Distributed Systems

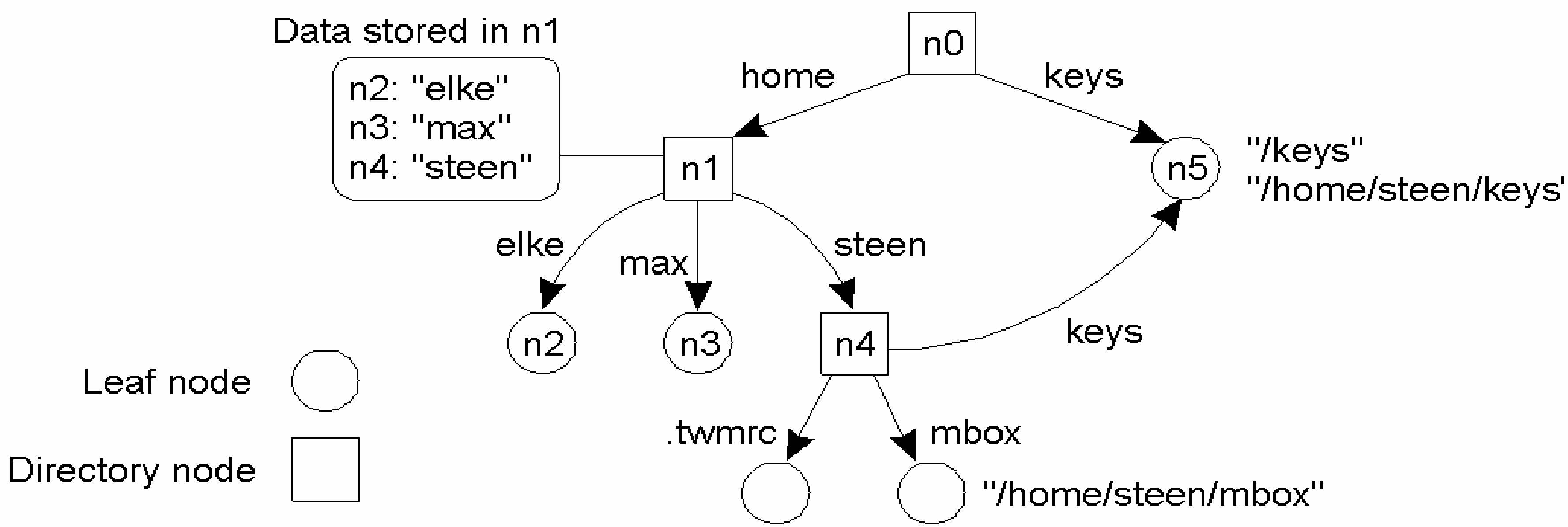
3. Naming

Names, Addresses, Identifiers

- Naming is about mapping between names, addresses, identifiers and the referred entities
- Names (a bit- or character-string referring to an entity)
  - E.g. John Smith or ftp-server
  - Can be *human-friendly* (or not) and *location dependent* (or not)
- Addresses (define access points)
  - Entities can be operated through an *access point*
  - The name of an access point is an address
  - E.g. phone number, or IP-address + port for a service
- Identifiers (*unique* identifiers)
  - A (true) identifier is a name with the following properties
    1. Each identifier refers to at most 1 entity and
    2. Each entity is referred to by at most 1 identifier
    3. An identifier always refers to the same entity (never reused)
  - E.g. John Smith + social security number, or MAC address

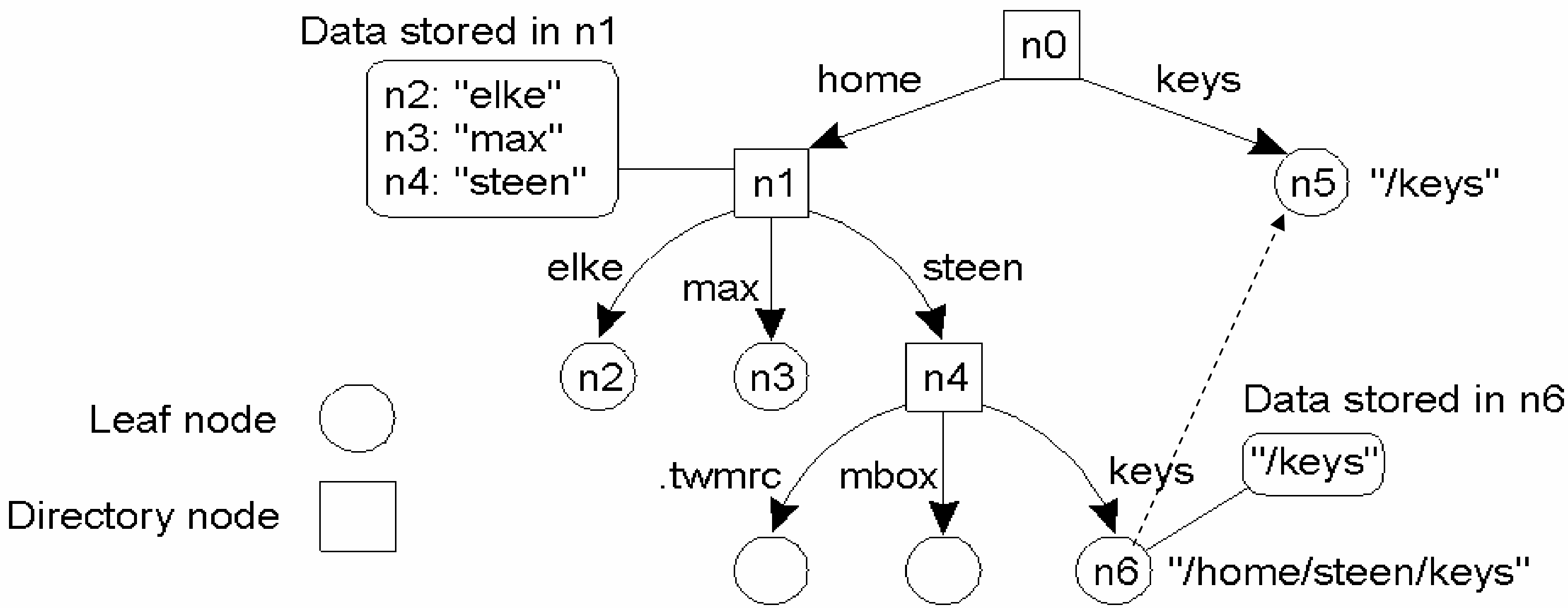
Name Spaces

- A set of related names, typically represented as a directed graph (maybe restricted to a DAG or a tree, or even a list)
  - *Path names* (sequence of edges) can be *absolute* and *relative*
  - *Name resolution* looks up the content of a referred node
  - Names can be *global* in a whole system (e.g. names of commands, like “cd”) or *local* (relative to an implicitly known node, e.g. home directory)



