

Naming

Chapter 4

Overview

- Names are used to share resources, to uniquely identify resources, to refer to locations, to communicate, etc.
- Important issue with naming is that a name can be resolved to the entity it refers to
- A naming system is used to resolve names
- In distributed systems the naming system is distributed across multiple machines and this distribution impacts the efficiency and scalability of the naming system
- Key topics covered:
 - General issues with respect to naming – Naming Entities
 - Locating mobile entities
 - Removing unreferenced entities

7/7/2005

2

Names, Identifiers, and Addresses

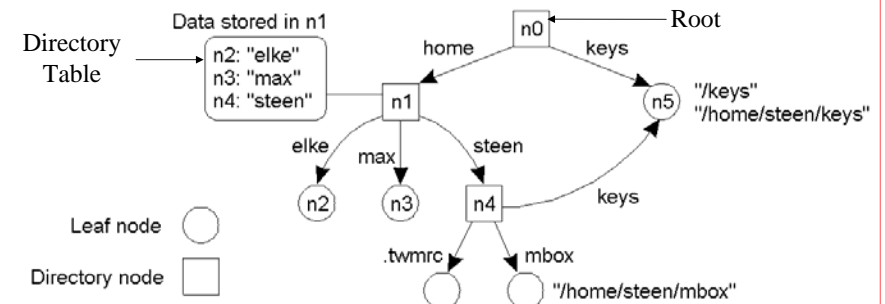
- Names are used to denote entities in a distributed system. To operate on an entity, we need to access it at an **access point**. Access points are entities that are named by means of an **address**.
- A **location-independent name** for an entity E, is independent from the addresses of the access points offered by E.
- **Pure name**: A name that has no meaning at all; it is just a random string. Pure names can be used for comparison only.
- **Identifier**: A name having the following properties:
 - Each 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 (prohibits reusing an identifier)
- An identifier need not necessarily be a pure name, i.e., it may have content.
- Identifiers make it easier to unambiguously refer to an entity.

7/7/2005

3

Name Spaces

- Names in a distributed system are organized into a **name space**.
- A name space can be represented as a labeled, directed graph with two types of nodes:
 - a leaf node represents a (named) entity, no outgoing edges, stores entity information, state, or attributes
 - a directory node is an entity that refers to other nodes, has a number of outgoing edges, each labeled with a name



7/7/2005

4

Name Resolution

- The process of looking up a name is called name resolution
- Closure mechanism: The mechanism to select the implicit context from which to start name resolution:
 - cis.uab.edu – start at a DNS name server
 - /home/puri/Cal.java – start at the local NFS file server
 - 138.26.64.5 – route to CIS web server
- A closure mechanism may also determine how name resolution should proceed

7/7/2005

5

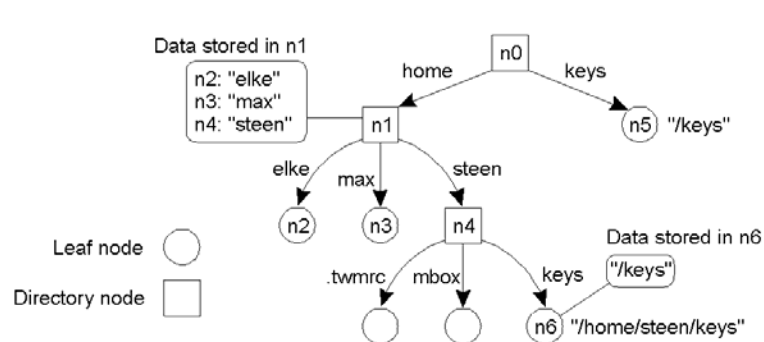
Name Linking

- Hard link: What we have described so far as a path name: a name that is resolved by following a specific path in a naming graph from one node to another.
- Soft link: Allow a node O to contain a name of another node:
 - First resolve O's name (leading to O)
 - Read the content of O, yielding *name*
 - Name resolution continues with *name*
- Observations:
 - The name resolution process determines that we read the content of a node, in particular, the name in the other node that we need to go to.
 - One way or the other, we know where and how to start name resolution given *name*

7/7/2005

6

Symbolic Link in a Naming Graph



7/7/2005

7

Merging Name Spaces (1)

- We have different name spaces that we wish to access from any given name space, name resolution can be used to merge different name spaces in a transparent way
- **Solution 1:** Introduce a naming scheme by which pathnames of different name spaces are simply concatenated (URLs).

