UNIT 160 – UPSC - Designing for Online and Distributed Environments

Technological innovation has modernize the life of human. Computers and networks are progressively able to support distributed collaborative multimedia applications.

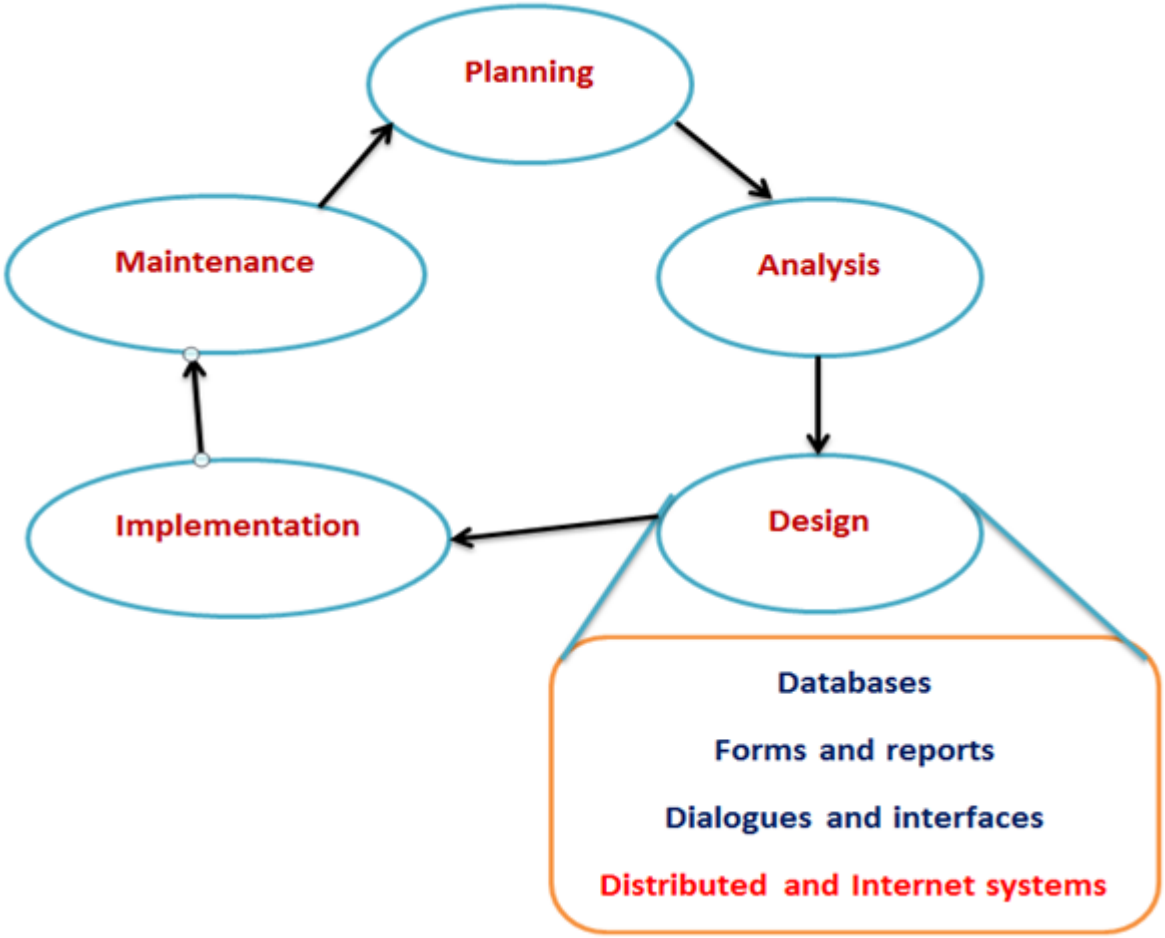
Process of Distributed and Internet Systems:

This process is similar to designing single-location systems.

Due to multi-location deployment, several design issues must be considered.

There is more opportunity for failure due to number of components.

Main issues involve in these systems are ensuring reliability, availability, survivability, performance. Figure: Distributed and Internet Systems



Designing Distributed System:

A distributed system is an application that performs a collection of protocols to synchronize the actions of multiple processes on a network, such that all components cooperate together to perform a single or small set of related tasks. There are numerous advantages of building distributed system such as the ability to connect remote users with remote resources in an open and scalable way. A distributed system can be much larger and more dominant given the combined capabilities of the distributed components, than combinations of stand-alone systems.