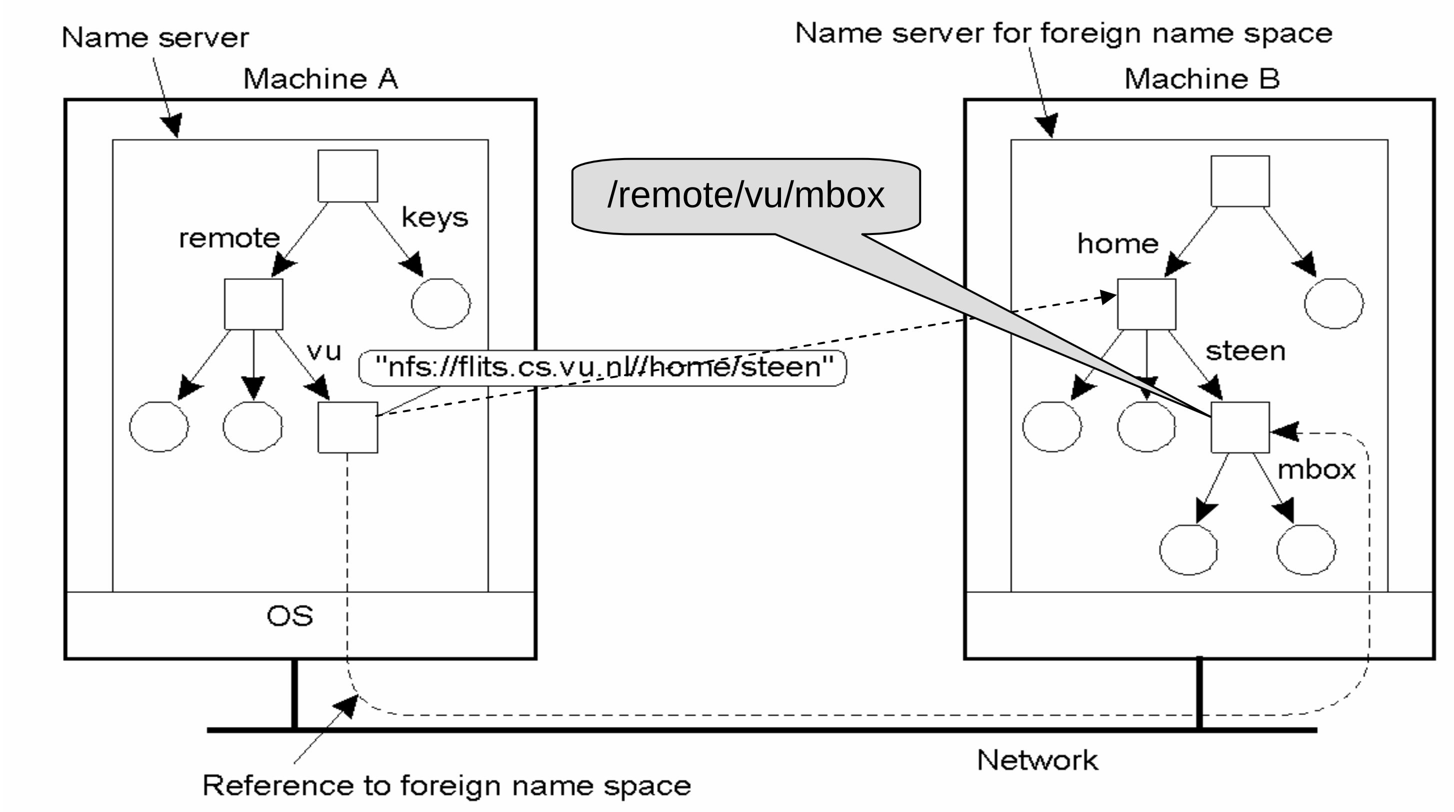
Alias names

- The same entity may have several *alias* names
  - *Hard links*: The same node is referred via several path names (see the 2 links leading to n5 on the previous slide)
  - *Symbolic links*: The referred node contains a further (abs.) reference, which can be used instead of the original path name

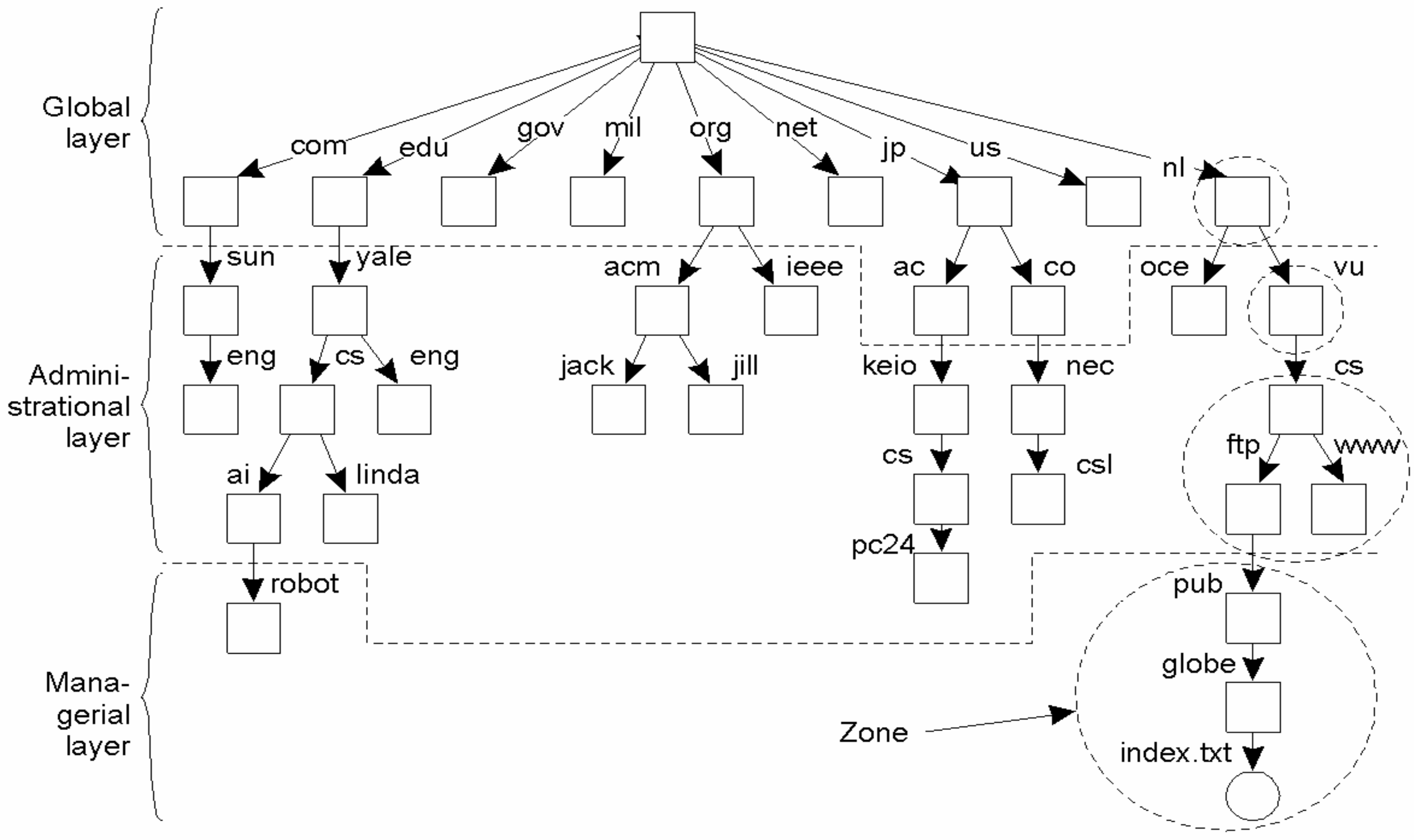


Linking and Mounting

- Mounting remote name spaces ("remote symbolic link") needs
  - ➢ A specific protocol (e.g. NFS, see later)
  - ➢ At a certain *mount point* of a given server
  - ➢ A name, containing access protocol, remote server, foreign mounting point



Name Space Distribution (1)



- Partitioning of the DNS name space into three layers
- More details of DNS see at “computer networks”

