Parallel & Distributed System

Naming Service

Md. Biplob Hosen

Lecturer, IIT-JU

Email: biplob.hosen@juniv.edu

Contents

- Names, Identifiers and Addresses
- **Flat Naming**
  - ✓ Broadcasting & Multicasting
  - ✓ Forwarding Pointers
  - ✓ Home-Based Approach
- **Structured Naming**
  - ✓ Namespace
  - ✓ Name Resolution
  - ✓ DNS Example
- **Attribute-Based Naming**

- Reference Books
  - ✓ Distributed Systems: Principles and Paradigms, 3rd Edition by Andrew S. Tanenbaum & Maarten van Steen, Publisher: Pearson Prentice Hall **[CH-05]**.

Naming - Overview

- **Names** are used to share resources, to uniquely identify entities, to refer to locations, and more.
- We need to **resolve** a name to the entity it refers to.
- **Name resolution** thus allows a process to access the named entity.
- The naming system itself be implemented in a distributed fashion.
- In **flat-naming systems**, entities are referred to by an identifier that, in principle, has no meaning at all. In addition, flat names bare no structure, implying that we need special mechanisms to trace the location of such entities.
- In practice, humans prefer to use readable names. **Structured names** allow for a highly systematic way of finding the server responsible for the named entity, as exemplified by the Domain Name System.
- Finally, in **attribute-based naming**, humans often prefer to describe entities by means of various characteristics.

Names, Identifiers and Addresses

- A **name** in a distributed system is a string of bits or characters used to refer to an entity.
- **Entity** is something that is operated on using some **access point**. The name of the access point is called an **address**.
- Entities: hosts, printers, disks, files, processes, users, mailboxes, web pages, graphical windows, messages, network connections, etc.
- An entity may change its access point over course of time. Thus the address cannot be treated as the name of the entity. Moreover an entity may have more than one access point.
- An **identifier** is an unambiguous reference to an entity. An identifier refers to at most one entity. Each entity is referred to by at most one identifier. An identifier always refers to the same entity.

Types of Naming Systems

- **Flat naming:** The identifier is simply a random bit string. Good for machines.
  1. Broadcasting and multicasting
  2. Forwarding pointers
  3. Home-based approaches

- **Structured naming:** Composed of simple human-readable names. Examples are file system naming and host naming on the Internet.

- **Attribute-based naming:** Allows an entity to be described by (attribute, value) pairs. This allows a user to search more effectively by constraining some of the attributes.

Broadcasting/Multicasting

- Consider a LAN that offers efficient broadcasting facility. A message containing the identity of the entity is broadcast to each machine.
- Only the machines that can offer access to that entity send a reply containing the address of the access point.
- **E.g. Address Resolution Protocol (ARP)** used on a LAN to find the data-link address of a machine given only the IP address.
- A more efficient approach for larger networks is multicasting, by which only a restricted group of machines receive the request.

Forwarding Pointers

- When an entity moves from A to B, it leaves behind in A a reference to its new location at B.
- The main advantage of this approach is its simplicity: as soon as an entity has been located using a traditional naming service, a client can look up the current address by following the chain of forwarding pointers.
- It is important to keep the forwarding chains relatively short and to ensure that the forwarding pointers are robust.
- Example: Remote objects that can move from host to host.
- A server stub contains either a local reference to the actual object or a local reference to a remote client stub for that object.
- Whenever an object moves from address A to B, it leaves behind a client stub in its place on A and installs a server stub that refers to it in B. This makes the migration completely transparent to a client.

