UNIT IV : Naming and DFS

(1) Name, Addresses and Identifiers.

1. A name is a string of bits or characters that is used to refer to an entity within the system.

2. An entity could be anything such as hosts, printers, files, etc.

3. To operate on an entity, we need to access it, for which we require access point

4. Access point is a special kind of entity and its name is called 'address.'
   eg: IP, port, phone no, etc.

- An entity can have more than one access point / address
- An entity can change its access points or addresses
- An address refers to atmost one entity.
- Addresses may be location dependent

5. Identifiers

- Identifier is a special name to uniquely identify an entity

- A true identifier has following three properties
  ① Each identifier refers to atmost one entity

  ② Each entity ~~refers to~~ is refered by atmost one identifier

  ③ An identifier always refers to the same entity (i.e it is never reused)

  Eg: MAC, SSN

- Identifiers allows us to unambiguously refer to an entity

- It allows processes to unambiguously interact with entities.

* Significance of Naming in DS.

- Naming place plays a crucial role in distributed systems because it enables entities to be identified, located and accessed efficiently.

- Names abstracts the complexity of accessing distributed resources such as files, servers or services.
- Instead of low level details like IP addresses, users and applications can interact with entities in human readable names.

- The seperation of names and addresses allow flexibility in distributed system as entities can be moved, replicated or replaced without affecting the way other components interact with them

- The naming system handles the resolution of names to addresses and this mapping can be dynamic to adapt to changes in addresses accordingly.

- Names in distributed systems can be mapped to multiple entities (replicas). If one replica fails, name can resolve to another operational instance thus improving fault tolerance.

- In DFS, naming determines how files and directories are organised across multiple servers.

* Flat naming

- Identifiers are convenient to uniquely represent entities.
- In many cases, identifiers are simply random bit strings which we conveniently refer to as Flat names or unstructured names
- Eg: SSN, MAC, ISBN, etc.
- Locating an entity when given only its name involves a process known as

"Name resolution"

OR

"Entity resolution"

- The goal is to map the identifier to the actual entity it represents.
- This is done through various mechanisms

(a) Simple solution { Broadcasting / Forwarding pointer
(b) Home based approach
(c) Distributed hash tables
(d) Hierarchical approaches.

(1) Broadcasting

- Broadcasting is an important technique used in certain senarios to aid in name resolution or entity resolution especially in decentralised peer-to-peer systems.

- Broadcasting involves sending a query or request for a specific identifier to all or subset of nodes in a nlw.

- Each node that receives the broadcasted query, checks if it holds the information about the requested identifier.

- If it does, the corresponding entity can be located and provide the response to the query

- This technique is generally used to locate entity associated with the given identifier where there might not be a centralized lookup.

Eg: ARP (Address resolution Protocol)

Disadvantage

It is not efficient for big nlws and it requires all entities to continuously listen for all incoming requests.

(2) Forwarding pointer

- Forwarding pointers are another approach used in name resolution or entity resolution process in distributed systems

- A forwarding pointer is essentially a reference or a piece of information that helps direct the requester to the location or node where the desired entity can be found.

- Instead of directly providing the actual location or entity associated with an identifier, a system can provide a forwarding pointer.

- This pointer points to another node or location that holds the actual information

- This approach adds a layer of indirection and allows for flexibility in managing the mapping b/w identifiers and entities.

- Forwarding pointer can be updated dynamically to accomodate changes in the system

