Chapter 3

Naming

3.1 Introduction

- Names are used to refer a wide variety of resources
  - Computers, remote objects and files as well as users
- Names facilitate communication and resource sharing
- Names are needed to
  - Request resource like website, shared files, exchange information through emails
  - Be descriptive
- Naming Facility: enables users and programs to assign character string names to objects and to refer them
- Locating facility: maps an object's name to the object's location
- The naming system helps to achieve the goals of: location transparency, facilitating transparent migration and replication objects and object sharing

3.2 Names, Identifiers and Addresses

Name

- Is a string of bits or characters that is used to refer to an entity. For example,
  - file names like */etc/passwd*,
  - URLs like *http://www.google.com/* and
  - Internet domain names like *www.google.com* etc.
- An entity can be: resources such as hosts, printers, disks, and files, explicit names such as processes, users, mailboxes, newsgroups, Web pages, graphical windows, messages, network connections, and so on.
- To operate on an entity, it is necessary to access it, for which we need an access point.

Address

- The name of an access point is called an address.
- Is a special kind of name(access point of an entity)
- An entity can offer more than one access point.
  - **Eg.** Website can have multiple ip addresses, a person can have multiple phone numbers (these access points can be changed in the course of time)
- It is tightly associated with an entity
  - In other words, an entity may easily change an access point, or an access point may be reassigned to a different entity (cause an invalid reference)
- A name for an entity that is independent from its addresses is often much easier and more flexible to use (such name is called location independent)

Identifiers

- An identifier is used to uniquely identify an entity.
- A true identifier has the following properties:
  - An identifier refers to at most one entity.
  - Each entity is referred by at most one identifier.
  - An identifier always refers to the same entity (i.e., it is never reused).
- A person's name can not be an identifier whereas Ethernet Address(physical Addresses) are identifiers
- Another type of names are human-friendly names (represented as a character string) such as files in Unix (255charaters), DNS names

3.3 Flat Naming

- refer to as unstructured names
- It does not contain any information on how to locate the access point of its associated entity.
- How do we locate an entity?
  - Simple solution
    - Broadcasting & Multicasting (applicable only in LAN)
  - Home Based Approach
  - Hierarchical Approach
  - Forwarding pointer
  - DHT: Chord System

3.3.1. Broadcasting

- Broadcast Domain
  - In LAN
  - In WLAN
- To locate entities in LAN ARP(Internet Address Resolution Protocol) is used when only the IP address of the entity is known
  - message containing the identifier of the entity is broadcast to each machine
  - each machine is requested to check whether it has that entity
  - Only the machines that can offer an access point for the entity send a reply message containing the address of that access point(Ethernet/MAC Address)

Drawbacks of broadcasting

- Inefficient when the network grows
- Network bandwidth wasted by request messages
- Too many hosts may be interrupted by requests they cannot answer

Solution:

- Using Multicasting
  - only a restricted group of hosts receives the request.

