3

PEER TO PEER SERVICES AND FILE SYSTEM

3.1 INTRODUCTION TO PEER TO PEER SYSTEMS

Peer-to-peer (P2P) is a decentralized communications model in which each party has the same capabilities and either party can initiate a communication session.

Unlike the client/server model, in which the client makes a service request and the server fulfills the request, the P2P network model allows each node to function as both client and server.

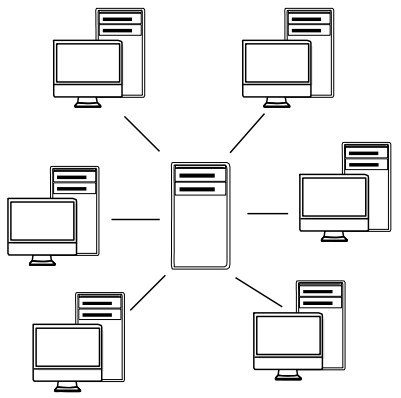
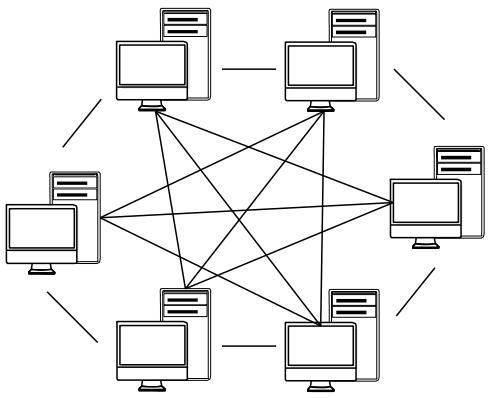
Server Based Communication	Peer to Peer Communication
	

Fig 3.1 Communication in Peer to peer and Client/ server model

P2P systems can be used to provide anonymized routing of network traffic, massive parallel computing environments, distributed storage and other functions. Most P2P programs are focused on media sharing and P2P is therefore often associated with software piracy and copyright violation.

Features of Peer to Peer Systems

- Large scale sharing of data and resources
- No need for centralized management
- Their design of P2P system must be in such a manner that each user contributes resources to the entire system.
- All the nodes in apeer-to-peer system have the same functional capabilities and responsibilities.
- The operation of P2P system does not depend on the centralized management.
- The choice of an algorithm for theplacement of data across many hosts and the access of the data must balance the workload and ensure the availability without much overhead.

Advantages of P2P systems over client/ server architecture

1) It is easy to install and so is the configuration of computers on the network.
2) All the resources and contents are shared by all the peers, unlike server-client architecture where Server shares all the contents and resources.
3) P2P is more reliable as central dependency is eliminated. Failure of one peer doesn‟t affect the functioning of other peers. In case of Client –Server network, if server goes down whole network gets affected.
4) There is no need for full-time System Administrator. Every user is the administrator of his machine. User can control their shared resources.
5) The over-all cost of building and maintaining this type of network is comparatively very less.

Disadvantages of P2P systems over client/ server architecture

1) In this network, the whole system is decentralized thus it is difficult to administer. That is one person cannot determine the whole accessibility setting of whole network.
2) Security in this system is very less viruses, spywares,trojans, etc malwares can easily transmitted this architecture.
3) Data recovery or backup is very difficult. Each computer should have its own back-up system
4) Lot of movies, music and other copyrighted files are transferred using this type of file transfer. P2P is the technology used in torrents.

P2P Middleware

- Middleware is the software that manages and supports the different components of a distributed system.
- In essence, it sits in the middle of the system.
- Middleware is usually off Middleware is usually off-the-shelf rather than specially written software.
- A key problem in Peer-to-Peer applications is to provide a way for clients to access data resources efficiently.
- Peer clients need to locate and communicate with any available resource, even though resources may be widely distributed and configuration may be dynamic, constantly adding and removing resources and connections.
- The P2P middleware must possess the following characteristics:
  - Global Scalability
  - Load Balancing
  - Local Optimization
  - Adjusting to dynamic host availability
  - Security of data
  - Anonymity, deniability, and resistance to censorship

3.1.1 Routing Overlays

A routing overlay is a distributed algorithm for a middleware layer responsible for routing requests from any client to a host that holds the object to which the request is addressed.

- Any node can access any object by routing each request through a sequence of nodes, exploiting knowledge at each of theme to locate the destination object.
- Global User IDs (GUID) also known as **opaque identifiers** are used as names, but they do not contain the location information.
- A client wishing to invoke an operation on an object submits a request including the object‟s GUID to the routing overlay, which routes the request to a node at which a replica of the object resides.

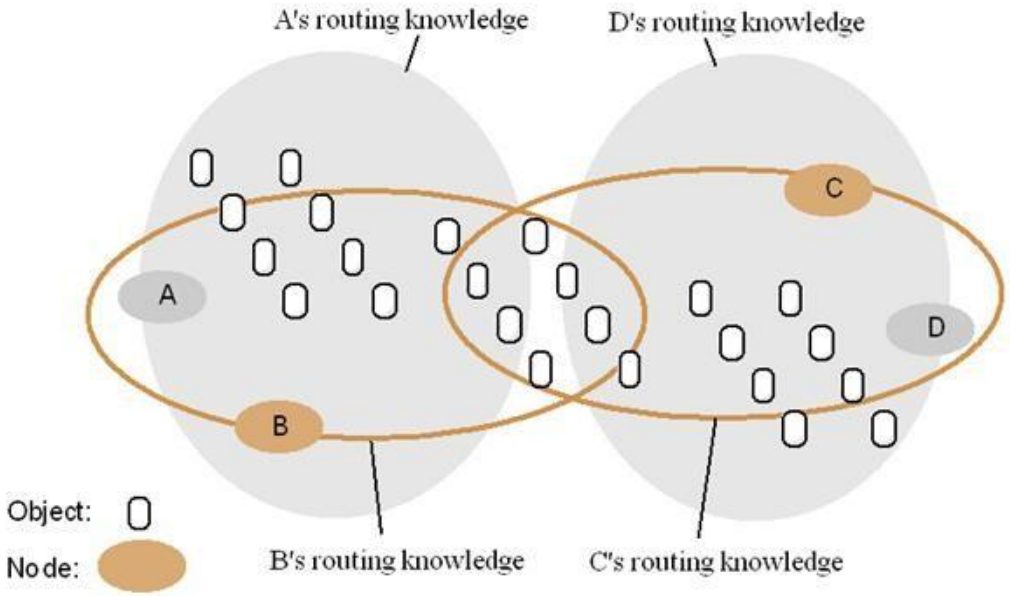


Fig 3.2: Information distribution in Routing Overlay

Differences between Overlay networks and IP routing

IP	Overlay Network
The scalability of IPV4 is limited to 2^{32} nodes and IPv6 is $2^{128.}$	Peer to peer systems can address more objects using GUID.
The load balancing is done based on topology.	Traffic patterns are independent of topology since the object locations are randomized.
Routing tables are updated asynchronously.	Routing tables are updated both synchronously and synchronously.
Failure of one node does not degrade the performance much. Redundancy is introduced in IP. n-fold replication is costly.	Routes and object references can be replicated n-fold, ensuring tolerance of n failures of nodes or connections.
Each IP can map to only one node.	Messages are routed to the nearest replica of the target object.
Secure addressing is possible only between trusted nodes.	Secure communication is possible between limited trusted systems.

Distributed Computation

- Distributed computing refers to multiple computer systems working on a single problem.
- Here a single problem is divided into many parts, and each part is solved by different computers.
- As long as the computers are networked, they can communicate with each other to solve the problem.
- The computers perform like a single entity.
- The ultimate goal of distributed computation is to maximize performance by connecting users and IT resources in a cost-effective, transparent and reliable manner.
- It also ensures fault tolerance and enables resource accessibility in the event that one of the components fails.

3.2 NAPSTER AND ITS LEGACY APPLICATION

- The Internet was originally built as a peer-to-peer system in the late 1960s to share computing resources within the US.
- Later the transfer of resources between systems took place by means of client server communication.
- Later Napster was developed for peer –to-peer file sharing especially MP3 files.
- They are not fully peer-to-peer since it used central servers to maintain lists of connected systems and the files they provided, while actual transactions were conducted directly between machines.

3.2.1 Working of Napster

- ✓ Each user must have Napster software in order to engage in file transfers.
- ✓ The user runs the Napster program. Once executed, this program checks for an Internet connection.
- ✓ If an Internet connection is detected, another connection between the user's computer and one of Napster's Central Servers will be established. This connection is made possible by the Napster file-sharing software.
- ✓ The Napster Central Server keeps a directory of all client computers connected to it and stores information on them as described above.

- ✓ If a user wants a certain file, they place a request to the Napster Centralised Server that it's connected to.
- ✓ The Napster Server looks up its directory to see if it has any matches for the user's request.
- ✓ The Server then sends the user a list of all that matches (if any) it as found including the corresponding, IP address, user name, file size, ping number, bit rate etc.
- ✓ The user chooses the file it wishes to download from the list of matches and tries to establish a direct connection with the computer upon which the desired file resides.
- ✓ It tries to make this connection by sending a message to the client computer indicating their own IP address and the file name they want to download from the client.
- ✓ If a connection is made, the client computer where the desired file resides is now considered the host.
- ✓ The host now transfers the file to the user.
- ✓ The host computer breaks the connection with the user computer when downloading is complete.

