Name Servers

Jonathan Walpole

CSE515 Distributed Computing Systems

Domain Name System (DNS)

- Internet hosts, routers like to use fixed length addresses (numbers)
  - IP address (32 bit) - used for addressing datagrams
- Humans like to use names
  - www.cse.ogi.edu
- Name servers, such as DNS
  - Map from names to IP addresses
  - Map from IP addresses to names

Early Internet Naming

- Flat namespace
  - /etc/hosts
  - SRI kept main copy
  - Downloaded regularly
- Problems
  - Count of hosts was increasing: machine per domain → machine per user
    - Many more downloads
    - Many more updates

Goals for a New Naming System

- Implement a wide area distributed database
  - Scalability
  - Decentralized maintenance
  - Robustness, fault-tolerance
  - Global scope
    - Names mean the same thing everywhere
  - Don’t need
    - Atomicity
    - Strong consistency

Why Not Centralize DNS?

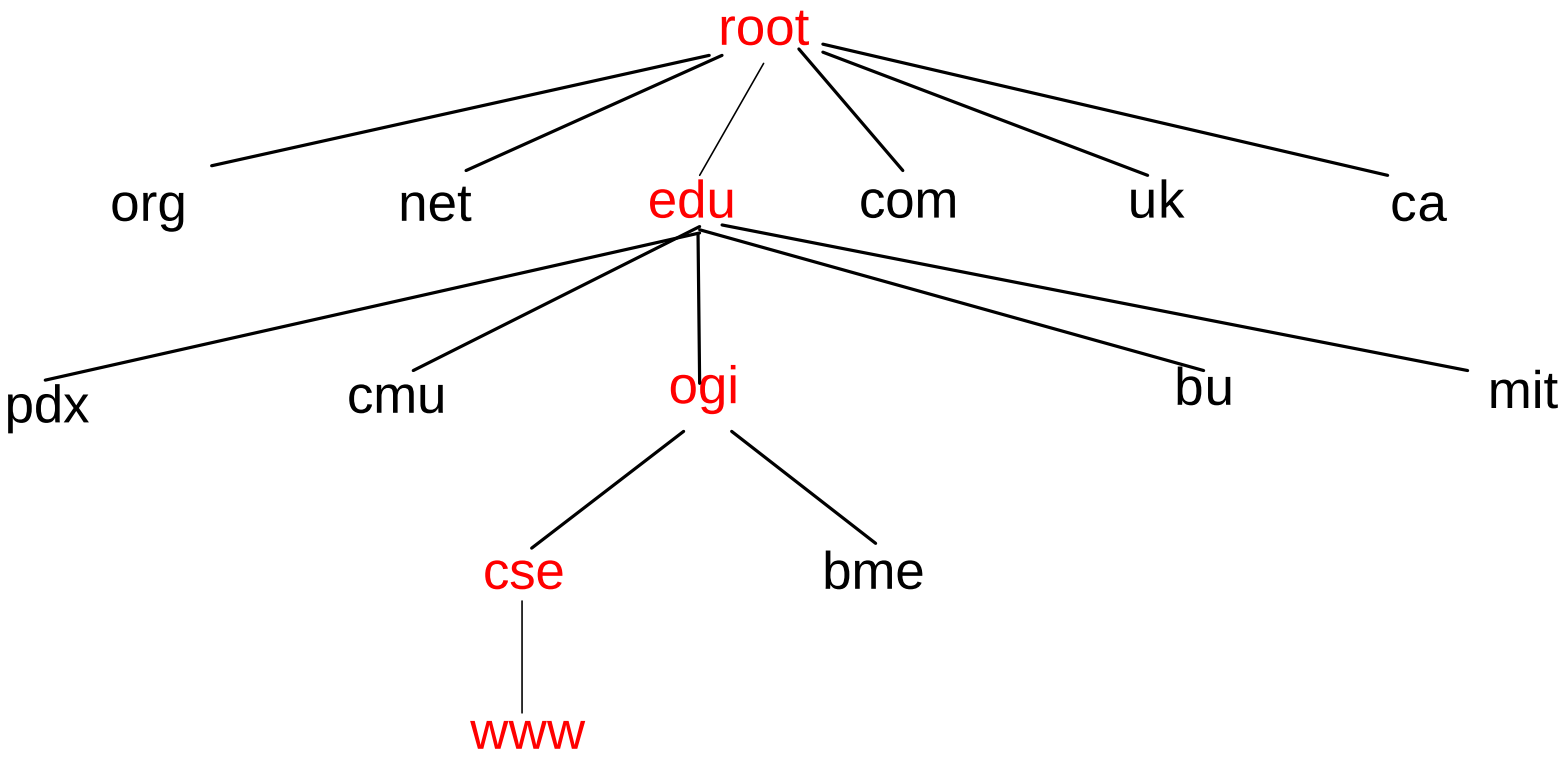
- Single server with all name-to-IP address mappings
  - single point of failure
  - traffic volume
  - distant centralized database (performance)
  - maintenance
  - doesn't *scale!*

DNS (Domain Name System)

- http://www.rfc-editor.org/rfc/rfc1034.txt
- http://www.rfc-editor.org/rfc/rfc1035.txt
- *distributed database* implemented in hierarchy of many *name servers*
- *decentralized control and management* of data
- *application-layer protocol* used by hosts and name servers
  - communicate to *resolve* names (name/address translation)
  - core Internet function implemented as application-layer protocol
- complexity at network's "edge"

DNS Name Space

- Hierarchical canonical name space
  - www.cse.ogi.edu



DNS Name Servers

- Authoritative name servers store parts of the database
- Names assigned to authoritative name servers
  - For a host, authority stores that host's IP address, name
  - Responds to queries for host's IP address
  - Perform name/address translation for that host's name
- Root name server knows authoritative servers for particular subdomains
  - Hierarchy organizes authoritative name servers
  - Reserving a domain gives you control of entry in root name server for particular names

DNS Name Lookup

- hierarchical lookup
  - Each host has a pointer to a local name server to query for unknown names
  - Each local name server knows root of hierarchy
  - Root points to sub-levels, sub-levels point to deeper sub-levels, … , deeper sub-levels point to leaf name server representing authority for unknown name