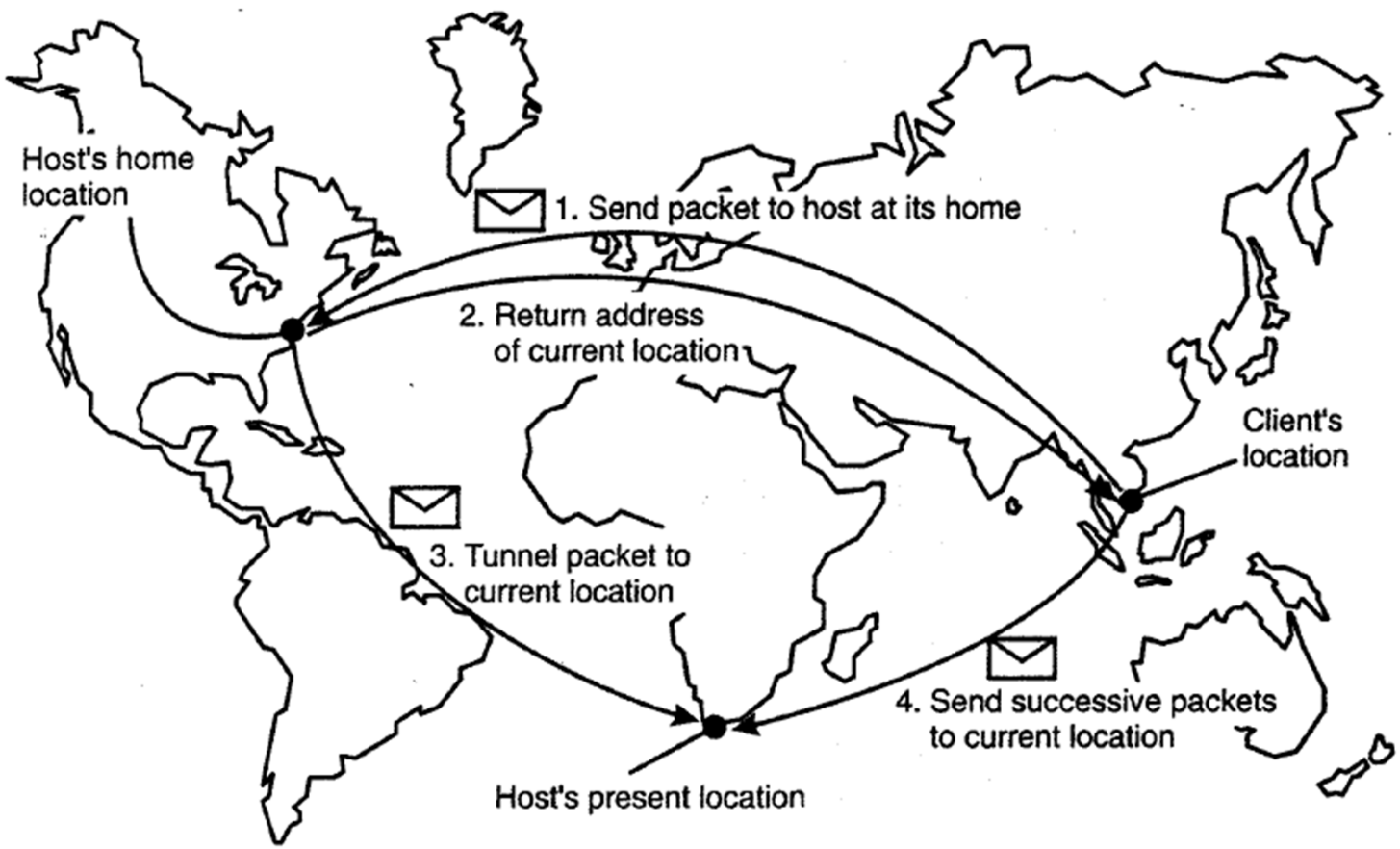
Multicasting

- Multicasting also used to locate entities in point-to-point networks.
  - **For example**, the Internet supports network-level multicasting by allowing hosts to join a specific multicast group - such groups are identified by a multicast address.
- When a host sends a message to a multicast address, the network layer provides a **best-effort service** to deliver that message to all group members.
- A multicast address can be used as a general location service for multiple entities.
  - For example, consider an organization where each employee has his or her own mobile computer.
  - ✓ When such a computer connects to the locally available network, it is dynamically assigned an IP address. In addition, it joins a specific multicast group.
  - ✓ When a process wants to locate computer A, it sends a "where is A?" request to the multicast group. If A is connected, it responds with its current IP address.

- Another way to use a multicast address is to associate it with a replicated entity, and to use multicasting to locate the nearest replica.
- When sending a request to the multicast address, each replica responds with its current (normal) IP address.
  - ✓A crude way to select the nearest replica is to choose the one whose reply comes in first.

3.3.2. Home-Based Approaches

- Broadcasting and Multicasting are not efficient in large scale networks
- This approach supports in locating mobile entities in large-scale networks by introduce a home location - which keeps track of the current location of an entity.
- In practice, the home location is often chosen to be the place where an entity was created.
- All communication to the Entity's IP address is initially directed to the mobile host's home agent.
- This home agent is located on the local-area network corresponding to the network address contained in the mobile host's IP address - a network-layer component.
- Whenever the mobile host moves to another network, it requests a temporary address that it can use for communication - this Care-of Address is registered at the home agent.



- Mobile IP is one example where it followed the home-based approach: Each mobile host uses a fixed IP address.

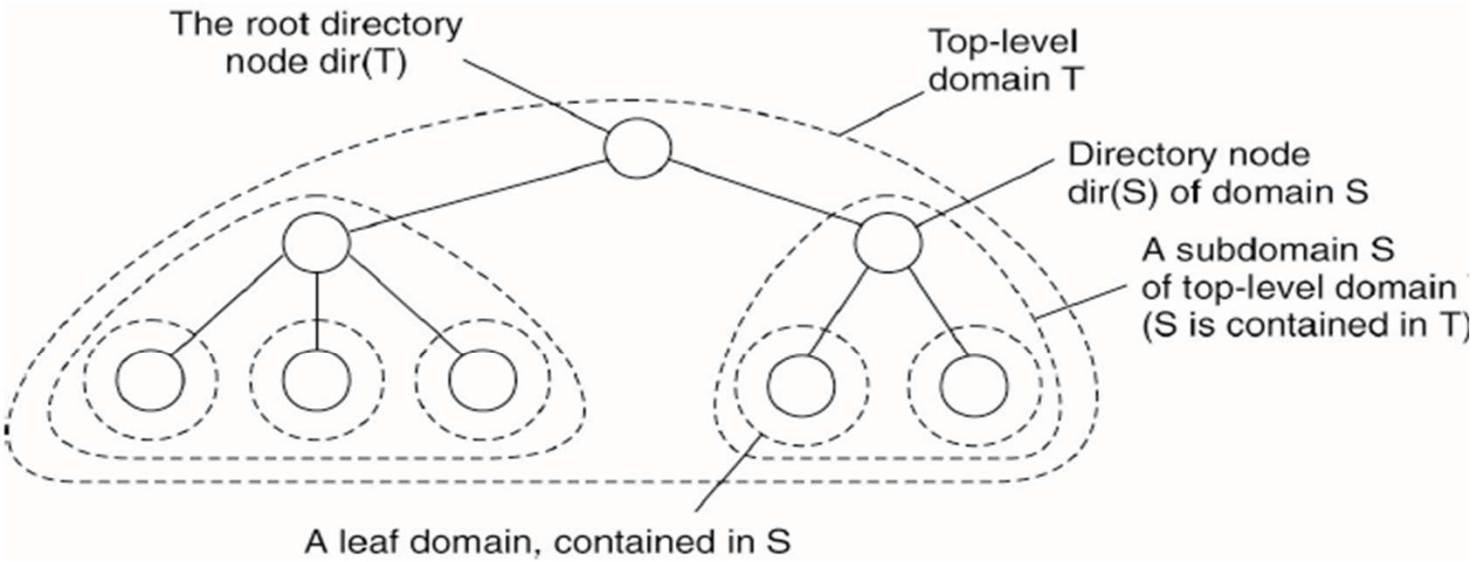
- The previous figure shows this principle: When the home agent receives a packet for the mobile host;
  - ✓ it looks up the host's current location
  - ✓ If the host is on the current local network, the packet is simply forwarded.
  - ✓ Otherwise, it is tunneled to the host's current location, that is, wrapped as data in an IP packet and sent to the care-of address.
  - ✓ At the same time, the sender of the packet is informed of the host's current location.

