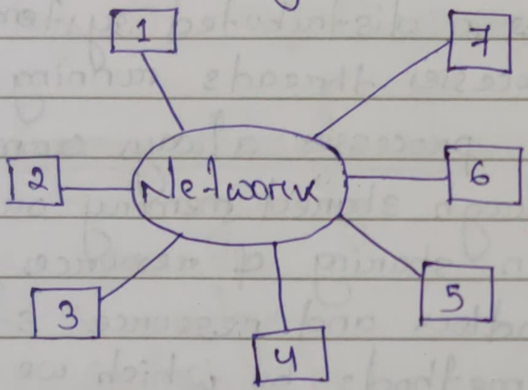
Distributed System

_/_/_

Ch-1

A distributed system is one in which components located at network computer communicate their actions only by passing messages.

Features of distributed systems :-
→ Concurrency of components
→ Independent of failure of components
→ Lack of global clock



If at any point of time a system fails, remaining systems will work properly.

Q. Difference between centralized system and distributed system.

centralized system	Distributed system
→ All resources accessible	→ Resources may not accesible
→ Component shared by users all the time.	→ Not shared
→ single point of control	→ Multiple point of control
→ single point of failure	→ Multiple point of failure
→ One component with non-autonomous parts	→ Multiple autonomous components.

Autonomous Components

A distributed system is a collection of autonomous computing elements that appears to its users as a single coherent system (within a system).

/ /

This definition refers to two features of distributed system. The first one is that a distributed system is a collection of computing elements each one able to behave independency of each other. The second element is that users.

Concurrency :-

It is the execution of multiple instructions at the same time. It happens in distributed system when there are several processes threads running in parallel. The running processes always communicate with each other through shared memory or message. Concurrency results in sharing of resources resulting in problems live deadlock and resource starvation.
There are 3 basic methods by which we can improve the performance of a system while concurrency.

① Reduce latency
② Hide latency
③ Increased throughput

Appraches for programming concurrency
1> Sequential
2> Declarative
3> Message Passing
4> Sharded state

Heterogeneity :-

In this all the documents are accessible that are available on the internet. Even though the documents are located in different types of machines.

_/_/_

Heterogeneity applies to networks, computer hardware, operating systems, programming languages. Middleware helps to solve the problem of heterogeneity. It also provides an uniform computational model for use by the programmer of server and distributed system. Middleware that maintains heterogeneity is common object request broker architecture (COBRA). It is a standard defined by the object management group.

Common characteristics

1. Heterogeneity
2. Openness
3. Security
4. Scalability
5. Failure handling
6. Concurrency
7. Transparency.

Openness

The openness of an distributed system is determined by its ability to offer new resource sharing services. Open systems are one in which there key interfaces for accesing shared resources and information on how they work at any given time can be made available through publication on broadcasting.

Security

Security is required for authentication of ~~[illegible]~~ front-end and privacy.

Scalability

It is required for distributed system to accomodate more users and respond faster.

_/_/_

failure handling
Fault tolerance ~~is~~ achieved by
a) Recovery (b) Duplicacy

Transparency
The idea of transparency is common in software systems as it allows for a degree of independence between clients and implementation. This may be positive because it prevents the system from dealing correctly with many complex partial failures that arrives when trying to implement services on its own.

Types of transparency

1. access	4. Replication	7. performance
2. location	5. failure	8. scaling
3. concurrency	6. mobility	

Access
It allows the same operations to be used to access local and remote resources.
Ex:- Navigation in the web, file system operations, SQL queries.

Location
It enable information objects to be accessed without knowing the physical location of the data.
Ex- pages in the web, tables in distributed database.
Users of the network file system can access files by name and do not need to know whether the file besides on a local on a remote disk.

