server

Q.9) Explain design issues of Distributed ~~[illegible]~~ servers.

1. There are several ways to organise servers. In case of Iterative servers, the server itself handles the request & if necessary returns a response to client. A concurrent server doesn't handle the request itself, but passes it to a separate thread or another process, after which it immediately waits for next incoming requests.

2. Another design issue is where clients contact a server. In all cases, clients send requests to an endpoint A.K.A. port where the server is running. There are many services that don't require a pre assigned port.

3. Another issue that needs to be taken into account is whether & how a server can be interrupted such as using out-of-bound data, etc.

4. A final important design issue is whether or not the server is stateless. A stateless server doesn't keep info on the state of its clients & can change its own state w/o informing the client.

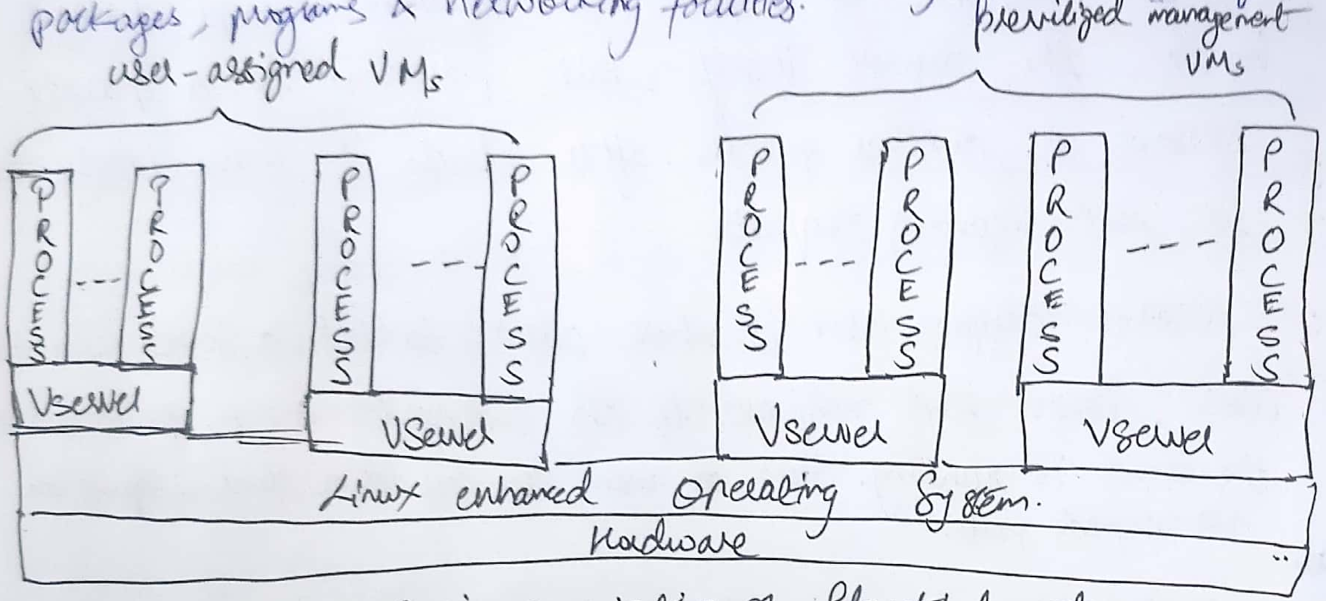
Q.10) Explain basic organisation of planet Lab.

Planet Lab is a collaborative D.S. in which different organisations donate one or more computers, adding up to a total hundreds of nodes. Together, these ~~components~~ computers for a 1-tier server cluster, where access, processing & storage can all take place on each node individually. Management of Planet Lab is by necessity almost entirely distributed.

In Planet Lab, an organisation donates 1 or more nodes where each node is a single computer or could be a cluster of machines. Each node is organized. There are 2 important components, The Virtual machine manager (VMM) which is an enhanced Linux OS, etc

Scanned by CamScanner

enhancements consist of adjustments for supporting the 2nd component namely vserver. A vserver can best be thought as a separate ~~env.~~ environment in which a group of processes run. processes from vserver are completely independent & can't directly share any resources, instead a vserver provides an env. consisting of its own software packages, programs & networking facilities.