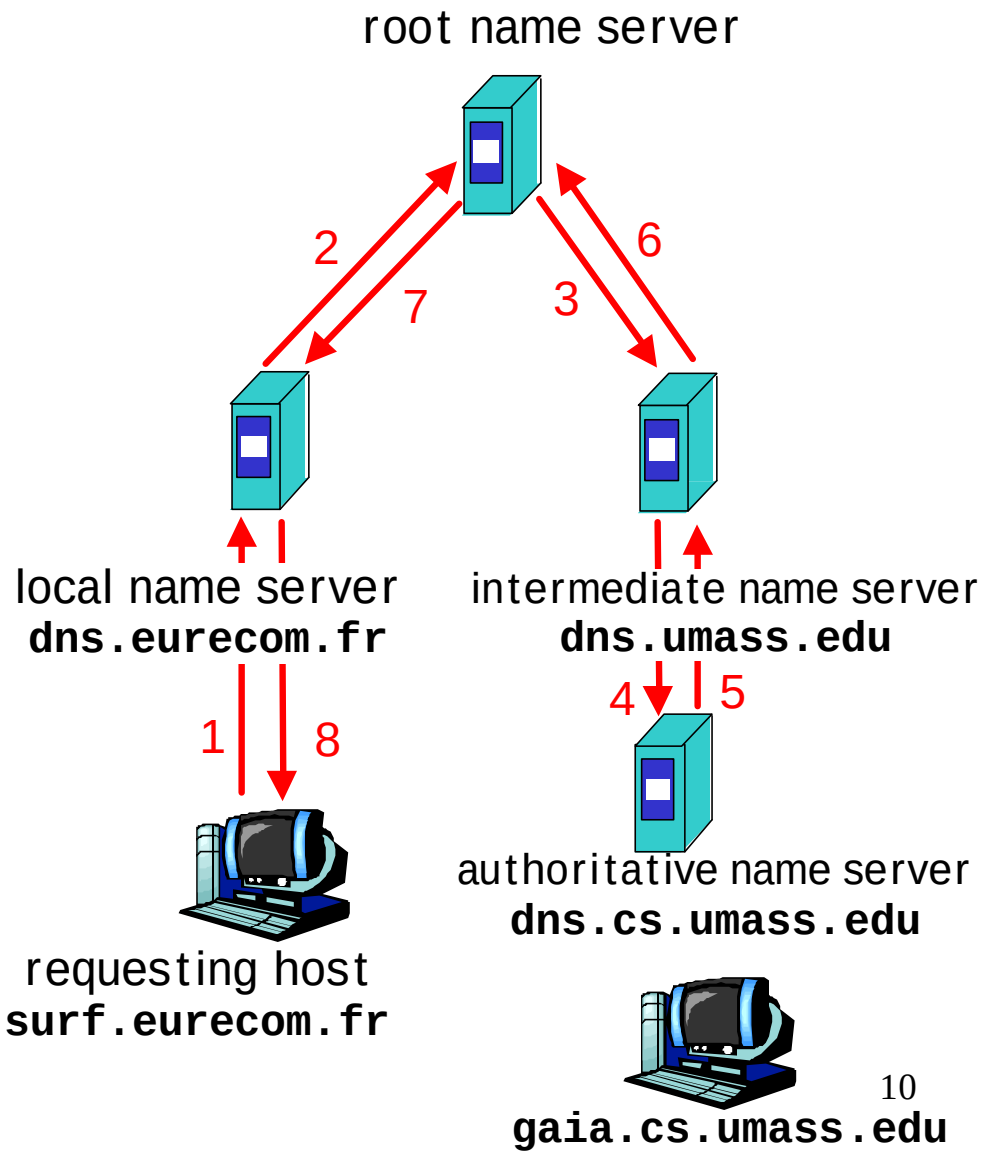
DNS Name Lookup Example

Root name servers:

- may not know authoratiative name server
- may know *intermediate name server:* who to contact to find authoritative name server
- multiple root name servers for fault-tolerance

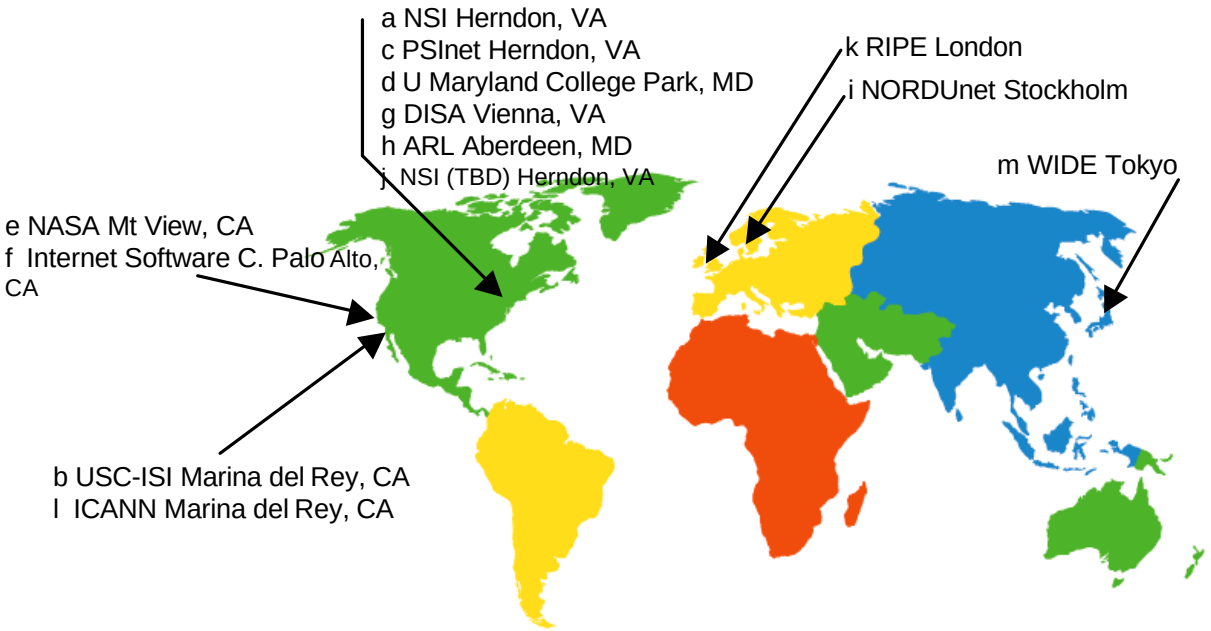
Example:

- surf.eurecom.fr wants to talk to gaia.cs.umass.edu
  - contact local dns server
  - local dns contacts root
  - root contacts authoritative (or next level to authoritative)



DNS Root Name Servers

- contacted by local name server that can not resolve any part of the name
- root name server:
  - contacts authoritative name server if name mapping not known
  - gets mapping
  - returns mapping to local name server



13 root name servers worldwide (all that fit in a 512 octet SOA record)

DNS Server Name Database

- DB contains tuples called resource records (RRs)
  - RR contains type, class and application data
    - Before types added, only one record type (A)
  - Classes = Internet (IN), Chaosnet (CH), etc.
  - Each class defines types, e.g. for IN:
    - A = address
    - NS = name server
    - CNAME = canonical name (for aliasing)
    - HINFO = CPU/OS info
    - MX = mail exchange
    - PTR = pointer for reverse mapping of address to name
- nslookup example

DNS Record Types

Resource records (RR) and their types

RR format: **(name, value, type, ttl)**

- Type=A
  - **name** is hostname
  - **value** is IP address
- Type=NS
  - **name** is domain (e.g. foo.com)
  - **value** is IP address of authoritative name server for this domain
- Type=CNAME
  - **name** is an alias name for some "canonical" (the real) name
  - **value** is canonical name
- Type=MX
  - **value** is hostname of mailserver associated with **name**

DNS Database

•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.

DNS DB Example

- 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

DNS MX Record Type

- MX records point to mail exchanger for a name
  - E.g. mail.acm.org is MX for acm.org
- Addition of MX record type proved to be a challenge
  - How to get mail programs to lookup MX record for mail delivery rather than A record?
  - Needed critical mass of such mailers

DNS Server Database Distribution

- Administrative hierarchy
  - Organized into regions known as “zones”
  - “.” as separator
  - zone = contiguous section of name space
- Zones created by convincing owner node to delegate a subzone
  - umass.edu zone delegates cs.umass.edu to a different set of authoritative name servers
  - Each zone contains multiple redundant servers
    - Primary (master) name server updated manually
    - Secondary (redundant) servers updated by zone transfer of name space
    - Provides fault-tolerance within zone
- Host name to address section
  - Top-level domains → edu, gov, ca, us, etc.
  - Sub-domains = subtrees
  - Human readable name = leaf → root path

DNS Client Lookups

- Each host has a resolver
  - Typically a library that applications can link gethostbyname()
  - Local name servers hand-configured (e.g. /etc/resolv.conf) or automatically configured (DHCP)
    - Can specify a file /etc/hosts
    - Can specify a name server by its IP address (i.e. 129.95.50.2)
  - Host queries local name server for unknown names

```
state > more /etc/resolv.conf
domain cse.ogi.edu
nameserver 129.95.40.4
nameserver 129.95.90.19
nameserver 129.95.40.2
state >
```