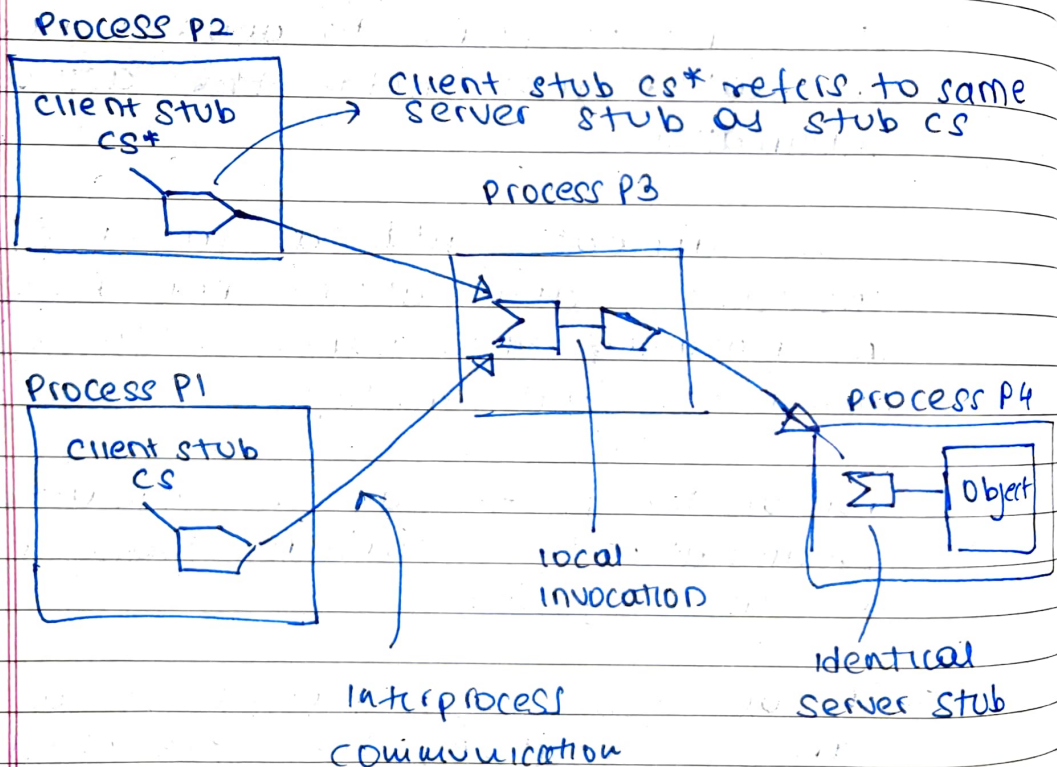
- If an entity is moved to a different location, the forwarding pointer can be updated to point to a new location ensuring that

request for the identifier are still correctly resolved.

- Forwarding pointers can be used for load balancing purposes

If multiple nodes can handle request for same identifier, a forwarding pointer can direct request to different nodes in a balanced manner, distributing the load evenly across the system.

- Fig: Principle of forwarding ptrs using client stub and server stub pairs.



(3) Home based Approaches

- Home based approaches involves associating a each identifier with a specific 'home' node or location where information about the corresponding entity is stored or managed.

- The use of broadcasting and forwarding pointers imposes scalability problems.

Broadcasting (multicasting) is difficult to implement efficiently in largescale n/ws

Long chains of forwarding pointers introduce performance problems and are susceptible to broken links.

- In home based approach, we introduce a home location that keeps track of current location of entity.

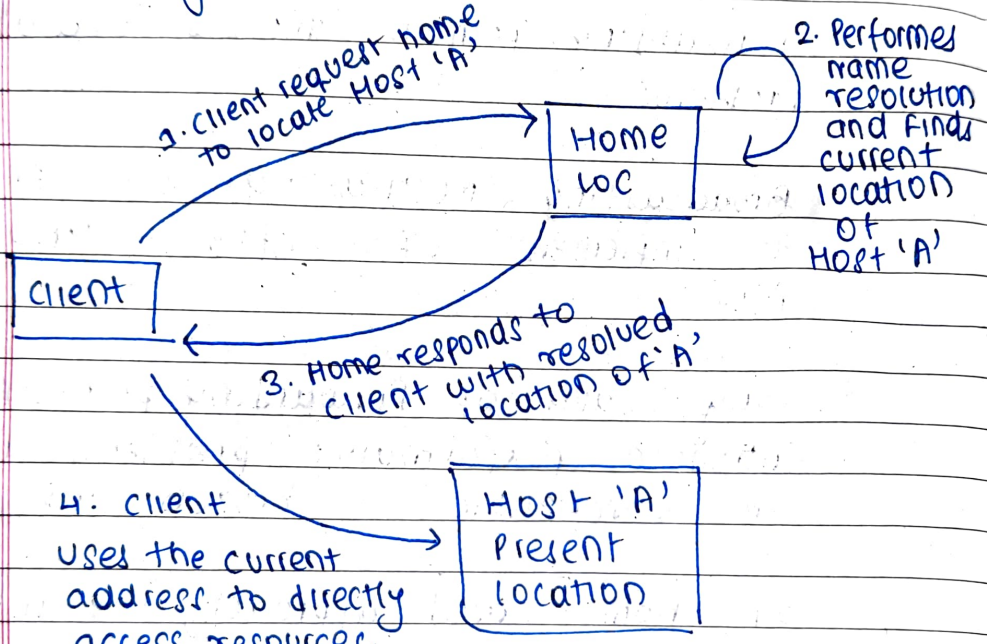
- Each identifier is assigned a home node during the entities creation or registration.