❖ Note that **the IP address** is effectively used as **an identifier** for the **mobile host.**

- Generally, the issues with Home-Based Approaches are
  - ✓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)

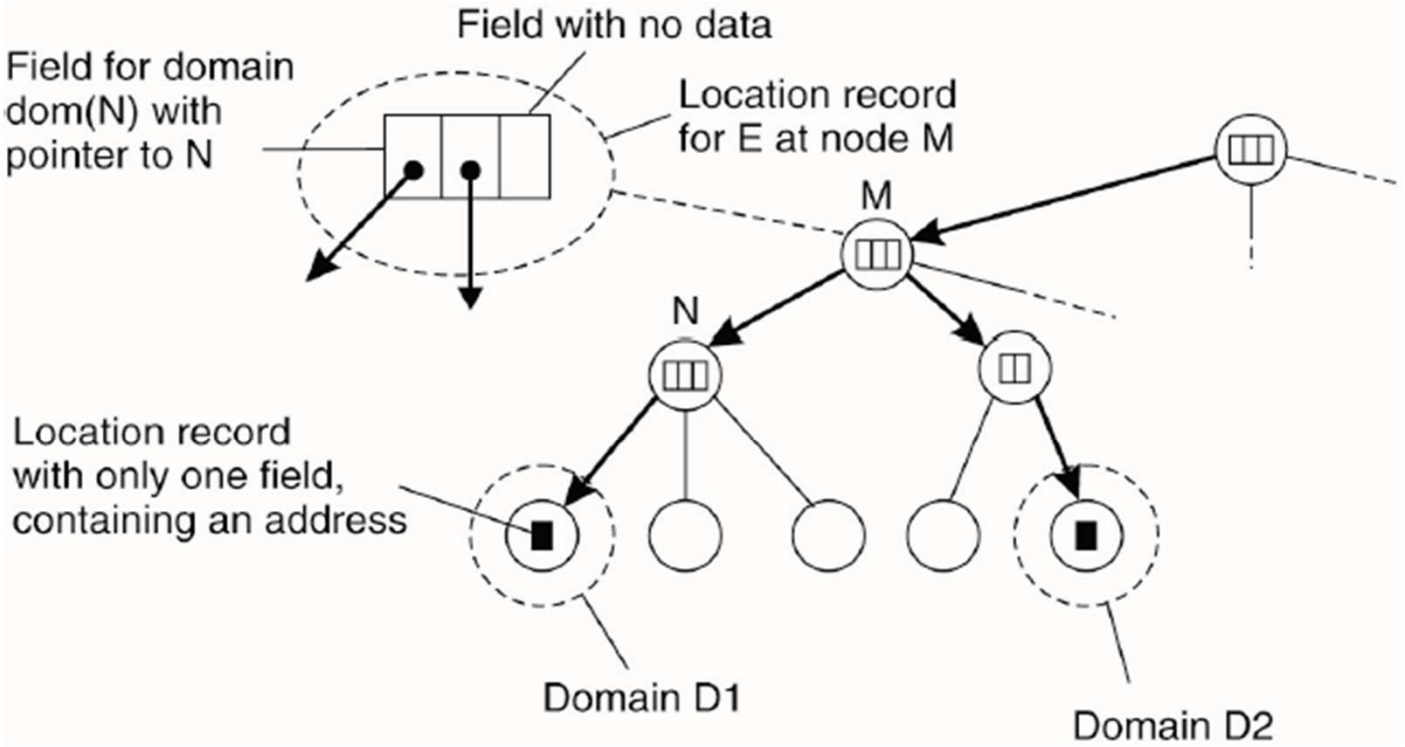
3.3.3 Hierarchical Approach

- In this scheme, a network is divided into a collection of domains
- There is a single top-level domain i.e. the root(directory) node
- Each domain can be subdivided into multiple, smaller sub domains. A lowest-level domain is called leaf domain. (typically corresponds to LAN or cell in a wireless network)
- Each domain D has an associated directory node dir(D) that keeps track of the entities in that domain.



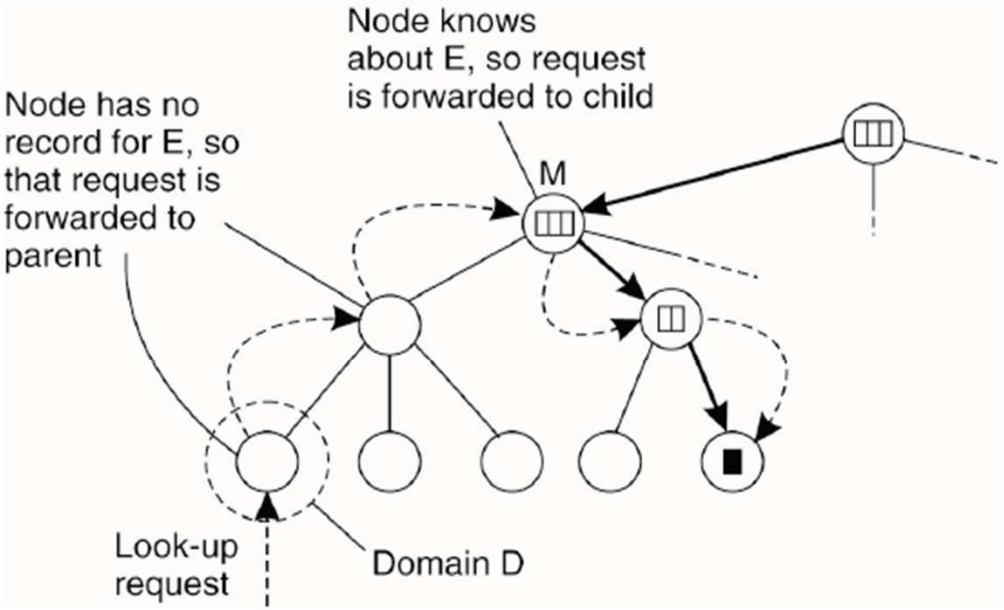
Hierarchical organization of a location service into domains, each having an associated directory node.