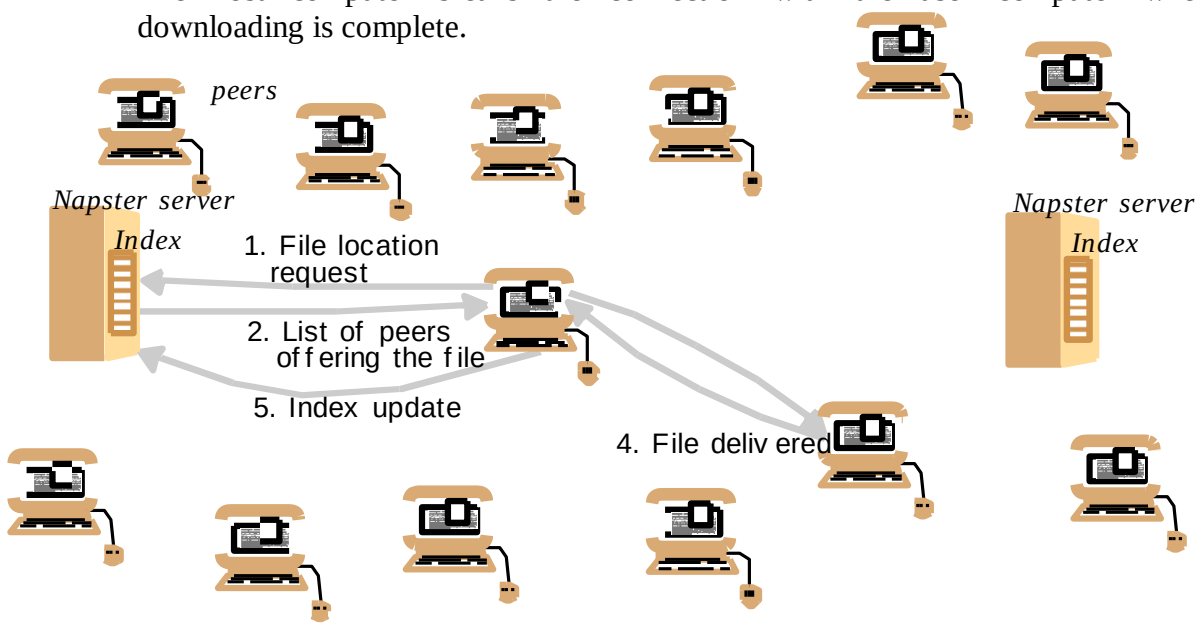


Fig 3.1: Napster

Limitations of Napster:

Discovery and addressing a file is difficult unless the location is distributed.

3.3 PEER TO PEER MIDDLEWARE

- ✓ Middleware is the software that manages and supports the different components of a distributed system.
- ✓ In essence, it sits in the middle of the system.
- ✓ Middleware is usually off Middleware is usually off-the-shelf rather than specially written software.
- ✓ A key problem in Peer-to-Peer applications is to provide a way for clients to access data resources efficiently.
- ✓ Peer clients need to locate and communicate with any available resource, even though resources may be widely distributed and configuration may be dynamic, constantly adding and removing resources and connections.
- ✓ The following are the functional requirements of the middleware:
    - ➢ Simplify construction of services across many hosts in wide network
    - ➢ Add and remove resources at will
    - ➢ Add and remove new hosts at will
    - ➢ Interface to application programmers should be simple and independent of types of distributed resources
- ✓ The following are the non functionalrequirements of the middleware:
    - ▪ Global Scalability
    - ▪ Load Balancing
    - ▪ Optimization for local interactions between neighboring peers
    - ▪ Accommodation to highly dynamic host availability
    - ▪ Security of data in an environment simplify construction of services across many hosts in wide network
    - ▪ Anonymity, deniability and resistance to censorship
- ✓ Sharing and balancing across large numbers of computers pose major design challenges to the development of middleware.
- ✓ In order to locate an object, the knowledge of object location must be distributed throughout network through replication.

3.4 ROUTING OVERLAY

A routing overlay is a distributed algorithm for a middleware layer responsible for routing requests from any client to a host that holds the object to which the request is addressed.

- Any node can access any object by routing each request through a sequence of nodes, exploiting knowledge at each of theme to locate the destination object.
- Global User IDs (GUID) also known as **opaque identifiers** are used as names, but they do not contain the location information.
- A client wishing to invoke an operation on an object submits a request including the object‟s GUID to the routing overlay, which routes the request to a node at which a replica of the object resides.
- They are actually sub-systems within the peer-to-peer middleware that are meant locating nodes and objects.
- They implement a routing mechanism in the application layer.
- They ensure that any node can access any object by routing each request thru a sequence of nodes

Features of GUID:

- ✓ They are pure names or opaque identifiers that do not reveal anything about the locations of the objects.
- ✓ They are the building blocks for routing overlays.
- ✓ They are computed from all or part of the state of the object using a function that deliver a value that is very likely to be unique. Uniqueness is then checked against all other GUIDs.
- ✓ They are not human understandable.

Process of Routing Overlay

- ∗ Client submits a request including the object GUID, routing overlay routes the request to a node at which a replica of the object resides.
- ∗ A node introduces a new object by computing its GUID and announces it to the routing overlay.
- ∗ Note that clients can remove an object and also nodes may join and leave the service

Types of Routing Overlays

1. DHT – Distributed Hash Tables. GUIDs are stored based on hash values.
2. DOLR – Distributed Object Location and Routing. DOLR is a layer over the DHT that maps GUIDs and address of nodes. GUIDs host address is notified using the Publish() operation.

3.5 PASTRY

Pastry is a generic, scalable and efficient substrate for peer-to-peer applications. Pastry nodes form a decentralized, self-organizing and fault-tolerant overlay network within the Internet.

- Pastry provides efficient request routing, deterministic object location, and load balancing in an application-independent manner.
- Furthermore, Pastry provides mechanisms that support and facilitate application-specific object replication, caching, and fault recovery.
- Each node in the Pastry network has a unique, uniform random identifier (nodeId) in a circular 128-bit identifier space.
- When presented with a message and a numeric 128-bit key, a Pastry node efficiently routes the message to the node with a nodeId that is numerically closest to the key, among all currently live Pastry nodes.
- The expected number of forwarding steps in the Pastry overlay network is O(log N), while the size of the routing table maintained in each Pastry node is only O(log N) in size (where N is the number of live Pastry nodes in the overlay network).
- At each Pastry node along the route that a message takes, the application is notified and may perform application-specific computations related to the message.
- Each Pastry node keeps track of its L immediateneighbors in the nodeId space called the **leaf set**, and notifies applications of new node arrivals, node failures and node recoveries within the leaf set.
- Pastry takes into account locality (proximity) in the underlying Internet.
- It seeks to minimize the distance messages travel, according to a scalar proximity metric like the ping delay.
- Pastry is completely decentralized, scalable, and self-organizing; it automatically adapts to the arrival, departure and failure of nodes.

Capabilities of Pastry

- **Mapping application objects to Pastry nodes**
  - Application-specific objects are assigned unique, uniform random identifiers (objIds) and mapped to the k, (k >= 1) nodes with nodeIds numerically closest to the objId.
  - The number k reflects the application's desired degree of replication for the object.
- **Inserting objects**
  - Application-specific objects can be inserted by routing a Pastry message, using the objId as the key.
  - When the message reaches a node with one of the k closest nodeIds to the objId, that node replicates the object among the other k-1 nodes with closest nodeIds (which are, by definition, in the same leaf set for k <= L/2).
- **Accessing objects**
  - Application-specific objects can be looked up, contacted, or retrieved by routing a Pastry message, using the objId as the key.
  - By definition, the message is guaranteed to reach a node that maintains a replica of the requested object unless all k nodes with nodeIds closest to the objId have failed.
- **Availability and persistence**
  - Applications interested in availability and persistence of application-specific objects maintain the following invariant as nodes join, fail and recover: object replicas are maintained on the k nodes with numerically closest nodeIds to the objId, for k> 1.
  - The fact that Pastry maintains leaf sets and notifies applications of changes in the set's membership simplifies the task of maintaining this invariant.
- **Diversity**
  - The assignment of nodeIds is uniform random, and cannot be corrupted by an attacker. Thus, with high probability, nodes with adjacent nodeIds are diverse in geographic location, ownership, jurisdiction, network attachment, etc.
  - The probability that such a set of nodes is conspiring or suffers from correlated failures is low even for modest set sizes.
  - This minimizes the probability of a simultaneous failure of all k nodes that maintain an object replica.

- **Load balancing**

  * Both nodeIds and objIds are randomly assigned and uniformly distributed in the 128-bit Pastry identifier space.

  * Without requiring any global coordination, this results in a good first-order balance of storage requirements and query load among the Pastry nodes, as well as network load in the underlying Internet.

- **Object caching**

  * Applications can cache objects on the Pastry nodes encountered along the paths taken by insert and lookup messages.

  * Subsequent lookup requests whose paths intersect are served the cached copy.

  * Pastry's network locality properties make it likely that messages routed with the same key from nearby nodes converge early, thus lookups are likely to intercept nearby cached objects.

  * This distributed caching offloads the k nodes that hold the primary replicas of an object, and it minimizes client delays and network traffic by dynamically caching copies near interested clients.