Continue…

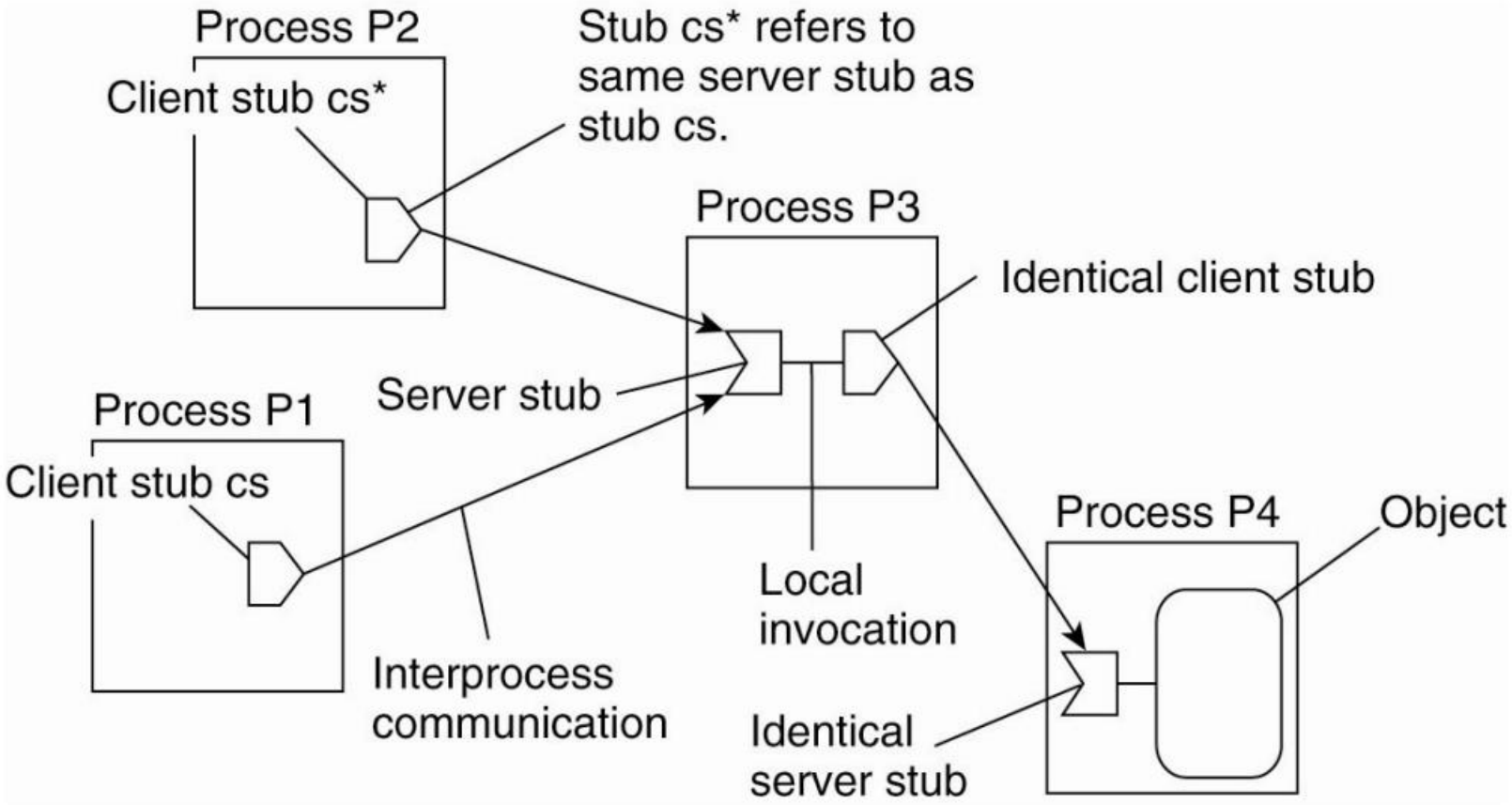


Fig. The principle of forwarding pointers using (client stub, server stub) pairs.

Continue…

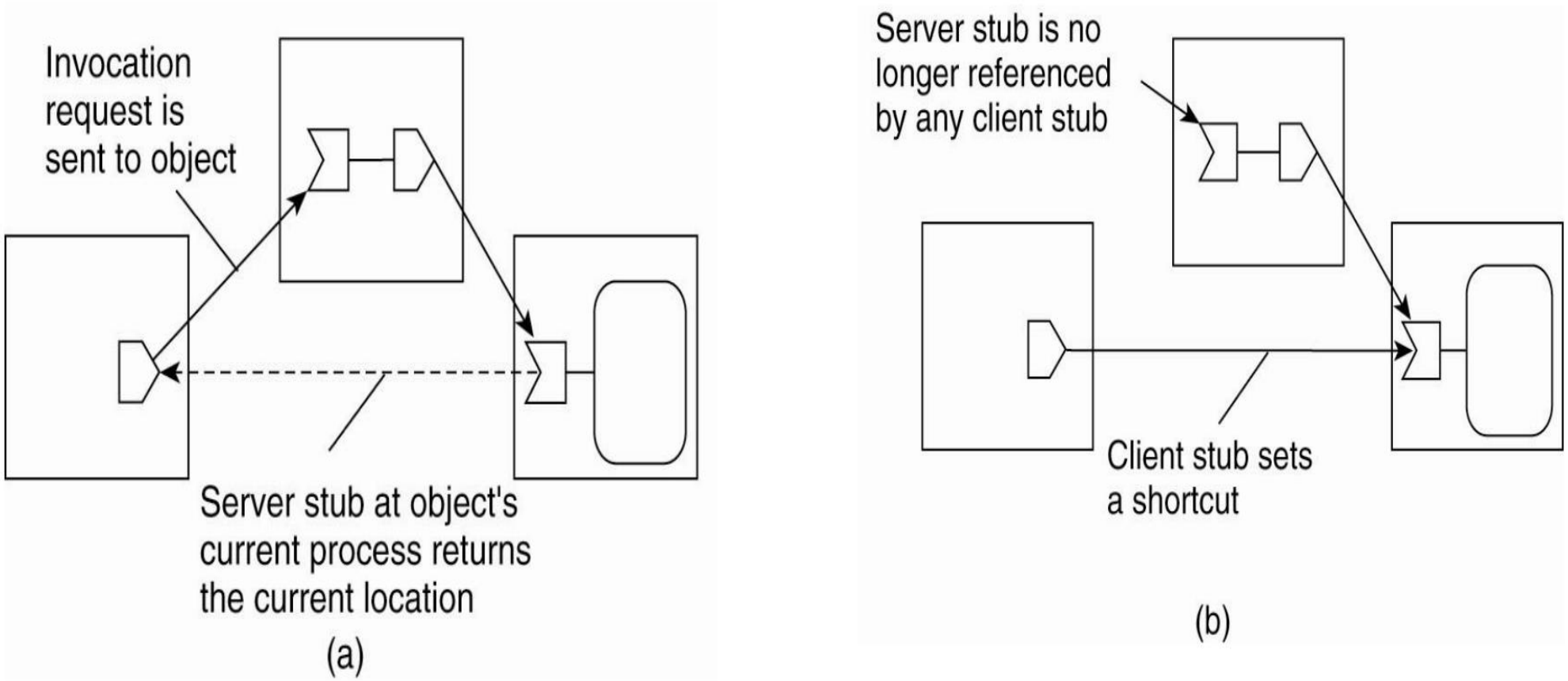


Fig. Redirecting a forwarding pointer by storing a shortcut in a client stub.

Locating Mobile Entities

- Traditional naming systems are primarily used for naming entities that have a fixed location. They are not well-suited for supporting name-to-address mappings that change regularly.
- Suppose an entity moves to a new address. How do we handle this?
- Record the new address in the original DNS database.
- **Problem:** If entity moves again, the update becomes a remote operation, possibly taking long time to complete.
- Record the new name, creating a symbolic link.
- **Problem:** Each lookup now takes two operations. Also each move of the entity adds another level of lookup.
- **Solution:** Separate naming from locating entities by introducing identifiers. An entity has a unique identifier that never changes.