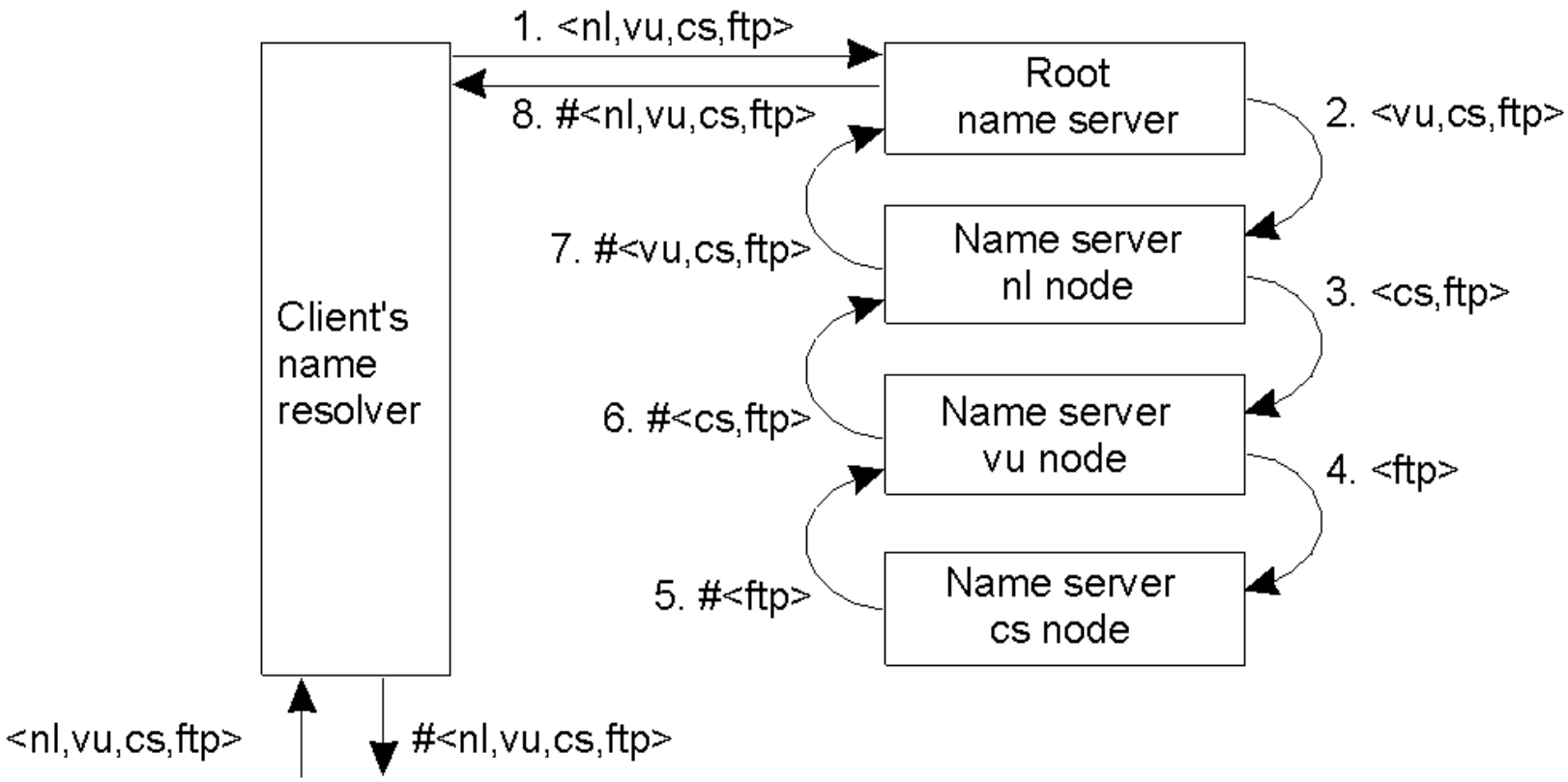
DNS Name Servers

- Configured with well-known root servers
  - Currently {a-m}.root-servers.net
- Local servers
  - Typically answer queries about local zone
  - Typically do a lookup of distant host names for local hosts

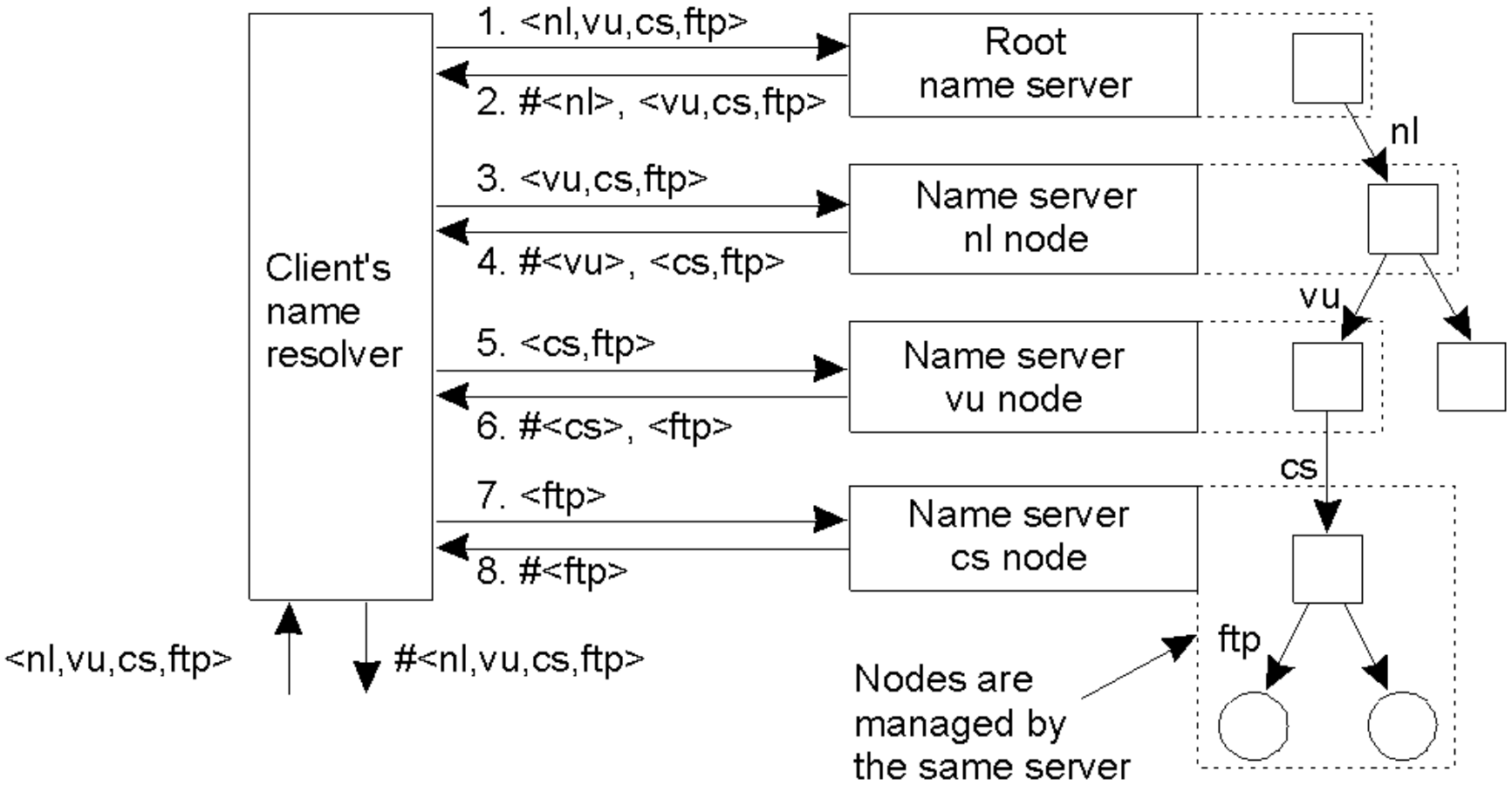
Lookup Methods

- Recursive queries
  - Server goes out and searches for more info on behalf of the client (recursive)
  - Only returns final answer or “not found”
  - Puts burden of name resolution on contacted name server
  - Heavy load?
    - Root server implosion
- Iterative
  - Server responds with as much as it knows (i.e. name of server to contact next)
  - “I don’t know this name, but ask this server”
  - Client iteratively queries additional servers