Basic organisation of PlanetLab node.

The linux VMM ensures that vserver are separated: processes in different vserver are executed concurrently & independently each making use of ~~•~~ S/w packages & programs available in their own env. The isolation b/w processes in different vservers is strict. The VM in PlanetLab run on top of Linux O.S..

Q.11) Different approaches to code migration

Code migration in distributed systems took place in form of process migration in which an entire process was moved from one machine to another. Moving a process is costly & is done only if their is a significant increase in performance. The overall performance is increased by moving processes on a heavily loaded machine to lightloaded machines.

Load distribution algorithms by which decisions concerning allocation & redistribution of tasks are made w.r.t a set of processors ~~are~~ play crucial role in compute-intensive systems. The support for code migration can also improve performance by exploiting parallelism, but w/o usual intricacies in web.

An alternative is to let server provide client implementation when the client binds to server. At that point, the client dynamically downloads the implementation, ~~[illegible]~~ goes through necessary initialisation steps & invoked the server

Q.12> Explain RPC mechanism & basic operations.

Many D.S. have been based on explicit message & exchange b/w processes. The procedures send & recieve messages with no everything. When a process on machine A Calls a procedure on machine B the calling process on A is suspended & execution of called fn takes place on B. Info can be transported from A to B in the parameter & can come back as procedure result. No message passing is visible to the user. This method is called Remote Procedure Calls (RPC).

Basic operation

count = read (fd, buf, nbytes);

where fd is an integer indicating a file, buf is array of char into which data are read. nbytes gives info on how many bytes to read.

After read procedure has finished running it puts the return value in a register, removed return address & transfers control back to caller. The caller then removed parameters from stack, returning stack to its original state. The parameters can be passed either ~~to~~ using call-by-value or call-by-reference.

Q.13> What is stream-oriented & multicast oriented communication?

A stream oriented communication, a data stream is a connection oriented communication facility that supports isochronous data transmission called a stream

A data stream is a sequence of data units. There are 2 types of data discrete & continuous. for stream oriented communication 3 transmission model are available:—

- Asynchronous transmission mode
- Synchronous transmission mode
- Isochronous transmission mode.

Multi-Cast Communications :

Sending data to multiple recievers is known as multicast-Communication.

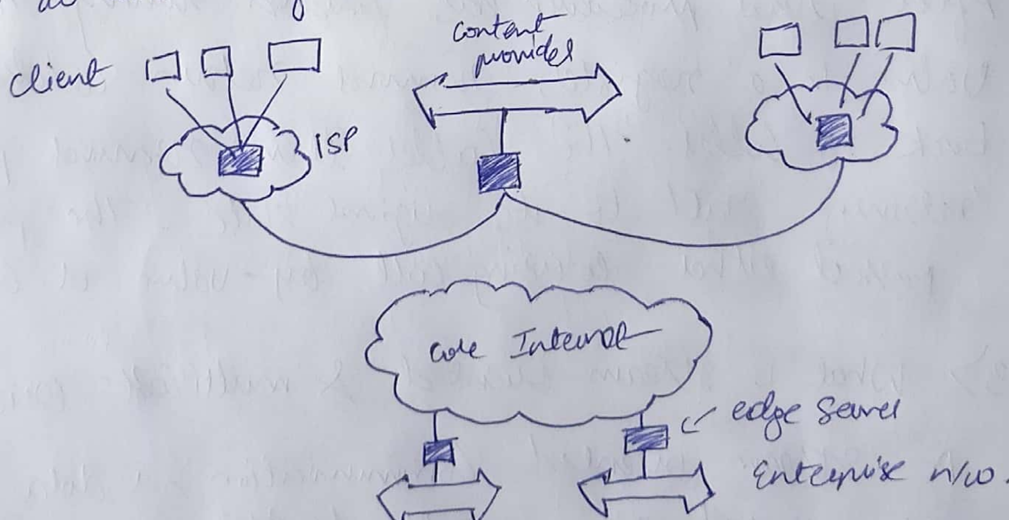
Multicast Communication Can be accomplished by Setting up explicit Communication paths, gossip-based information dissemination provides simple ways.

Q.15> Explain Hybrid architecture.