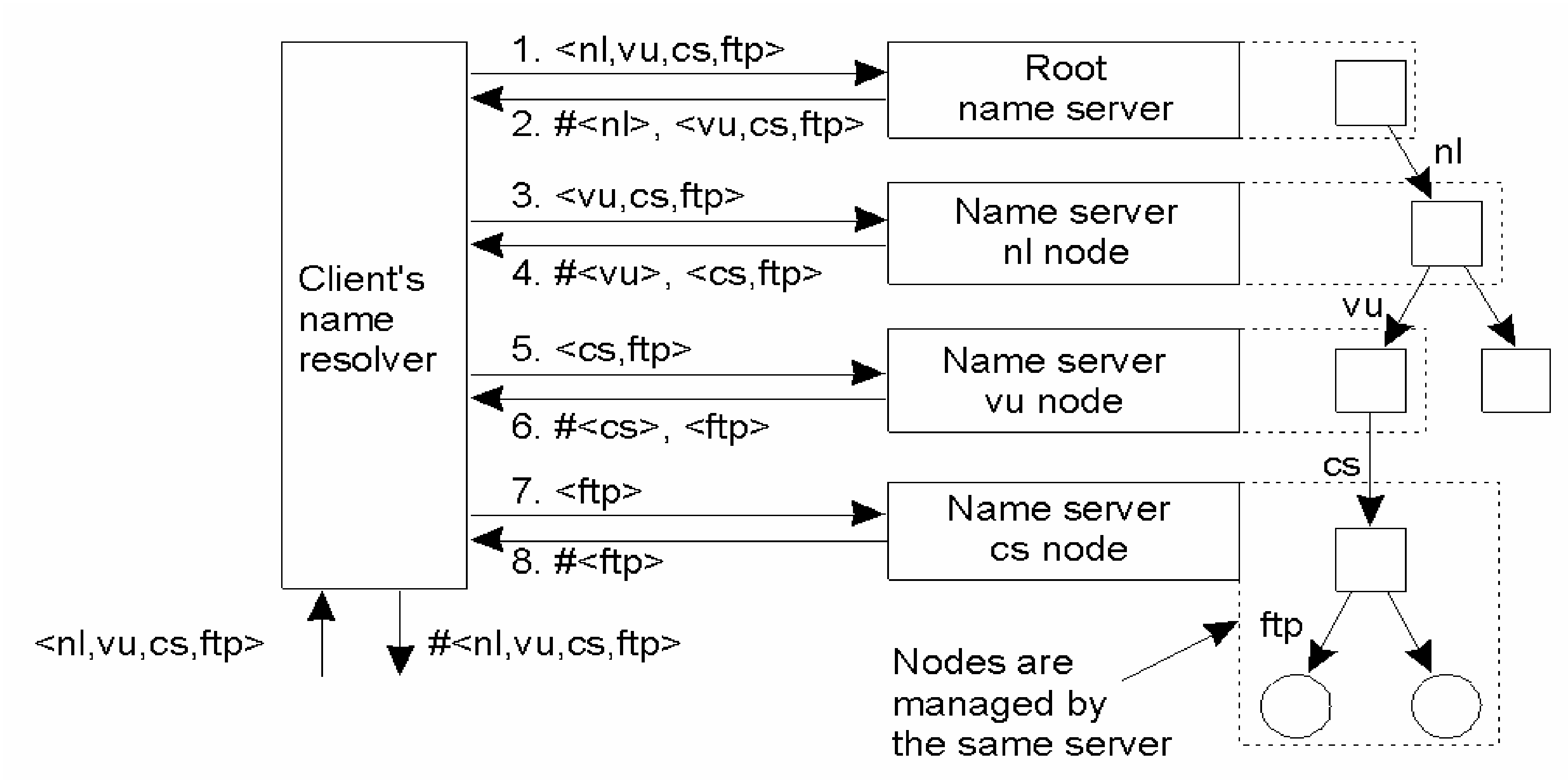
Name Space Distribution (2)

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, an administrational layer, and a managerial layer

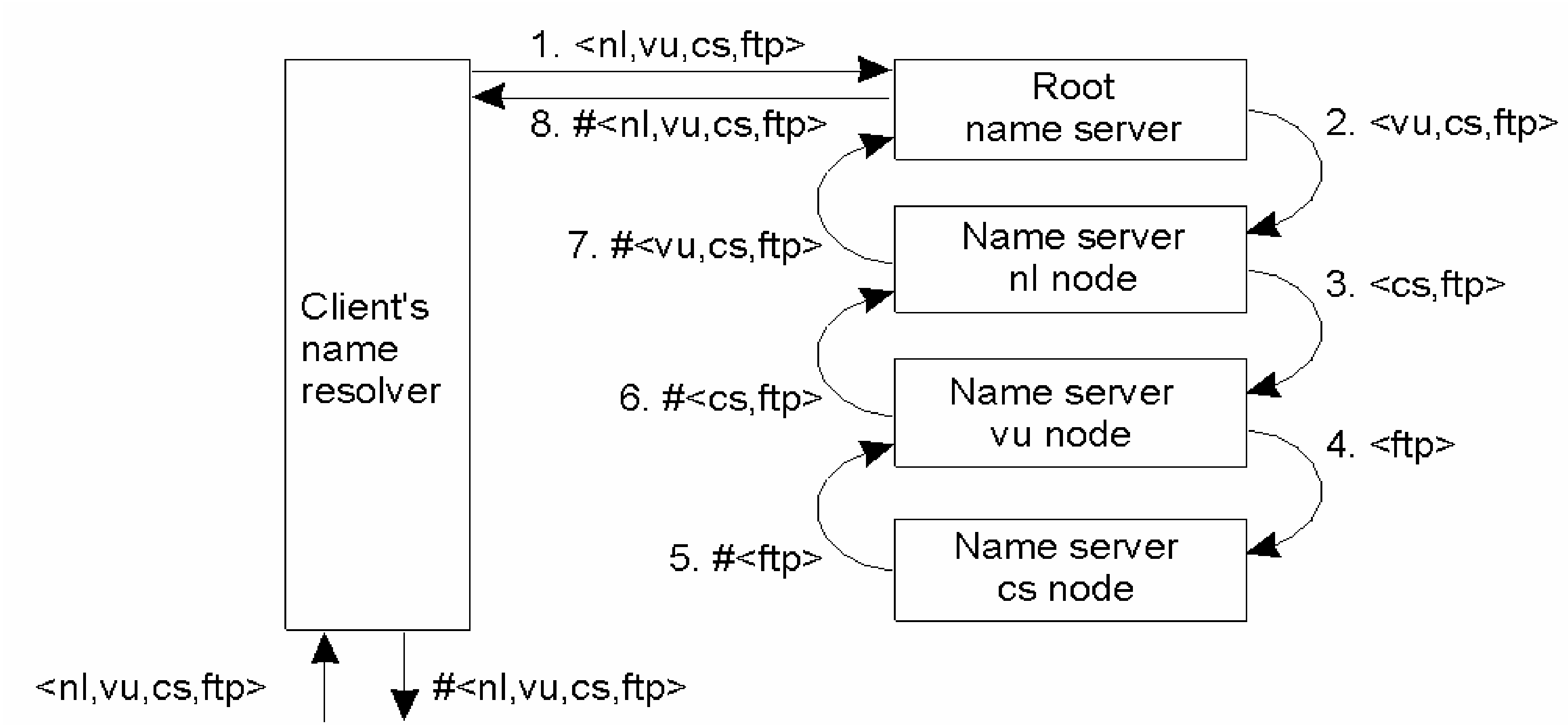
Implementation of Name Resolution (1)

- The principle of iterative name resolution
  - The address of the root server must be well knwon



Implementation of Name Resolution (2)

- The principle of recursive name resolution
  - \+ Enables efficient caching and reduces communication
  - \- Causes higher performance demand on the name servers

