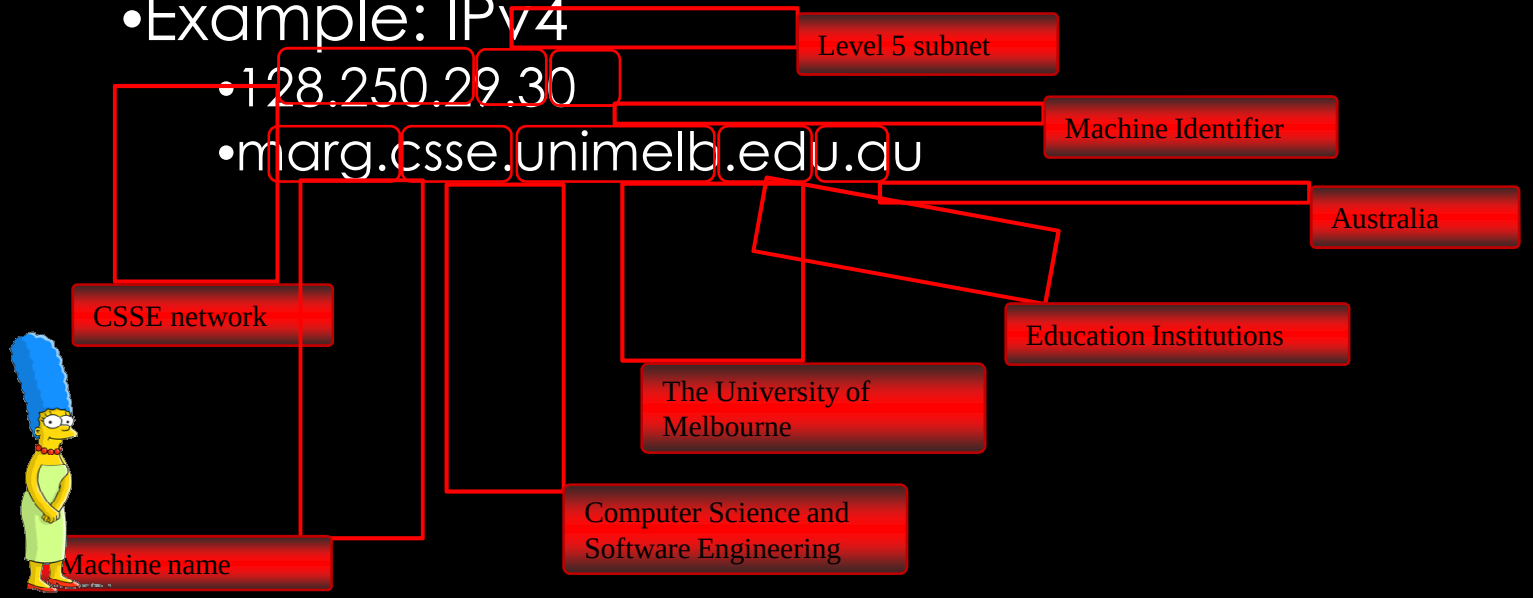
Naming services, name space, Name resolution, name servers

SUDHA M.R.
III year CSE - B

Names or Codes, or Numbers?

- Names (when meaningful) are easier to remember than codes or numbers…
- Number (or sequence codes) are more useful for structuring data and locating resources by a program..
- Example: IPv4
  - 128.250.29.30
  - marg.csse.unimelb.edu.au



Introduction

- In a distributed system, names are used to refer to a wide variety of resources such as:
  - Computers, services, remote objects, and files, as well as users.
- Naming is fundamental issue in DS design as it facilitates communication and resource sharing.
  - A name in the form of URL is needed to access a specific web page.
  - Processes cannot share particular resources managed by a computer system unless they can name them consistently
  - Users cannot communicate within one another via a DS unless they can name one another, with email address.
- Names are not the only useful means of identification: descriptive attributes are another.

What are Naming Services?

•The naming facility 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 facility, 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.

What are Naming Services?

- Definition

  In a Distributed System, a Naming Service is a specific service whose aim is to provide a consistent and uniform naming of resources, thus allowing other programs or services to localize them and obtain the required metadata for interacting with them.