- **Efficient, scalable information dissemination**

* Applications can perform efficient multicast using reverse path forwarding along the tree formed by the routes from clients to the node with nodeId numerically closest to a given objId.

* Pastry's network locality properties ensure that the resulting multicast trees are efficient; i.e., they result in efficient data delivery and resource usage in the underlying Internet.

3.5.1 Pastry's Routing Algorithm

The routing algorithm involves the use of a routing table at each node to route messages efficiently. This is divided into two stages:

- **First Stage:**

  * In this stage a routing scheme is used, that routes messages correctly but inefficiently without a routing table.

  * Each active node stores a leaf set – a vector L (of size 2l) containing the GUIDs and IP addresses of the nodes whose GUIDs are numerically closest on either side of its own (l above and l below).

- Leaf sets are maintained by Pastry as nodes join and leave. Even after a node failure, they will be corrected within a short time.
- In the Pastry system that the leaf sets reflect a recent state of the system and that they converge on the current state in the face of failures up to some maximum rate of failure.
- The GUID space is treated in circular fashion.
- GUID 0"s lower neighbour is 2128–1.
- Every leaf set includes the GUIDs and IP addresses of the current node"s

  immediateneighbours.
- A Pastry system with correct leaf sets of size at least 2 can route messages to any GUID trivially as follows: any node A that receives a messageM with destination address D routes the message by comparing D with its own GUIDA and with each of the GUIDs in its leaf set and forwarding M to the node amongstthem that is numerically closest to D.

➢ **Second Stage:**

- This has a full routing algorithm, which routes a request to any node in O(log N) messages.
- Each Pastry node maintains a tree-structured routing table with GUIDs and IP addresses for a set of nodes spread throughout the entire range of 2^{128} possible GUID values, with increased density of coverage for GUIDs

  numerically close to its own.
- The structure of routing table is: GUIDs are viewed as hexadecimal values and the table classifies GUIDs based on their hexadecimal prefixes.
- The table has as many rows asthere are hexadecimal di gits in a GUID, so for the prototype Pastry system that weare describing, there are 128/4 = 32 rows.
- Any row n contains 15 entries – one foreach possible value of the nth

  hexadecimal digit, excluding the value in the local node "s GUID.
- Each entry in the table points to one of the potentially many nodes whose GUIDs have the relevant prefix.
- The routing process at any node A uses the information in its routing table R and leaf set L to handle each request from an application and each incoming messagefrom another node according to the algorithm.

Pastry's Routing Algorithm

To handle a message M addressed to a node D (where R[p,i] is the element at column i, row p of the routing table):

1. If (L-l < D <Ll) { // the destination is within the leaf set or is the current node.
2. Forward M to the element Li of the leaf set with GUID closest to D or the current node A.
3. } else { // use the routing table to despatch M to a node with a closer GUID
4. Find p, the length of the longest common prefix ofD and A,.and i, the (p+1)th hexadecimal digit of D.
5. If (R[p,i] ● null) forward M to R[p,i] // route M to a node with a longer common prefix.
6. else { // there is no entry in the routing table.
7. Forward M to any node in L or R with a common prefix of length p but a GUID that is numerically closer.

}

}

Locality

- The Pastry routing structure is redundant.
- Each row in a routin g table contains 16 entries.
- The entries in the ithrow give the addresses of 16 nodes with GUIDs with i–1 initial hexadecimal digits thatmatch the current node "s GUID and an ith di git that takes each of the possible hexadecimal mal values.
- A well-populated Pastry overlay will contain many more nodesthan can be contained in an individual routing table; whenever a new routing table isbeing constructed a choice is made for each position between several candidates based on a proximity neighbour selection algorithm.
- A locality metric is used to compare candidates and the closest available node is chosen.
- Since the information available is not comprehensive, this mechanism cannot produce globally optimal routings, but simulations have shown that it results in routes that are on average only about 30–50% longer than the optimum.

Fault tolerance

- The Pastry routing algorithm assumes that allentries in routing tables and leaf sets refer to live, correctly functioning nodes.
- All nodessend „heartbeat" messages i.e., messages sent at fixed time intervals to indicate that thesender is alive to neighbouring nodes in their leaf sets, but information about failednodes detected in this manner may not be disseminated sufficiently rapidly to eliminaterouting errors.
- Nor does it account for malicious nodes that may attempt to interfere with correct routing.
- To overcome these problems, clients that depend upon reliable messagedelivery are expected to employ an at-least-once delivery mechanism and repeat their requests several times in the absence of a response.
- This willallow Pastry a longer time window to detect and repair node failures.
- To deal with any remaining failures or malicious nodes, a small degree of randomness is introduced into the route selection algorithm.

Dependability

- Dependability measures include the use of acknowledgements at each hop in the routing algorithm.
- If the sending host does not receive an acknowledgement after aspecified timeout, it selects an alternative route and retransmits the message.
- The nodethat failed to send an acknowledgement is then noted as a suspected failure.
- To detect failed nodes each Pastry node periodically sends aheartbeat message to its immediate neighbour to the left (i.e., with a lower GUID) in the leaf set.
- Each node also records the time of the last heartbeat message received from its immediate neighbour on the right (with a higher GUID).
- If the interval since the lastheartbeat exceeds a timeout threshold, the detecting node starts a repair procedure thatinvolves contacting the remaining nodes in the leaf set with a notification about thefailed node and a request for suggested replacements.
- Even in the case of multiplesimultaneous failures, this procedure terminates with all nodes on the left side of thefailed node having leaf sets that contain the l live nodes with the closest GUIDs.

- Suspectedfailed nodes in routing tables are probed in a similar manner to that used for the leaf setand if they fail to respond, their routing table entries are replaced with a suitablealternative obtained from a nearby node.
- A simple gossip protocol is used to periodically exchange routing table information betweennodes in order to repair failed entries and prevent slow deterioration of the localityproperties.
- The gossip protocol is run about every 20 minutes.

3.6 TAPESTRY

- Tapestry is a decentralized distributed system.
- It is an overlay network that implements simple key-based routing.
- Each node serves as both an object store and a router that applications can contact to obtain objects.
- In a Tapestry network, objects are "published" at nodes, and once an object has been successfully published, it is possible for any other node in the network to find the location at which that objects is published.
- Identifiers are either NodeIds, which refer to computers that perform routing operations, or GUIDs, which refer to the objects.
- For any resource with GUID G there is a unique root node with GUID RG that is numerically closest to G.
- Hosts H holding replicas of G periodically invoke publish(G) to ensure that newly arrived hosts become aware of the existence of G.

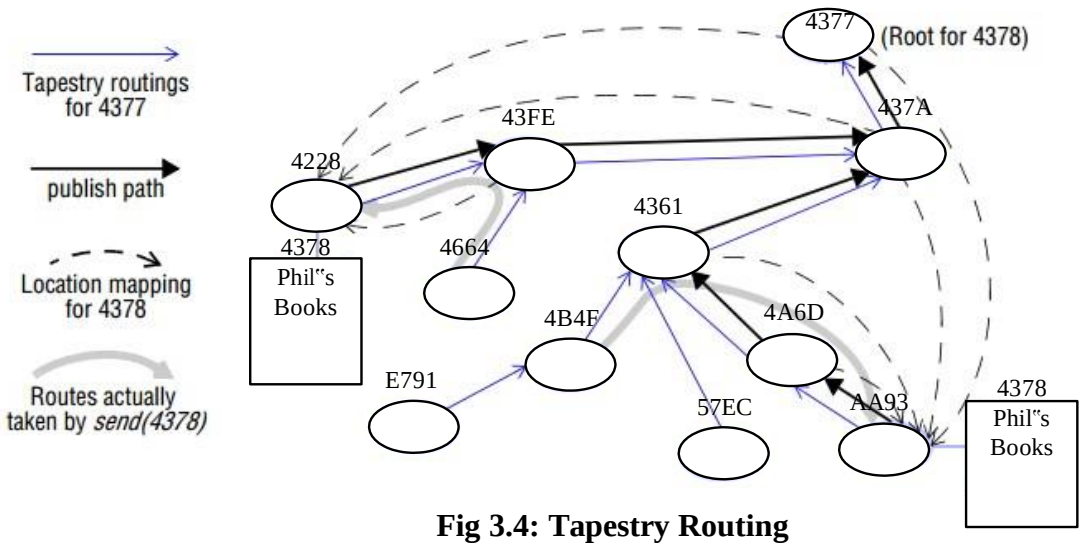


Fig 3.4: Tapestry Routing

Unstructured peer-to-peer Networks

- ✓ In structured approaches, there is an overall global policy governing the topology of the network, the placement of objects in the network and the routing or search functions used to locate objects in the network.
- ✓ There is a specific data structure underpinning the associated overlay and a set of algorithms operating over that data structure.
- ✓ These algorithms are efficient and time bounded.
- ✓ In unstructured approaches, there is no overall control over the topology or the placement of objects within the network.
- ✓ The overlay iscreated in an ad hoc manner, with each node that joins the network following simple, local rules to establish connectivity.
- ✓ In particular, a joining node will establishcontact with a set of neighbours knowing that the neighbours will also be connected tofurtherneighbours and so on, forming a network that is fundamentally decentralized andself-organizing g and hence resilient to node failure.
- ✓ To locate a given object, it is thennecessary to carry out a search of the resultant network topology.
- ✓ This approachcannot offer guarantees of being able to find the object and performance will beunpredictable.
- ✓ There is a real risk of generating excessive message traffic tolocate objects.