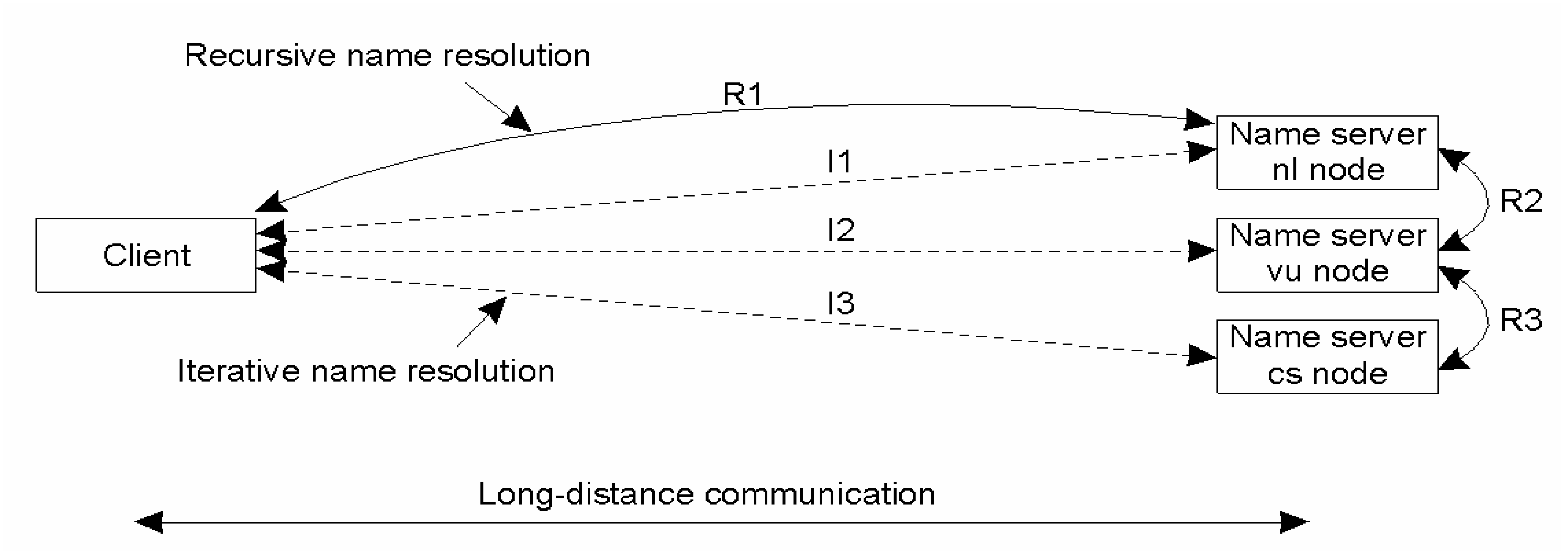


Implementation of Name Resolution (3)

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>
nl	<vu,cs,ftp>	#<vu>	<cs,ftp>	#<cs> #<cs,ftp>	#<vu> #<vu,cs> #<vu,cs,ftp>
root	<nl,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

Implementation of Name Resolution (4)



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

The X.500 Name Space (1)

- More than a naming service
  - A directory service with *search*
  - Items can be found based on *properties* (not only full names)
  - X.500 defines attribute-value pairs, such as:

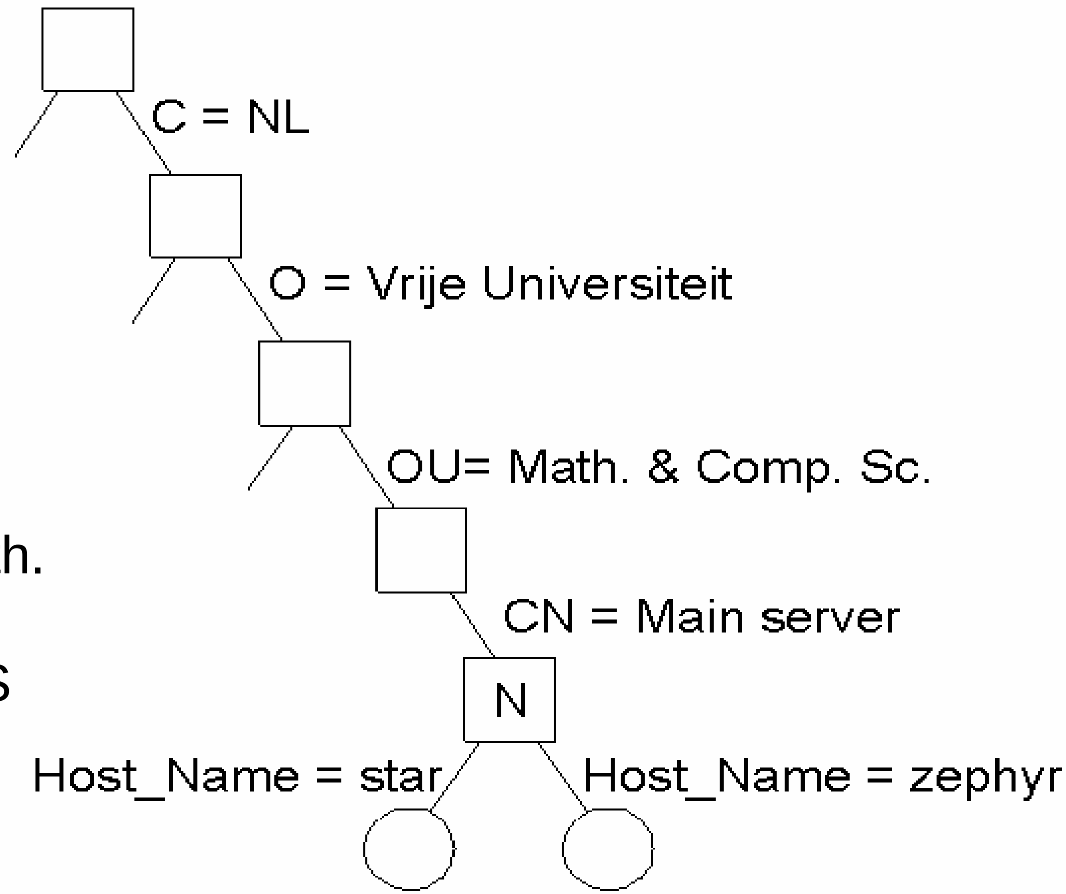
Attribute	**Abbr.**	**Value**
Country	C	NL
Locality	L	Amsterdam
Organization	L	Vrije Universiteit
OrganizationalUnit	OU	Math. & Comp. Sc.
CommonName	CN	Main server
Mail_Servers	--	130.37.24.6, 192.31.231,192.31.231.66
FTP_Server	--	130.37.21.11
WWW_Server	--	130.37.21.11

The X.500 Name Space (2)

- The collection of directory entries
  - Form the Directory Information Base (DIB)
  - Naming attributes (Country etc.) are called Relative Distinguished Names (RDN)
  - Each record is uniquely named, by a sequence of RDNs
- Directory Information Tree (DIT)
  - Naming graph
  - Each node represents a directory entry and an X.500 record
  - With *read* we can read a certain record
  - With *list* we can read all outgoing edges of the node
- A simplified version of X.500 is generally used
  - Known as Lightweight Directory Access Protocol (LDAP)

The X.500 Name Space (3)

- Part of the directory information tree
  - ➢ The X.500 name /C=NL/O=Vrije Universiteit/OU=Math. & Comp. Sc.
  - ➢ is analog to the DNS name nl.vu.cs



The X.500 Name Space (4)