- A location record for entity *E* in the directory node *dir(D)* contains the entity's current address in that domain.
- In contrast, the directory node *dir(D)* for the next higher-level domain *D'* that contains *D,* will have a location record for *E* containing only a pointer to *N*
- The root will have a location record for each entity, where each location record stores a pointer to the directory node of the next lower-level sub domain



An example of storing information of an entity having two addresses in different leaf domains.

An entity may have multiple addresses, for example if it is replicated. If an entity has an address in leaf domain *D* 1 and *D2* respectively, then the directory node of the smallest domain containing both *D* 1 and *D* 2, will have two pointers, one for each sub domain containing an address.



Looking up a location in a hierarchically organized location service.

✓A client wishing to locate an entity *E,* issues a lookup request to the directory node of the leaf domain D

✓If the directory node does not store a location record for the entity, then the node forwards the request to its parent

✓If the parent also has no location record for *E,* the lookup request is forwarded to a next higher level

3.3.4 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
  - its simplicity: a client can look up the current address by following the chain of forwarding pointers.
- There are also drawbacks.
  - If no special measures are taken, a chain for a high mobile entity can become so long that locating that entity is prohibitively expensive.
  - All intermediate locations in a chain will have to maintain their part of the chain of forwarding pointers as long as needed.
  - Vulnerability to broken links. As soon as any forwarding pointer is lost, the entity can no longer be reached.

3.3.5 DHT: Chord System

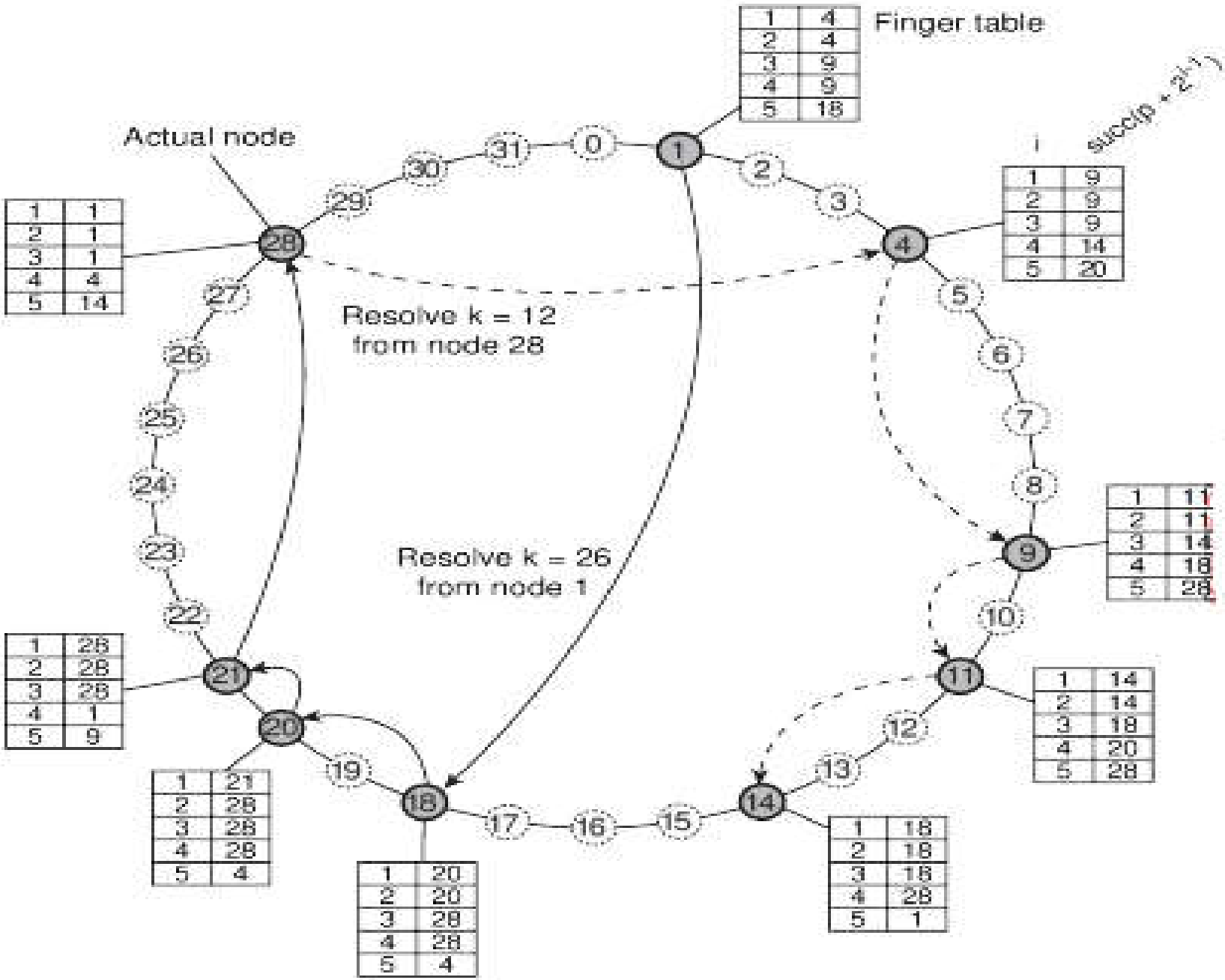
- Chord uses an m-bit identifier space to assign randomly chosen identifiers to nodes as well as keys to specific entities.
- The number m of bits is usually 128 or 160; depending on the hash function is used.
- An entity with key *k* falls under the jurisdiction of the node with the smallest identifier id >= *k*. This node is referred to as the successor of k and denoted as *succ(k)*.
- The main issue in DHT-based systems is to efficiently resolve a key *k* to the address of *succ*(k).
  - **non-scalable approach**
    - Let each node p keep track of the successor *succ*(p+1) as well as its predecessor *pre*(p).
    - In that case, whenever a node p receives a request to resolve key k, it will simply forward the request to one of its two neighbors-unless *pred*(p)<*k*≤p in which case node p should return its own address to the process that initiated the resolution of the key k.