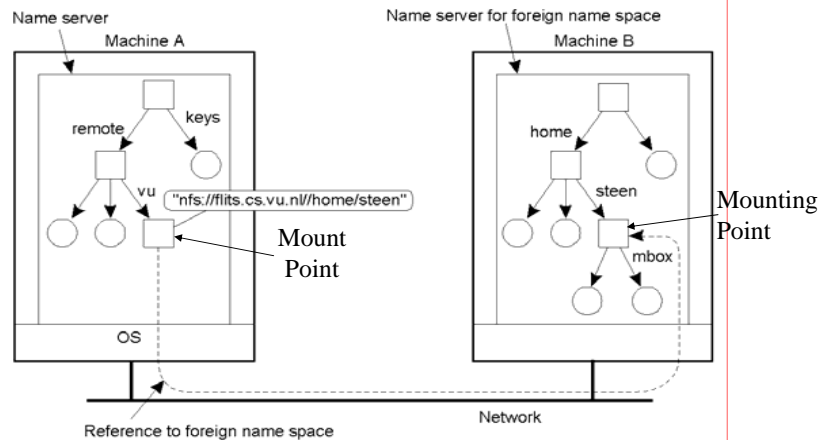
ftp://ftp.cis.uab.edu/pub/puri	
ftp	Name of protocol used to talk with server
://	Name space delimiter
ftp.cis.uab.edu	Name of a node representing an FTP server, and containing the IP address of that server
/	Name space delimiter
pub/puri	Name of a (context) node in the name space rooted at the context node mapped to the FTP server

7/7/2005

8

Merging Name Spaces (2)

Solution 2: Introduce nodes that contain the name of a node in a “foreign” name space, along with the information how to select the initial context in that foreign name space

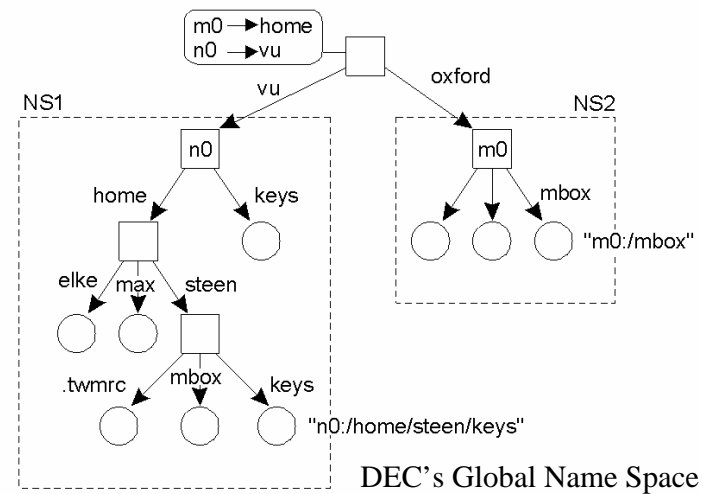


7/7/2005

9

Merging Name Spaces (3)

Solution 3: Use only *full pathnames*, in which the starting context is explicitly identified, and merge by adding a new root node



7/7/2005

10

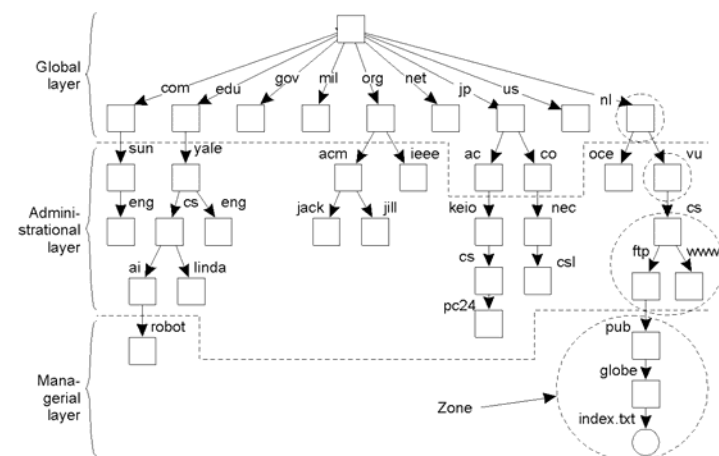
Name Space Implementation

- Basic issue: Distribute the name resolution process as well as name space management across multiple machines, by distributing nodes of the naming graph.
- Consider a hierarchical naming graph and distinguish three levels:
 - Global level: Consists of the high-level directory nodes. Main aspect is that these directory nodes have to be jointly managed by different administrations
 - Administrational level: Contains mid-level directory nodes that can be grouped in such a way that each group can be assigned to a separate administration.
 - Managerial level: Consists of low-level directory nodes within a single administration. Main issue is effectively mapping directory nodes to local name servers.