They are some specific class of distributed Systems in which client-Server Solutions are combined with decentralised Solutions.

Edge-Server Systems

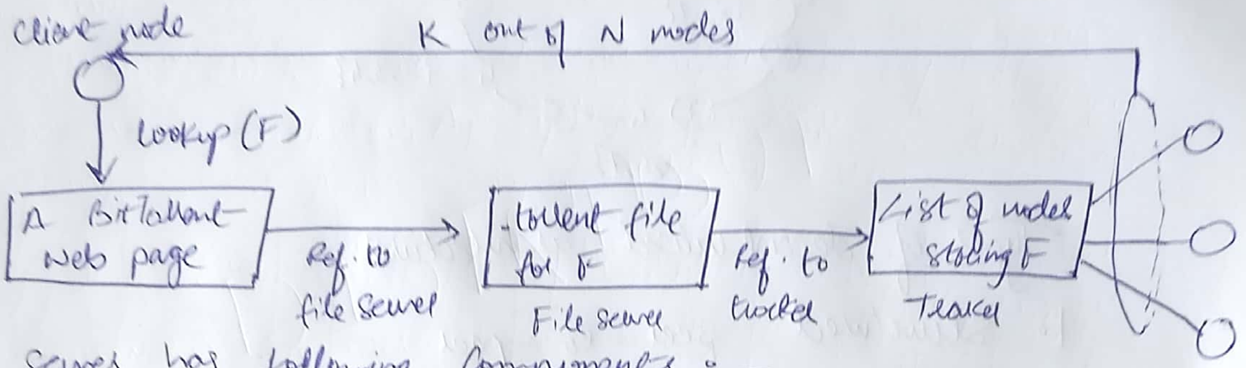
These systems are deployed on the Internet where servers are placed "at the edge" of the network. This edge is formed by the boundary between enterprise network and the actual Internet for eg as provided by ISPs. Likewise as endusers at home connect to internet through ISPs, the ISP can be considered as residing at the end of the Internet.



End users, or clients in general, connect to internet by means of edge server.

Collaborative Distributed Systems

Hybrid architectures are notably deployed in CDS. Bit-Torrent file sharing system is a peer to peer file downloading system. The basic idea is that when an enduser is looking for a file f, he downloads chunks of file from other users until the downloaded chunks can be assembled together yielding the complete file. An important design goal was to ensure collaboration.



each server has following components :-

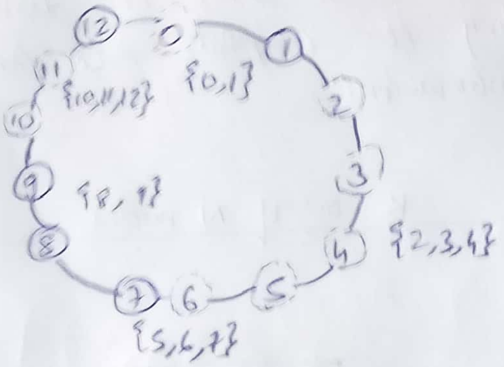
1. A component that can redirect client requests to other services.
2. A component for analysing access patterns
3. A component for managing the replication of web pages.

servers communicate with the broker completely analogous to what one would expect in a client - server system. The broker can be replicated, this type of replication is widely applied.

Q.16) Describe structured peer to peer architecture. What are the types of decentralised architecture?

Structured peer-to-peer Architecture

In a structured peer-to-peer architecture, the overlay n/w is constructed using a deterministic procedure. The most used procedure is to organize the processes through a distributed hash Table (DHT). In a DHT-based system, data items are assigned a random key from a large identifier space such as 128-bit or 160-bit identifier. For example, in chord system, the nodes are logically comprised in a ring such that a data item with key K is mapped to the node with the smallest identifier id ≥ K. This node is referred to as the successor of key K & denoted as succ(K). To lookup data item, an application running on an arbitrary node calls the fn lookup(K) which returns the n/w address of succ(K). At that point the application can contact the node to obtain a copy of the data item.



mapping of data items onto nodes in chord

Types of decentralised architectures :-

1. Structured peer-to-peer
2. unstructured peer-to-peer
3. Overlay Network
4. Superpeers
5. Client-server combined with peer-to-peer.

Scanned by CamScanner