- Efficient approach
  - Chord node maintains a finger table containing s ≤ m entities.
  - If FT_p denotes the finger table of node p, then

$$FT_p[i]=succ(p+2^{i-1}) \bmod (128 \text{ or } 160)$$

  - To lookup a key *k*, the node p will then immediately forward the request to node q with index j in the p's finger table where:
    - q= FTp[j] ≤ k < FT_p[j+1] (forward the request to p successor FTp[j])

      or
    - q= FTp[1] where p < k < FTp[1].(1st address will be returned)


Finger table
Actual node
succ(p + 2^(i-1))
Resolve k = 12
from node 28
Resolve k = 26
from node 1

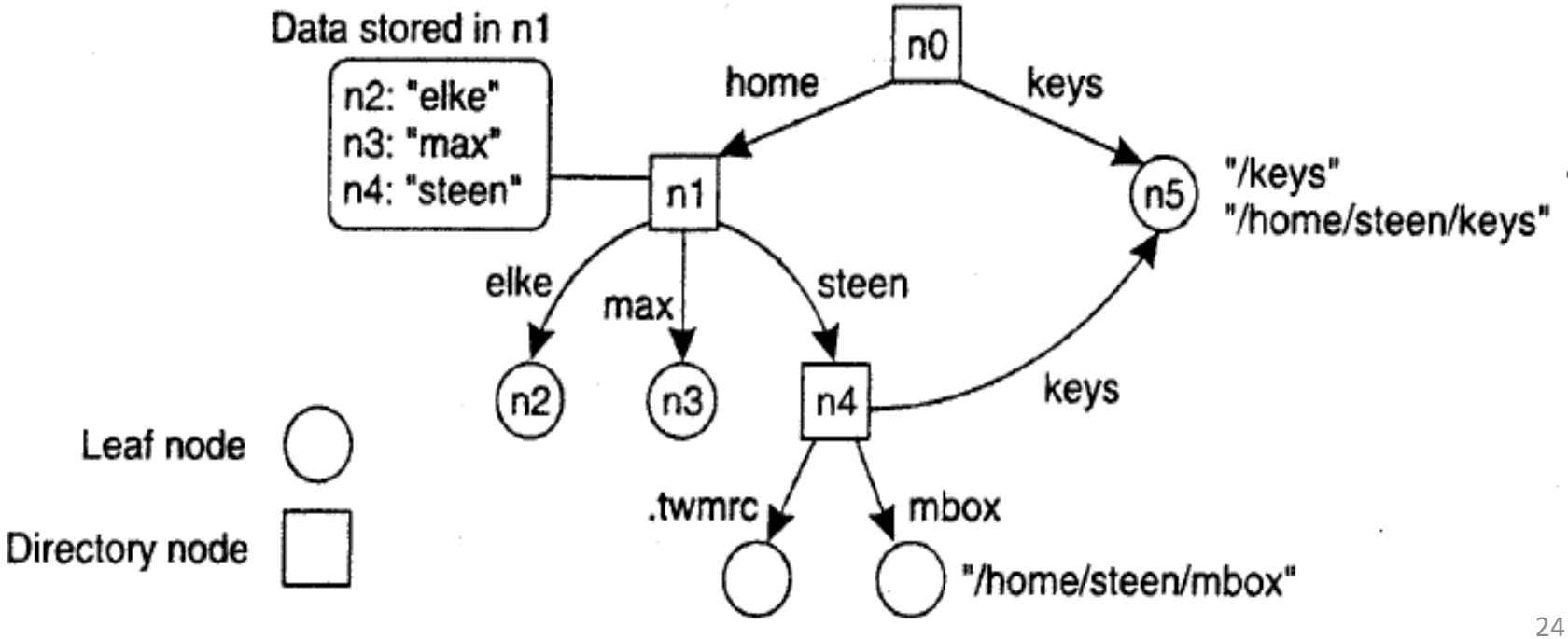
3.4 Structured Naming

- Flat names are good for machines, but are generally not very convenient for humans to use.
- Structured names are composed from simple, human-readable names.
- Used in file naming and host naming on the internet

Name Space

- Names are commonly organized in a **name space.**
- Name spaces for structured names can be represented as a labeled, directed graph with two types of nodes
  - A **leaf node** represents a named entity and has no outgoing edges. It stores information on the entity it is representing. For example, its address-so that a client can access it.
  - A **directory node** has a number of outgoing edges, each labeled with a name. Each node in a naming graph is considered as yet another entity in a distributed system and has an associated identifier.