7/7/2005

11

Name Space Distribution (1)



An example partitioning of the DNS name space, including Internet-accessible files, into three layers.

7/7/2005

12

Name Space Distribution (2)

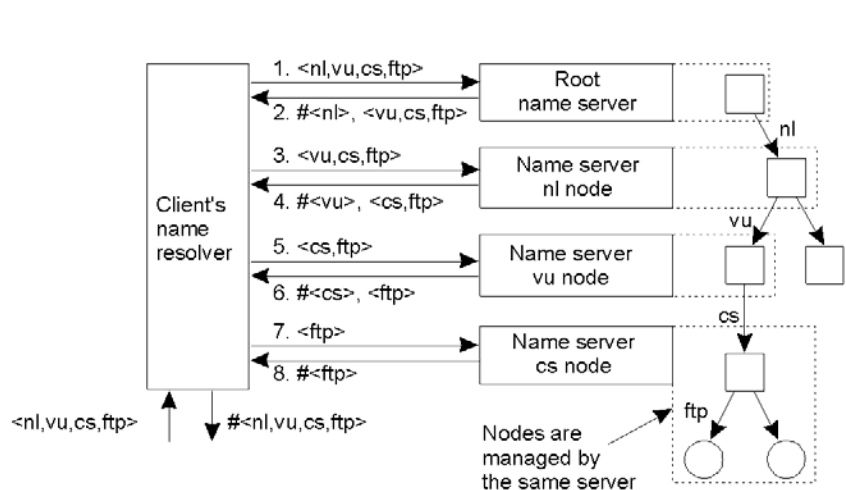
Item	Global	Administrational	Managerial
Geographical scale of network	Worldwide	Organization	Department
Total number of nodes	Few	Many	Vast numbers
Responsiveness to lookups	Seconds	Milliseconds	Immediate
Update propagation	Lazy	Immediate	Immediate
Number of replicas	Many	None or few	None
Is client-side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

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

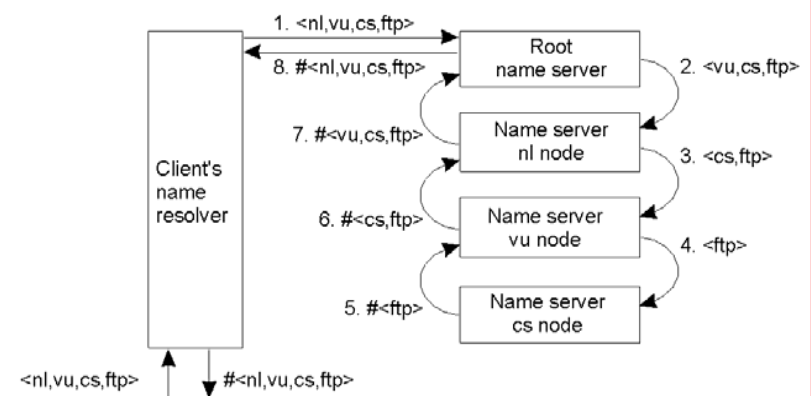
Implementing Name Resolution

- Distribution of name space across multiple name servers affects the implementation of name resolution
- There are two ways to implement name resolution:
 - Iterative name resolution
 - Recursive name resolution