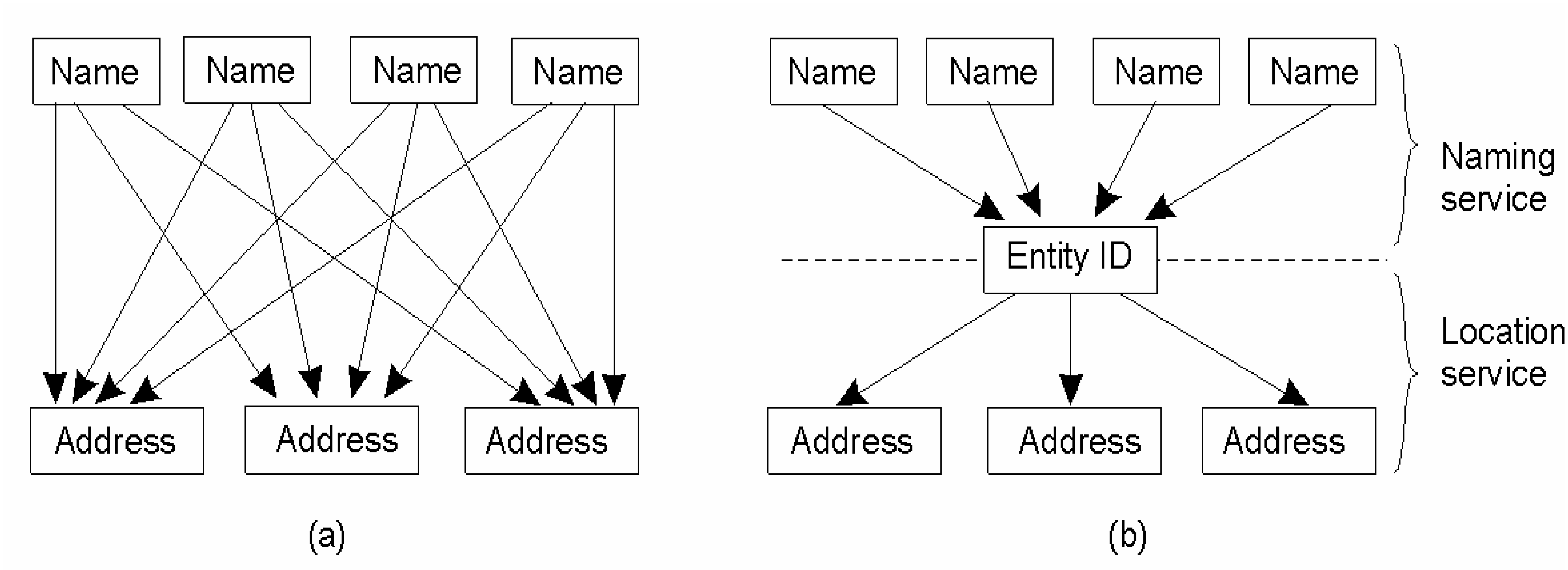
- Two directory entries having *Host_Name* as Relative Distinguished Name (RDN)

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

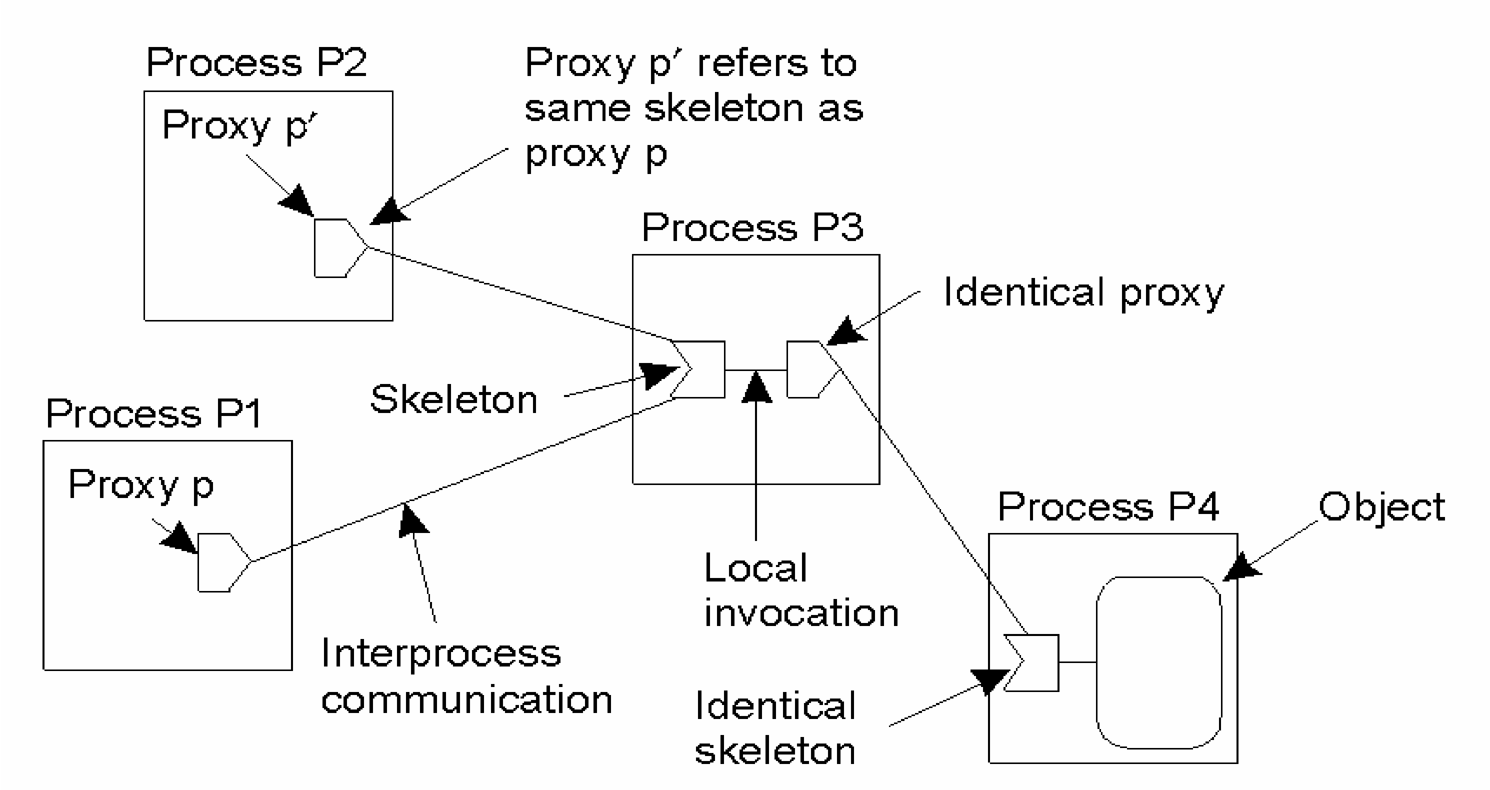
Naming versus Locating Entities

- DNS and X.500 assume restricted change in the system
- For highly mobile entities additional location services are needed
    - ➢ A simple example is ARP (IP → MAC adr. based on LAN-broadcast)



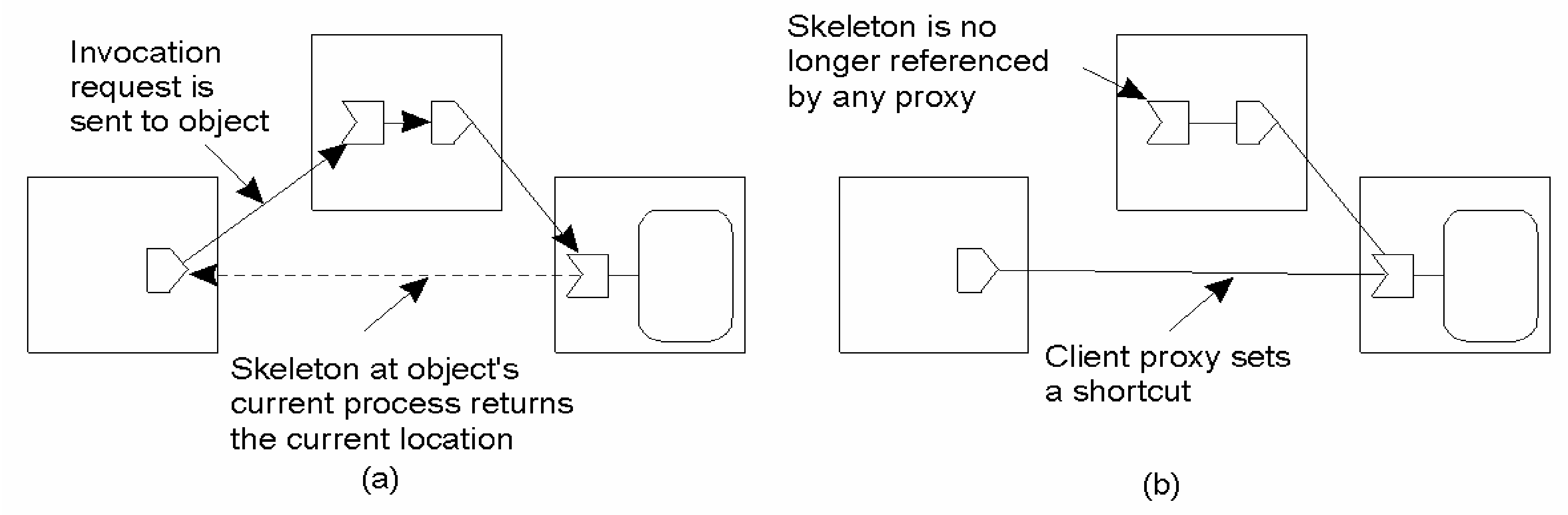
a) Direct mapping between user-friendly names and addresses
b) 2-level mapping using (non-user-friendly) identities