Differences between structured and unstructured networks

Structured Network	Unstructured Network
This guarantees to locate objects and can offer time and complexity bounds on this operation. So it has relatively low message overhead.	This is self-organizing and resilient to node failure.
This needs complexoverlay structures, which can bedifficult and costly to achieve, especially in highly dynamic environments.	Probabilistic and hence cannot offer absolute guarantees on locating objects; prone to excessive messaging overhead which can affect scalability

Gnutella

- ✓ The Gnutella network is a fully decentralized, peer-to-peer application layer network that facilitates file sharing; and is built around an open protocol developed to enable host discovery, distributed search, and file transfer.

- ✓ It consists of the collection of Internet connected hosts on which Gnutella protocol enabled applications are running.
- ✓ The Gnutella protocol makes possible all host-to-host communication through the use of messages.

GUID	Type	TTL	Hops	Payload Size
16 bytes	1 byte	1 byte	1 byte	4 bytes
23 bytes				

Fig : 3.5: Message Format

- ✓ A message consists of the following five fields:
  - The GUID field provides a unique identifier for a message on the network; the Type field indicates which type of message is being communicated.
  - TTL field enumerates the maximum number of hops that this message is allowed to traverse.
  - Hops field provides a count of the hops already traversed.
  - Payload Size field provides a byte count of all data expected to follow the message.
  - A host communicates with its peers by receiving, processing, and forwarding messages, but to do so it must follow a set of rules which help to ensure reasonable message lifetimes.
- ✓ The rules are as follows:
  1. Prior to forwarding a message, a host will decrement its TTL field and increment its Hops field. If the TTL field is found to be zero following this action, the message is not forwarded.
  2. If a host encounters a message with the same GUID and Type fields as a message already forwarded, the new message is treated as a duplicate and is not forwarded a second time.

Types of Message

There are only five types of messages

- ✦ Ping
- ✦ Pong
- ✦ Query

- ✦ Query-Hit
- ✦ Push

Ping and Pong messages facilitate host discovery, Query and Query-Hit messages make possible searching the network, and Push messages ease file transfer from firewalled hosts.Because there is no central index providing a list of hosts connected to the network, a disconnected host must have offline knowledge of a connected host in order to connect to the network.

- ✓ Once connected to the network, a host is always involved in discovering hosts, establishing connections, and propagating messages that it receives from its peers, but it may also initiate any of the following six voluntary activities: searching for content, responding to searches, retrieving content, and distributing content. A host will typically engage in these activities simultaneously.
- ✓ A host will search for other hosts and establish connections so as to satisfy a maximum requirement for active connections as specified by the user, or to replace active connections dropped by it or its peer.
- ✓ Consequently, a host tends to always maintain the maximum requirement for active connections as specified by the user, or the one connection that it needs to remain connected to the network.
- ✓ To engage in host discovery, a host must issue a Ping message to the host of which it has offline knowledge.
- ✓ That host will then forward the ping message across its open connections, and optionally respond to it with a Pong message.
- ✓ Each host that subsequently receives the Ping message will act in a similar manner until the TTL of the message has been exhausted.
- ✓ A Pong message may only be routed along the reverse of the path that carried the Ping message to which it is responding.
- ✓ After having discovered a number of other hosts, the host that issued the initial ping message may begin to open further connections to the network.
- ✓ Doing so allows the host to issue ping messages to a wider range of hosts and therefore discover a wider range of other hosts, as well as to begin querying the network.
- ✓ In searching for content, a host propagates search queries across its active connections.
- ✓ Those queries are then processed and forwarded by its peers.

- ✓ When processing a query, a host will typically apply the query to its local database of content, and respond with a set of URLs pointing to the matching files.
- ✓ The propagation of Query and Query-Hit messages is identical to that of Ping and Pong messages.
- ✓ A host issues a Query message to the hosts to which it is connected.
- ✓ The hosts receiving that Query message will then forward it across their open connections, and optionally respond with a Query-Hit message.
- ✓ Each host that subsequently receives the Query message will act in a similar manner until the TTL of the message has been exhausted.
- ✓ Query-Hit messages may only be routed along the reverse of the path that carried the Query message to which it is a response.
- ✓ The sharing of content on the Gnutella network is accomplished through the use of the HTTP protocol.
- ✓ Due to the decentralized nature of the architecture, each host participating in the Gnutella network plays a key role in the organization and operation of the network.
- ✓ Both the sharing of host information, as well as the propagation of search requests and responses are the responsibility of all hosts on the network rather than a central index.

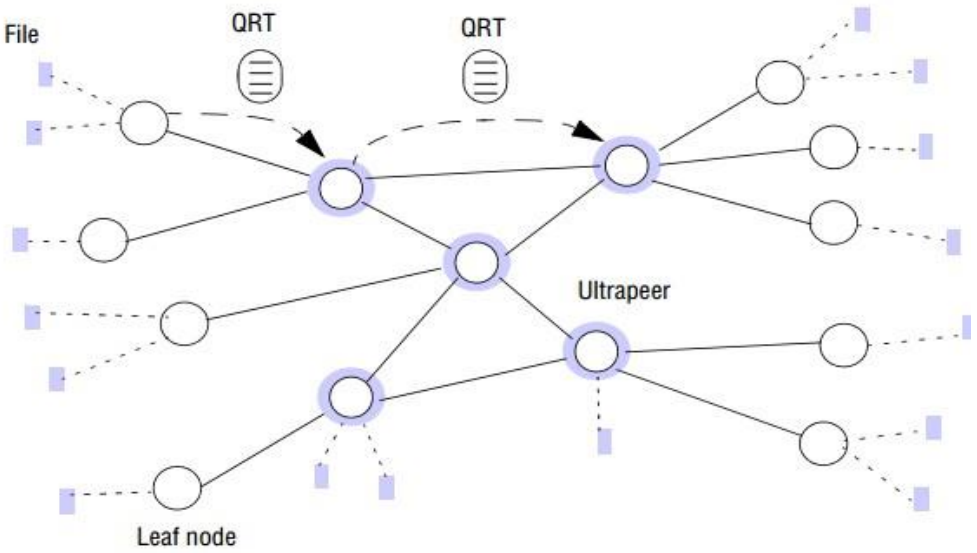


Fig 3.6: Elements in Gnutella Network

- ✓ The Gnutella network architecture avoids creating the single-point-of-failure that, theoretically, threatens other networks.

- ✓ The Gnutella network exhibits a great deal of fault-tolerance, continuing to function even as subsets of hosts become unavailable.
- ✓ Each host in the Gnutella network is capable of acting as both a server and client, allowing a user to distribute and retrieve content simultaneously.
- ✓ This spreads the provision of content across many hosts, and helps to eliminate the bottlenecks that typically occur when placing enormous loads on a single host.
- ✓ In this network, the amount and variety of content available to a user scales with the number of users participating in the network.

3.7 DISTRIBUTED FILE SYSTEMS

- ✓ The purpose of a distributed file system (DFS) is to allow users of physically distributed computers to share data and storage resources by using a common file system.
- ✓ A typical configuration for a DFS is a collection of workstations and mainframes connected by a local area network (LAN).
- ✓ A DFS is implemented as part of the operating system of each of the connected computers.

Distributed file systems support the sharing of information in the form of files and hardware resources.

- ✓ The following are the modules of a file systems:
  - ∗ Directory module: Relates file names to the disk.
  - ∗ File module: Relates file ID to the files
  - ∗ Access Control module: Manages access rights
  - ∗ File access module: Controls read or write operation on the files
  - ∗ Block module: Allocates disk blocks and helps in accessing them
  - ∗ Device module: Manages disk I/O and buffering.

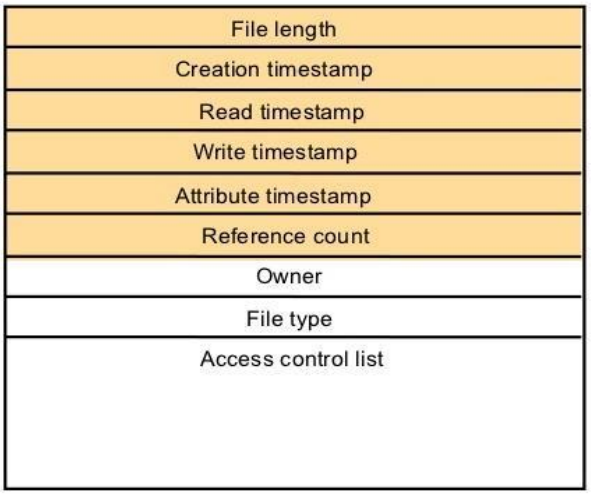


Fig 3.6: File Attribute Record Structure

File Attribute Record Structure

* File length specifies the size of the file.
* Creation Timestamp is the time at which the file was created.
* Read Timestamp is the time at which the last read operation was made.
* Write Timestamp is the time at which the last write operation was made.
* Reference Count is the number or links to the file.
* Owner is the creator of the file who has full access rights.
* File type describes the content of the file.
* Access Control List contains the list of users and the access rights given to them.