- Key benefits
  - Resource localization
  - Uniform naming
  - Device independent address (e.g., you can move domain name/web site from one server to another server seamlessly).

Name management

- Name management is separated from other services largely because of the openness of distributed systems, which brings the following motivations:
- Unification: It is often convenient for resources managed by different services to use the same naming scheme. URIs are a good example of this.
- Integration: It is not always possible to predict the scope of sharing in a distributed system. It may become necessary to share and therefore name resources that were created in different administrative domains. Without a common name service, the administrative domains may use entirely different naming conventions.

General name service requirements and goals

- To handle an essentially arbitrary number of names and to serve an arbitrary number of administrative organizations: For example, the system should be capable of handling the names of all the documents in the world.
- A long lifetime: Many changes will occur in the organization of the set of names and in the components that implement the service during its lifetime.
- High availability: Most other systems depend upon the name service; they can't work when it is broken.
- Fault isolation: Local failures should not cause the entire service to fail.
- Tolerance of mistrust: A large open system cannot have any component that is trusted by all of the clients in the system.

Desirable features of a good naming system

- Location transparency : Name of the object should not reveal anything about the physical location of the object.
- Location independence: The name of an object need not be changed when the object location changes. An object at any node can issue an access request without the knowledge of its own physical location.
- Scalability: DS vary in size from one node to several nodes. Size may change dynamically. Naming system should thus be scalable.
- Uniform naming convention: In most DS, different naming systems are used. Example, file names typically differ from username and process names. Instead, a good naming system should use uniform naming convention.

- Multiple user-defined names for the same object: Users should have the flexibility to assign multiple user-defined names to the same object. In this case, it should be possible for a user to change or delete his or her name for the object without affecting those of other users.
- Group naming: Should allow many different objects to be identified by the same name. It can be useful for broadcasting or for conferencing with other applications.
- Performance: Performance measurement is given by the amount of time needed to map the object's name to its attributes. i.e, the number of messages exchanged in a name mapping operation should be as small as possible.
- Fault tolerance: Naming systems should be capable of tolerating the faults that occur due to node failure or communication link failure. It should at least continue functioning in a degraded form.
- Replication transparency: Replica are created to improve performance and reliability. A naming system should support the use of multiple copies of the same object. It should be able to locate the nearest replica.

Name – definition

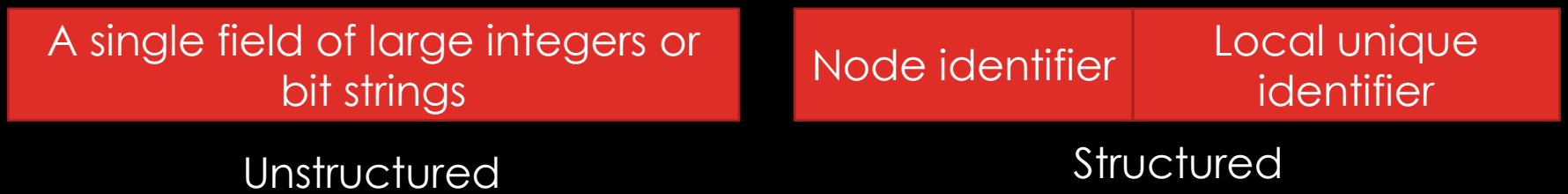
•A name is a string composed of a set of symbols chosen from a finite alphabet.
•For example, TEMPUS, #173#4879#5965, /a/b/c, 25A2368DM197, etc.
•A name is also called an identifier because it is used to denote or identify an object.
•A name may also be thought of as a logical object that identifies a physical object to which it is bound from among a collection of physical objects. Therefore, the correspondence between names and objects is the relation of binding logical and physical objects for the purpose of object identification.
•The OSI definition of a name is a very general one. It covers:

§Data items which identify objects by their location. Such names are called addresses .

§ It also covers names which are assigned to objects. Such names are called titles .