Forwarding Pointers (1)



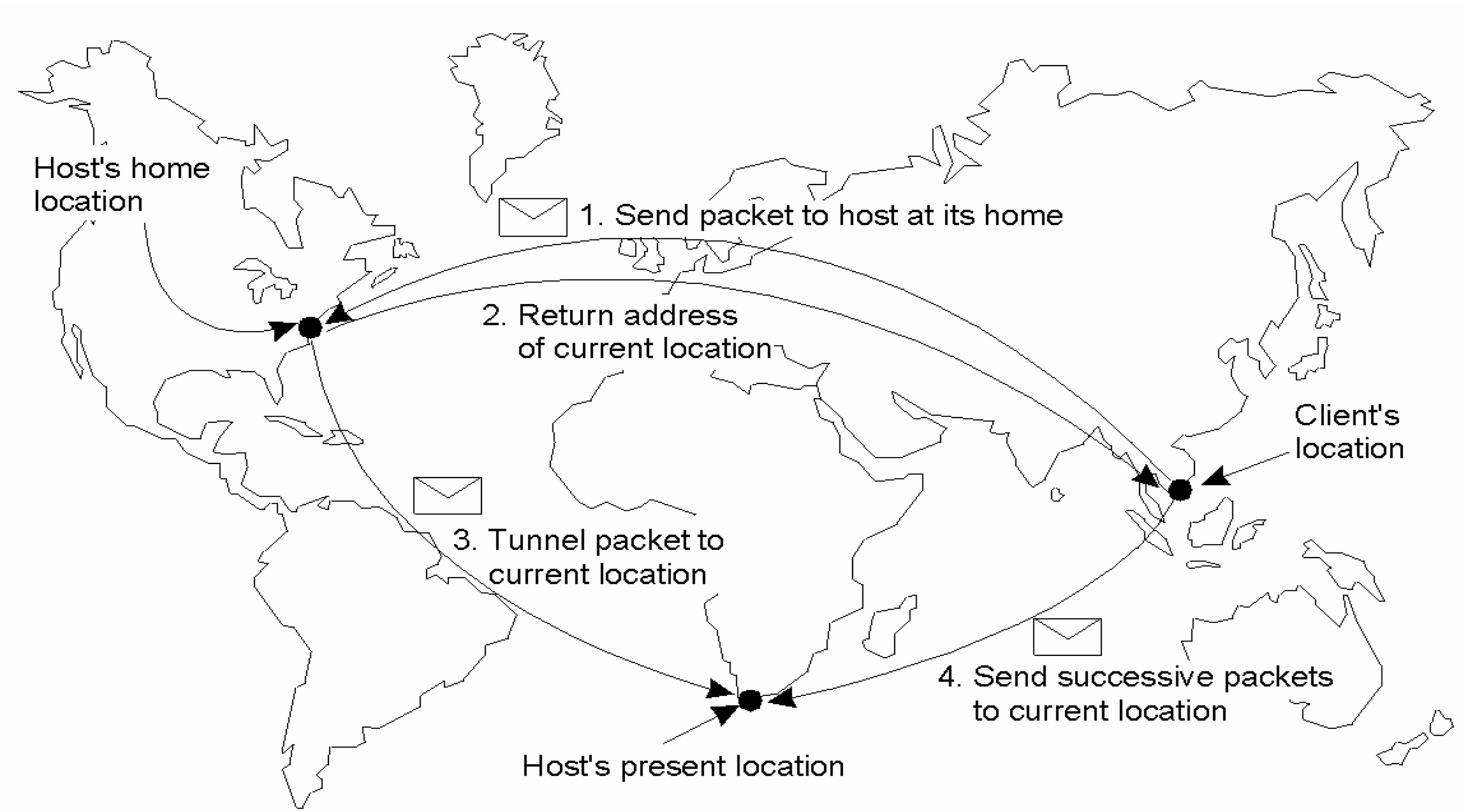
- A moving entity leaves behind a reference (*proxy* or *stub*) at a client pointing to the place at the server (*skeleton* or *scion*)
- Long SSP chains can be built (stub → skeleton → stub → skeleton …)

Forwarding Pointers (2)



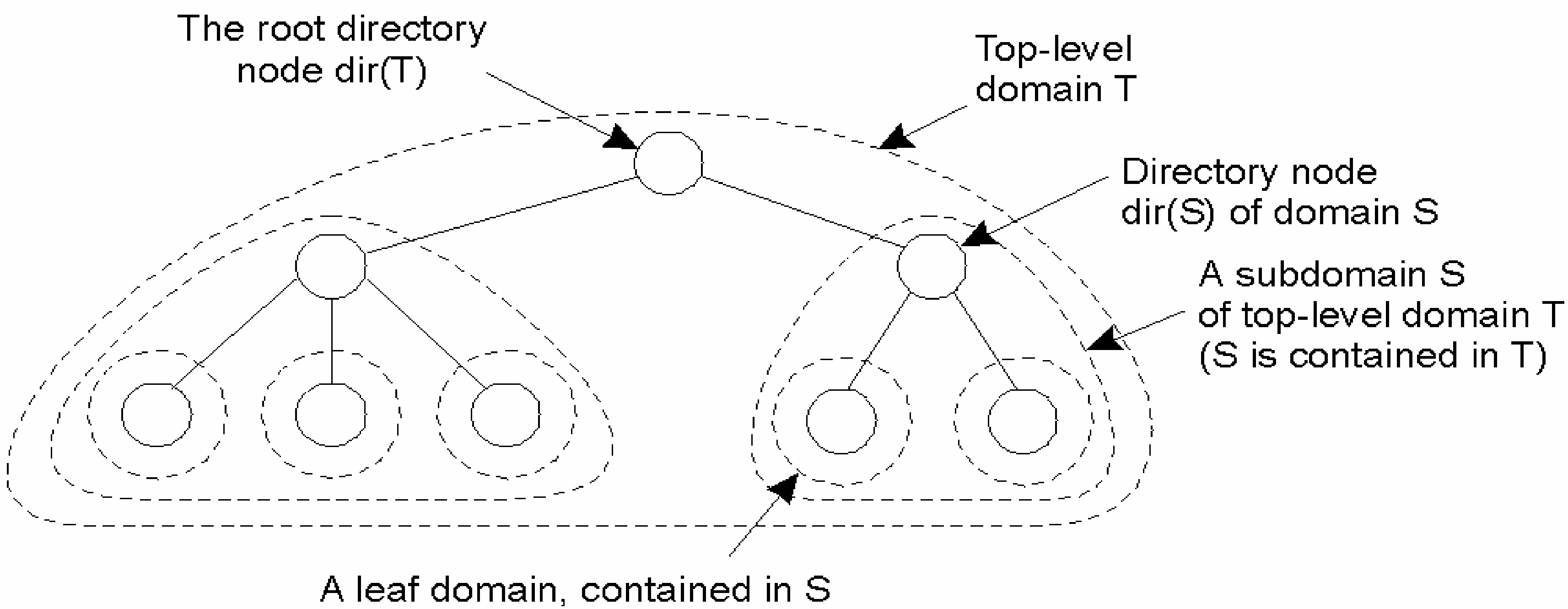
- Redirecting a forwarding pointer, by storing a shortcut in a proxy
- The response can be sent directly, or along the whole reverse path, thus updating all intermediate proxies
- Pointer forwarding is fully transparent, but vulnerable to errors