3.7.1 Unix File Systems

The time sharing Unix file system is a model for the distributed file systems. The following some of the unix commands to access files:

- filedes= open(name, mode): Opens an already existing file.
- filedes=creat(name, mode): Creates a new file
- status= close(filedes): Closes a file.
- count= read(filedes, buffer, n): Transfers n bytes from the filedes to buffer and returns the number of bytes transferred.
- count= write(filedes, buffer, n): Transfers n bytes from the filedes to buffer and returns the number of bytes transferred. This advances the write file pointer.

- pos= lseek(filedes, offset, whence): Moves the read-write pointer to offset depending on whence.
- status=unlink(name): renames the specified file.
- status=unlink(name1, name2): Adds a new name (name2) for a file (name1).
- status= stat(name, buffer): Gets the file attributes for file into the buffer.

3.7.2 Characteristics of a File system

- ✓ File systems are responsible for the organization, storage, retrieval, naming, sharing and protection of files.
- ✓ Data and attributes are the primary characteristics of a file system.
- ✓ The data consist of a sequence of data items .
- ✓ The attributes are held as a single record containing information such as the length of the file, timestamps, file type, owner‟s identity and access control lists.
- ✓ A directory is a special file that provides a mapping from text names to internal file identifiers.
- ✓ The term metadata is used to refer to all of the extra information stored by a file system that is needed for the management of files.
- ✓ It includes file attributes , directories and all the other persistent information used by the file system.
- ✓ The file system also maintains the access control lists.

3.7.3 Requirements of distributed file systems

Transparency is operating in such a way as to not be perceived by users. The distributed file system demands the following transparency requirements:

- **Login Transparency:** User can log in at any host with uniform login procedure and perceive a uniform view of the file system.
- **Access Transparency:** Client process on a host has uniform mechanism to access all files in system regardless of files are on local/remote host.
- **Location Transparency:** The names of the files do not reveal their physical location.
- **Concurrency Transparency:** An update to a file should not have effect on the correct execution of other process that is concurrently sharing a file.
- **Replication Transparency:** Files may be replicated to provide redundancy for availability and also to permit concurrent access for efficiency.

Apart from the above transparency properties, the file systems also need the following:

- **Fault Tolerance:** It is a design that enables a system to continue operation, possibly at a reduced level, rather than failing completely, when some part of the system fails.
- **Network Transparency:** Same access operation as if they are local files.
- **Location Independence:** The file name should not be changed when the physical location of the file changes.
- **User Mobility:** User should be able to access the file from any where.
- **File Mobility:** Moves files from one place to the other in a running system.

Examples:

- ✓ **The GOOGLE File System:** A scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients.
- ✓ **The CODA distributed file system:** Developed at CMU, incorporates many distinguished features which are not present in any other distributed file system.

3.8 FILE SERVICE ARCHITECTURE

File service architecture offers a clear separation of the main concerns in providing access to files is obtained by structuring the file service as three components:

- A flat file service
- A directory service
- A client module.

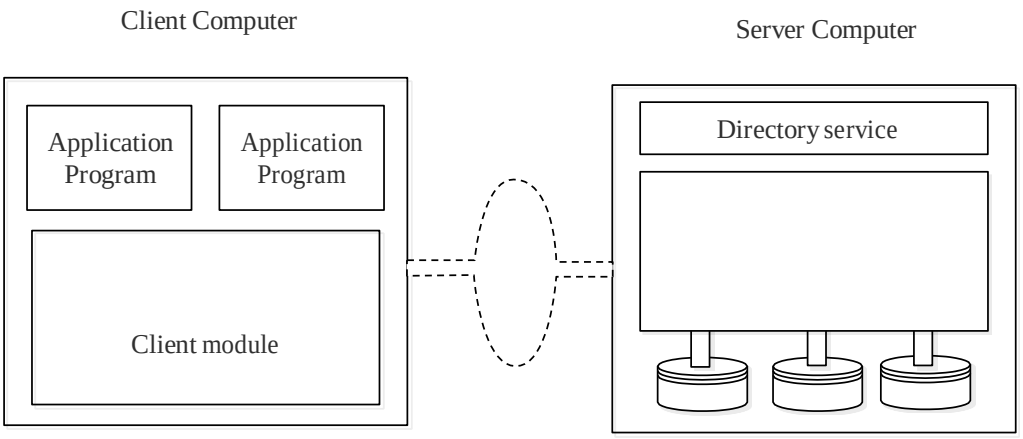


Fig 3.9: File Service Architecture

Flat file service:

- ✓ This is concerned with the implementation of operations on the contents of file.
- ✓ Unique File Identifiers (UFIDs) are used to refer to files in all requests for flat file service operations.
- ✓ UFIDs are long sequences of bits chosen so that each file has a unique among all of the files in a distributed system.

Directory service:

- ✓ This provides mapping between text names for the files and their UFIDs.
- ✓ Clients may obtain the UFID of a file by quoting its text name to directory service.
- ✓ Directory service supports functions needed generate directories, to add new files to directories.

Client module:

- ✓ It runs on each computer and provides integrated service (flat file and directory) as a single API to application programs.
- ✓ It holds information about the network locations of flat-file and directory server processes; and achieve better performance through implementation of a cache of recently used file blocks at the client.

The following are the operations in file service:

- Read(FileId, i, n)→Data :Reads a sequence of up to n items from a file starting at item i and returns it in Data.
- Write(FileId, i, Data):Write a sequence of Data to a file, starting at item i, extending the file if necessary.
- Create() →FileId: Creates a new file of length0 and delivers a UFID for it.
- Delete(FileId): Removes the file from the file store.
- GetAttributes(FileId) →Attr:Returns the file attributes for the file.
- SetAttributes(FileId, Attr):Sets the file attributes .

➢ **Access control in files:**

In distributed implementations, access rights checks have to be performed at the server because the server RPC interface is an otherwise unprotected point of access to files.

- **Hierarchic file system**

  A hierarchic file system such as the one that UNIX provides consists of a number of directories arranged in a tree structure.

- **File Group**

  - ✓ A file group is a collection of files that can be located on any server or moved between servers while maintaining the same names.
  - ✓ A similar construct is used in a UNIX file system.
  - ✓ It helps with distributing the load of file serving between several servers.
  - ✓ File groups have identifiers which are unique throughout the system.
  - ✓ To construct a globally unique ID we use some unique attribute of the machine on which it is created, e.g. IP number, even though the file group may move subsequently.

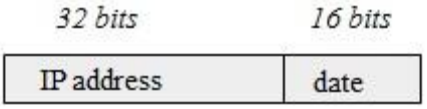


Fig 3.9: File group ID

3.9 ANDREW FILE SYSTEM (AFS)

- This was developed by Carnegie Mellon University as part of Andrew distributed computing environments (in 1986).
- The public domain implementation is available on Linux (LinuxAFS)
- It was adopted as a basis for the DCE/DFS file system in the Open Software Foundation (OSF, www.opengroup.org) DEC (Distributed Computing Environment).
- Like NFS, AFS provides transparent access to remote shared files for UNIX programs running on workstations.
- AFS is implemented as two software components that exist at UNIX processes called Vice and Venus.
- The files available to user processes running on workstations are either local or shared.
- Local files are handled as normal UNIX files.
- They are stored on the workstation‟s disk and are available only to local user processes.

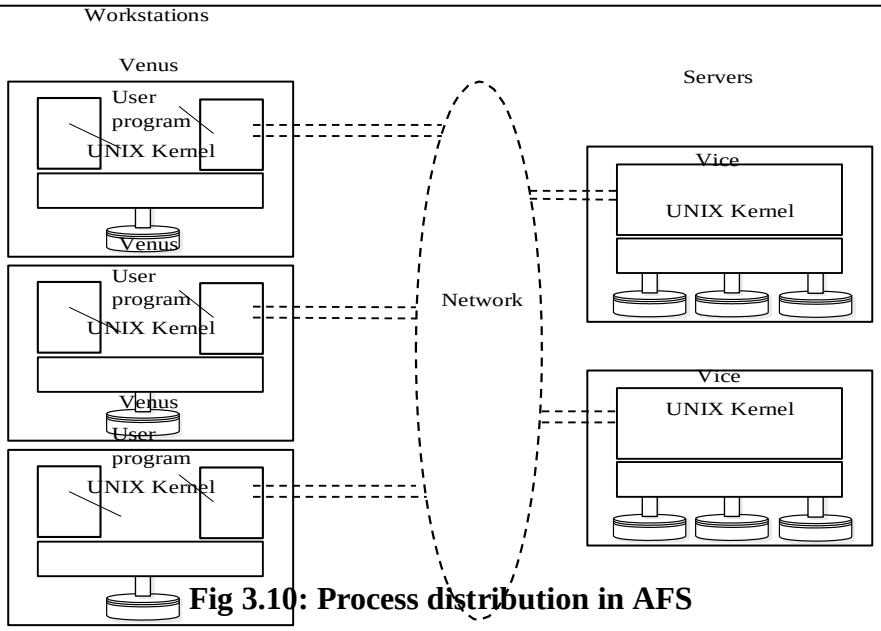


Fig 3.10: Process distribution in AFS

- Shared files are stored on servers, and copies of them are cached on the local disks of workstations.
- The UNIX kernel in each workstation and server is a modified version of BSD UNIX.
- The modifications are designed to intercept open, close and some other file system calls when they refer to files in the shared name space and pass them to the Venus process in the client computer.