Iterative Name Resolution



Recursive Name Resolution



Advantages: (1) better caching of results (2) reduced communication costs

Main drawback: higher performance demand on each name server

Overview of Recursive Resolution

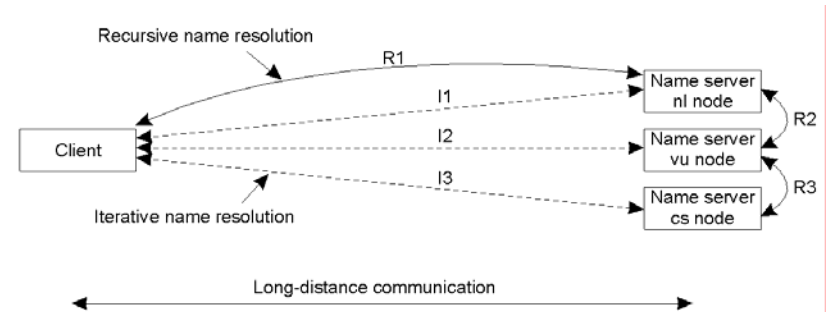
Server for node	Should resolve	Looks up	Passes to child	Receives and caches	Returns to requester
cs	<ftp>	#<ftp>	--	--	#<ftp>
vu	<cs,ftp>	#<cs>	<ftp>	#<ftp>	#<cs> #<cs, ftp>
ni	<vu,cs,ftp>	#<vu>	<cs,ftp>	#<cs> #<cs,ftp>	#<vu> #<vu,cs> #<vu,cs,ftp>
root	<ni,vu,cs,ftp>	#<nl>	<vu,cs,ftp>	#<vu> #<vu,cs> #<vu,cs,ftp>	#<nl> #<nl,vu> #<nl,vu,cs> #<nl,vu,cs,ftp>

Recursive name resolution of <nl, vu, cs, ftp>. Name servers cache intermediate results for subsequent lookups.

7/7/2005

17

Comparison of Communication Costs



The comparison between recursive and iterative name resolution with respect to communication costs.

7/7/2005

18

Examples

- The Domain Name System (DNS)
 - One of the largest distributed naming services in use
 - Primarily used for looking up host addresses and mail servers
 - Comparable to a telephone book for looking up phone numbers (traditional naming service)
- X.500
 - Based on directory services, a directory service allows an entity to be looked up based on a description of properties, instead of a full name
 - Similar to using “yellow pages”

7/7/2005

19

The DNS Name Space

Type of record	Associated entity	Description
SOA	Zone	Holds information on the represented zone
A	Host	Contains an IP address of the host this node represents
MX	Domain	Refers to a mail server to handle mail addressed to this node
SRV	Domain	Refers to a server handling a specific service
NS	Zone	Refers to a name server that implements the represented zone
CNAME	Node	Symbolic link with the primary name of the represented node
PTR	Host	Contains the canonical name of a host
HINFO	Host	Holds information on the host this node represents
TXT	Any kind	Contains any entity-specific information considered useful

The most important types of resource records forming the contents of nodes in the DNS name space.

7/7/2005

20

DNS Implementation (1)

An excerpt from the DNS database for the zone *cs.vu.nl*.

Name	Record type	Record value
cs.vu.nl	SOA	star (1999121502,7200,3600,2419200,86400)
cs.vu.nl	NS	star.cs.vu.nl
cs.vu.nl	NS	top.cs.vu.nl
cs.vu.nl	NS	solo.cs.vu.nl
cs.vu.nl	TXT	"Vrije Universiteit - Math. & Comp. Sc."
cs.vu.nl	MX	1 zephyr.cs.vu.nl
cs.vu.nl	MX	2 tornado.cs.vu.nl
cs.vu.nl	MX	3 star.cs.vu.nl
star.cs.vu.nl	HINFO	Sun Unix
star.cs.vu.nl	MX	1 star.cs.vu.nl
star.cs.vu.nl	MX	10 zephyr.cs.vu.nl
star.cs.vu.nl	A	130.37.24.6
star.cs.vu.nl	A	192.31.231.42
zephyr.cs.vu.nl	HINFO	Sun Unix
zephyr.cs.vu.nl	MX	1 zephyr.cs.vu.nl
zephyr.cs.vu.nl	MX	2 tornado.cs.vu.nl
zephyr.cs.vu.nl	A	192.31.231.66
www.cs.vu.nl	CNAME	soling.cs.vu.nl
ftp.cs.vu.nl	CNAME	soling.cs.vu.nl
soling.cs.vu.nl	HINFO	Sun Unix
soling.cs.vu.nl	MX	1 soling.cs.vu.nl
soling.cs.vu.nl	MX	10 zephyr.cs.vu.nl
soling.cs.vu.nl	A	130.37.24.11
laser.cs.vu.nl	HINFO	PC MS-DOS
laser.cs.vu.nl	A	130.37.30.32
vucs-das.cs.vu.nl	PTR	0.26.37.130.in-addr.arpa
vucs-das.cs.vu.nl	A	130.37.26.0

7/7/2005

DNS Implementation (2)

Name	Record type	Record value
cs.vu.nl	NIS	solo.cs.vu.nl
solo.cs.vu.nl	A	130.37.21.1

Part of the description for the *vu.nl* domain which contains the *cs.vu.nl* domain.