Home-Based Approach

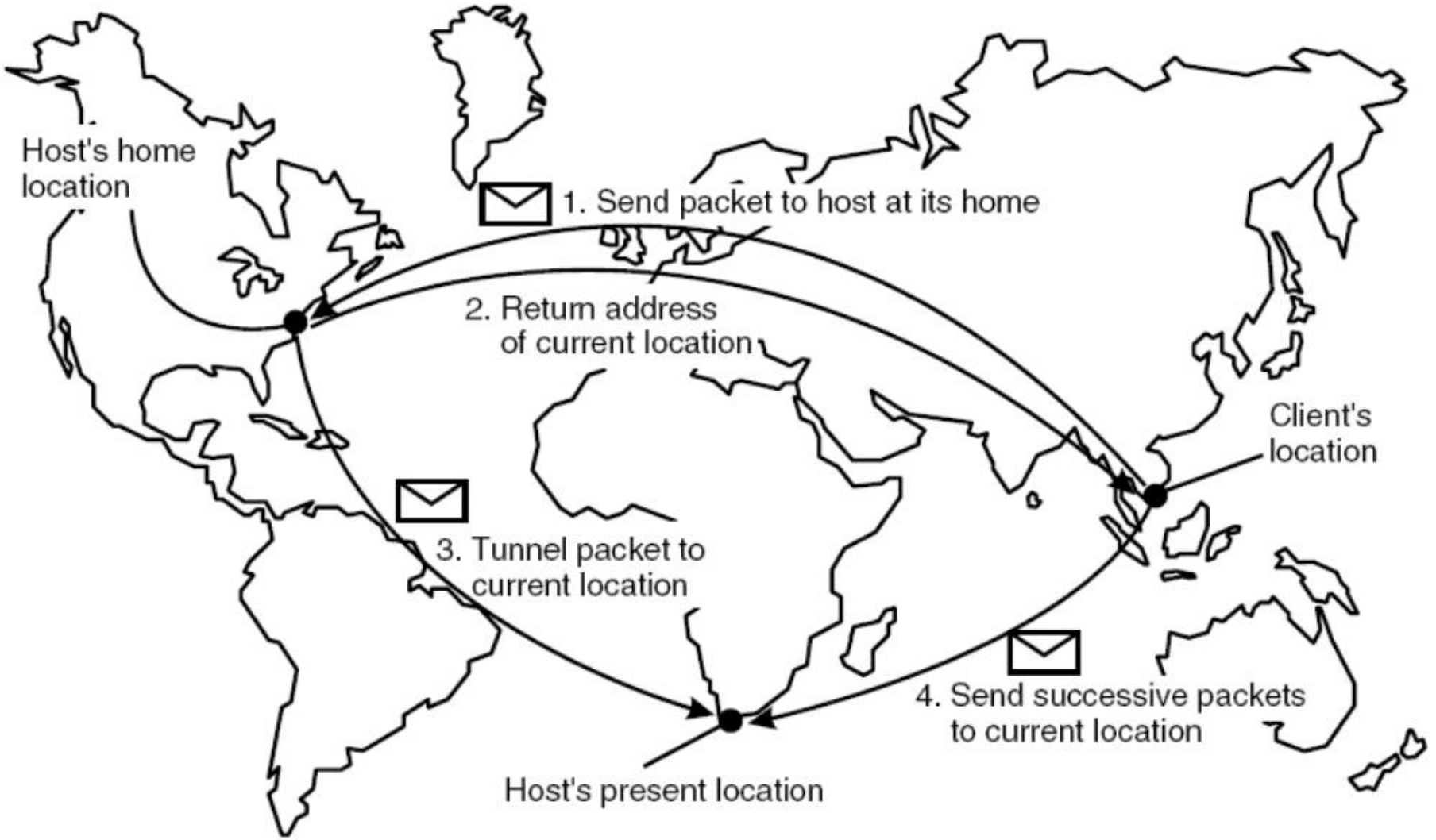


Fig. The principle of Mobile IP

Continue…

Problems with home-based approaches:

- Home address has to be supported as long as entity lives.
- Home address is fixed - unnecessary burden if entity permanently moves.
- Poor geographical scalability (the entity may be next to the client).

Structured Naming

- Flat names are good for machines, but are generally not very convenient for humans to use.
- As an alternative, naming systems generally support structured names that are composed from simple, human-readable names.
- Focus on:
  - Name spaces
  - Name resolution
  - Name space implementation
  - Example: Domain Name System (DNS)

Name Spaces

- Names are commonly organized into what is called a name space. Name spaces for structured names can be represented as a labeled, directed graph.