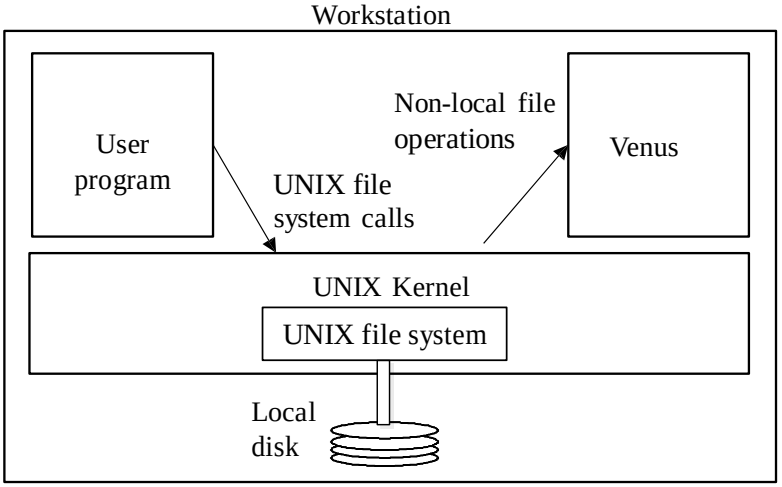


Fig 3.11: System call interception

Components of Vice service interface

- Fetch(fid)→attr, data: Returns the attributes (status) and, optionally, the contents of
  the file identified by fid and records a callback promise on it.
- Store(fid, attr, data): Updates the attributes and (optionally) the contents of a specified file.
- Create() →fid: Creates a new file and records a callback promise on it.
- Remove(fid) Deletes the specified file.
- SetLock(fid, mode): Sets a lock on the specified file or directory. The mode of the lock may be shared or exclusive. Locks that are not removed expire after 30 minutes.
- ReleaseLock(fid):Unlocks the specified file or directory.
- RemoveCallback(fid): Informs the server that a Venus process has flushed a file from
  its cache.
- BreakCallback(fid:) Call made by a Vice server to a Venus process; cancels the call back promise on the relevant file.

3.10 FILE ACCESSING MODELS

File access models are methods used for accessing remote files and the unit of data access. A distributed file system may use one of the following models to service a client‟s file access request when the accessed file is remote:

1. **Remote service model**
   - Processing of a client‟s request is performed at the server‟s node.
   - Thus, the client‟s request for file access is delivered across the network as a message to the server, the server machine performs the access request, and the result is sent to the client.
   - This need to minimize the number of messages sent and the overhead per message.
2. **Data-caching model**
   - This model attempts to reduce the network traffic of the previous model by caching the data obtained from the server node.
   - This takes advantage of the locality feature of the found in file accesses.

- A replacement policy such as LRU is used to keep the cache size bounded.
- This model reduces network traffic it has to deal with the cache coherency problem during writes, because the local cached copy of the data needs to be updated, the original file at the server node needs to be updated and copies in any other caches need to be updated.
- The data-caching model offers the possibility of increased performance and greater system scalability because it reduces network traffic, contention for the network, and contention for the file servers. Hence almost all distributed file systems implement some form of caching.
- In file systems that use the data-caching model, an important design issue is to decide the **unit of data transfer.**
- This refers to the fraction of a file that is transferred to and form clients as a result of single read or write operation.

3. File-level transfer model

- In this model when file data is to be transferred, the entire file is moved.
- File needs to be transferred only once in response to client request and hence is more efficient than transferring page by page which requires more network protocol overhead.
- This reduces server load and network traffic since it accesses the server only once. This has better scalability.
- Once the entire file is cached at the client site, it is immune to server and network failures.
- This model requires sufficient storage space on the client machine.
- This approach fails for very large files, especially when the client runs on a diskless workstation.
- If only a small fraction of a file is needed, moving the entire file is wasteful.

4. Block-level transfer model

- File transfer takes place in file blocks.
- A file block is a contiguous portion of a file and is of fixed length (can also be a equal to a virtual memory page size).
- This does not require client nodes to have large storage space.
- It eliminates the need to copy an entire file when only a small portion of the data is needed.

- ❖ When an entire file is to be accessed, multiple server requests are needed, resulting in more network traffic and more network protocol overhead. NFS uses block-level transfer model.

5. **Byte-level transfer model**

- ❖ Unit of transfer is a byte.
- ❖ Model provides maximum flexibility because it allows storage and retrieval of an arbitrary amount of a file, specified by an offset within a file and length.
- ❖ The drawback is that cache management is harder due to the variable-length data for different access requests.

6. **Record-level transfer model**

- ❖ This model is used with structured files and the unit of transfer is the record.

3.11 FILE-SHARING SEMANTICS

- ✓ Multiple users may access a shared file simultaneously.
- ✓ An important design issue for any file system is to define when modifications of file data made by a user are observable by other users.
- ✓ The UNIX semantics is implemented in file systems for single CPU systems because it is the most desirable semantics and because it is easy to serialize all read/write requests.
- ✓ Implementing UNIX semantics in a distributed file system is not easy.
- ✓ One may think that this can be achieved in a distributed system by disallowing files to be cached at client nodes and allowing a shared file to be managed by only one file server that processes all read and write requests for the file strictly in the order in which it receives them.
- ✓ However, even with this approach, there is a possibility that, due to network delays, client requests from different nodes may arrive and get processed at the server node in an order different from the actual order in which the requests were made.
- ✓ Also, having all file access requests processed by a single server and disallowing caching on client nodes is not desirable in practice due to poor performance, poor scalability, and poor reliability of the distributed file system.
- ✓ Hence distributed file systems implement a more relaxed semantics of file sharing.
- ✓ Applications that need to guarantee UNIX semantics should provide mechanisms (e.g. mutex lock etc) themselves and not rely on the underlying semantics of sharing provided by the file system.

3.12 NAMING IN DISTRIBUTED SYSTEMS

- A Name is a string of bits used to refer to an entity.
- We operate on an entity through its Access Point. The Address is the name of the access point.
- The naming faciliy of a distributed operating system enables users and programs to assign character-string names to objects and subsequently use these names to refer to those objects.
- The locating faciliy, which is an integral part of the naming facility, maps an object's name to the object's location in a distributed system.
- The naming and locating facilities jointly form a naming system that provides the users with an abstraction of an object that hides the details of how and where an object is actually located in the network.
- It provides a further level of abstraction when dealing with object replicas.
- Given an object name, it returns a set of the locations of the object's replicas.
- The naming system plays a very important role in achieving the goal of
    - location transparency,
    - facilitating transparent migration
    - replication of objects, v object sharing.

Example:

- ✓ Telephone as Access Point to a person.
- ✓ The Telephone Number then becomes the address of the person.
- ✓ Person can have several telephone numbers.
- ✓ Entity can have several addresses

3.12.1 Identifiers:

- Identifiers are special names that uniquely identify an entity.
- An identifier refers to at most one entity.
- Each entity is referred to at most one identifier.
- An identifier always refers to the same entity (never reused).

3.12.2 Namespaces:

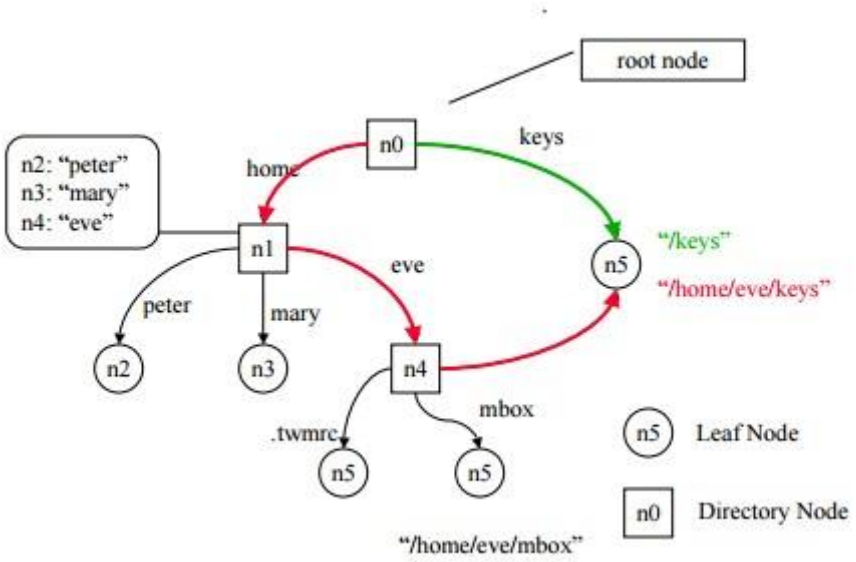


Fig 3.12: Namespaces

- Names are organized into Name Space.
- Entities in a structured name space are named by a path name.
- Leaf nodes represent named entities (e.g., files) and have only incoming edges .
- Directory nodes have named outgoing edges and define the path used to find a leaf node

Attribute based naming

- This allows a user to search for an entity whose name is not known.
- Entities are associated with various attributes, which can have specific values.
- By specifying a collection of <attribute, value> pairs, a user can identify one (or more) entities
- Attribute based naming systems are also referred to as **directory services**, as opposed to naming systems.

3.12.3 Naming resolution

It is the process of looking up a name. The path name of the file is given as

N:<label-1, label-2,…, label-n>

Name resolution is the process of mapping an object'sname to the object's properties, such as its location.