_/_/_

Concurrency
It enables several processes to operate concurrently using shared information objects without interference between them. The shared items are all accessed at the same time.
Ex :- DBMS, ATM network
Replication
It enables multiple instances of information objects to be used to increase reliability and performance without unowledge of the replicas by users on application programs.
Ex:- distributed DBMS, Mirroring webpages
Failure
It enables the recovery of the faults and also allows useres and applications to complete their task despite the failure of the other components.
Ex- dbms
Mobility
It allows the movement of information objects within a system without affecting the operations of users on application programs.
Performance
It allows the system to be reconfigured to improve performance as loads vary.
Ex- distributed OS
Scaling
It allows the system and applications to expand in a scale without change to the system structure on the application algorithm.
Ex- distributed OS

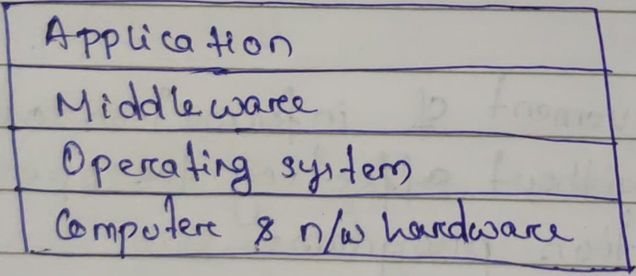
Example of distributed systems

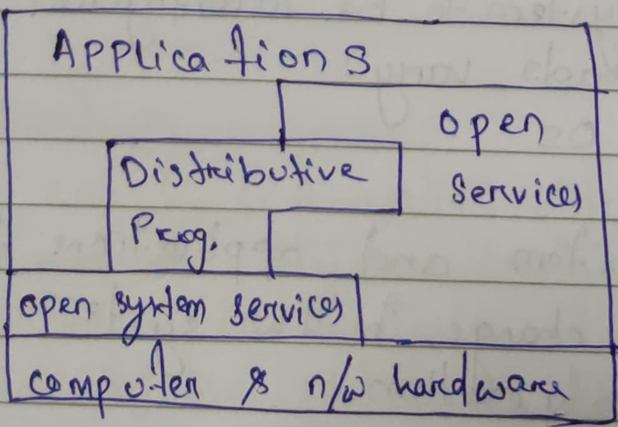
1. Local area network and Intranet
2. Data base management system
3. Automatic teller machine (ATM) network
4. Internet
5. Mobile and ubiquitous computing

Specific / Basic design issues :-

1. Naming
2. Communication
3. Software structure
4. System architecture
5. workload allocation
6. Consistency Maintenance

Software structure





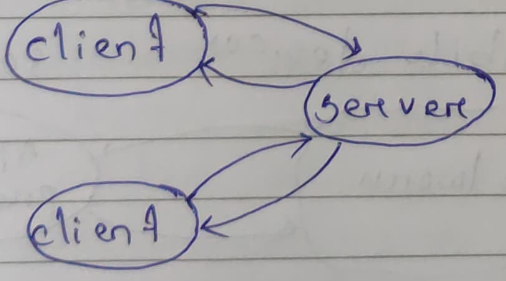
→ Layeres and dependencies in distributive system

__/__/__

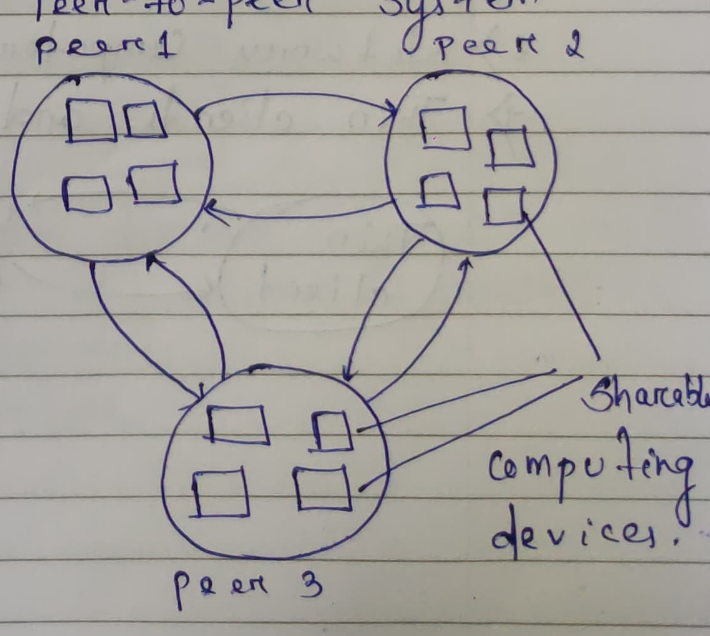
- → Kernel only perform basic resource management with the addition of interprocess communication. A new class of software component called open services to provide all other shared resources and services.
- → Application program may use operating system, kernel services, distributed programming support and open services.
- → Distributive programming support includes run time support for language facilities that allow program written in conventional language to work together.
- → Middleware provides run time support for programming languages such as interpreters and libraries.
- → Operating system is the main system software to manage basic resources to provide user and application services for example memory allocation and protection, process and disk scheduling, user authentication, file management, clock facilities.