- In some systems, there might be a central entity that assigns and manages home nodes

- Home based approach offers a balance b/w centralization and distribution

- However there are challenges

1. Home node becomes a single point of failure
2. Home node might struggle to handle high traffic
3. Performance overhead as it requires contacting home node which can introduce latency.



4. Poor geographical scalability as home server might be far away but entity might be next to client.

(4) Distributed Hash Tables (DHTs)

- DHTs are widely used technique for resolution of flat names in distributed systems

- DHTs provide a decentralized and efficient way to manage the mapping blw identifiers and entities across nlws.

- In DHTs, each node is responsible for storing and managing a portion of the data.

- When a client wants to retrieve or store data, it sends a request to the nlw.

- The request is then forwarded to an appropriate node based on the key of the data being requested.

The node then responds to the request and either retrieves or stores the data.

- Each identifier is hashed to produce a numeric value.
This numeric value acts as key

- There are many DHT systems, one of them is Chord (peer-to-peer)

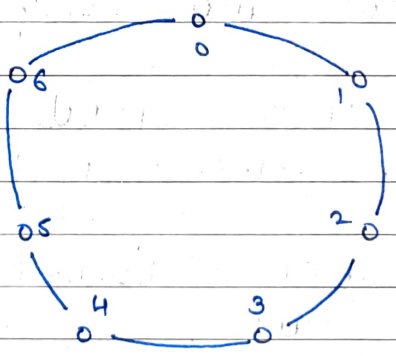
CHORD PROTOCOL

- Chord is a protocol and algorithm for peer-to-peer distributed hash table

- Chord specifies how keys are assigned to nodes and how a node can discover value of key by first locating node responsible for that key.

- Chord assigns an m-bit identifier (randomly chosen) to each node. A node can be contacted through its nlw address.

- A node generates its identifier by picking a random value from the hash space.
The node joins the DHT and determines its successor and predecessor in the table.



- Internally chord uses a consistent hash function for mapping keys to node locations. The nodes are hashed according to their IP address while key value pairs are hashed on

their keys ring
- nodes are arranged in a virtual ring. Chord
- key (k) is assigned to a node whose identifier is equal to or greater than the keys identifier..

- This node is called as successor (k) and its first node clockwise from k.
- Each node knows how to contact its 'successor' node in virtual ring, so all nodes can be visited in linear order until we encounter node that contains the key

- Each node in DHT maintains a finger table that allows systems to lookup values in O(log N) time rather than linear O(N) time.

- Finger table contains references (pointers) to other nodes in a ring to facilitate efficient routing.

- When a node receives a query for 'k' key, it first checks if k falls within its immediate range of responsibilities

- If not, then the node uses finger table to forward the query to the nearest predecessor of k in the table.
- This reduces the no. of hops compared to sequential search

(5) Hierarchical Approaches

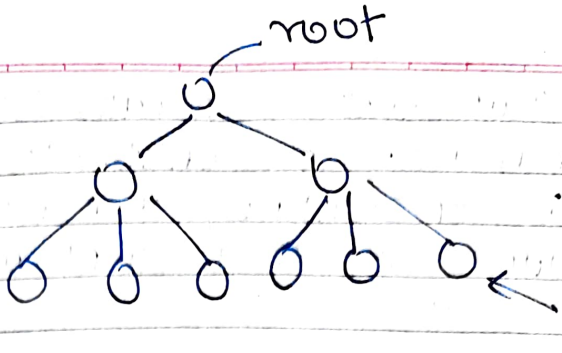
- The hierarchical approach for flat name resolution is a strategy used in distributed systems to manage and resolve flat names by organising the resolution process into multiple levels or tiers.