System-oriented name

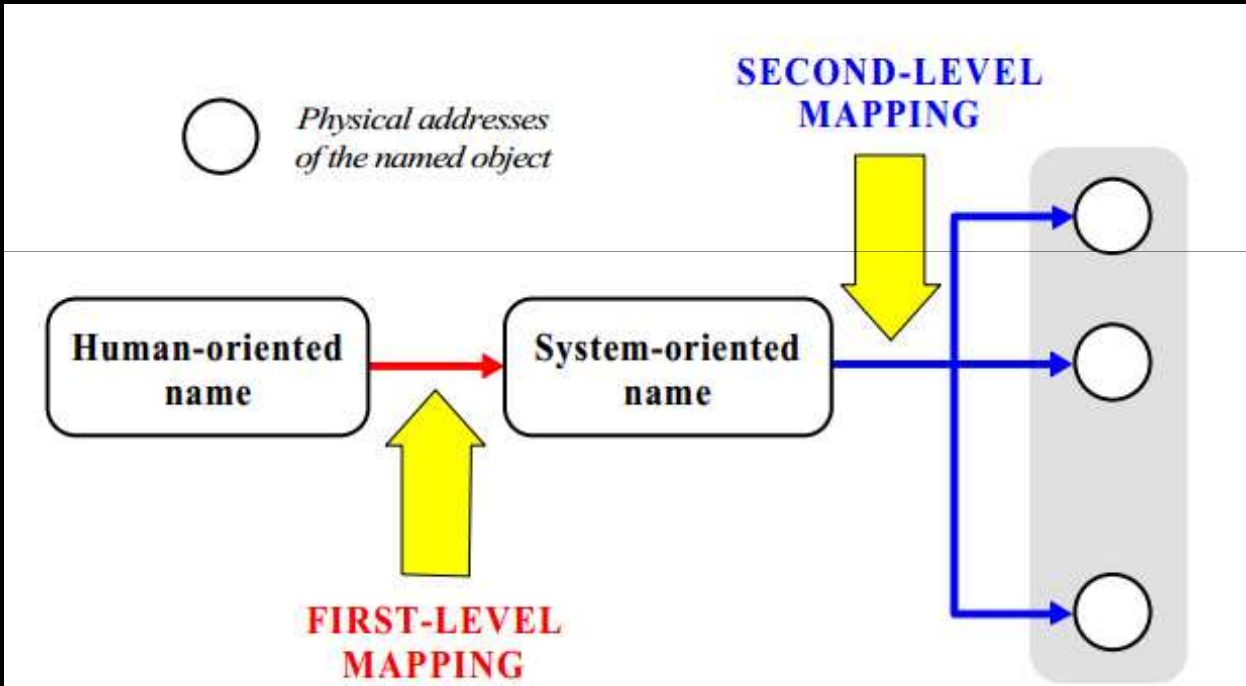
- Fixed size bit pattern that can easily be manipulated and stored by machines. Also known as low-level names.
- Characterized by large integers or bit strings of variable length.
- System oriented names are also referred as unique identifiers and are automatically generated.
- These are hard to guess and provide good security.
- Centralized approach is used for generating structured and unstructured names. A standard and uniform global identifier name is generated for each object in the system by a centralized global unique identifier generator.



Human oriented names

•Human oriented names are generally a character string and is meaningful to its user. It is defined by the user.

•Are not unique for an object and are variable in length.
•Known as high level names and are not easily manipulated, stored and used by the machines.



A simple naming model based on the use of human-oriented and system-oriented names in a distributed system.

Requirements for name spaces

- Allow simple but meaningful names to be used
- Potentially infinite number of names
- Structured
  - to allow similar subnames without clashes
  - to group related names
- Allow re-structuring of name trees
  - for some types of change, old programs should continue to work
- Management of trust

The set of names complying with a given naming convention is said to form a name space

Flat Name space

- The simplest name space is a flat name space where names are character strings exhibiting no structure.
- Names defined in a flat name space are called primitive or flat names. Since flat names do not have any structure, it is difficult to assign unambiguous meaningful names to a large set of objects.
- Therefore, flat names are suitable for use either for small name spaces having names for only a few objects or for system-oriented names that need not be meaningful to the users.

PARTITIONED NAME SPACE

When there is a need to assign unambiguous meaningful names to a large set of objects, a naming convention that partitions the name space into disjoint classes is normally used.

•Each partition of a partitioned name space is called a domain of the name space.

•A name defined in a domain is called a simple name. In a partitioned name space, all objects cannot be uniquely identified by simple names, and hence compound names are used for the purpose of unique identification.