System architecture

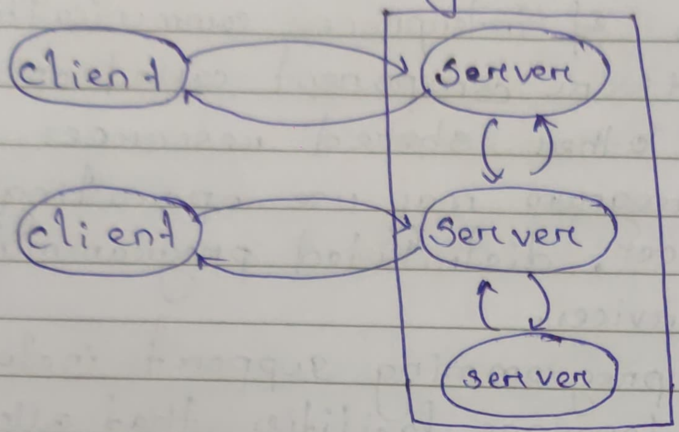
1> client server



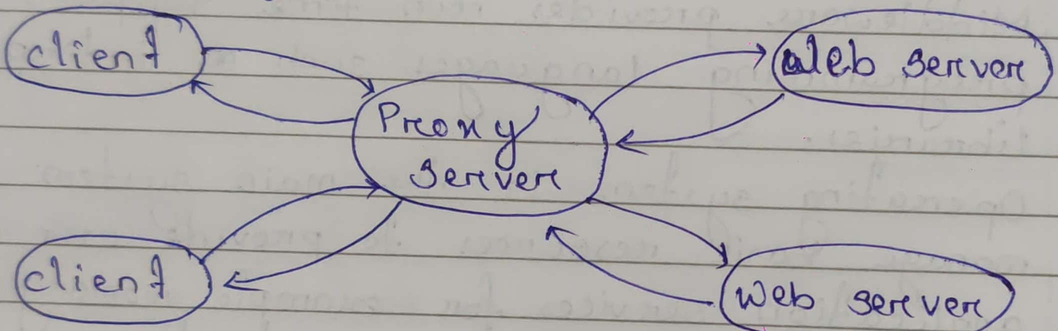
2> Peer-to-peer system



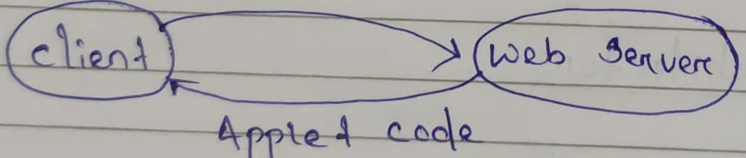
3> Services provided by multiple servers



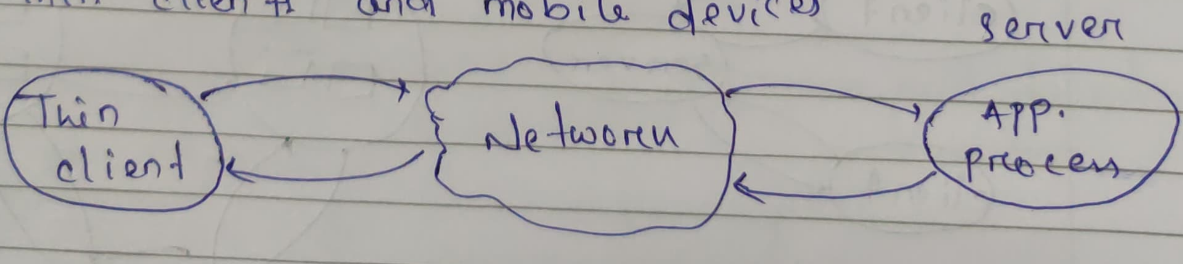
4> Web proxy server



5> Mobile code and agents



6> Network Computers
7> Thin clients and mobile devices


Thin client
Network
server
APP. process