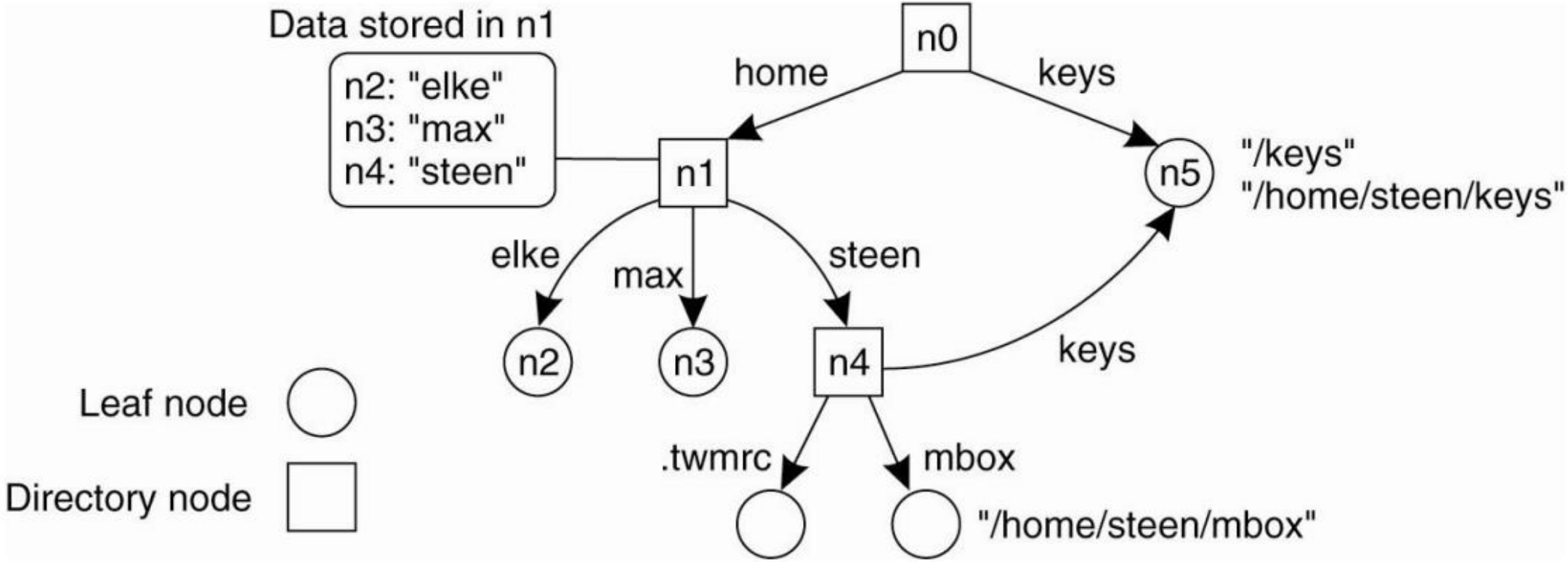


Fig. A general naming graph with a single root node.

Name Resolution

- Name spaces offer a convenient mechanism for storing and retrieving information about entities by means of names.
- More generally, given a path name, it should be possible to look up any information stored in the node referred to by that name.
- The process of looking up a name is called name resolution.
- To explain how name resolution works, let us consider a path name such as N:[$label_1$, $label_2$, ..., $label_n$].
- Resolution of this name starts at node N of the naming graph, where the name $label_1$ is looked up in the directory table, and which returns the identifier of the node to which $label_1$ refers.
- Resolution then continues at the identified node by looking up the name $label_2$ in its directory table, and so on.
- Assuming that the named path actually exists, resolution stops at the last node referred to by $label_n$, by returning that node's content.

Closure Mechanism

- Knowing how and where to start name resolution is generally referred to as **closure mechanism**. Essentially, a closure mechanism deals with selecting the initial node in a name space from which name resolution is to start.
- Consider, for example, the string “0031205983778 4”. Many people will not know what to do with these numbers, unless they are told that the sequence is a telephone number. That information is enough to start the resolution process, in particular, by dialing the number.
- As another example, consider the use of global and local names in distributed systems. A typical example of a local name is an environment variable.
- For example, in Unix systems, the variable named HOME is used to refer to the home directory of a user. Each user has its own copy of this variable, which is initialized to the global, system-wide name corresponding to the user’s home directory. The closure mechanism associated with environment variables ensures that the name of the variable is resolved by looking it up in a user-specific table.

Name Space Implementation

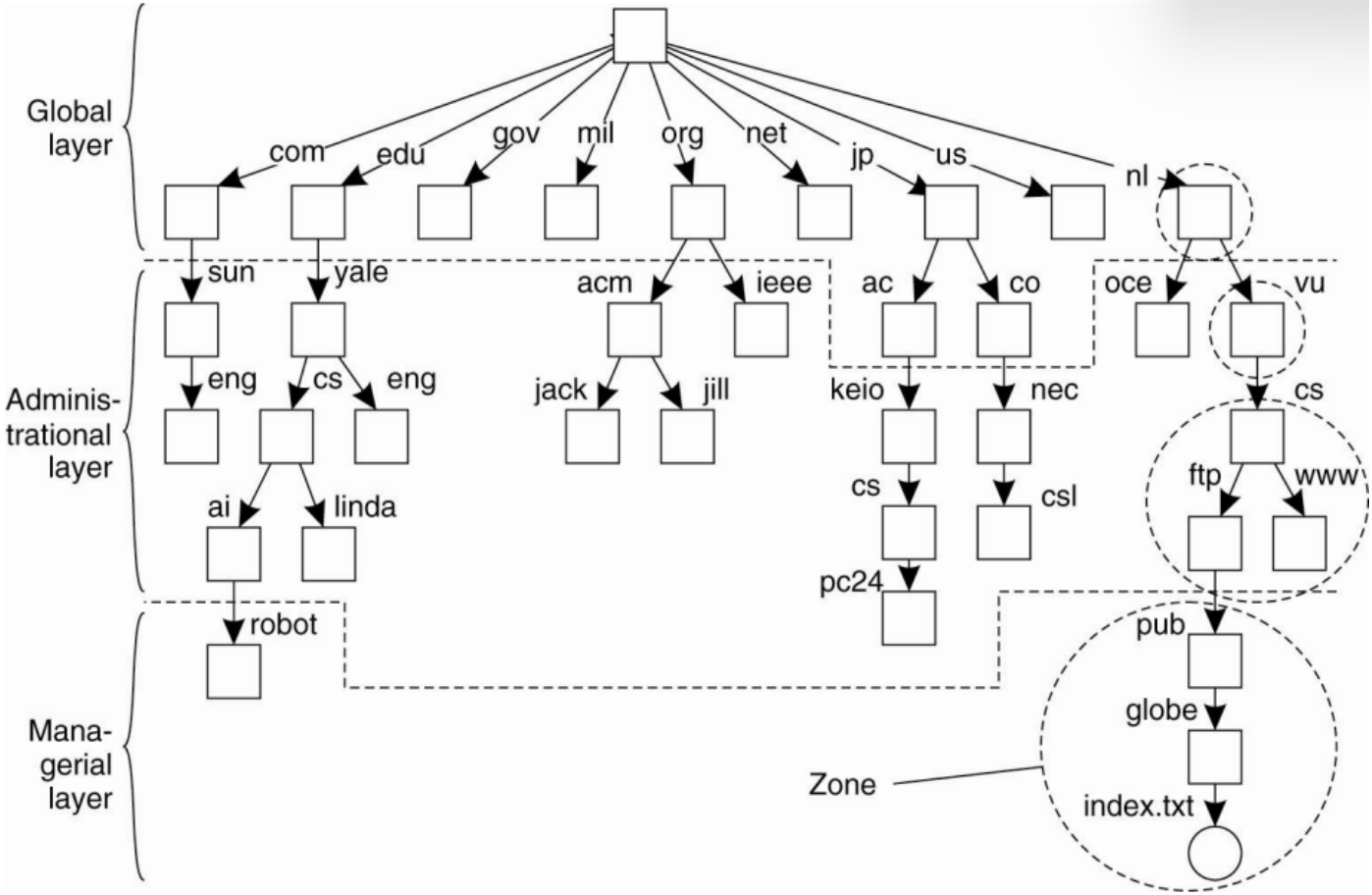
- A name space is implemented by a naming service that allows users to add, remove and look up names.
- It is implemented by name server(s).
- For a distributed system limited to a LAN, it may be feasible to implement the name service with a single name server.
- However, in a large-scale distributed system spread across a large geographical area, it is necessary to distribute the implementation over many name servers.