Hierarchical name space

- A commonly used type of partitioned name space is the hierarchical name space, in which the name space is partitioned into multiple levels and is structured as an inverted tree.
- Each node of the name space tree corresponds to a domain of the name space. In this type of name space, the number of levels may be either fixed (Eg. Grapevine, Xerox Clearinghouse) or arbitrary (Eg.DARPA).
- For instance, telephone numbers fully expanded to include country and area codes form a four-level hierarchical name space and network addresses in computer networks form a three-level hierarchical name space where the three levels are for network number, node number, and socket number.

Names and resources

•Currently, different name systems are used for each type of resource:

resource	*name*	*identifies*
file	pathname	file within a given file system
process	process id	process on a given computer
port	port number	IP port on a given computer

•Uniform Resource Identifiers (URI) offer a general solution for any type of resource. There two main classes:

URL Uniform Resource Locator (URL)

- •typed by the protocol field (http, ftp, nfs, etc.)
- •part of the name is service-specific
- •resources cannot be moved between domains

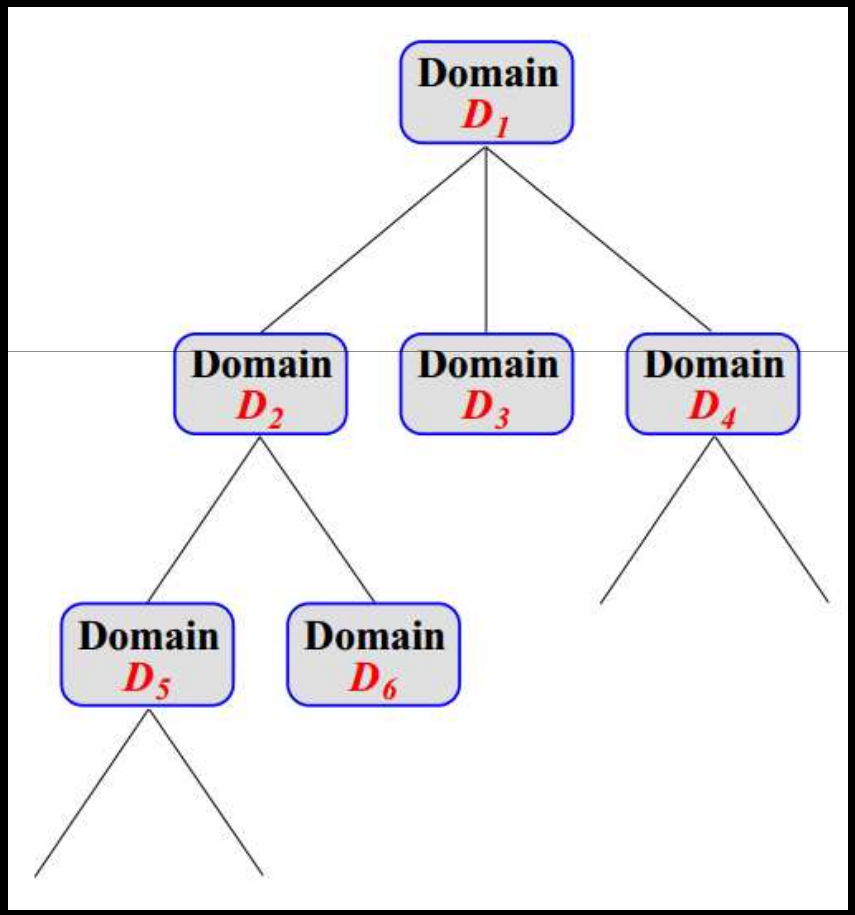
URN Uniform Resource Name (URN)

- •requires a universal resource name lookup service - a DNS-like system for all resources

Name server

- Name spaces are managed by name servers. A name server is a process that maintains information about named objects and provides facilities that enable users to access that information.
- The name servers that store the information about an object are called the authoritative name servers of that object.
- For example, in a hierarchical name space, it is sufficient that each name server store only enough information to locate the authoritative name servers for the root domain of the name tree.
- The authoritative name servers of the root domain, in turn, should know the locations of the authoritative name servers of the domains that branch out from the root domain.
- In general, the authoritative name servers of a domain should know the locations of the authoritative name servers of only those domains that branch out from that domain.