A distributed system must have the following characteristics:

1. Fault-Tolerant: It can recover from component failures without performing incorrect actions.
2. Highly Available: It can restore operations, permitting it to resume providing services even when some components have failed.
3. Recoverable: Failed components can restart themselves and rejoin the system, after the cause of failure has been repaired.
4. Consistent: The system can coordinate actions by multiple components often in the presence of concurrency and failure. This underlies the ability of a distributed system to act like a non-distributed system.

5. Scalable: It can operate correctly even as some aspect of the system is scaled to a larger size. For example, we might increase the size of the network on which the system is running. This increases the frequency of network outages and could degrade a "non-scalable" system. Similarly, we might increase the number of users or servers, or overall load on the system. In a scalable system, this should not have a significant effect.
6. Predictable Performance: The ability to provide desired responsiveness in a timely manner.
7. Secure: The system authenticates access to data and services (Birman, Kenneth, 2005)

In distributed systems design, handling failures is vital task. Failures has two apparent categories that include hardware and software. Hardware failures were major concern till the decade of 80's, but since then internal hardware reliability has improved extremely. Decreased heat production and power consumption of smaller circuits, reduction of off-chip connections and wiring, and high-quality manufacturing systems have a positive role in enhancing hardware reliability. Presently, problems are most often related with connections and mechanical devices, i.e., network failures and drive failures.

Software failures are major issue in distributed systems. Even with rigorous testing, software bugs account for a substantial fraction of unplanned downtime (estimated at 25-35%). Residual bugs in mature systems can be classified into two main categories (Gray, J. and Reuter, A., 1993).

Types of failures that can occur in a distributed system:

Halting failures: A component simply stops. Tester will not be able to detect the failure except by timeout. It either stops sending "I'm alive" (heartbeat) messages or fails to respond to requests. User's computer freezing is a halting failure.

Fail-stop: A halting failure with some kind of notification to other components. A network file server telling its clients it is about to go down is a fail-stop. Omission failures: Failure to send/receive messages primarily due to lack of buffering space, which causes a message to be discarded with no notification to either the sender or receiver. This can happen when routers become overloaded.

Network failures: A network link breaks.

Network partition failure: A network fragments into two or more disjoint sub-networks within which messages can be sent, but between which messages are lost. This can occur due to a network failure.

Timing failures: A temporal property of the system is violated. For example, clocks on different computers which are used to coordinate processes are not synchronized; when a message is delayed longer than a threshold period, etc.

Byzantine failures: This captures several types of faulty behaviors including data corruption or loss, failures caused by malicious programs, etc. (Birman, Kenneth, 2005).

Heisenbug: it is a type of bug that seems to disappear or alter its characteristics when it is observed or researched. A common example is a bug that occurs in a release-mode compile of a program, but not when researched under debug-mode.

Bohrbug: A bug that does not disappear or alter its characteristics when it is researched. A Bohrbug typically manifests itself reliably under a well-defined set of conditions.

Heisenbugs tend to be more widespread in distributed systems than in local systems. Because it is difficult for programmers to obtain a coherent and comprehensive view of the interactions of synchronized processes.

Distributed systems use:

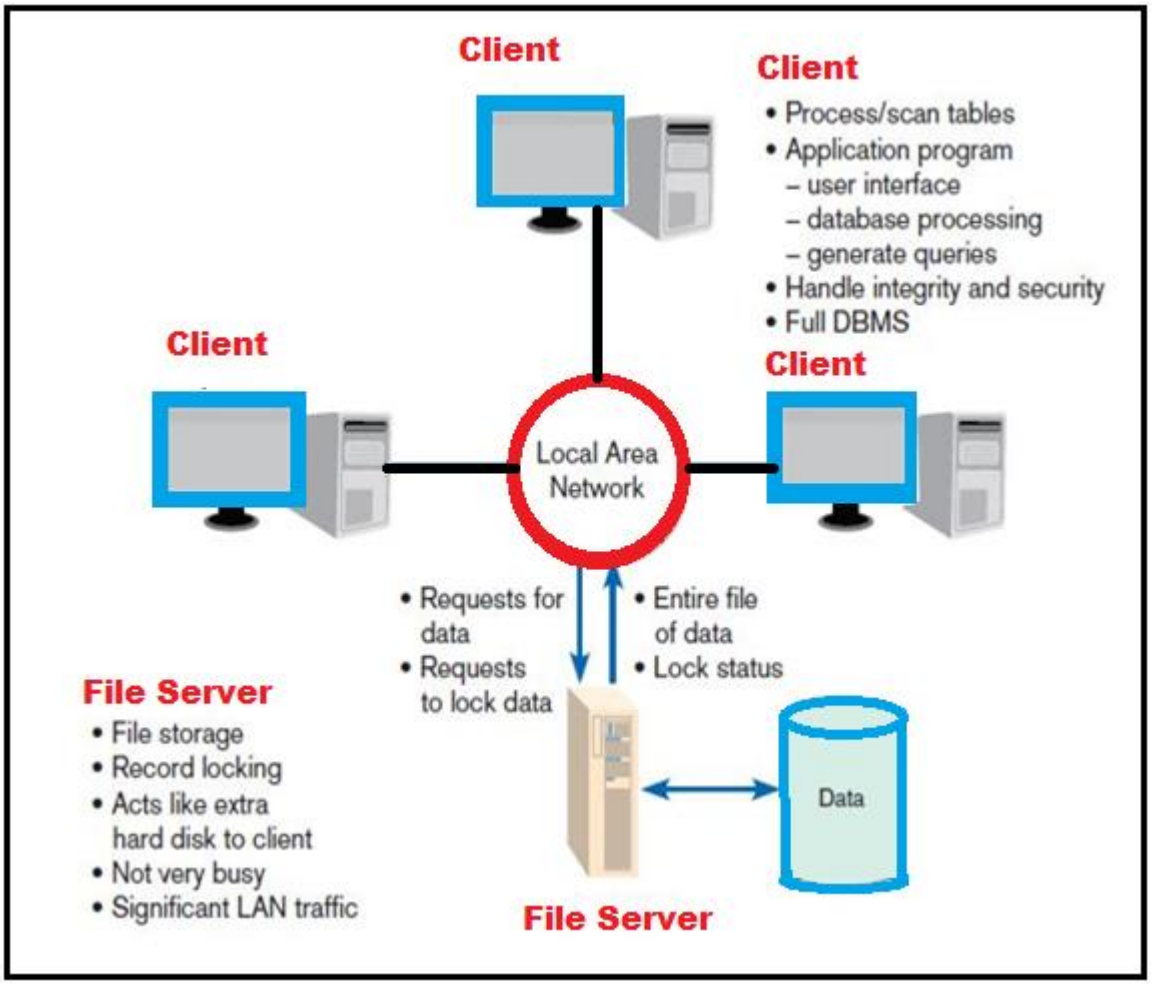
- LAN-based file server architecture.
- Client/server architecture.

Designing system for LAN:

LAN (Local Area Networks): Key components of LAN are the cabling, hardware, and software used to connect workstations, computers, and file servers located in a confined geographical area. Typically within one building or campus.

File server: It is a device that manages file operations and is shared by each client PC attached to a LAN.

File server model:



Drawbacks of file server model:

There is excessive data movement. Entire data table must be transferred, instead of individual records.

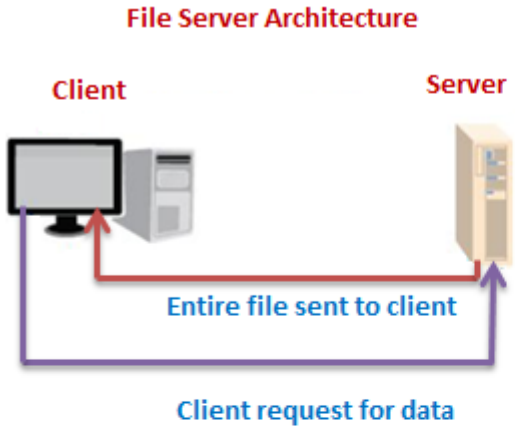
It needs for powerful client workstations. Each client workstation must devote memory to a full DBMS.

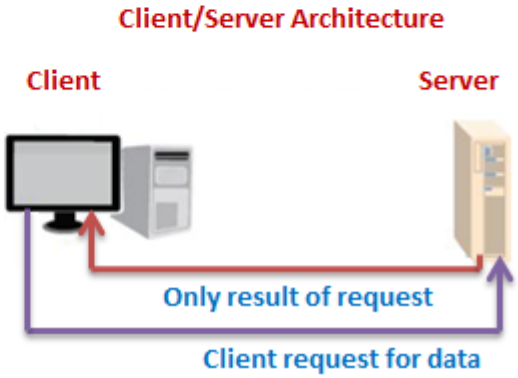
There is decentralized data control. Record concurrency control, recovery, and security are complicated.

Designing Systems for a Client/Server Architecture:

Client/server architecture: A LAN-based computing environment is in which central database server or engine performs all database commands sent to it from client workstations, and application programs on each client concentrate on user interface functions.