Recursive Name Resolution



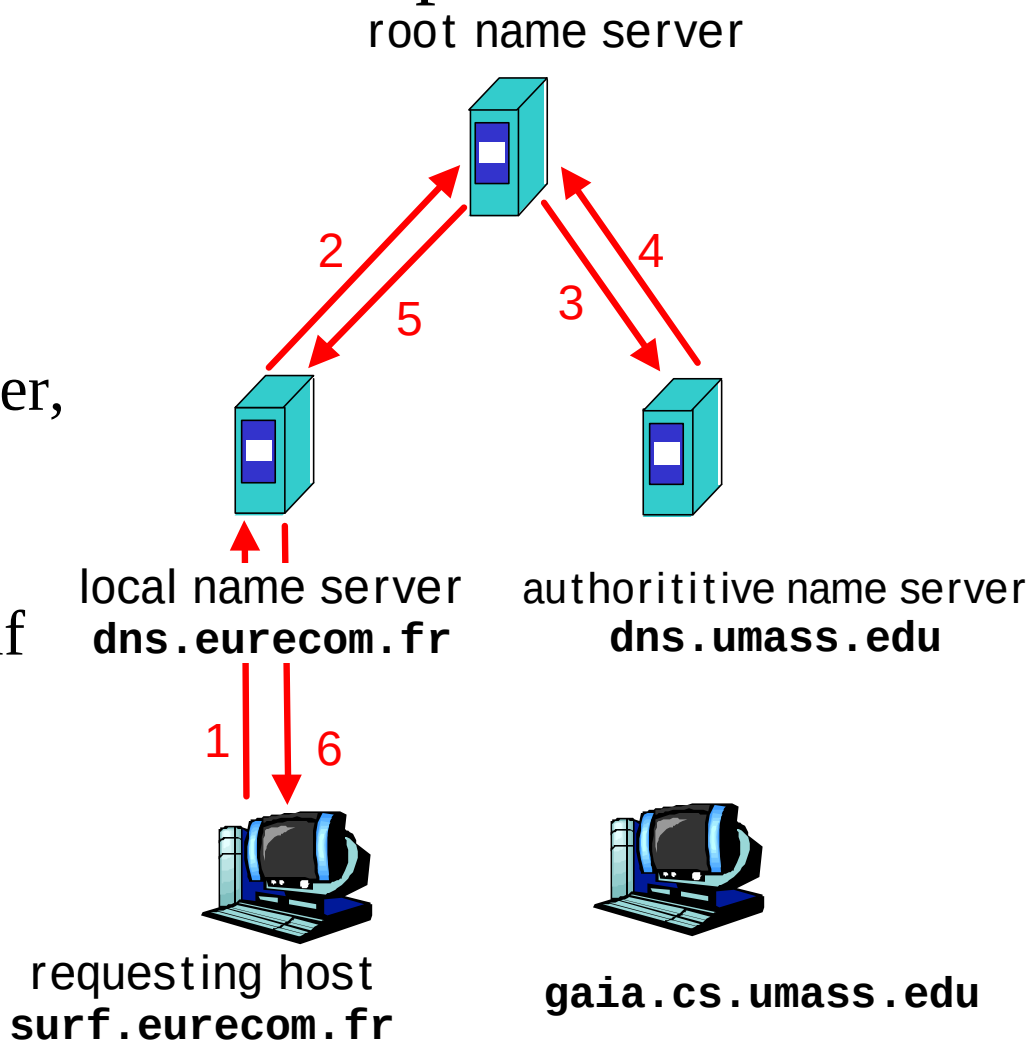
Iterative Name Resolution



Recursive DNS Example

host **`surf.eurecom.fr`**
wants IP address of
`gaia.cs.umass.edu`

1. Contacts its local DNS server, **`dns.eurecom.fr`**
2. **`dns.eurecom.fr`** contacts root name server, if necessary
3. root name server contacts authoritative name server, **`dns.umass.edu,`** if necessary



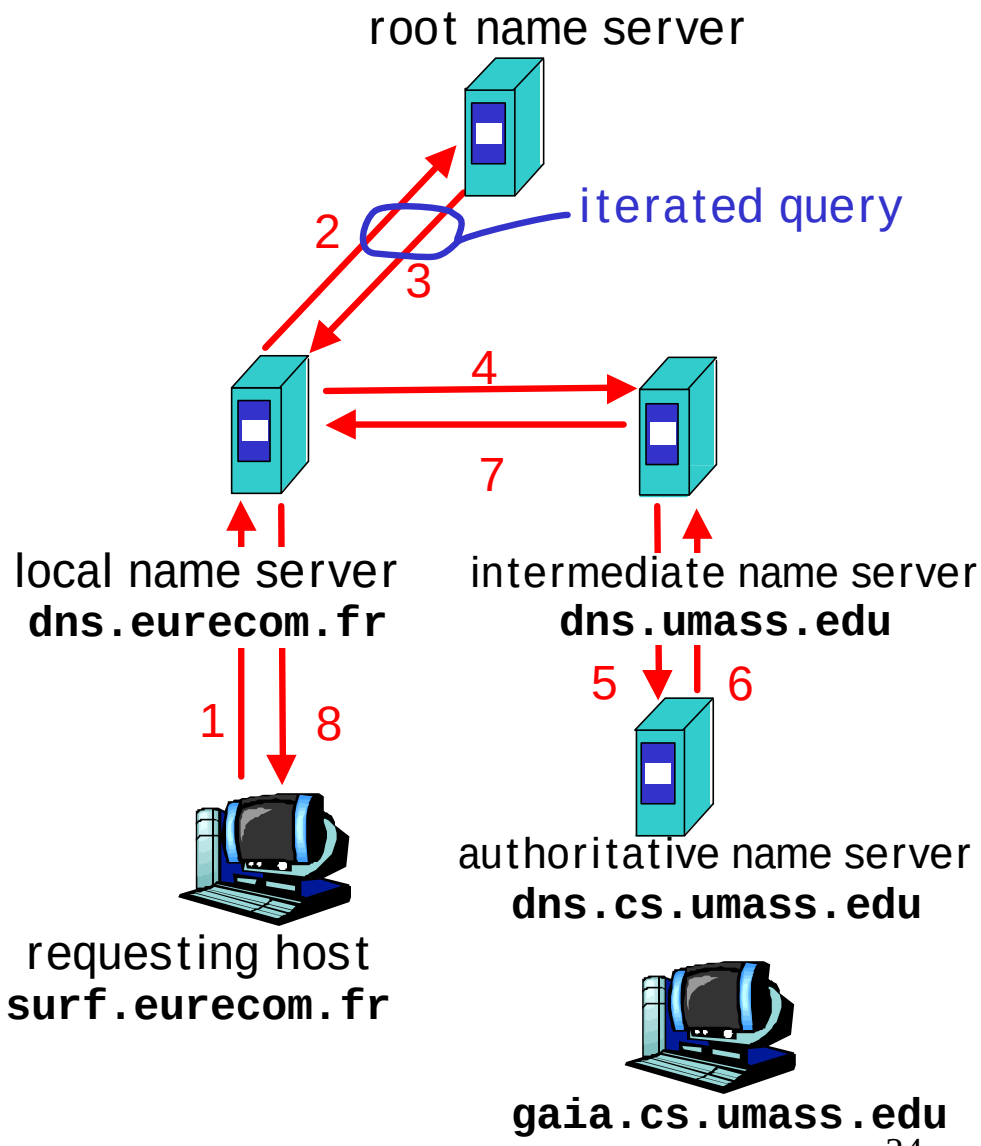
DNS Iterated Queries

recursive query:

- puts burden of name resolution on contacted name server
- heavy load?
- root servers now disable recursive queries (RFC 2010)

iterated query:

- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



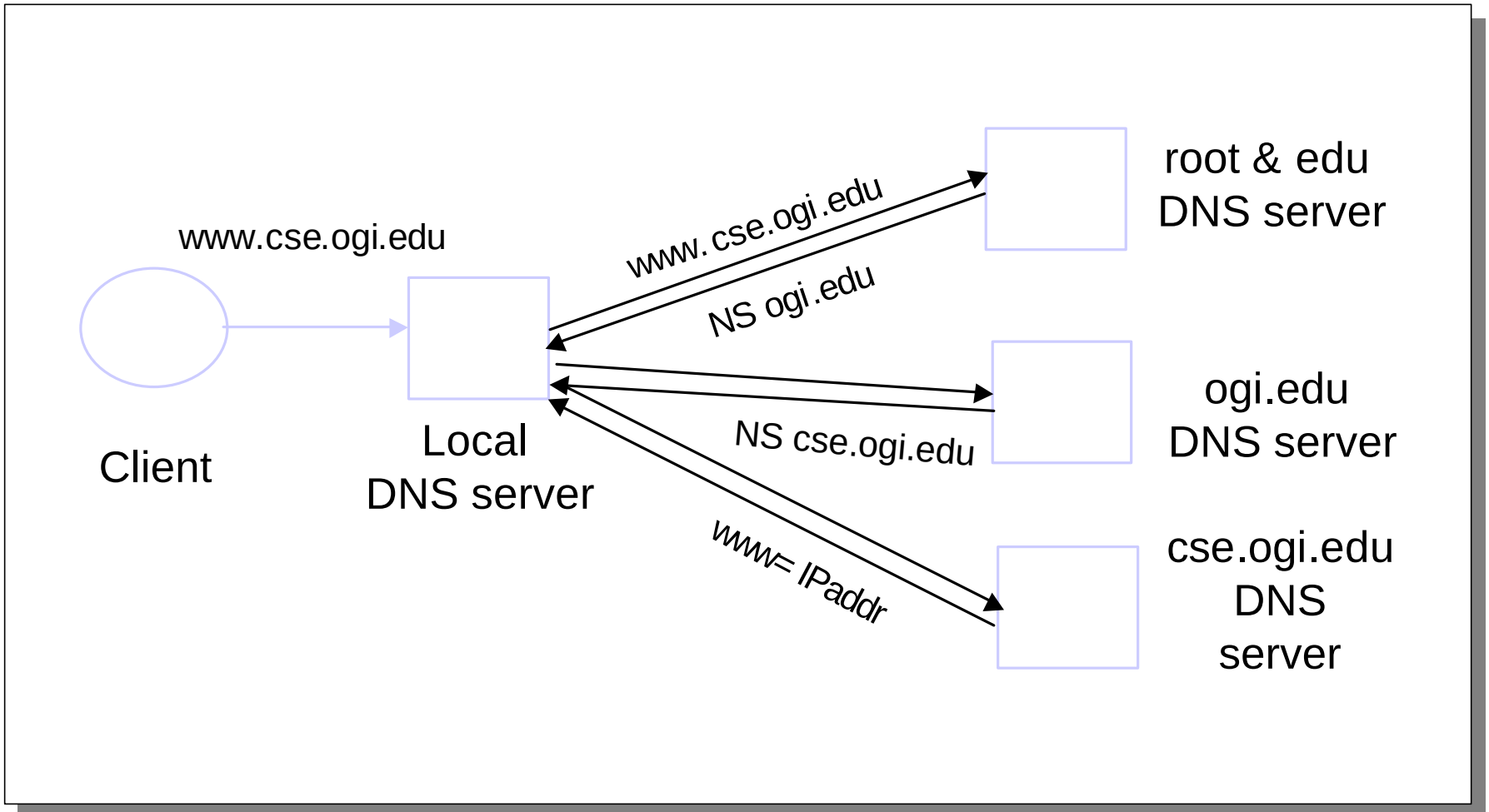
Typical Resolution

- Client does recursive request to local name server
- Local name server does iterative requests to find name
- Local name server has knowledge of root servers
- Steps for resolving www.ogi.edu
  - Application calls gethostbyname()
  - Resolver contacts local name server (S_1)
  - S_1 queries root server (S_2) for (www.ogi.edu)
  - S_2 returns NS record for ogi.edu (S_3)
  - S_1 queries S_3 for www.ogi.edu
  - S_3 returns A record for www.ogi.edu
- Can return multiple addresses → what does this mean?