Domains of a hierarchical name space



In the name space tree of Figure above all name servers must know the locations of the authoritative name servers of domain D1;

The authoritative name servers of domain D1 need only know the locations of the authoritative name servers of domains D2, D3, and D4;

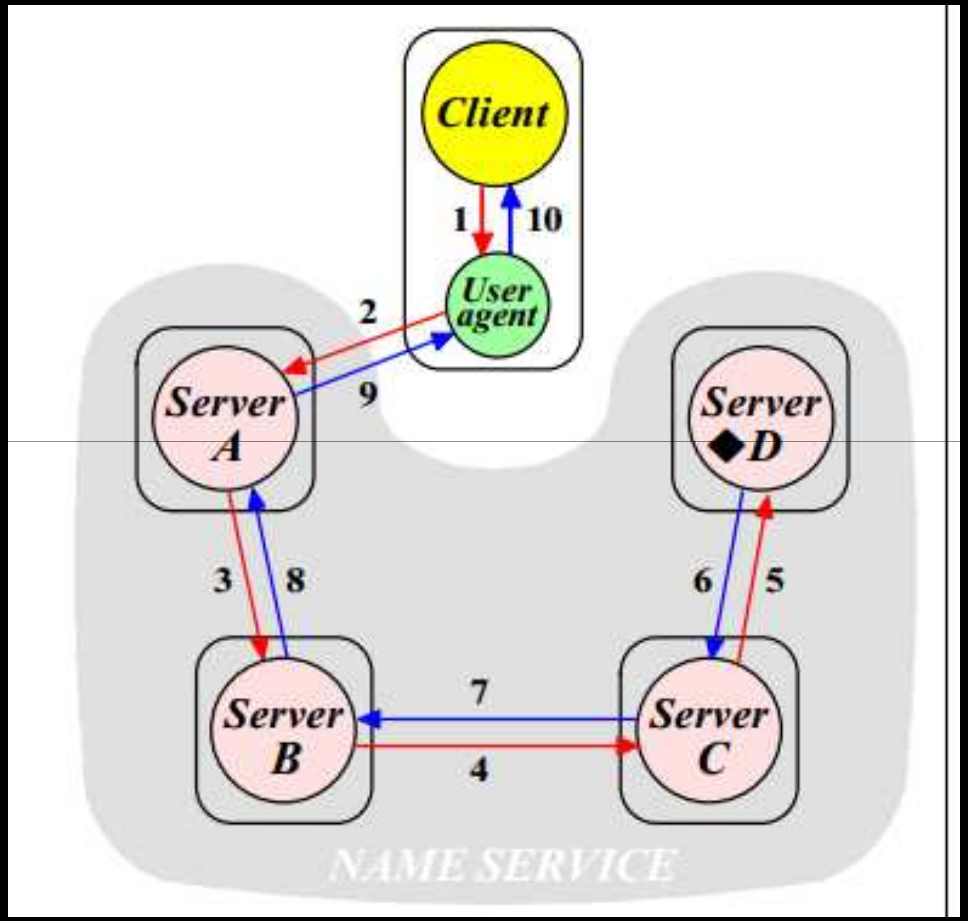
The authoritative name servers of domain D2 need only know the locations of the authoritative name servers of domains D5 and D6.

9.2 Name Resolution

- A name service stores a collection of one or more naming contexts, sets of bindings between textual names and attributes for objects such as computers, services, and users.
- The major operation that a name service supports is to resolve names.

Recursive resolution

•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.