Name Space Distribution

- Name spaces for a large-scale, possibly worldwide distributed system, are usually organized hierarchically.
- Assume a name space has a single root node. To implement such a name space, it is convenient to partition it into three logical layers.
- The **global layer** is formed by highest-level nodes, that is, the root node and other directory nodes logically close to the root, namely its children. Nodes in the global layer are often characterized by their stability, in the sense that directory tables are rarely changed. Such nodes may represent organizations, or groups of organizations, for which names are stored in the name space.
- The **administrational layer** is formed by directory nodes that together are managed within a single organization. A characteristic feature of the directory nodes in the administrational layer is that they represent groups of entities that belong to the same organization or administrational unit. For example, there may be a directory node for each department in an organization, or a directory node from which all hosts can be found. The nodes in the administrational layer are relatively stable, although changes generally occur more frequently than to nodes in the global layer.

Continue…

- The **managerial layer** consists of nodes that may typically change regularly. For example, nodes representing hosts in the local network belong to this layer. For the same reason, the layer includes nodes representing shared files such as those for libraries or binaries.



- Another important class of nodes includes those that represent user-defined directories and files. In contrast to the global and administrational layer, the nodes in the managerial layer are maintained not only by system administrators, but also by individual end users of a distributed system.

Continue…

- A comparison between name servers for implementing nodes from a large-scale name space partitioned into a global layer, an administrational layer, and a managerial layer:

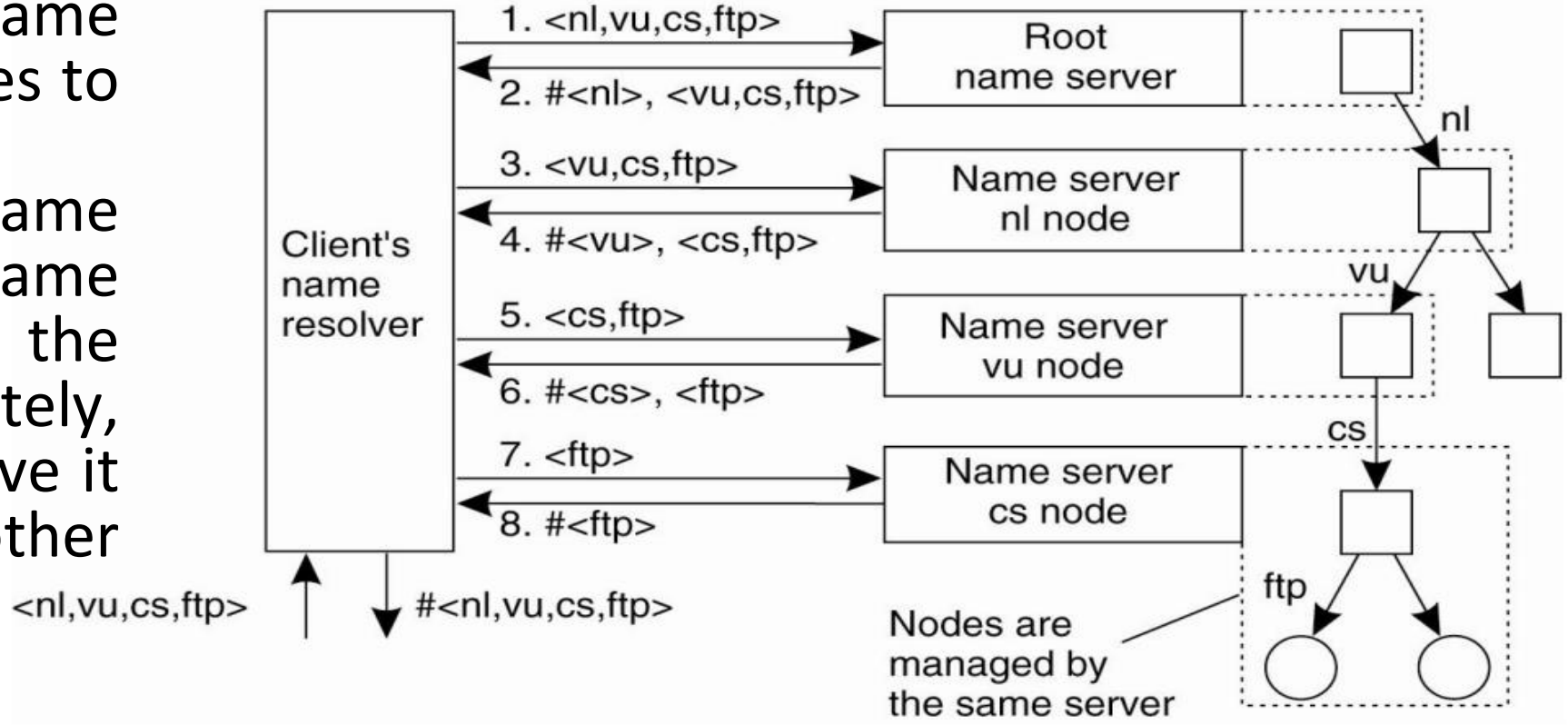
Item	Global	Administrational	Managerial
Geographical scale of network	Worldwide	Organization	Department
Total number of nodes	Few	Many	Vast numbers
Responsiveness to lookups	Seconds	Milliseconds	Immediate
Update propagation	Lazy	Immediate	Immediate
Number of replicas	Many	None or few	None
Is client-side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