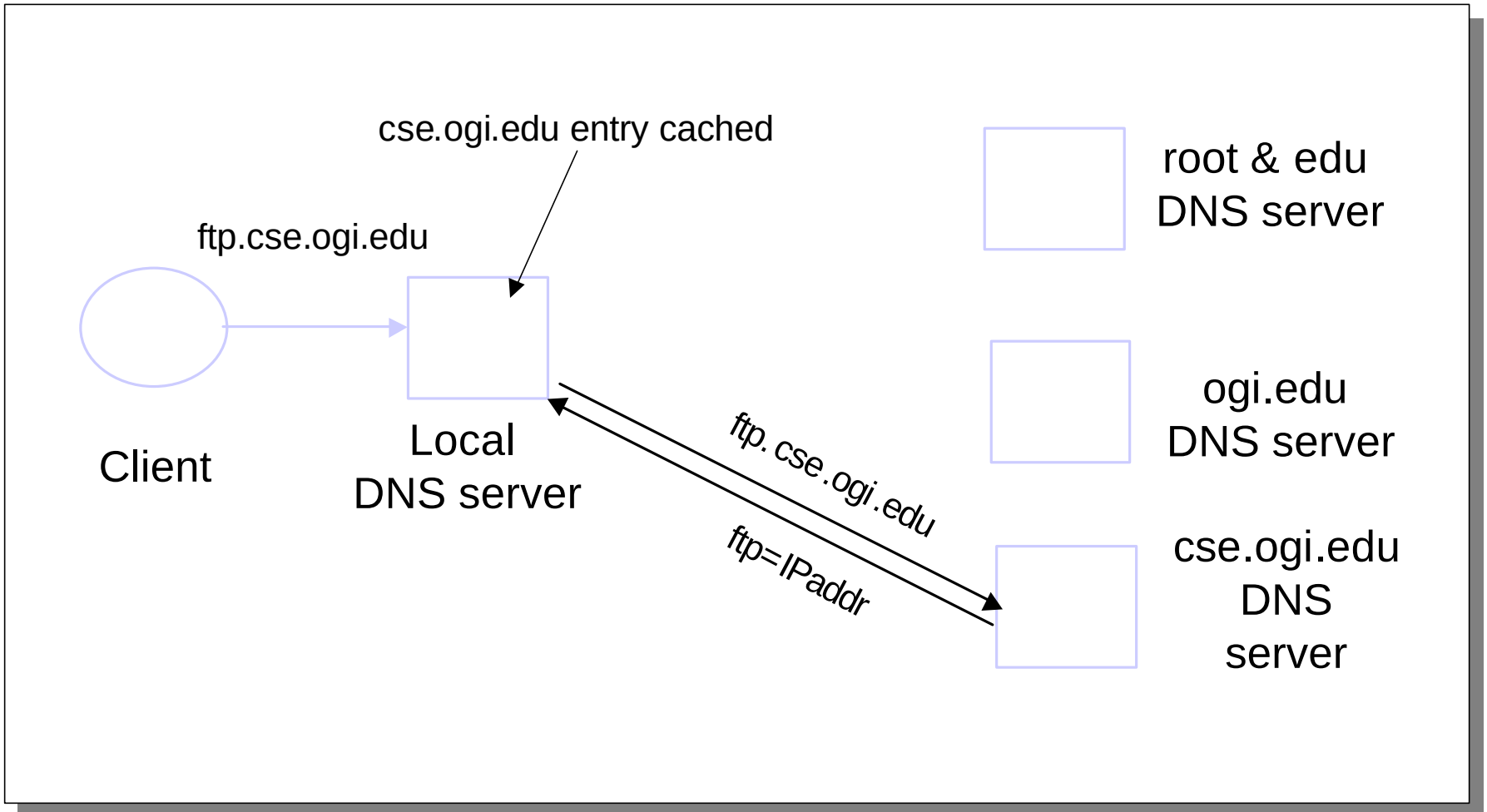
DNS Caching

- DNS responses are cached
  - Quick response for repeated translations
  - Other queries may reuse some parts of lookup
    - NS records for domains
- DNS negative queries are also cached
  - Don’t have to repeat past mistakes
  - E.g. misspellings
- Cached data periodically times out
  - Soft state
  - Lifetime (TTL) of data controlled by owner of data
  - TTL passed with every record
  - TTL affects DNS-based load balancing techniques
- update/notify mechanisms under design by IETF
  - RFC 2136
  - http://www.ietf.org/html.charters/dnsind-charter.html

DNS Lookup Example



Subsequent Lookup Example



A word about iterated queries and caching

- Why not do iterative queries from host and recursive queries from local DNS server?
  - Win2k client
    - Does iterative queries from host
  - Caching implications?

Implications of Lookup Methods on Caching

•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.

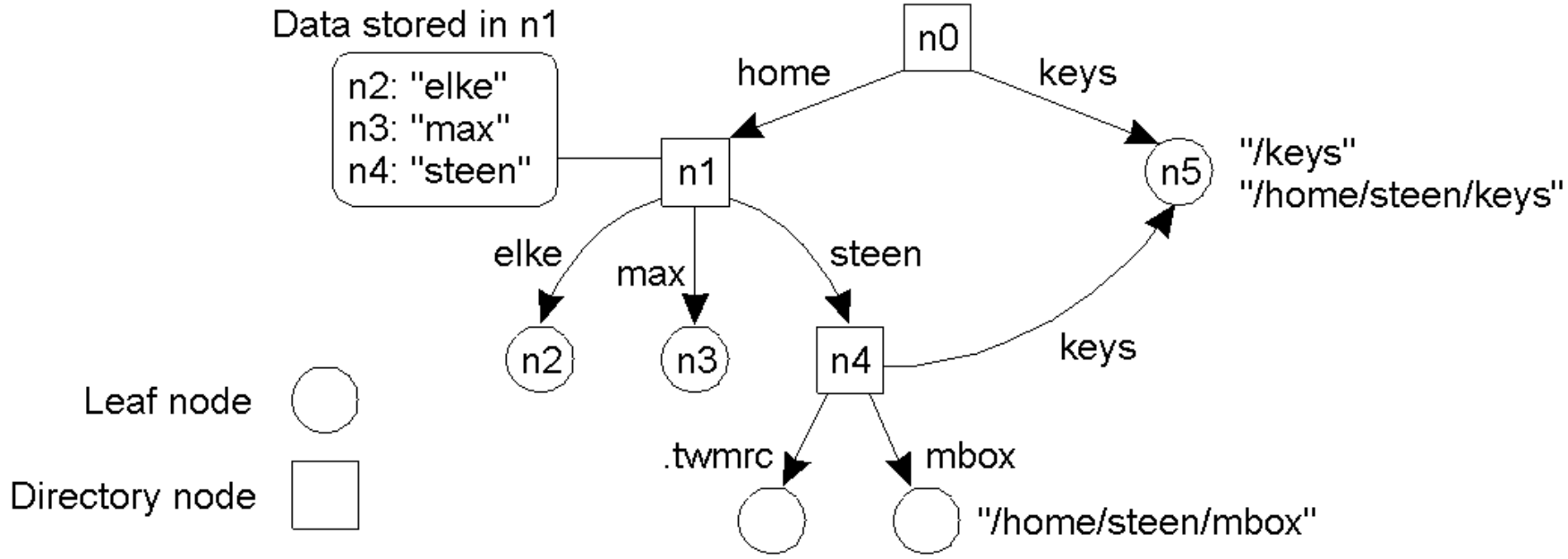
DNS issues

- Static configuration (root server list)
- Inflexible name space
- Lack of support for mobile entities
- Lack of exponential back-off / congestion control
- UDP used for queries
  - Need reliability → Why not TCP?
- Vulnerability of 13 TLD servers

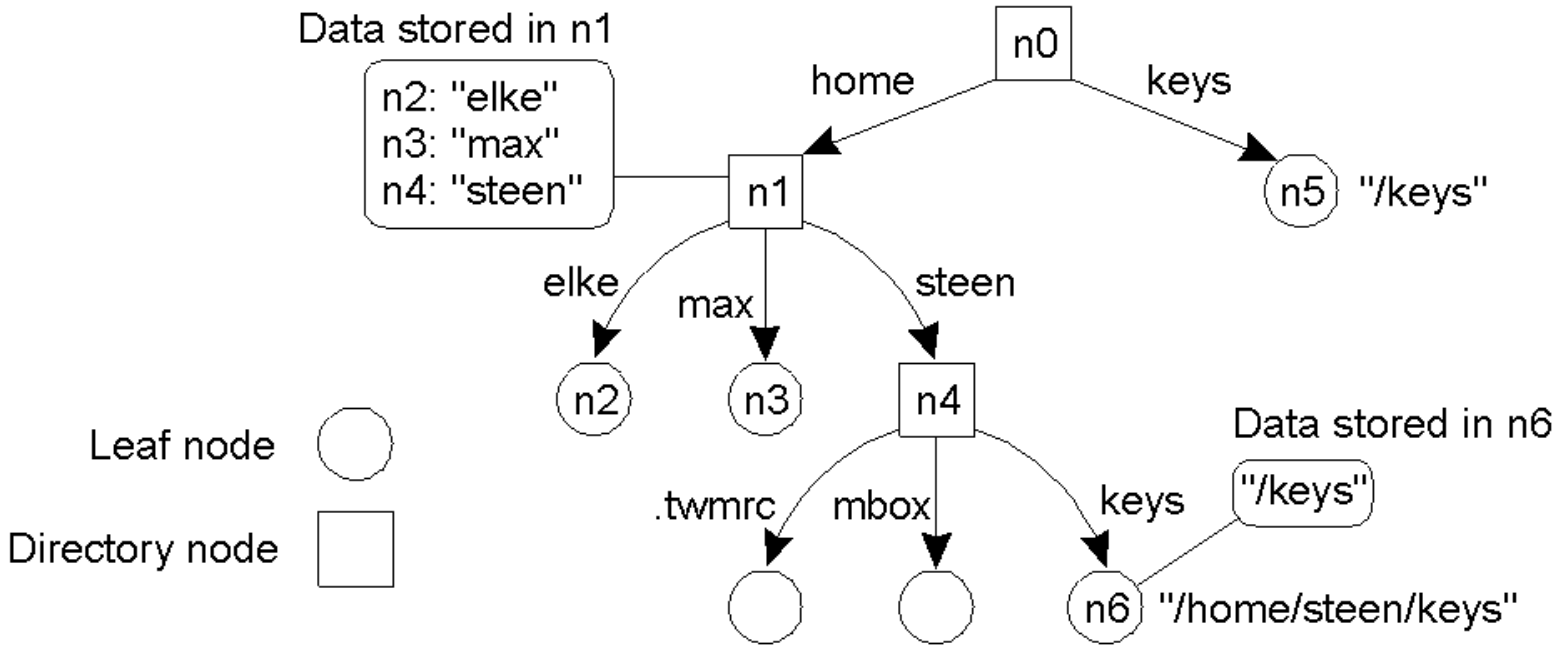
General Naming Issues

Name Spaces (1)

- A general naming graph with a single root node.