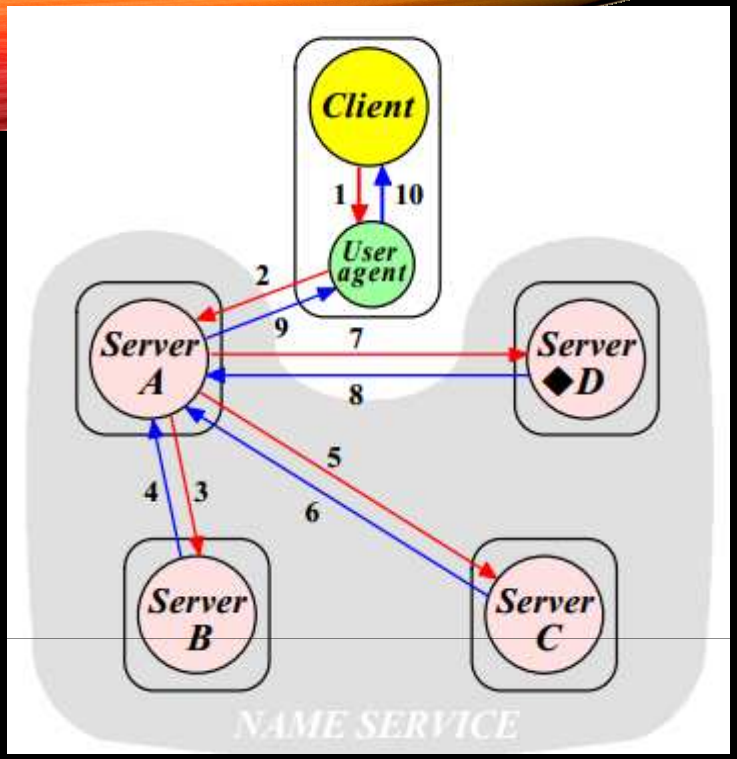


As an example, if the name */a/b/c/d* is to be resolved, the name agent sends it to the name server (say SA) of the root context (/) and waits for a reply.

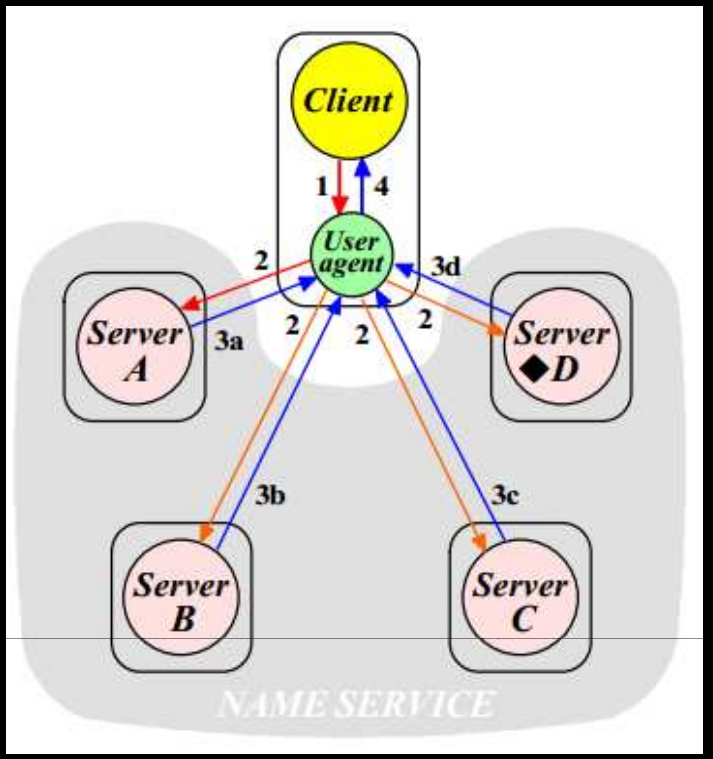
Then SA searches for the component name *a* in the root context, extracts the corresponding binding information, sends the remaining pathname *b/c/d* to the name server (say SB) of the next context (/a), and waits for a reply. Then SB extracts from context */a* the binding information corresponding to the component name *b*, sends the remaining pathname c to the name server (say SC) of the next context (/a/b), and waits fora reply. Then SC extracts from context /a/b the authority attribute corresponding to the component name c and returns it to S B,. which in turn returns it to SA, and finally S, returns it to the name agent.

ITERATIVE 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.
- As in the recursive process, the name agent first sends the name to be resolved to the name server that stores the first context needed to start the resolution of the given name.
- The server resolves as many components of the name as possible. If the name is completely resolved, the authority attribute of the named object is returned by the server to the name agent.
- Otherwise, the server returns to the name agent the unresolved portion of the name along with the location information of another name server that the name agent should contact next.



Parallel iterative resolution protocol



Sequential iterative resolution protocol

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.