7/7/2005

22

The X.500 Name Space (1)

Attribute	Abbr.	Value
Country	C	NL
Locality	L	Amsterdam
Organization	L	Vrije Universiteit
OrganizationalUnit	OU	Math. & Comp. Sc.
CommonName	CN	Main server
Mail_Servers	--	130.37.24.6, 192.31.231,192.31.231.66
FTP_Server	--	130.37.21.11
WWW_Server	--	130.37.21.11

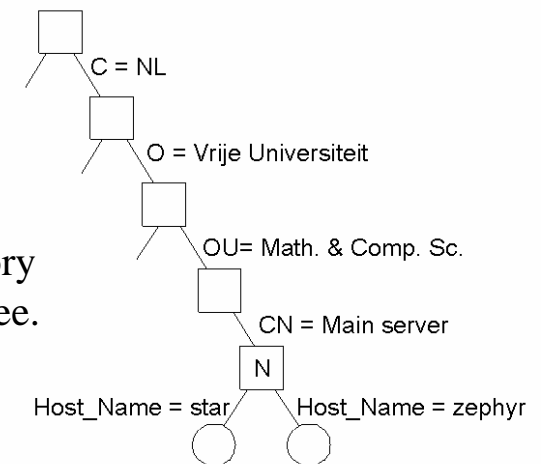
A simple example of a X.500 directory entry using X.500 naming conventions.

7/7/2005

23

The X.500 Name Space (2)

Part of the directory information tree.



7/7/2005

24

The X.500 Name Space (3)

Attribute	Value
Country	NL
Locality	Amsterdam
Organization	Vrije Universiteit
OrganizationalUnit	Math. & Comp. Sc.
CommonName	Main server
Host_Name	star
Host_Address	192.31.231.42

Attribute	Value
Country	NL
Locality	Amsterdam
Organization	Vrije Universiteit
OrganizationalUnit	Math. & Comp. Sc.
CommonName	Main server
Host_Name	zephyr
Host_Address	192.31.231.66

Two directory entries having *Host_Name* as RDN.

7/7/2005

25

Locating Mobile Entities

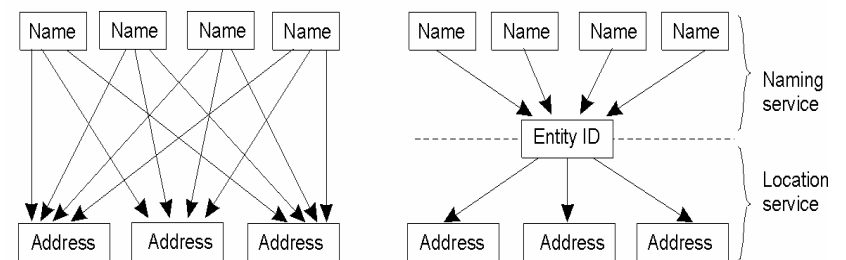
Naming and Locating Entities (1)

- Location service: Solely aimed at providing the addresses of the current locations of entities.
- Assumption: Entities are mobile, so that their current address may change frequently.
- Naming service: Aimed at providing the content of nodes in a name space, given a (compound) name. Content consists of different (attribute,value) pairs.
- Assumption: Node contents at global and administrative level is relatively stable for scalability reasons.
- Observation: If a traditional naming service is used to locate entities, we also have to assume that node contents at the managerial level is stable, as we can use only names as identifiers (think of Web pages)
- Problem: It is not realistic to assume stable node contents down to the local naming level
- Solution: Decouple naming from locating entities

7/7/2005

27

Naming and Locating Entities (2)



(a)
Direct, single level mapping
between names and addresses.

(b)
T-level mapping using identities.

Name: Any name in a traditional naming space

Entity ID: A true identifier

Address: Provides all information necessary to contact an entity

Observation: An entity's name is now completely independent from its location.