Implementation of Name Resolution

- The distribution of a name space across multiple name servers affects the implementation of name resolution.
- To explain the implementation of name resolution in large-scale name services, we assume for the moment that name servers are not replicated and that no client-side caches are used.
- Each client has access to a **local name resolver**, which is responsible for ensuring that the name resolution process is carried out.
- Referring to the previous figure, assume the path name **root:[nl, vu, cs, ftp, pub, globe, index.html]** is to be resolved.
- Using a URL notation, this path name would correspond to **ftp: //ftp.cs.vu.nl/pub/globe/index.html**.
- There are now two ways to implement name resolution:
  1. Iterative Name Resolution
  2. Recursive Name Resolution

Iterative Name Resolution

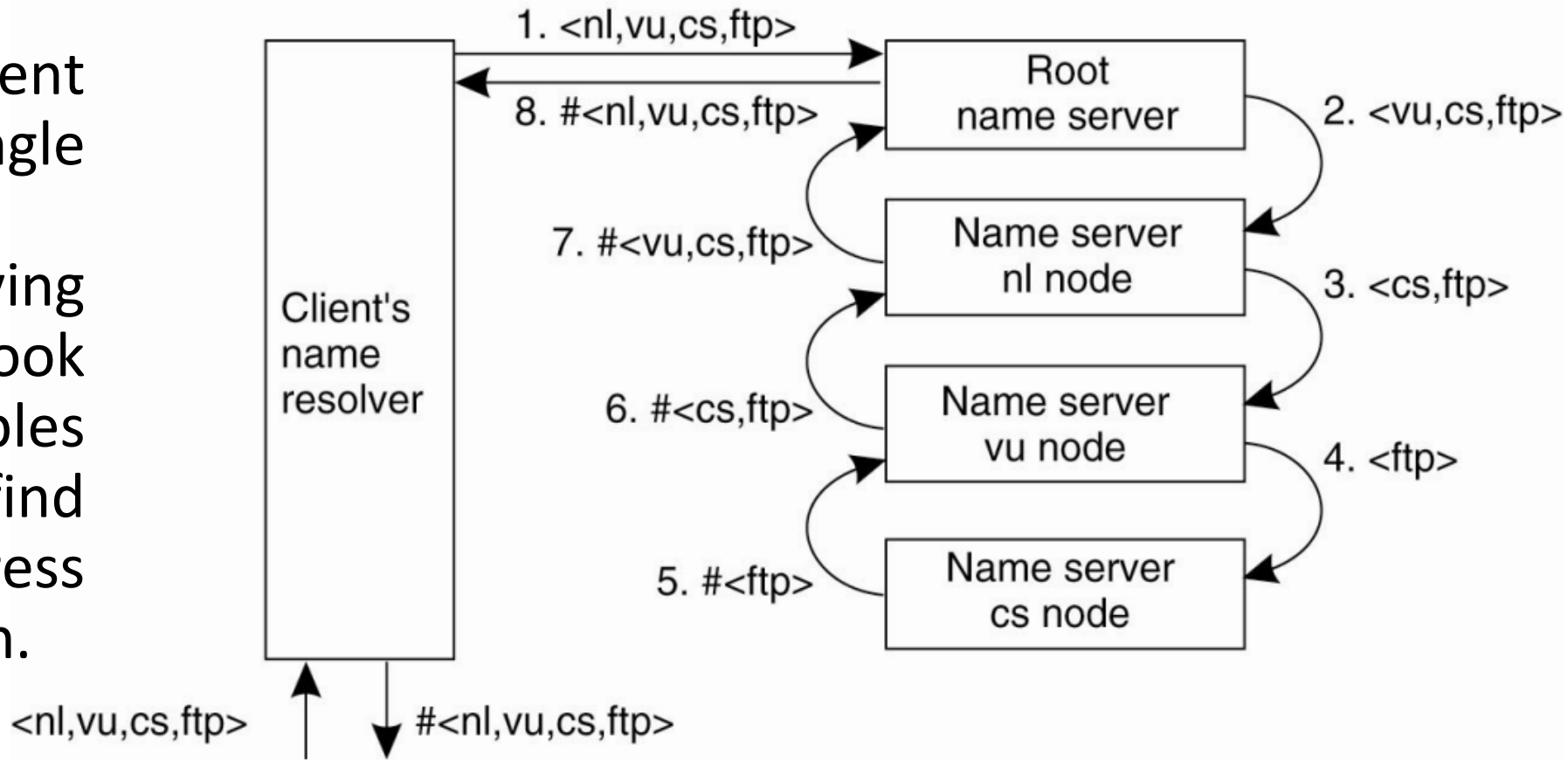
- The client will present a name to the local name server which tries to resolve it.
- If the local name server has the name it will return the result immediately, if it does not have it will suggest another server.