File Server vs. Client/Server





Designing Systems for a Client/Server Architecture:

Application processing is divided between client and server. Client manages the user interface. Database server is responsible for data storage and query processing.

Database engine: The (back-end) portion of the client/server database system running on the server that provides database processing and shared access functions.

Client: The (front-end) portion of the client/server database system that provides the user interface and data manipulation functions.

Application program interface (API): Software building blocks that are used to ensure that common system capabilities, such as user interfaces and printing, as well as modules are standardized to facilitate data exchange between clients and servers.

Client/Server Advantages:

1. Leverages benefits of microcomputer technology
2. Processing performed close to data source
-Improves response time
-Reduces network traffic
3. Facilitates use of GUIs
4. Encourages acceptance of open systems

Differences between client server and file server:

Characteristic	**File Server**	**Client/Server**
Processing	Client only	Both client and server
Concurrent data access	Low-managed by each client	High-managed by server
Network usage	Large files and data transfers	Efficient data transfer
Database security and integrity	Low- managed by each client	High-managed by server
Software maintenance	Low-software changes just on server	Mixed-some new parts must be delivered to each client
Hardware and system software flexibility	Client and server decoupled and be mixed	Need for greater coordination between client and server

Distributed systems design is apparently a challenging endeavor. In Client-server applications, the server offers some service such as processing database queries or sending out current stock prices. The client uses the service provided by the server, either displaying database query results to the user or making stock purchase recommendations to an investor. The communication that occurs between the client and the server must be consistent. That is, no data can be dropped and it must reach on the client side in the same order in which the server sent it.

There are numerous servers user encounter in a distributed system. For instance, file servers manage disk storage units on which file systems reside. Database servers house databases and make them available to clients. Network name servers implement a mapping between a symbolic name or a service description and a value such as an IP address and port number for a process that provides the service.

In distributed systems, there can be many servers of a particular type such as multiple file servers or multiple network name servers. The term service is used to signify a set of servers of a particular type. It is said that a binding occurs when a process that needs to access a service becomes related with a particular server which provides the service. There are many binding policies that define how a particular server is chosen. For example, the policy could be based on locality (a Unix NIS client starts by looking first for a server on its own machine); or it could be based on load balance (a CICS client is bound in such a way that uniform responsiveness for all clients is attempted).

A distributed service may use data replication, where a service maintains multiple copies of data to permit local access at multiple locations, or to increase availability when a server process may have crashed. Caching is a related notion and very common in distributed systems. A process has cached data if it maintains a copy of the data locally, for quick access if it is needed again. A cache hit is when a request is fulfilled from cached data instead of the primary service.

Caching is akin to replication, but cached data can become stale. Therefore, there may need to be a policy for validating a cached data item before using it. If a cache is actively refreshed by the primary service, caching is identical to replication (Birman, Kenneth, 2005). The communication between client and server needs to be reliable.

Examples of distributed systems:
-Web
-Email
-DNS
-Peer-to-peer systems (file sharing, CDNs, cycle sharing)

Distributed File Systems (NFS, ...)
-Sensor Networks
-Akamai CDN

2. Designing Internet Systems:

Internet systems development is an area in which designer can help streamline Web coding and promote website security. It involves the creation, planning, maintenance and modeling of Internet-based computer systems.

Online system design include following process.
1. Standards:
-Design is simpler due to standards
-Naming: BIND
-Translation: Hypertext Transfer Protocol (HTTP)
-Formatting: Hypertext Markup Language (HTML)

2. Separating Content and Display:
-HTML has limitations due to format orientation of tags.
-Extensible Markup Language (XML) has been developed to separate content from display: Ability to create custom languages.

3. Future Evolution:
-Move from desktop PCs to thin clients: Most processing and data storage occurs on the server.

4. Aids to Site Consistency:
-Cascading Style Sheets: A set of style rules that tells a Web browser how to present a document
-Extensible Style Language (XSL): Specification for separating style from content when generating HTML documents

5. Design Issues Related to Site Management:
-Customer Loyalty and Trustworthiness:

Conveyed by:
-Design quality
-Up-front disclosure
-Comprehensive, correct and current content
-Connected to the rest of the Web
-Data security
-Personalization

Web Pages Must Live Forever:
-Customer Bookmarks
-Links from Other Sites
-Search Engine Referrals
-Old Content Adds Value

-System Security

It has been well established in technical studies that effective design is the result of understanding how a system fits within the framework of an organization.