Home-Based Approaches



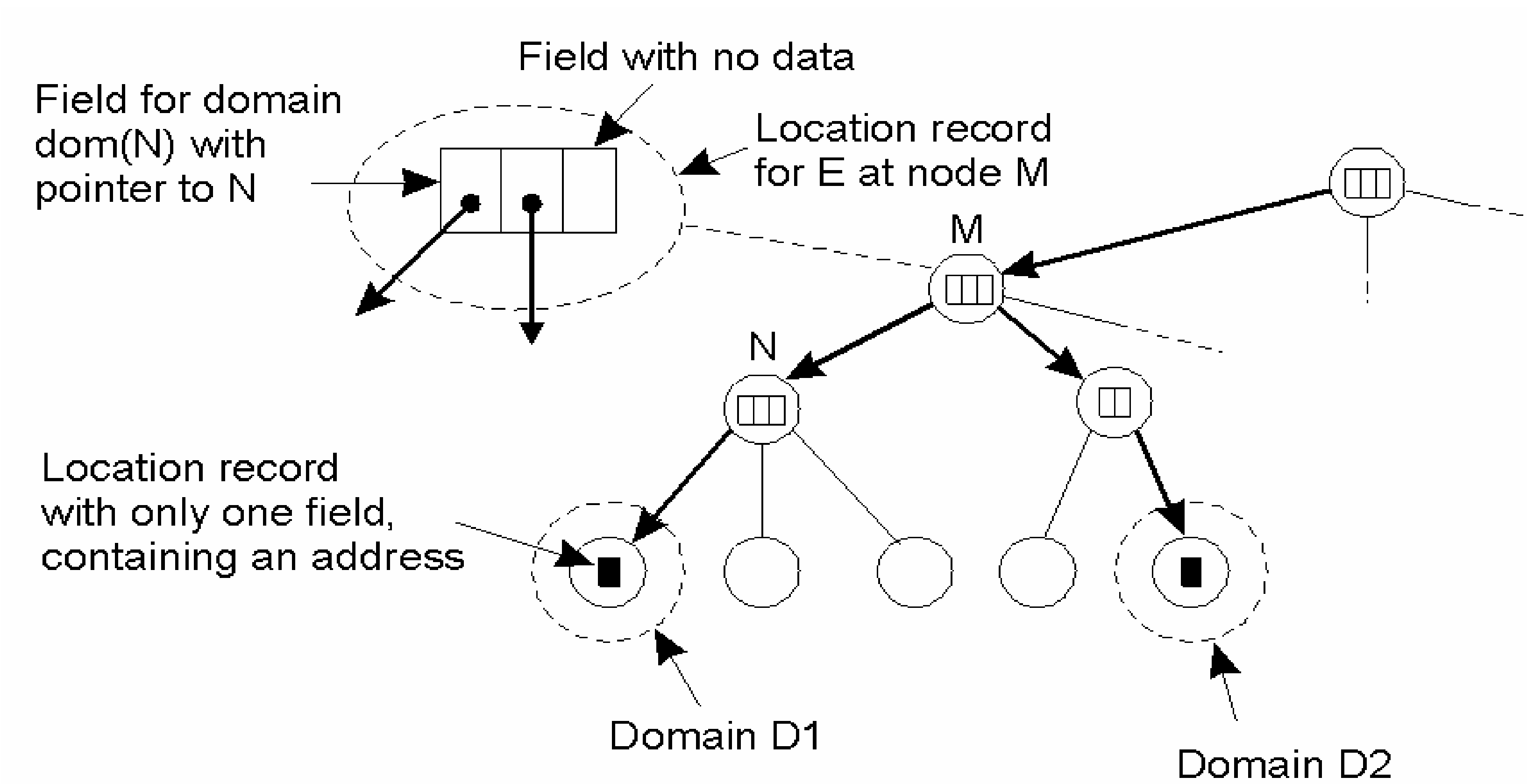
- This is also the principle of Mobile IP (see "Computer Networks")

Hierarchical Approaches (1)



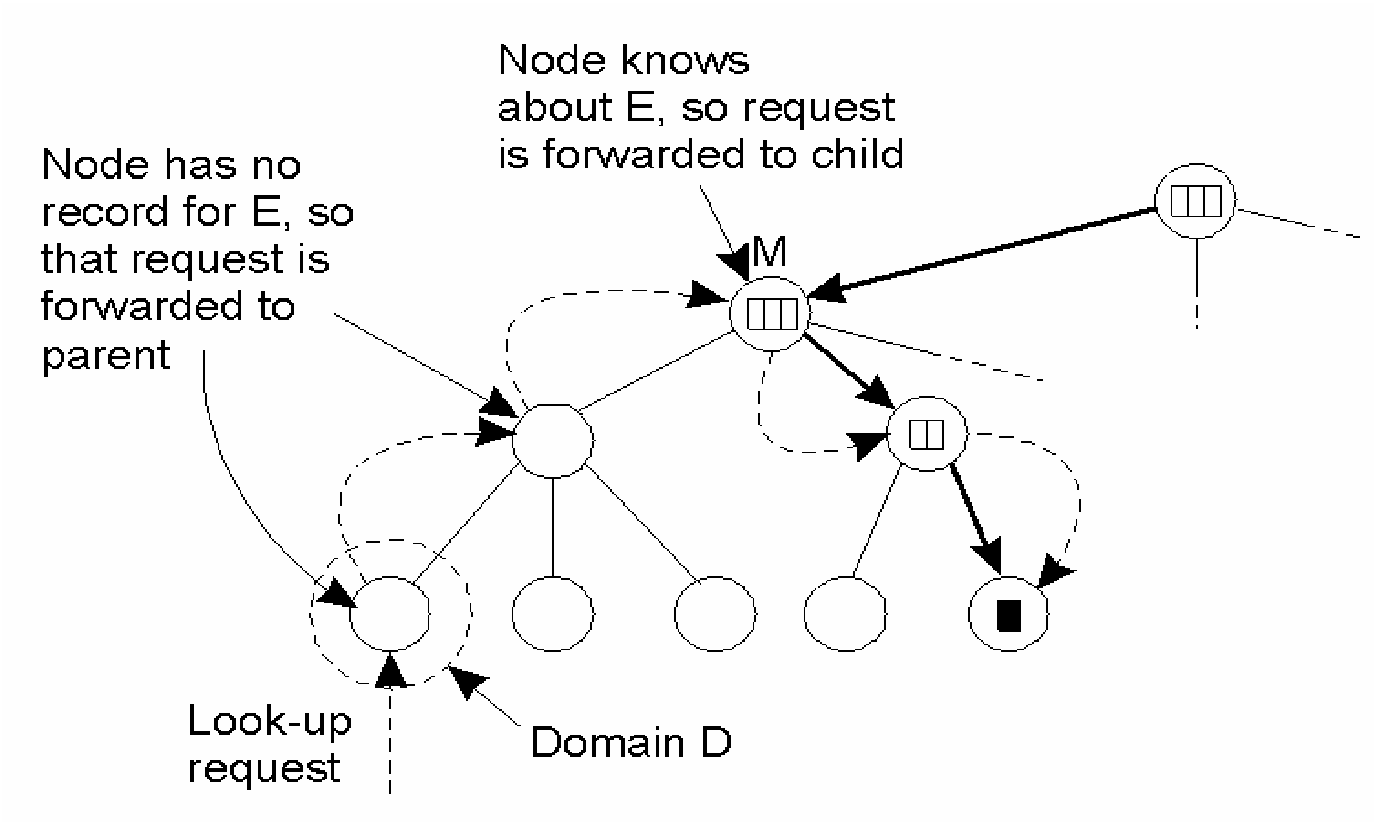
- A generalization of the 2-tired home based approach
- Hierarchical organization of a location service into domains (like DNS)
- The leaf level corresponds e.g. to LANs or mobile phone cells