- ✓ Since an object's properties are stored and maintained by the authoritative name servers of that object, name resolution is basically the process of mapping an object's name to the authoritative name servers of that object.
- ✓ Once an authoritative name server of the object has been located, operations can be invoked to read or update the object's properties.
- ✓ Each name agent in a distributed system knows about at least one name server apriori.
- ✓ To get a name resolved, a client first contacts its name agent, which in turn contacts a known name server, which may in turn contact other name servers.

Absolute and Relative Names

- A current working context is also known by the shorter names current context or working context.
- According to this concept, a user is always associated with a context that is his or her current context.
- A user can change his or her current context whenever he or she desires.

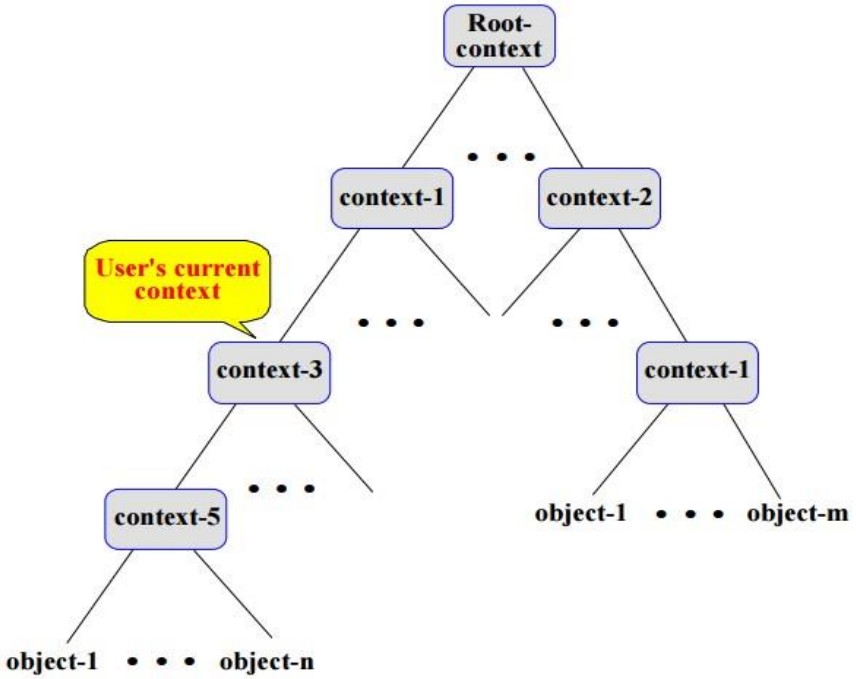


Fig 3.12: Tree structured name space

- An **absolute name** begins at the root context of the name space tree and follows a path down to the specified object, giving the context names on the path.

- On the other hand, a relative name defines a path from the current context to the specified object. It is called a relative name because it is "relative to" (start from) the user's current context.

In this method, a user may specify an object in any of the following ways:

1. Using the full (absolute) name
2. Using a relative name
3. Changing the current context first and then using a relative name

Implementing Name Spaces

Three layers used to implement such distributed name spaces

- Global layer: root node and its children
- Administrational layer: directory nodes within a single organization
- Managerial layer

Item	Global	Administrational	Managerial
Geographical scale of network	Worldwide	Organization	Department
Total number of nodes	Few	Many	Vast numbers
Responsiveness of lookups	Seconds	Milliseconds	Immediate
Updata propagation	Lazy	Immediate	Immediate
Number of replicas	Many	None or few	None
Is client-side caching applied?	Yes	Yes	Sometimes

Name Space Distribution

Iterative name resolution

- Recursive name resolution puts a higher performance demand on each name server.
- Too high for global layer name servers
- In this method, the name agent forwards the name resolution request to the name server that stores the first context needed to start the resolution of the given name.
- After this, the name servers that store the contexts of the given pathname are recursively activated one after another until the authority attribute of the named

object is extracted from the context corresponding to the last component name of the pathname.

- The last name server returns the authority attribute to its previous name server, which then returns it to its own previous name server, and so on.
- Finally, the fast name server that received the request from the name agent returns the authority attribute to the name agent

Advantages of recursive name resolution:

- Caching is more effective
- Communication costs may be reduced

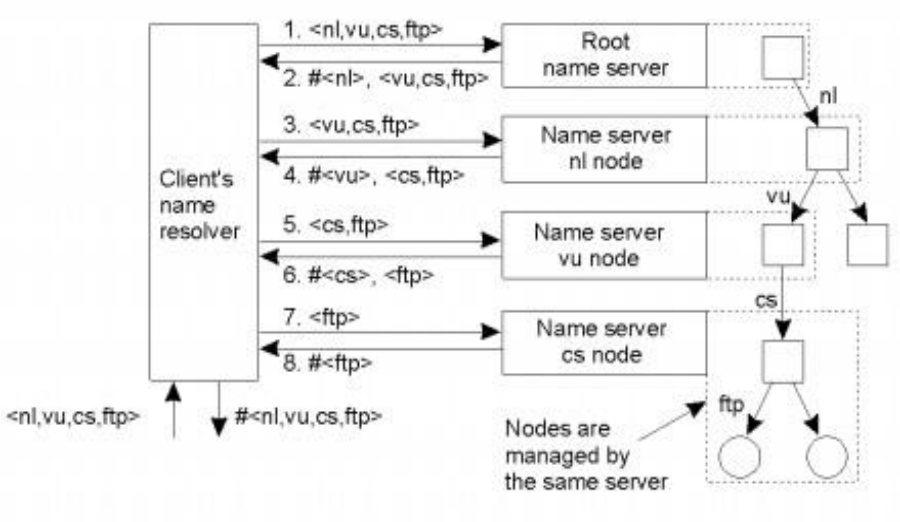


Fig 3.12: Principle of iterative name resolution

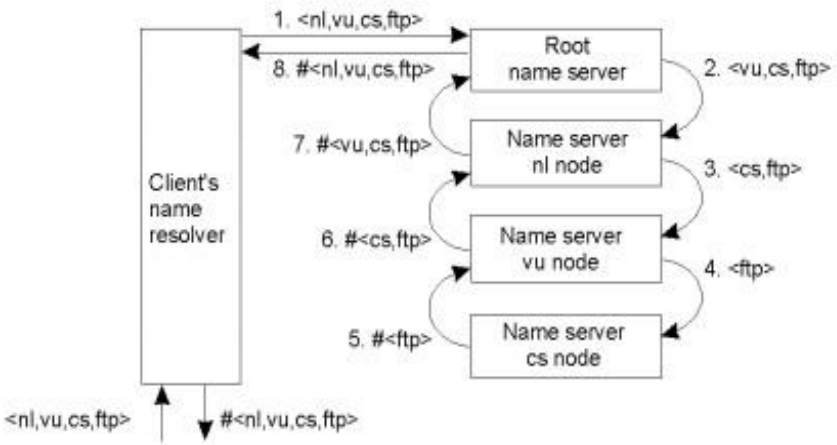


Fig 3.13: Principle of iterative name resolution

Recursive Name resolution:

- In this method, name servers do not call each other directly.
- Rather, the name agent retains control over the resolution process and one by one calls each of the servers involved in the resolution process.
- To continue the name resolution, the name agent sends a name resolution request along with the unresolved portion of the name to the next name server.
- The process continues until the name agent receives the authority attribute of the named object

3.12.4: Name Caches

Caching is an important technique for building responsive, scalable distributed systems. A cache can be maintained either by the client or the server or by both.

Types of Name caches

- **Client name caches**

  Caching at the client is an effective way of pushing the processing workload from the server out to client devices,if a client has the capacity.

- **Server based cache**

  If result data is likely to be reused by multiple clients or if the client devices do not have the capacity then caching at the server is more effective.

- **Multi-level caching**

  Caches can be maintained at multiple levels. For example, caches can be maintained at all clients and all servers. Use of a cache at one level reduces the number of requests handled at the levels below.

3.12.5: Lightweight Directory Access Protocol (LDAP)

LDAP is a standard technology for building computer network directories. A network directory is a specialized database that stores information about devices, applications, people and other aspects of a computer network.

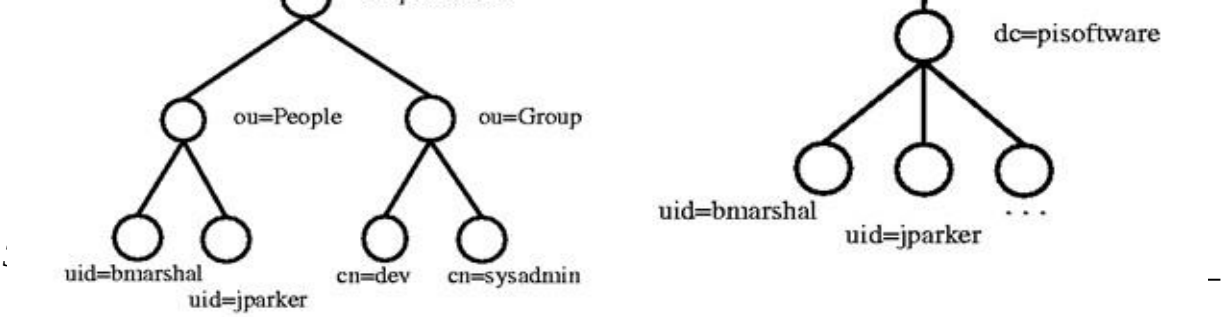


Fig 3.14: LDAP Tree structure

Working of LDAP:

- Client starts an LDAP session by connecting to an LDAP server.
- The default port on TCP is 389.
- Client sends operation requests to the server and the server sends responses in turn.
- With some exceptions the client need not wait for a response before sending the next request. Server may send the responses in any order.
- The client may request the following operations:
  - Start TLS : Optionally protect the connection with Transport Layer Security (TLS), to have a more secure connection
  - Bind - authenticate and specify LDAP protocol version
  - Search - search for and/or retrieve directory entries
  - Compare - test if a named entry contains a given attribute value
  - Add a new entry
  - Delete an entry
  - Modify an entry
  - Modify Distinguished Name (DN) - move or rename an entry
  - Abandon - abort a previous request

- Extended Operation - generic operation used to define other operations
- Unbind - close the connection (not the inverse of Bind)

– In addition the server may send "Unsolicited Notifications" that are not responses to any request, e.g. before it times out a connection.