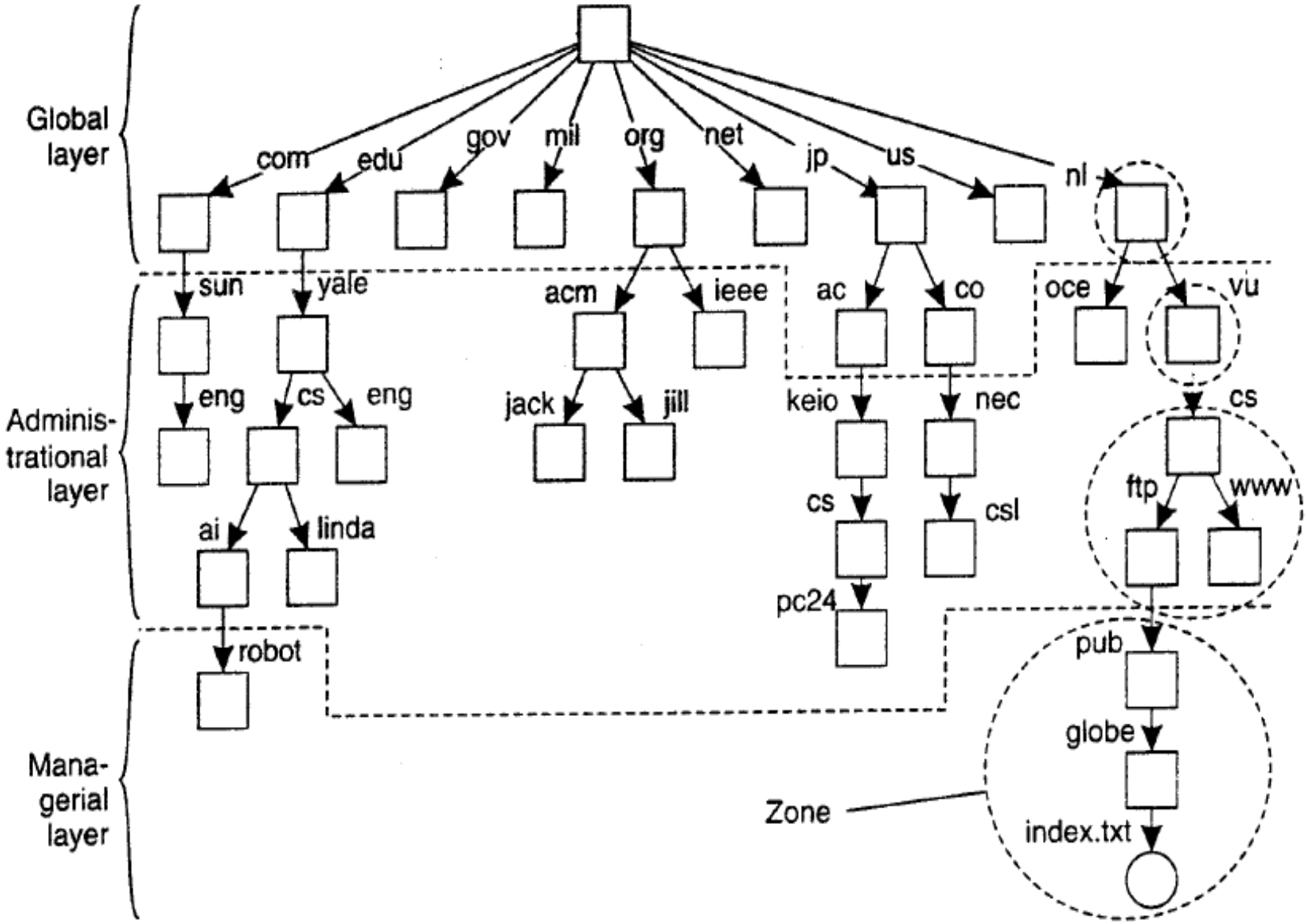
- A directory node stores a table in which an outgoing edge is represented as a pair *(edge label, node identifier).*
    - Such a table is called a directory table.
- A directory node that has only outgoing and no incoming edges is called the root node of the naming graph Eg. *n0*
- *A naming graph can have several root node,* for simplicity, many naming systems have only one



- Each path in a naming graph can be referred to by the sequence of labels corresponding to the edges in that path such as *N:<label-1, label-2... label-n>* where *N* refers to the first node in the path. Such a sequence is called a path name.
  - Ex. *n0:<home, steen, mbox>* is the actual path name, it is common practice to use its string representation */home/steen/mbox.*
- If the first node in a path name is the root of the naming graph, it is called an absolute path name. Otherwise, it is called a relative path name.
- A global name is a name that denotes the same entity, no matter where that name is used in a system. It is always interpreted with respect to the same directory node.
- In contrast, a local name is a name whose interpretation depends on where that name is being used. A local name is essentially a relative name whose directory in which it is contained is (implicitly) known.
- Each node also has exactly one associated (absolute) path name.

Name Space Distribution

- Name spaces for a large-scale, possibly worldwide distributed system, are usually organized hierarchically.
- To effectively implement such a name space, it is convenient to partition it into three logical layers.
  - The *global layer* formed by highest-level nodes, that is, the root node and other directory nodes logically close to the root, namely its children.
    - They are stable, i.e. 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.
    - 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
    - they are relatively stable, although changes occur more frequently than to nodes in the global layer.
  - The managerial layer consists of nodes that may typically change regularly. For example, nodes representing hosts in the local network
    - includes nodes representing shared files
    - maintained not only by system administrators, but also by individual end users of the DS