7/7/2005

28

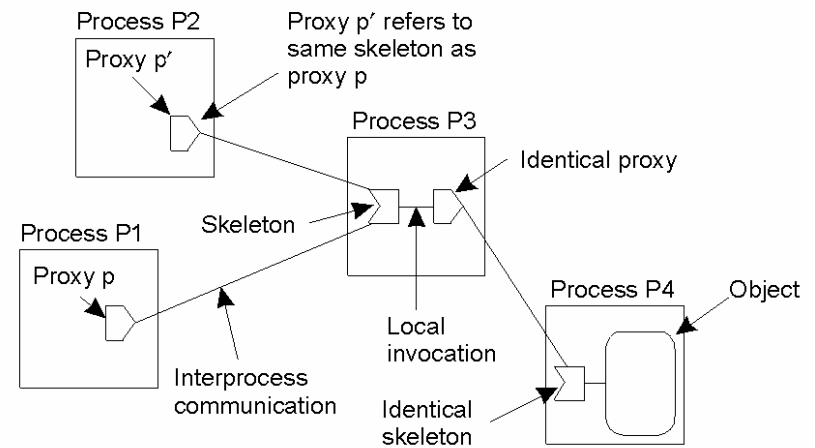
Solutions for Locating Entities

- **Broadcasting:** Simply broadcast the ID, requesting the entity to return its current address.
 - Can never scale beyond local-area networks (think of ARP/RARP)
 - Requires all processes to listen to incoming location requests
- **Forwarding pointers:** Each time an entity moves, it leaves behind a pointer telling where it has gone to.
 - Dereferencing can be made entirely transparent to clients by simply following the chain of pointers
 - Update a client's reference as soon as present location has been found
 - Geographical scalability problems:
 - Long chains are not fault tolerant
 - Increased network latency at dereferencing
 - Essential to have separate chain reduction mechanisms

7/7/2005

29

Forwarding Pointers (1)

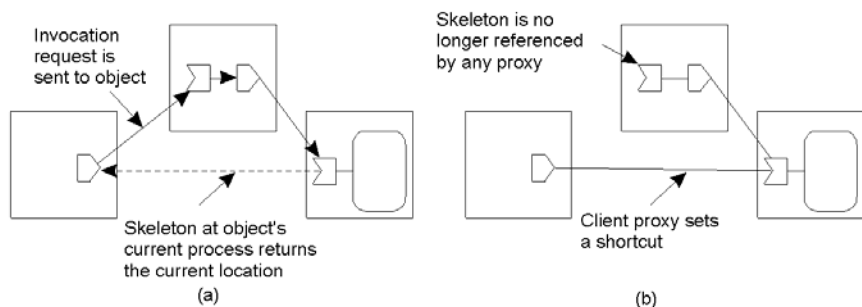


The principle of forwarding pointers using (*proxy*, *skeleton*) pairs.

7/7/2005

30

Forwarding Pointers (2)



Redirecting a forwarding pointer, by storing a shortcut in a proxy.

7/7/2005

31

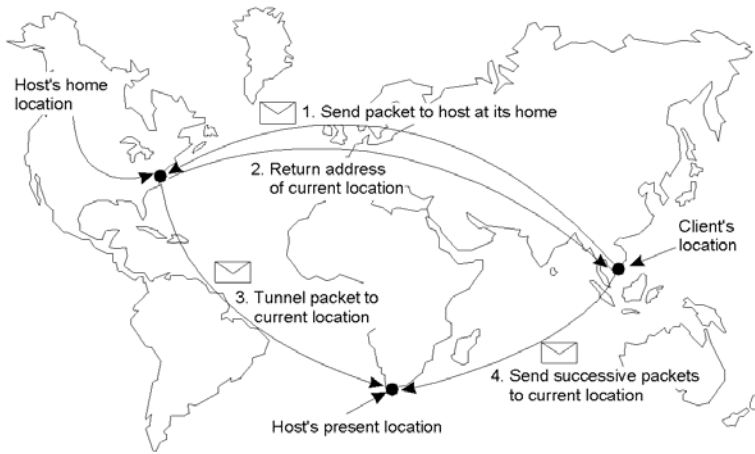
Home-Based Approaches (1)

- **Single-tiered scheme:** Let a home keep track of where the entity is:
 - An entity's home address is registered at a naming service
 - The home registers the foreign address of the entity
 - Clients always contact the home first, and then continues with the foreign location
- **Two-tiered scheme:** Keep track of visiting entities:
 - Check local visitor register first
 - Fall back to home location if local lookup fails
- **Problems with home-based approaches:**
 - The home address has to be supported as long as the entity lives.
 - The home address is fixed, which means an unnecessary burden when the entity permanently moves to another location
 - Poor geographical scalability (the entity may be next to the client)

7/7/2005

32

Home-Based Approaches (2)



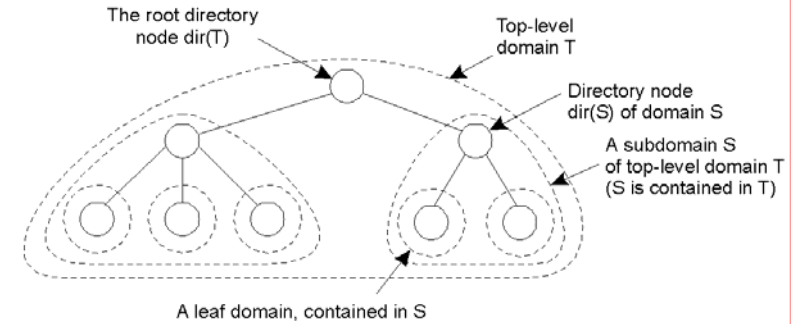
The principle of Mobile IP.