- Flat names lack inherent structure (eg: random strings, numbers) making direct resolution challenging in large scale systems.

- A hierarchical approach breaks down the flat name resolution into multiple levels, where responsibility of managing names is distributed across different domains or tiers.

- The resolution process starts at root level and moves down through hierarchy until the flat name is resolved to its associated entity or address.

Eg: Domain name systems. (DNS)
While domain names are structured, the IP addresses they resolve can be treated as flatnames
DNS resolves names hierarchically starting from root servers
↓
top level domains
↓
authoritative name servers



(*) Structured Names.

- Flat names are good for machines but generally not very convinient for humans to use.

- Structured names are designed to be human readable and meaningful, resembling hierarchical organisation or categorization.

(I) Name spaces

- Names are commonly organised in namespaces

- Namespaces define a set of possible names that can be ~~identified~~ used to identify resources within a distributed system.

- Namespaces for structured names can be represented as labelled, directed graph with two types of nodes:

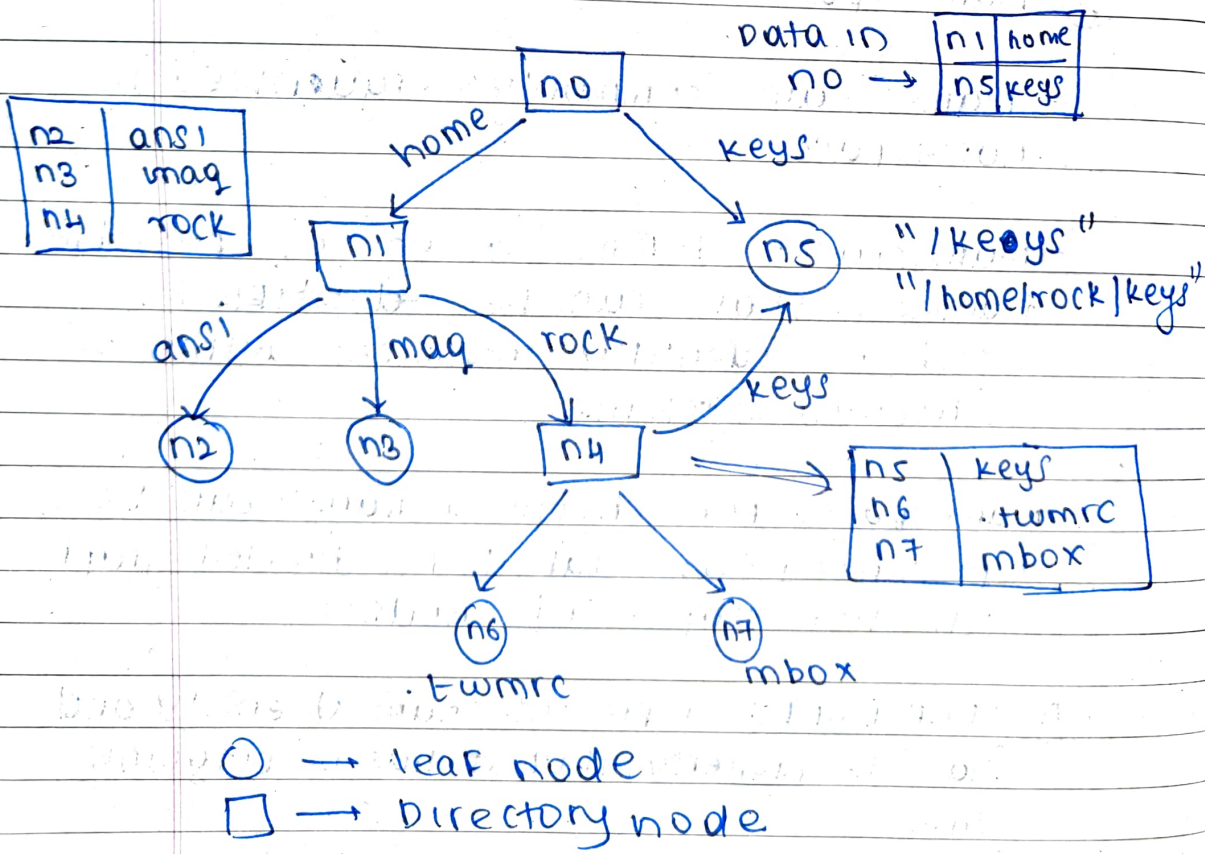
(A) LEAF NODE: represents named entity and has the property that it has no outgoing edges

A leaf node generally stores information of entity it is res representing, such as address - so that client can access it.