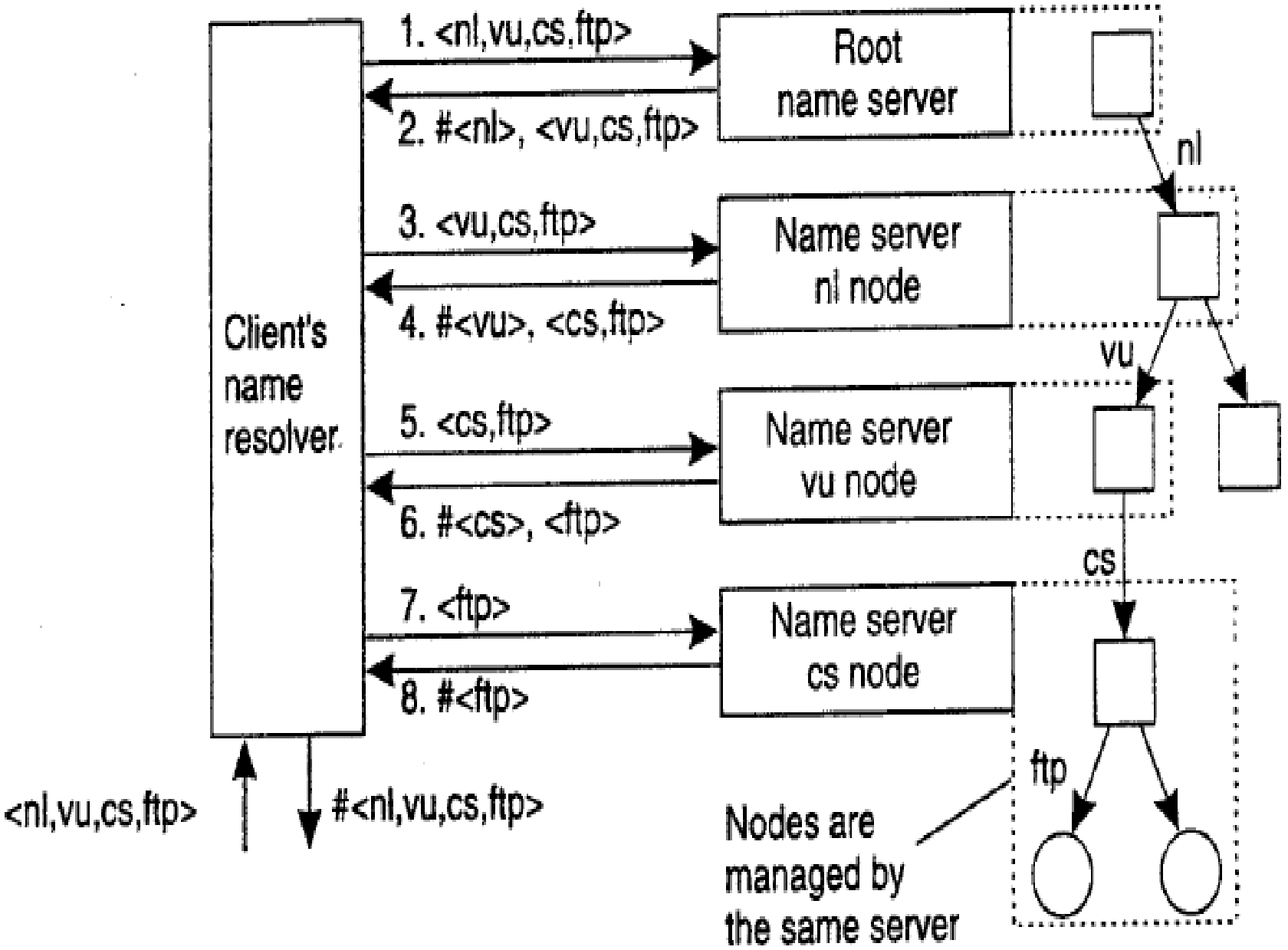

Global layer
Adminis-trational layer
Mana-gerial layer
com
edu
gov
mil
org
net
jp
us
nl
sun
yale
acm
ieee
ac
co
oce
vu
eng
cs
eng
jack
jill
keio
nec
cs
ftp
www
ai
linda
cs
csl
pc24
robot
pub
globe
index.txt
Zone

Name Resolution

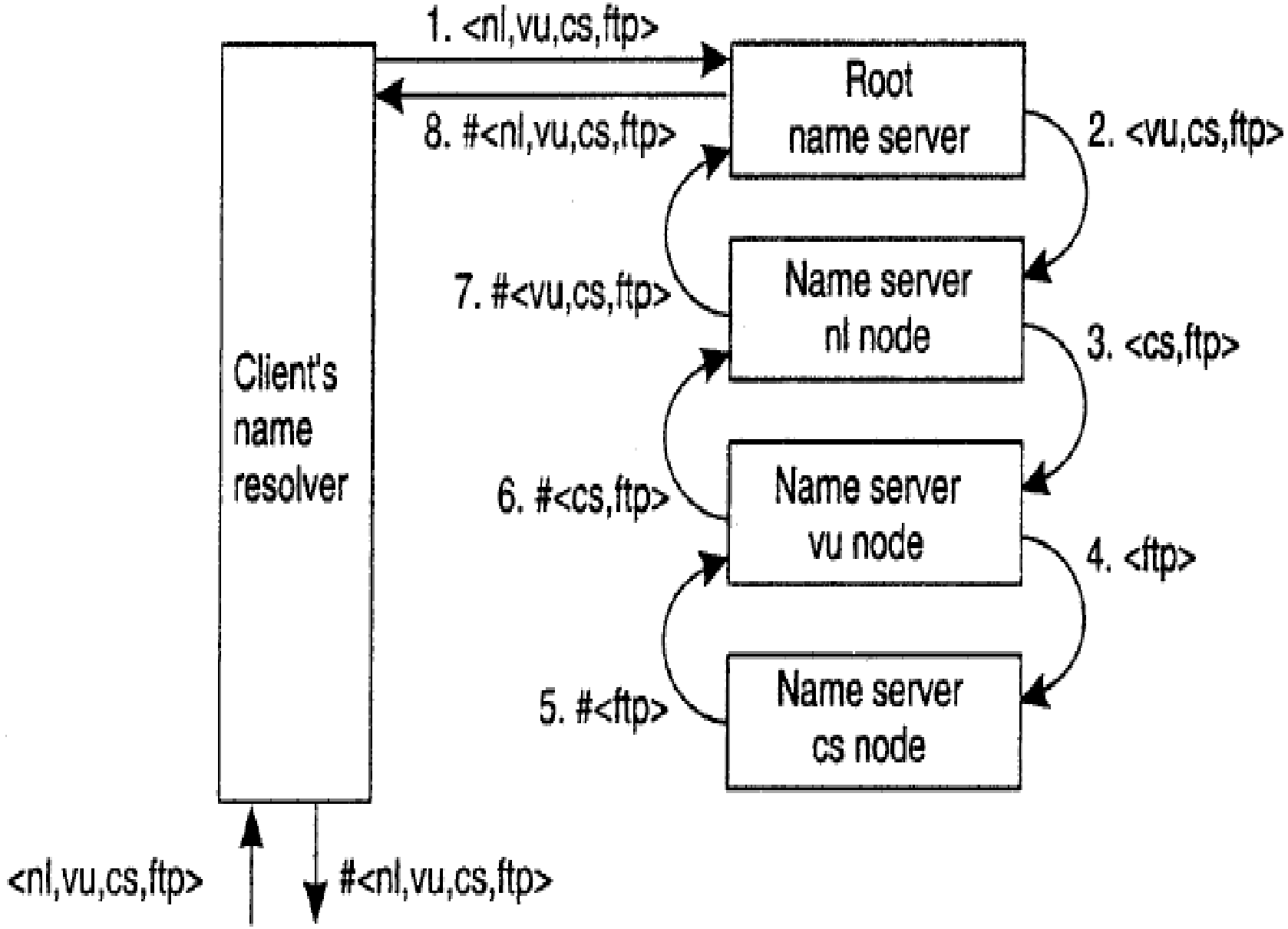
- It is a process of mapping an object's name to the object's properties, such as its location. (Since an object's properties are stored and maintained by the authoritative name servers of that object)
- Once an authoritative name server of the object has been located, operations can be invoked to read or update the object's properties.
- Each name agent in a distributed system knows about at least one name server a prior.
- 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.