7/7/2005

33

Hierarchical Location Services (HLS)

Build a large-scale search tree for which the underlying network is divided into hierarchical domains. Each domain is represented by a separate directory node.



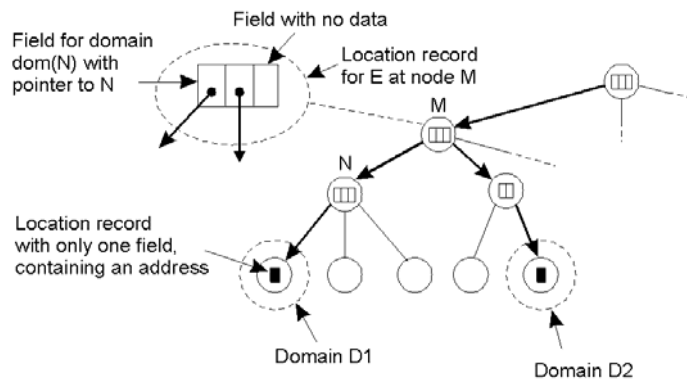
Hierarchical organization of a location service into domains, each having an associated directory node.

7/7/2005

34

HLS: Tree Organization

- The address of an entity is stored in a leaf node, or in an intermediate node
- Intermediate nodes contain a pointer to a child if and only if the subtree rooted at the child stores an address of the entity
- The root knows about all entities



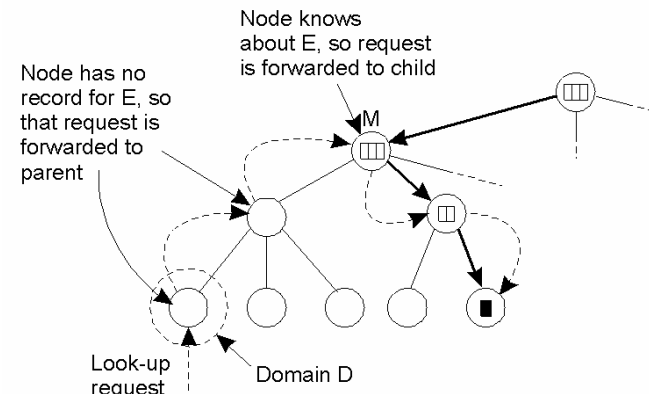
Storing information of an entity having two addresses in different leaf domains

7/7/2005

35

HLS: Lookup Operation

- Start lookup at local leaf node
- If node knows about the entity, follow downward pointer, otherwise go one level up
- Upward lookup always stops at root

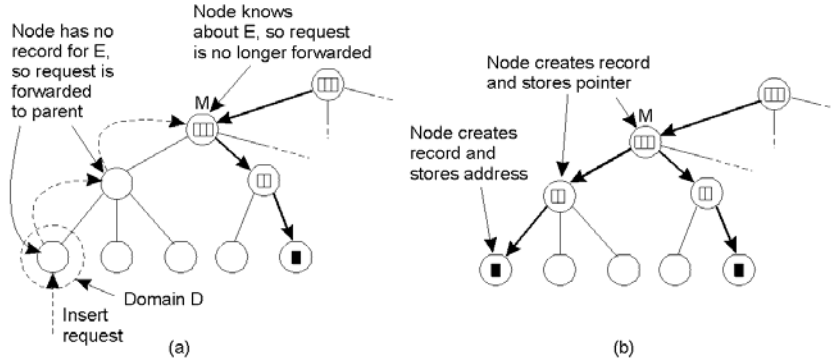


Looking up a location in a hierarchically organized location service.

7/7/2005

36

HLS: Insert Operation



- a) An insert request is forwarded to the first node that knows about entity E .
- b) A chain of forwarding pointers to the leaf node is created.