Alternatively it can store the state of entity like a complete file in case of file systems.

(B) DIRECTORY NODE - It has no. of outgoing edges each labelled with a name.

A directory node stores a table in which an outgoing edge is represented as a pair (edge label : node identifier) such table is called directory table.



- In such directed acyclic graph organisation, a node can have more than incoming edge but the graph is NOT permitted to have a cycle.
- There are also namespaces that do not have such restrictions

(*) Attributed based Naming

- Flat and structured names generally provide a unique and location independent way of referring to entities.

- However, along with location independence and human ~~Friendly~~ Friendliness, efficient search for entities is also an important criteria

- Attribute-based naming in distributed systems is a flexible approach where entities are identified and located based on their attributes rather than fixed names or hierarchical paths.

- In this approach, an entity is assumed to have an ~~association~~ associated collection of attributes, where each attribute say something about that entity.

- These attributes provide metadata about the entity making it easier to search for a resource based on specific criteria

- It is upto the naming system to return one or more entities that ∞ meet the user's description.

For eg, user might search for printer with attributes like

{
  type : laser
  location : floor 3
  brand : HP
}

- This approach is used in
  - CORBA
  - JNDI (Java Naming and Directory Interface)
  - UDDI

- Attribute-based naming systems are also known as directory services.

- A directory service is a system or software designed to store, manage and retrieve information about entities (like users, devices, services, etc) in a n/w.

- setting values consistently is a crucial problem

- To elevate some problem, Resource Description Framework (RDF) is used to unify the way resources are described.

- In RDF, each resource is described using a triplet {Subject, Predicate, Object}

eg: {Person, name, Alice} describes a person whose name is alice.

- ~~In RDF, each resource~~

- In RDF, each subject, predicate and object can be a resource itself.

---

* Lightweight Directory Access Protocol (LDAP)

- A common approach a tackling distributed directory services is to combine structured naming with attribute based naming.

- Eg MS Active Directory Service, etc

- Many such systems rely on LDAP

- LDAP defines a standard ~~protocol~~ method for accessing and updating information in a directory

- It has been widely accepted method for directory access

- LDAP is derived from OSI's X.500 directory service

- LDAP is particularly popular in enterprise environments for managing user authentication and access control.

- An LDAP directory service consist of number of directory entities - a collection of {attribute, value} pairs similar to DNS resource records.

- The collection of all directory entities is called Directory Information Base (DIB)

- An important aspect of DIB is that, each record is uniquely named so that it can be looked up

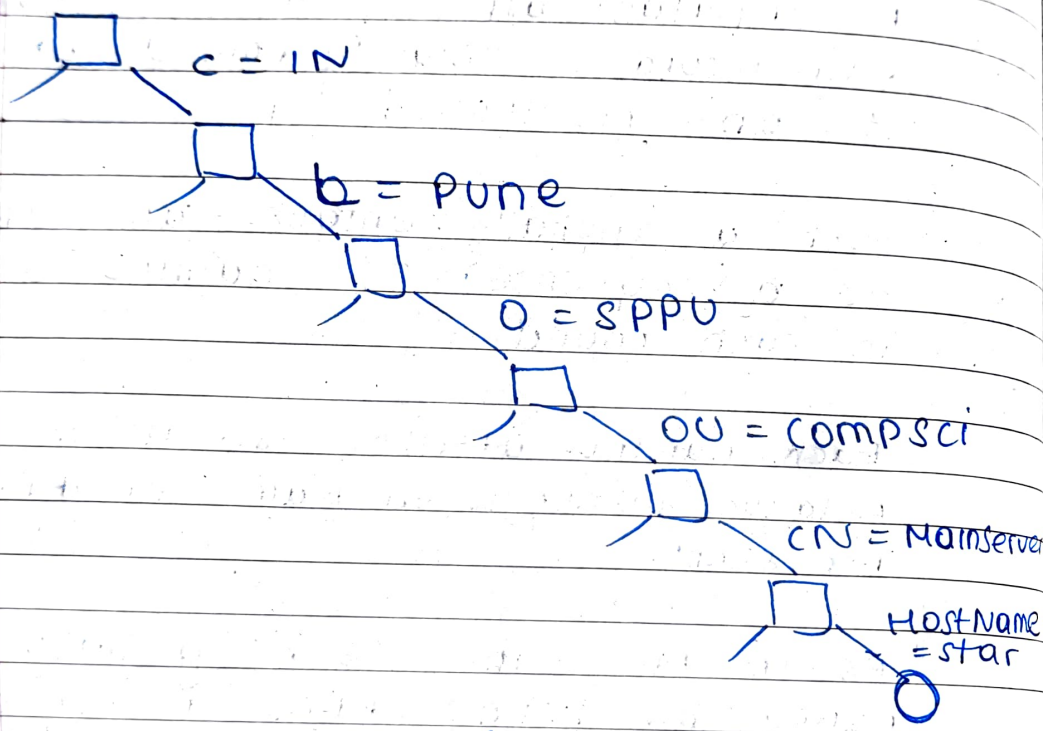
- Such a globally unique name appears as a sequence of naming attributes in each round.