- DNS support navigation, resolution continuous until the name is resolved or name found to be unbound.

Recursive Name Resolution

- The client contacts a single server.
- On receiving query, it will look through it tables (caches) to find the IP address for the domain.

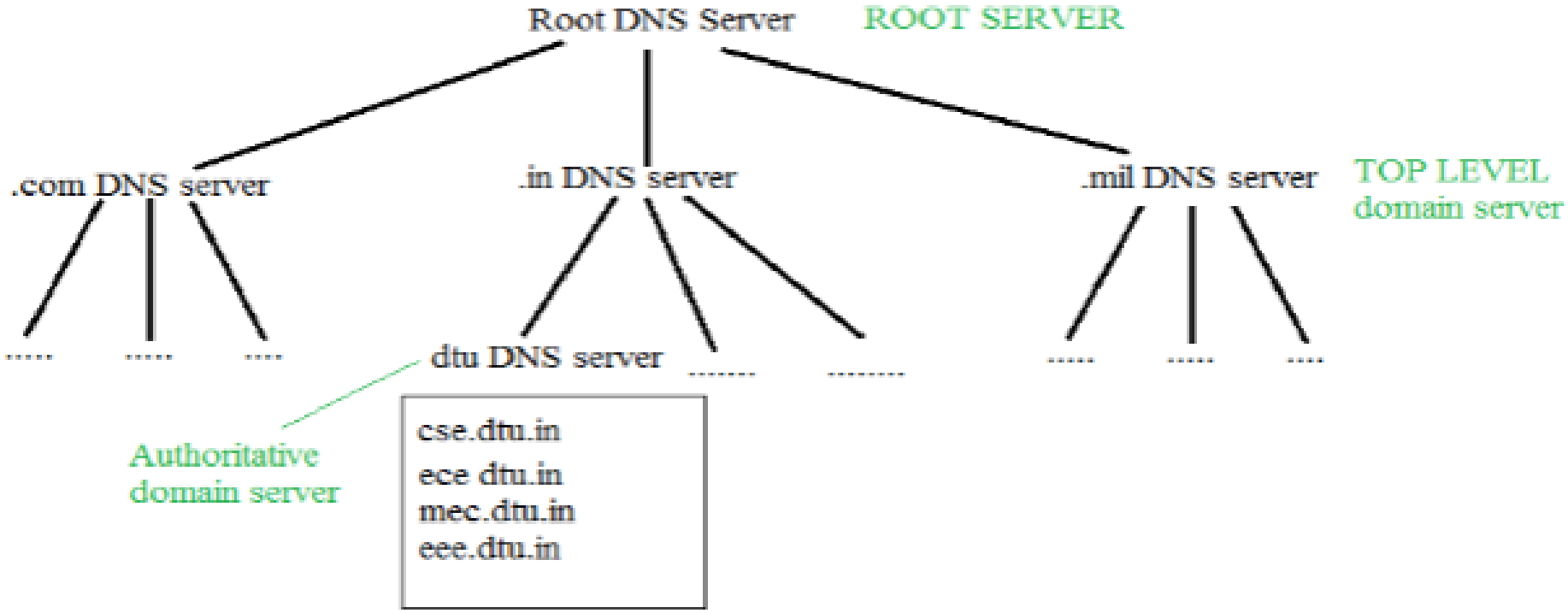


- If the server does not store the name, the server contacts a peer, which in turn attempts to resolve it.
- This procedure continues recursively until the name is resolved.

Domain Name Service

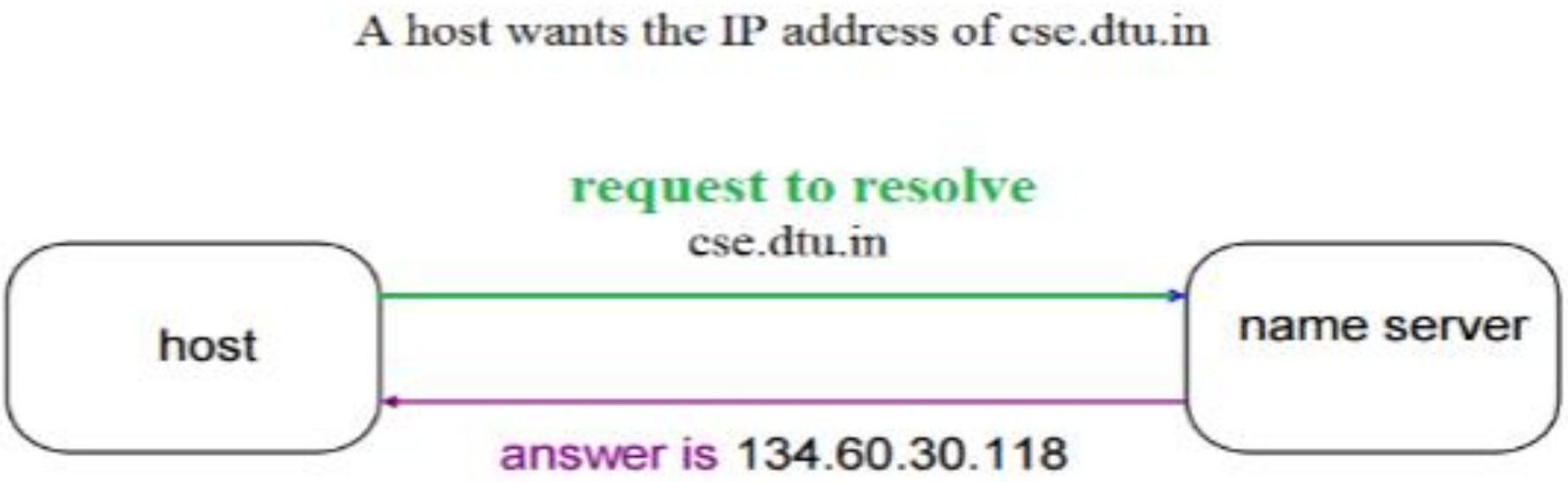
- One of the largest distributed system in use today.
- Used for mapping hostnames to IP addresses on the Internet.
- Uses an iterative lookup by default, with recursive lookup as an option.
- **Name Servers** are server programs which hold information about the domain tree's structure and set information.
- **Resolvers** are programs that extract information from name servers in response to client requests.
- Resolvers must be able to access at least one name server and use that name server's information to answer a query directly, or pursue the query using referrals to other name servers.