Hierarchical Approaches (2)



- A replicated entity having two addresses in different leaf domains
- The higher level directories store only pointers

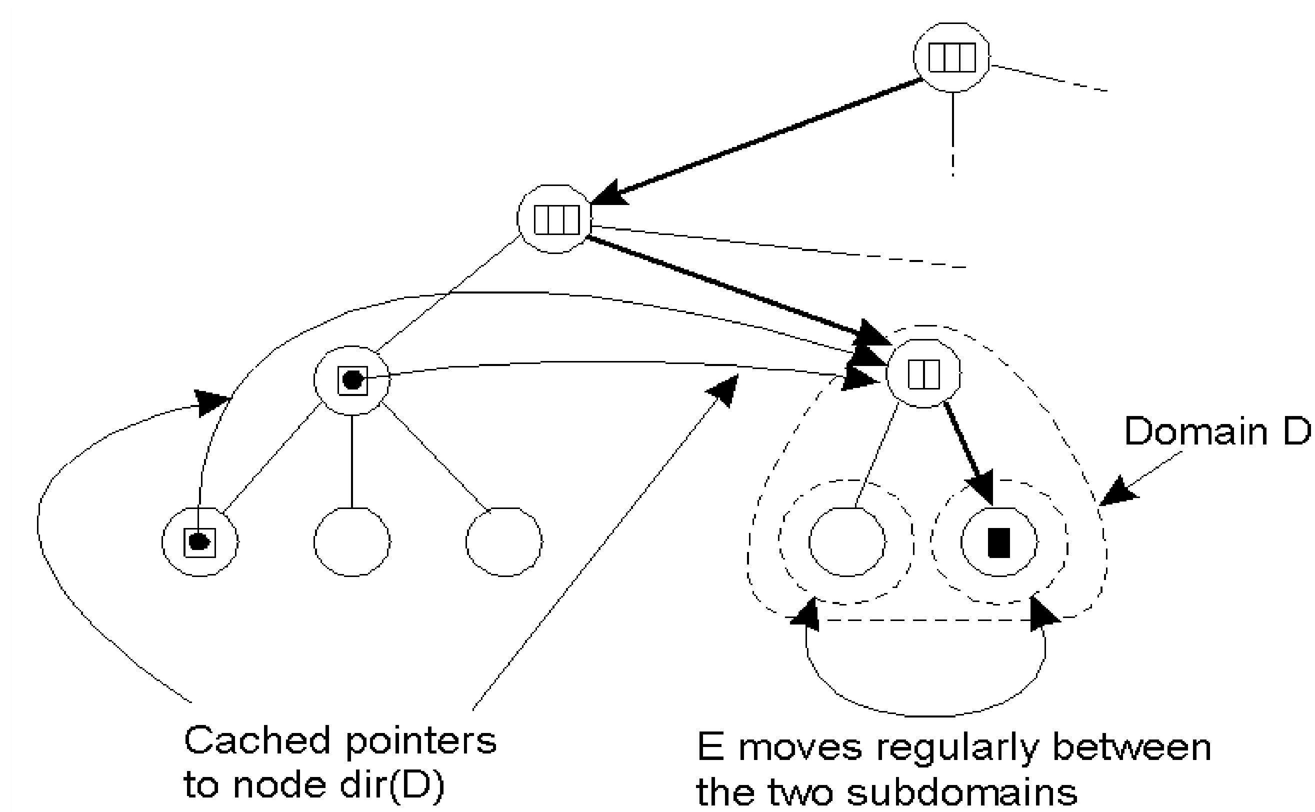
Hierarchical Approaches (3)



- Looking up a location in a hierarchically organized location service
- Lookup searches in an increasing ring, thus exploiting *locality*

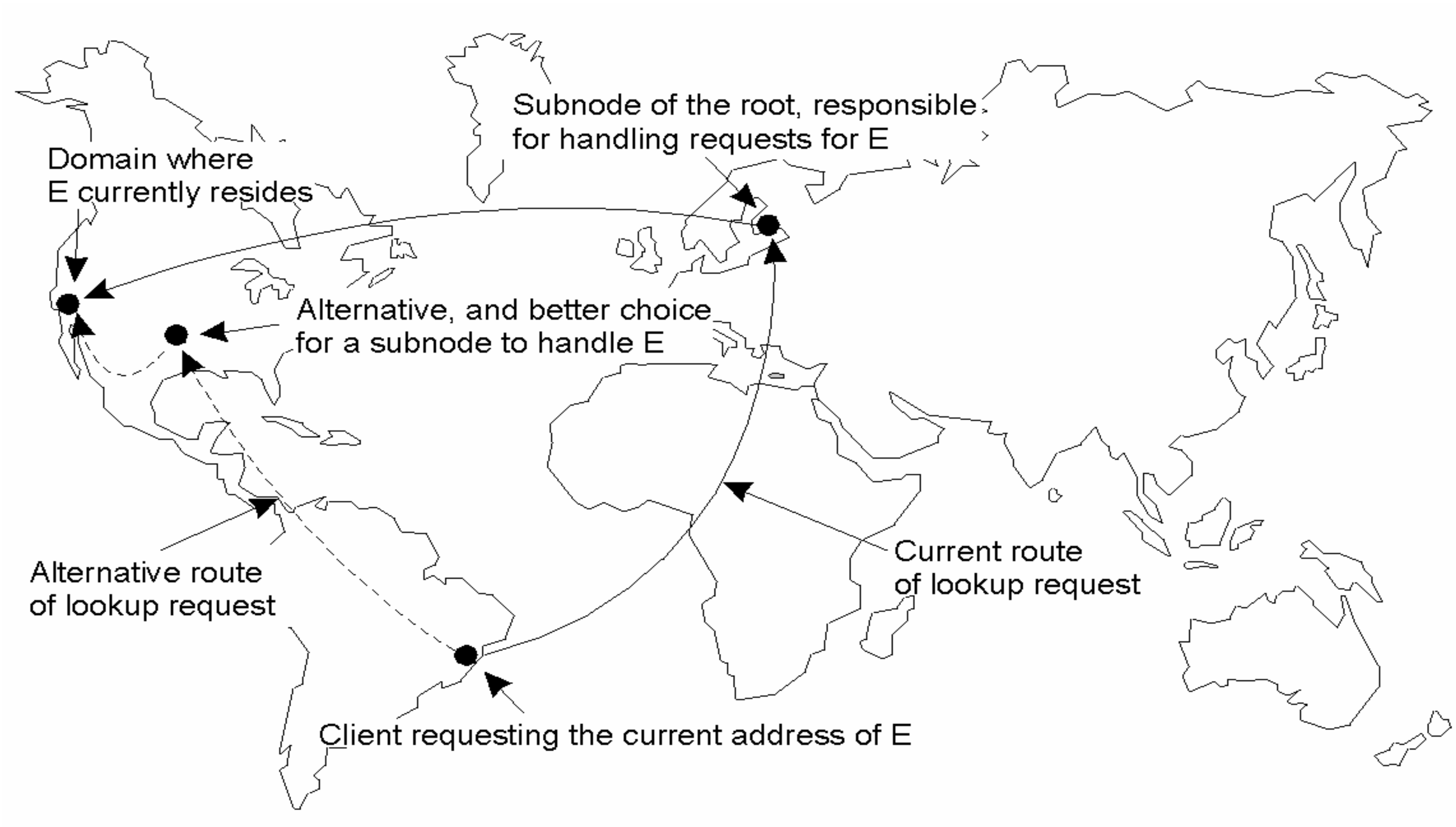
Pointer Caches

- Caching is only effective if changes are rare
- It is better to cache addresses to directories than to individual entities



Scalability Issues

- The root node must be partitioned if the system is big
  - ➢ Its sub-nodes could be placed uniformly distributed



The Problem of Unreferenced Objects

- Unreachable entities should be removed
- Distributed garbage collection is hard
  - Reference counting is simple, but cannot handle cycles