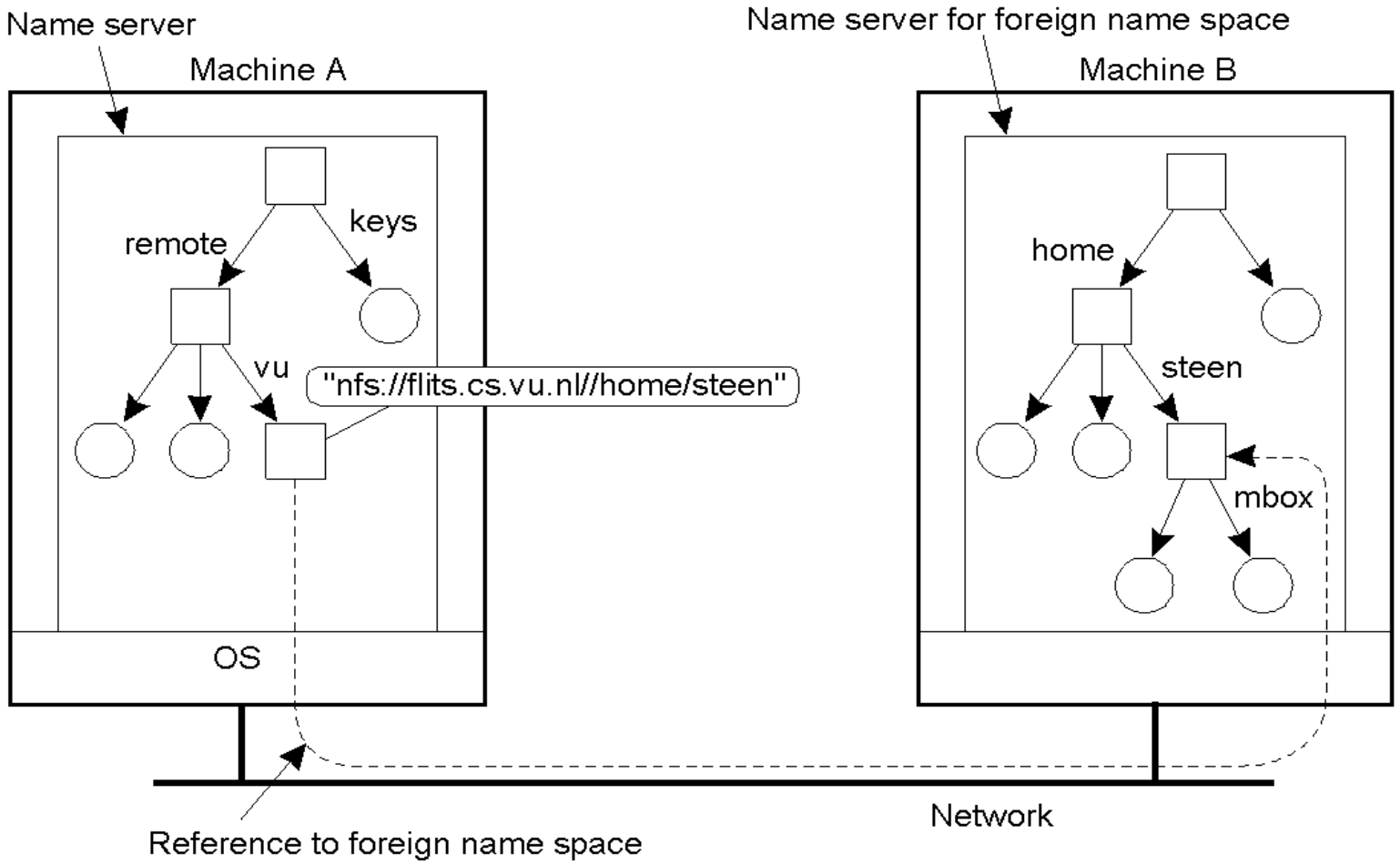


Linking and Mounting (1)

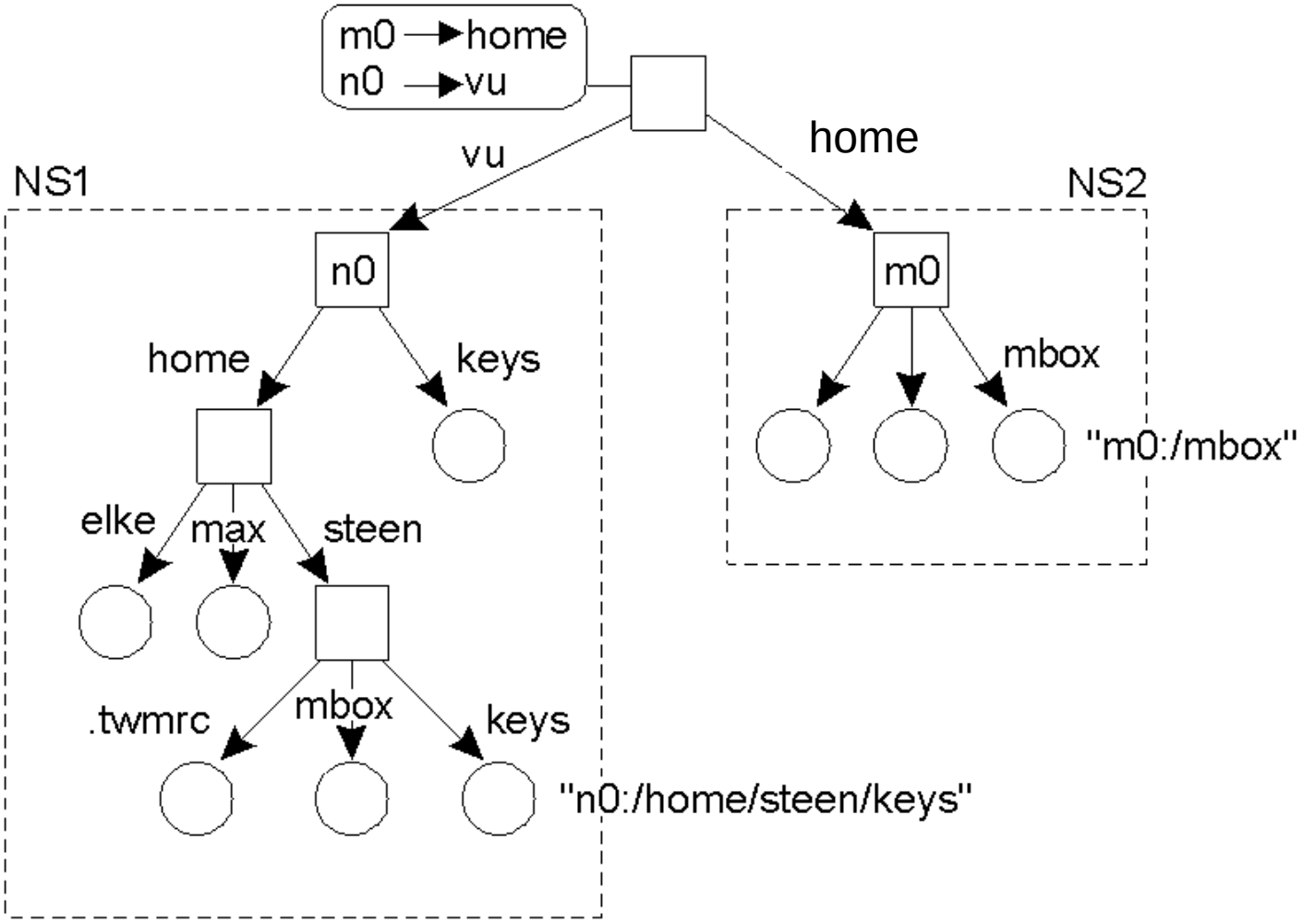


- The concept of a symbolic link explained in a naming graph.