Organization of Domain



Name-to-Address Resolution

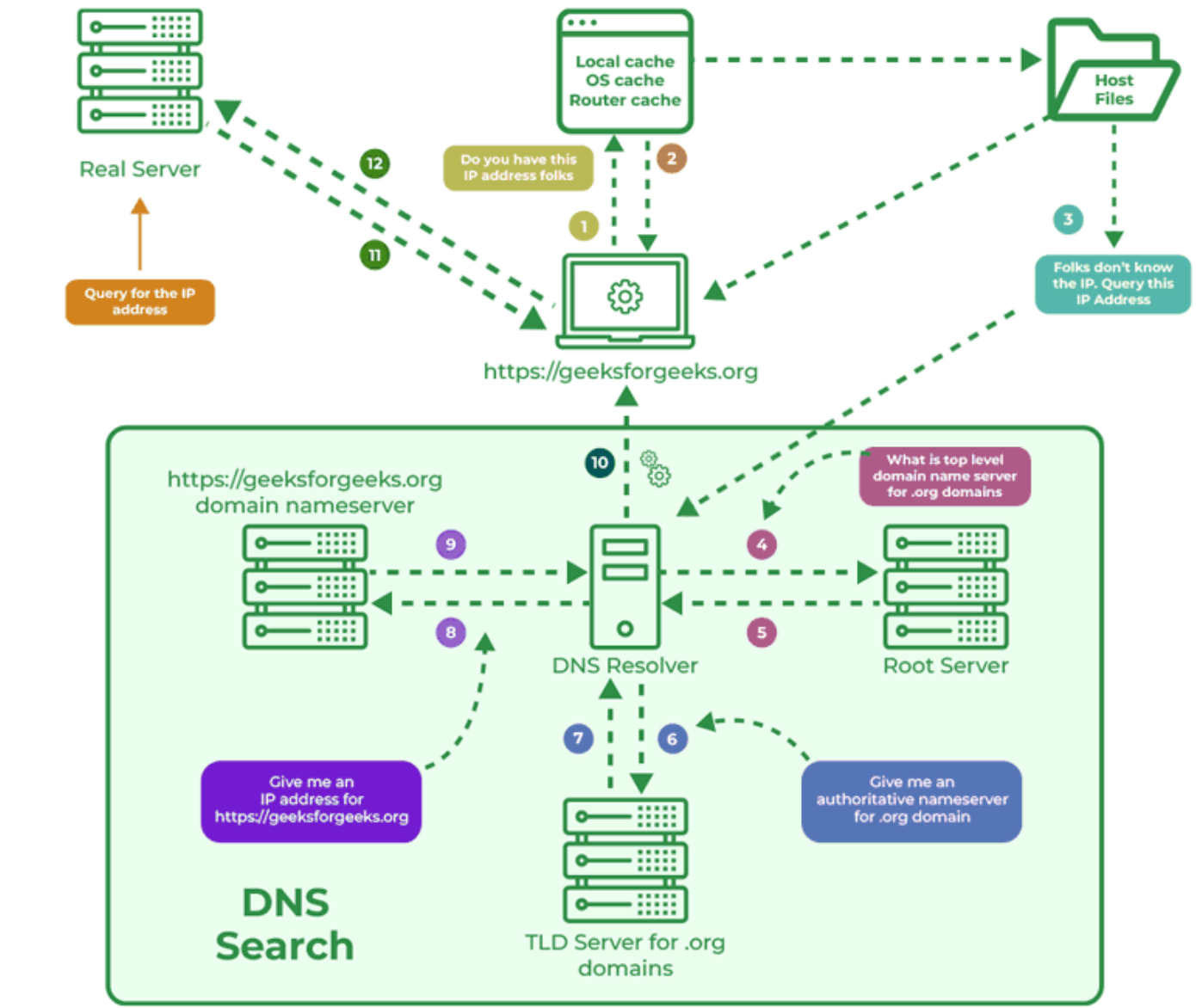
- The host requests the DNS name server to resolve the domain name. And the name server returns the IP address corresponding to that domain name to the host so that the host can connect to that address.



Continue…

- The client machine sends a request to the local name server (resolver), which, if the root does not find the address in its database, sends a request to the root name server, which in turn, will route the query to a top-level domain (TLD) or authoritative name server.

How Does DNS Works



Attribute-based Naming

- As more information becomes available, it becomes important to effectively search for entities.
- The user should be able to search based just on some attributes.
- Each entity has certain attributes. Each attribute says something about the entity.
- Users can search by constraining the attributes.
- Attribute-based naming systems are also know as directory services.
- More general model is using resource description framework (RDF).
- Resources are described as triplets consisting of a subject, predicate and an object. E.g. (person, name, Alice).

Hierarchical Implementations: LDAP

- LDAP (Lightweight Directory Access Protocol).
- A simplified protocol implementing the X.500 directory services.
- LDAP is an application level protocol, implemented directly on top of TCP.
- Used by email programs for contact information but can also be used to look for encryption certificates, pointers to printers or other services, sharing passwords between services etc.
- LDAP defines the protocol to be used between servers and clients or servers and servers.
- LDAP clients starts a LDAP session by contacting a LDAP server on default TCP port 389.

Thank You ☺