7/7/2005

37

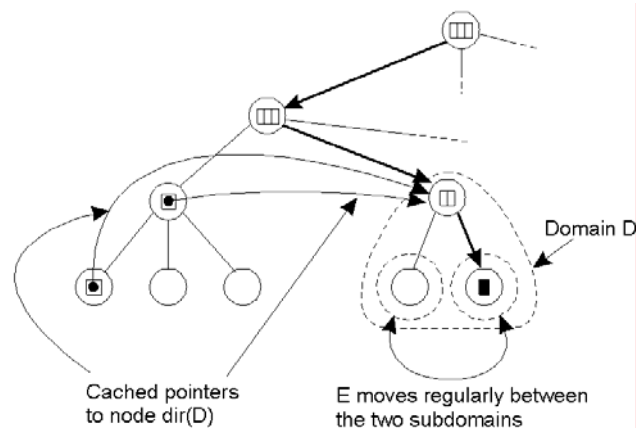
HLS: Record Placement

- Observation: If an entity E moves regularly between leaf domains D_1 and D_2 , it may be more efficient to store E 's contact record at the least common ancestor LCA of $\text{dir}(D_1)$ and $\text{dir}(D_2)$.
 - Lookup operations from either D_1 or D_2 are on average cheaper
 - Update operations (i.e., changing the current address) can be done directly at LCA
 - Note: assuming that E generally stays in $\text{dom}(LCA)$, it does make sense to cache a pointer to LCA

7/7/2005

38

Pointer Caches (1)

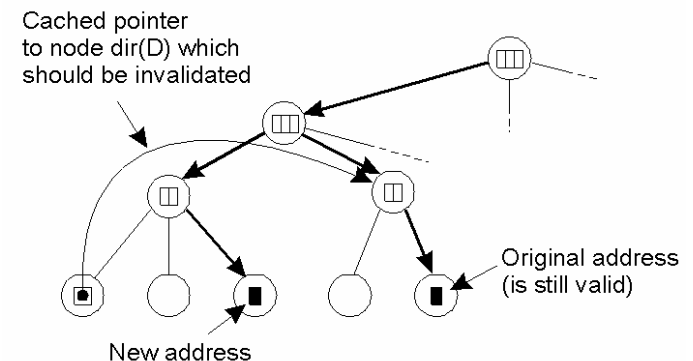


Caching a reference to a directory node of the lowest-level domain in which an entity will reside most of the time.

7/7/2005

39

Pointer Caches (2)



A cache entry that needs to be invalidated because it returns a nonlocal address, while such an address is available.

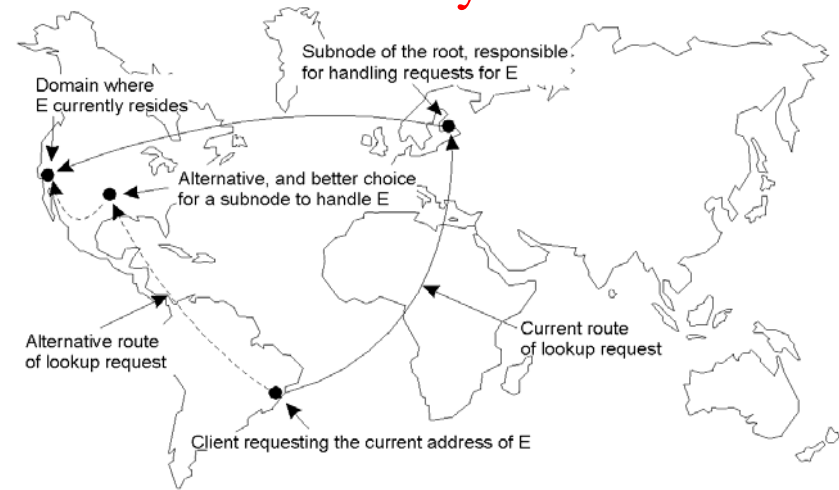
7/7/2005

40

HLS: Scalability Issues

- Size scalability: Again, we have a problem of overloading higher-level nodes:
 - Only solution is to partition a node into a number of subnodes and evenly assign entities to subnodes
 - Naive partitioning may introduce a node management problem, as a subnode may have to know how its parent and children are partitioned.
- Geographical scalability: We have to ensure that lookup operations generally proceed monotonically in the direction of where we'll find an address:
 - If entity E generally resides in California, we should not let a root subnode located in France store E's contact record.
 - Unfortunately, subnode placement is not that easy, and only a few tentative solutions are known

Scalability Issues



The scalability issues related to uniformly placing subnodes of a partitioned root node across the network covered by a location service.