Directory Structure

- ✓ Directory is a tree of directory entries.
- ✓ Each entry consists of a set of attributes.
- ✓ An attribute has: a name , an attribute type or attribute description and one or more values
- ✓ Attributes are defined in a schema.
- ✓ Each entry has a unique identifier called Distinguished Name (DN).
- ✓ The DN consists of its Relative Distinguished Name (RDN) constructed from some attribute(s) in the entry.
- ✓ It is followed by the parent entry's DN.
- ✓ Think of the DN as a full filename and the RDN as a relative filename in a folder.
- ✓ DN may change over the lifetime of the entry.
- ✓ To reliably and unambiguously identify entries, a UUID might be provided in the set of the entry's operational attributes.

REVIEW QUESTIONS

PART - A

1. Define peer-to-peer communications.

Peer-to-peer (P2P) is a decentralized communications model in which each party has the same capabilities and either party can initiate a communication session.

2. Give the features of Peer to Peer Systems

- Large scale sharing of data and resources
- No need for centralized management
- Their design of P2P system must be in such a manner that each user contributes resources to the entire system.
- All the nodes in apeer-to-peer system have the same functional capabilities and responsibilities.
- The operation of P2P system does not depend on the centralized management.
- The choice of an algorithm for theplacement of data across many hosts and the access of the data must balance the workload and ensure the availability without much overhead.

3. List the advantages of P2P systems over client/ server architecture

1. It is easy to install and so is the configuration of computers on the network.
2. All the resources and contents are shared by all the peers, unlike server-client architecture where Server shares all the contents and resources.
3. P2P is more reliable as central dependency is eliminated. Failure of one peer doesn‟t affect the functioning of other peers. In case of Client –Server network, if server goes down whole network gets affected.
4. There is no need for full-time System Administrator. Every user is the administrator of his machine. User can control their shared resources.
5. The over-all cost of building and maintaining this type of network is comparatively very less.

4. List the disadvantages of P2P systems over client/ server architecture

1. In this network, the whole system is decentralized thus it is difficult to administer. That is one person cannot determine the whole accessibility setting of whole network.
2. Security in this system is very less viruses, spywares,trojans, etc malwares can easily transmitted this architecture.

3. Data recovery or backup is very difficult. Each computer should have its own back-up system
4. Lot of movies, music and other copyrighted files are transferred using this type of file transfer. P2P is the technology used in torrents.

5. Give the characteristics of peer-to-peer middleware.

❖ The P2P middleware must possess the following characteristics:

- Global Scalability
- Load Balancing
- Local Optimization
- Adjusting to dynamic host availability
- Security of data
- Anonymity, deniability, and resistance to censorship

6. Define routing overlay.

A routing overlay is a distributed algorithm for a middleware layer responsible for routing requests from any client to a host that holds the object to which the request is addressed.

7. Give the differences between Overlay networks and IP routing

IP	Overlay Network
The scalability of IPV4 is limited to 2^{32} nodes and IPv6 is $2^{128.}$	Peer to peer systems can address more objects using GUID.
The load balancing is done based on topology.	Traffic patterns are independent of topology since the object locations are randomized.
Routing tables are updated asynchronously.	Routing tables are updated both synchronously and synchronously.
Failure of one node does not degrade the performance much. Redundancy is introduced in IP. n-fold replication is costly.	Routes and object references can be replicated n-fold, ensuring tolerance of n failures of nodes or connections.
Each IP can map to only one node.	Messages are routed to the nearest replica of the target object.
Secure addressing is possible only between trusted nodes.	Secure communication is possible between limited trusted systems.

8. What is Napster?

Napster was developed for peer –to-peer file sharing especially MP3 files.They are not fully peer-to-peer since it used central servers to maintain lists of connected systems and the files they provided, while actual transactions were conducted directly between machines.

9. Give the features of GUID.

- They are pure names or opaque identifiers that do not reveal anything about the locations of the objects.
- They are the building blocks for routing overlays.
- They are computed from all or part of the state of the object using a function that deliver a value that is very likely to be unique. Uniqueness is then checked against all other GUIDs.
- They are not human understandable.

10. Give the types of routing overlays.

DHT – Distributed Hash Tables. GUIDs are stored based on hash values.

DOLR – Distributed Object Location and Routing. DOLR is a layer over the DHT that maps GUIDs and address of nodes. GUIDs host address is notified using the Publish() operation.

11. **Define pastry**.

Pastry is a generic, scalable and efficient substrate for peer-to-peer applications. Pastry nodes form a decentralized, self-organizing and fault-tolerant overlay network within the Internet.

12. Give the Capabilities of Pastry

- Mapping application objects to Pastry nodes
- Inserting objects
- Accessing objects
- Availability and persistence
- Diversity
- Load balancing
- Object caching
- Efficient, scalable information dissemination

13. What is tapestry?

Tapestry is a decentralized distributed system.It is an overlay network that implements simple key-based routing. Each node serves as both an object store and a router that applications can contact to obtain objects.

14. Give the differences between structured and unstructured networks

Structured Network	Unstructured Network
This guarantees to locate objects and can offer time and complexity bounds on this operation. So it has relatively low message overhead.	This is self-organizing and resilient to node failure.
This needs complexoverlay structures, which can bedifficult and costly to achieve, especially in highly dynamic environments.	Probabilistic and hence cannot offer absolute guarantees on locating objects; prone to excessive messaging overhead which can affect scalability

15. What is Gnutella?

The Gnutella network is a fully decentralized, peer-to-peer application layer network that facilitates file sharing; and is built around an open protocol developed to enable host discovery, distributed search, and file transfer.

16. What is DFS?

The purpose of a distributed file system (DFS) is to allow users of physically distributed computers to share data and storage resources by using a common file system. A typical configuration for a DFS is a collection of workstations and mainframes connected by a local area network (LAN). Distributed file systems support the sharing of information in the form of files and hardware resources.

17. What are the requirements of distributed file systems?

Transparency is operating in such a way as to not be perceived by users. The distributed file system demands the following transparency requirements:

- Login Transparency: User can log in at any host with uniform login procedure and perceive a uniform view of the file system.
- Access Transparency: Client process on a host has uniform mechanism to access all files in system regardless of files are on local/remote host.
- Location Transparency: The names of the files do not reveal their physical location.

- Concurrency Transparency: An update to a file should not have effect on the correct execution of other process that is concurrently sharing a file.
- Replication Transparency: Files may be replicated to provide redundancy for availability and also to permit concurrent access for efficiency.

18. What are the components in file structures?

File service architecture offers a clear separation of the main concerns in providing access to files is obtained by structuring the file service as three components:

- A flat file service
- A directory service
- A client module.

19. What is AFS?

This was developed by Carnegie Mellon University as part of Andrew distributed computing environments (in 1986).The public domain implementation is available on Linux (LinuxAFS). It was adopted as a basis for the DCE/DFS file system in the Open Software Foundation (OSF, www.opengroup.org) DEC (Distributed Computing Environment).Like NFS, AFS provides transparent access to remote shared files for UNIX programs running on workstations

20. Define file access modes.

File access models are methods used for accessing remote files and the unit of data access.

21. What is naming and locating facility?

The naming faciliy of a distributed operating system enables users and programs to assign character-string names to objects and subsequently use these names to refer to those objects. The locating faciliy, which is an integral part of the naming facility, maps an object's name to the object's location in a distributed system.

22. Define naming resolution.

Name resolution is the process of mapping an object'sname to the object's properties, such as its location.

23. What is absolute name?

An absolute name begins at the root context of the name space tree and follows a path down to the specified object, giving the context names on the path.

24. What is relative name?

A relative name defines a path from the current context to the specified object. It is called a relative name because it is "relative to" (start from) the user's current context.

25. What are the layers in name spaces?

Three layers used to implement such distributed name spaces

- Global layer: root node and its children
- Administrational layer: directory nodes within a single organization
- Managerial layer

26. What is LDAP?

LDAP is a standard technology for building computer network directories. A network directory is a specialized database that stores information about devices, applications, people and other aspects of a computer network.

PART – B

1. Explain about peer to peer communication.
2. Explain the working of routing overlays.
3. Describe Napster.
4. Write in detail about peer to peer middleware.
5. Explain about pastry.
6. Describe tapestry.
7. Write in detail about distributed file system.
8. Describe file service architecture.
9. Write in detail about Andrew file system.
10. Describe the file accessing models.
11. Elucidate the file sharing semantics.
12. Describe the naming in distributed systems.