Each naming attribute is called a Relative Distinguished Name or RDN for short.

- As in DNS, the use of globally unique names by listing RDNs in sequence leads to hierarcy of the collection of directory entries which is referred to as Directory Information Tree (DIT)

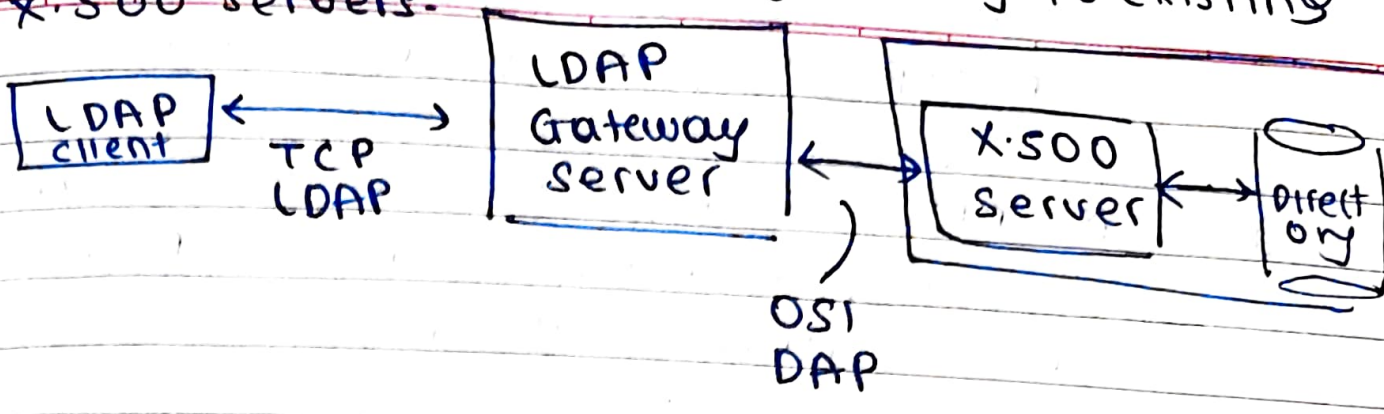
- DIT essentially forms a graph (naming graph) of an LDAP directory service in which each node represents directory service

Attribute	value
country	IN
City	Pune
Org	SPPU
Org unit	comp sci
Common Name	Main Server
Host Name	star
Host Addr	137.37.20.10



answer = search ("& (C=IN)(L=Pune)
(O=SPPU) (OU=*)(CN=Mainser
)

LDAP is a seperate protocol that has a feature which allows it to work in hybrid setups acting as gateway to existing X.500 servers.



(1) - clients called Directory User Agent (DUA), are similar to name resolvers, contact the LDAP server

(2) LDAP server known as Directory service agent (DSA) organises data in tree (DIT = directory information tree) with named nodes (DIB = directory information base)

(3) LDAP server looks up entries based on attributes.
In case of large scale directory, DIT is partitioned and distribute across several DSAs.