Implementation of Name Resolution

- There are two ways to implement name resolution.
  - *iterative name resolution*
  - *recursive name resolution*
- *Iterative name resolution*
  - a name resolver hands over the complete name to the root name server.
  - The root server will resolve the path name as far as it can, and return the result to the client. Ex. Assume the absolute path name: *root: <nl, VU, CS, ftp, pub, globe, index.html>* is to be resolved. URL: *ftp://ftp.cs.vu.nl/pub/globe/index.html*
  - the client would normally hand only the path name *root: <nl, VU, CS, ftp>* to the name resolver
  - In practice, the last step, namely contacting the FTP server and requesting it to transfer the file with path name *ftp:<pub, globe, index.himl>,* is carried out separately by the client process.


1. <nl,vu,cs,ftp>
2. #<nl>, <vu,cs,ftp>
3. <vu,cs,ftp>
4. #<vu>, <cs,ftp>
5. <cs,ftp>
6. #<cs>, <ftp>
7. <ftp>
8. #<ftp>
Client's name resolver
Root name server
Name server nl node
Name server vu node
Name server cs node
nl
vu
cs
ftp
<nl,vu,cs,ftp>
#<nl,vu,cs,ftp>
Nodes are managed by the same server

- Recursive Name *resolution*
  - Instead of returning each intermediate result back to the client's name resolver, with recursive name resolution, a name server passes the result to the next name server it finds.
  - contacting the FTP server and asking it to transfer the indicated file is generally carried out as a separate process by the client
  - The main drawback of recursive name resolution is that it puts a higher performance demand on each name server.
  - advantages to recursive name resolution
    - caching results is more effective compared to iterative name resolution.
    - communication costs may be reduced.


1. <nl,vu,cs,ftp>
8. #<nl,vu,cs,ftp>
Root
name server
2. <vu,cs,ftp>
7. #<vu,cs,ftp>
Name server
nl node
3. <cs,ftp>
6. #<cs,ftp>
Name server
vu node
4. <ftp>
5. #<ftp>
Name server
cs node
Client's
name
resolver
<nl,vu,cs,ftp>
#<nl,vu,cs,ftp>

Name Services

- A *name service* stores information about a collection of textual names, in the form of bindings between the names and the attributes of the entities they denote, such as users, computers, services and objects.
- The major operation that a name service supports is to resolve a name
- The common name services are
  - Grapevine
  - Global Name service
  - The more familiar is internet DNS

DNS

- is a name service design whose main naming database is used across the Internet.
- The objects named by the DNS are primarily computers – for which mainly IP addresses are stored as attributes
- provides a way for hosts to use their name to request the IP address of a specific server
- It made locating servers easier by associating a name with an IP address
- The Internet DNS name space is partitioned both organizationally and according to geography.

- The names are written with the highest-level domain on the right.
- The original top-level organizational domains (also called *generic domains*) in use across the Internet are:
  - *com* – Commercial organizations
  - *edu* – Universities and other educational institutions
  - *gov* – governmental agencies
  - *mil* – military organizations
  - *net* – Major network support centres
  - *org* – Organizations not mentioned above
  - *int* – International organizations
  - *biz* and *mobi are the new top level domains*

- In addition, every country has its own domains:
  - *us* – United States
  - *uk* – United Kingdom
  - *fr* – France
  - *et* – Ethiopia etc.
- A DNS server contains a table that associates hostnames in a domain with corresponding IP addresses.
- When a client has the name of server, such as a web server, but needs to find the IP address, it sends a request to the DNS server on port 53. The client uses the IP address of the DNS server configured in the DNS settings of the host's IP configuration.
- When the DNS server receives the request, it checks its table to determine the IP address associated with that web server.
- If the local DNS server does not have an entry for the requested name, it queries another DNS server within the domain.
- When the DNS server learns the IP address, that information is sent back to the client.
- If the DNS server cannot determine the IP address, the request will time out and the client will not be able to communicate with the web server.

3.5 Attribute-Based Naming

- This approach requires that a user can provide merely a description of what he/she is looking for.
- Describing an entity in terms of (attribute, value) pairs in distributed system, generally referred to as attribute-based naming.
- In this approach, an entity is assumed to have an associated collection of attributes. Each attribute says something about that entity.
- By specifying which values a specific attribute should have, a user essentially constrains the set of entities that he is interested in.
- It is up to the naming system to return one or more entities that meet the user's description.
- Attribute-based naming systems are also known as directory services, whereas systems that support structured naming are generally called naming systems.
- With directory services, entities have a set of associated attributes that can be used for searching.
- In some cases, the choice of attributes can be relatively simple. For example, in an e-mail system, message can be tagged with attributes for the sender, recipient, subject and so on.