Linking and Mounting (2)

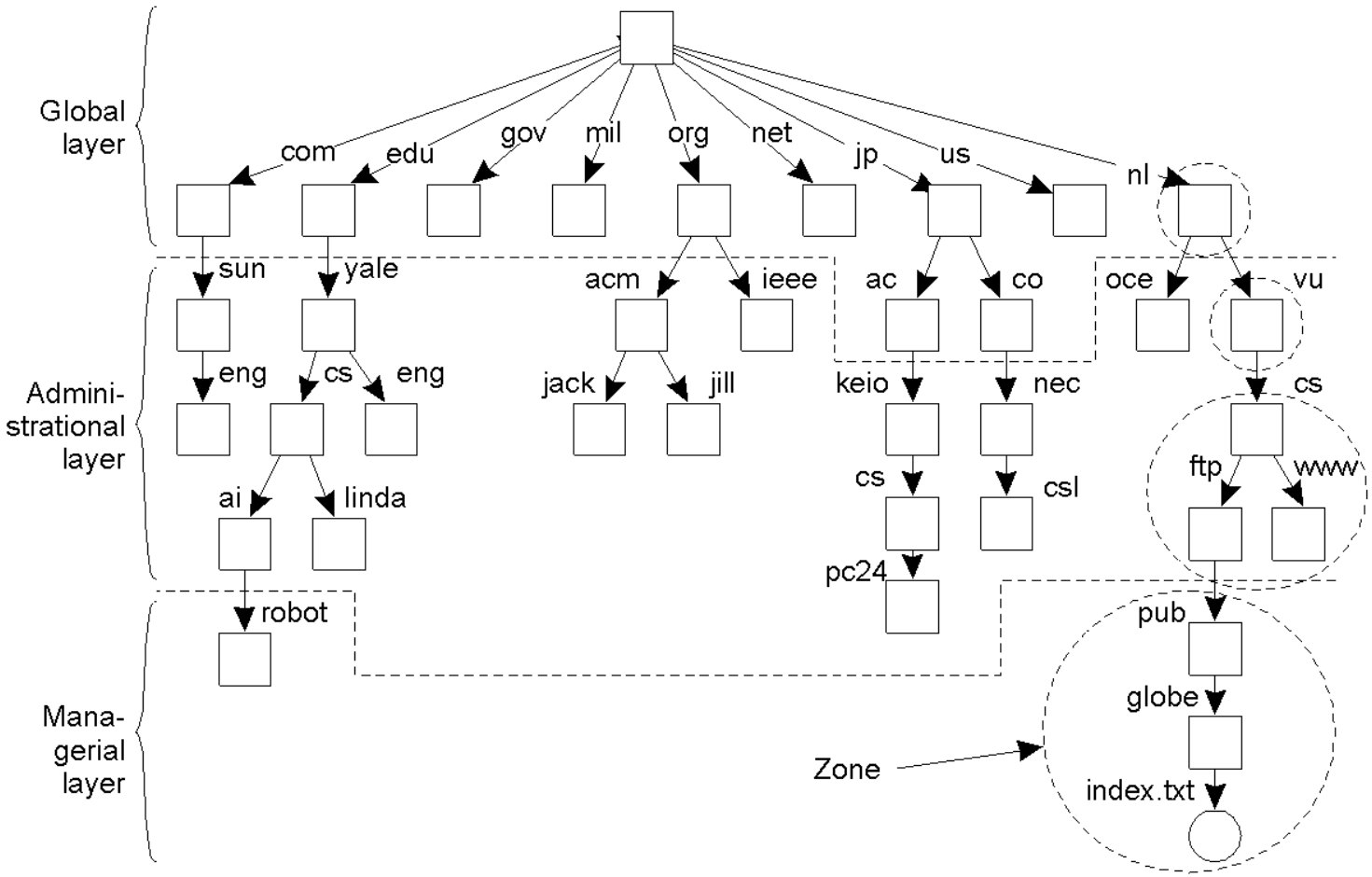


Linking and Mounting (3)



- Organization of the DEC Global Name Service

Name Space Distribution (1)



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

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 administrational layer, and a managerial layer.

Directory Services

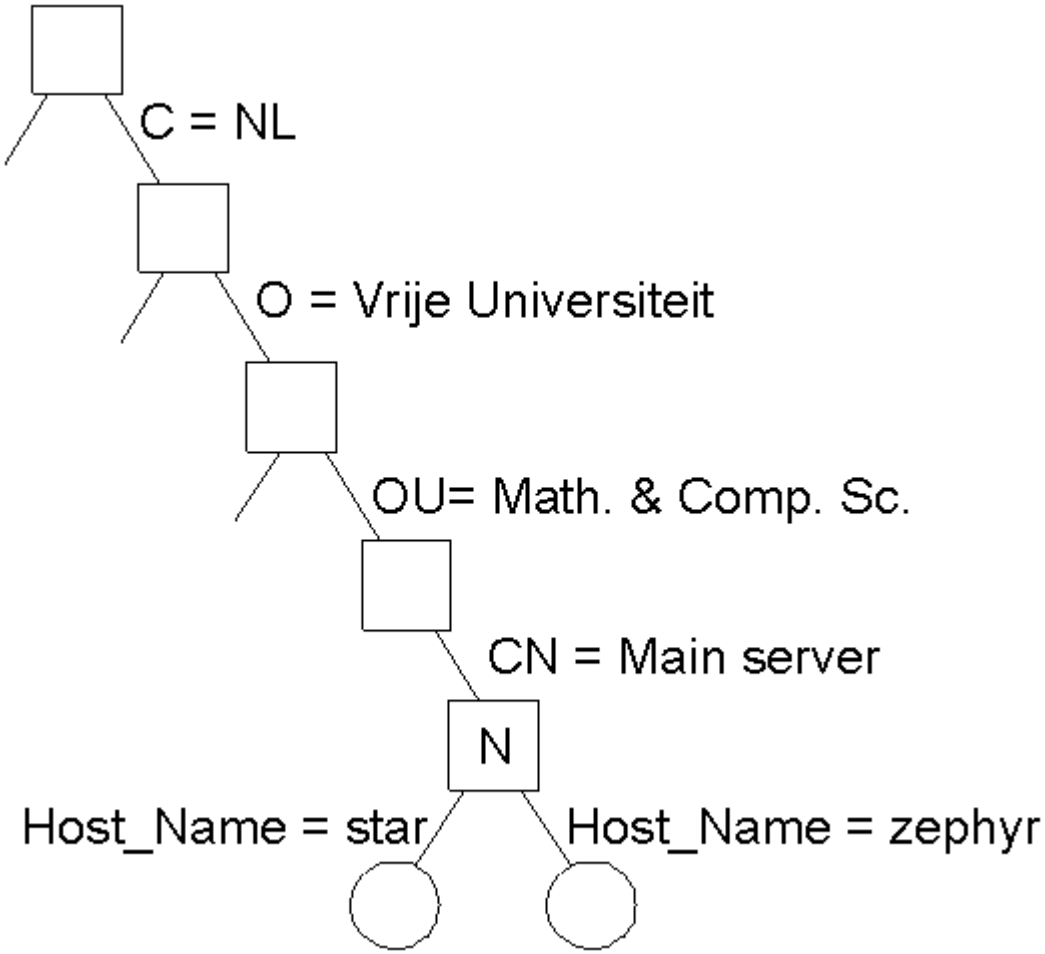
The X.500 Name Space (1)

•**Attribute**	•**Abbr.**	•**Value**
•Country	•C	•NL
•Locality	•L	•Amsterdam
•Organization	•O	•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.

The X.500 Name Space (2)

- Part of the directory information tree.



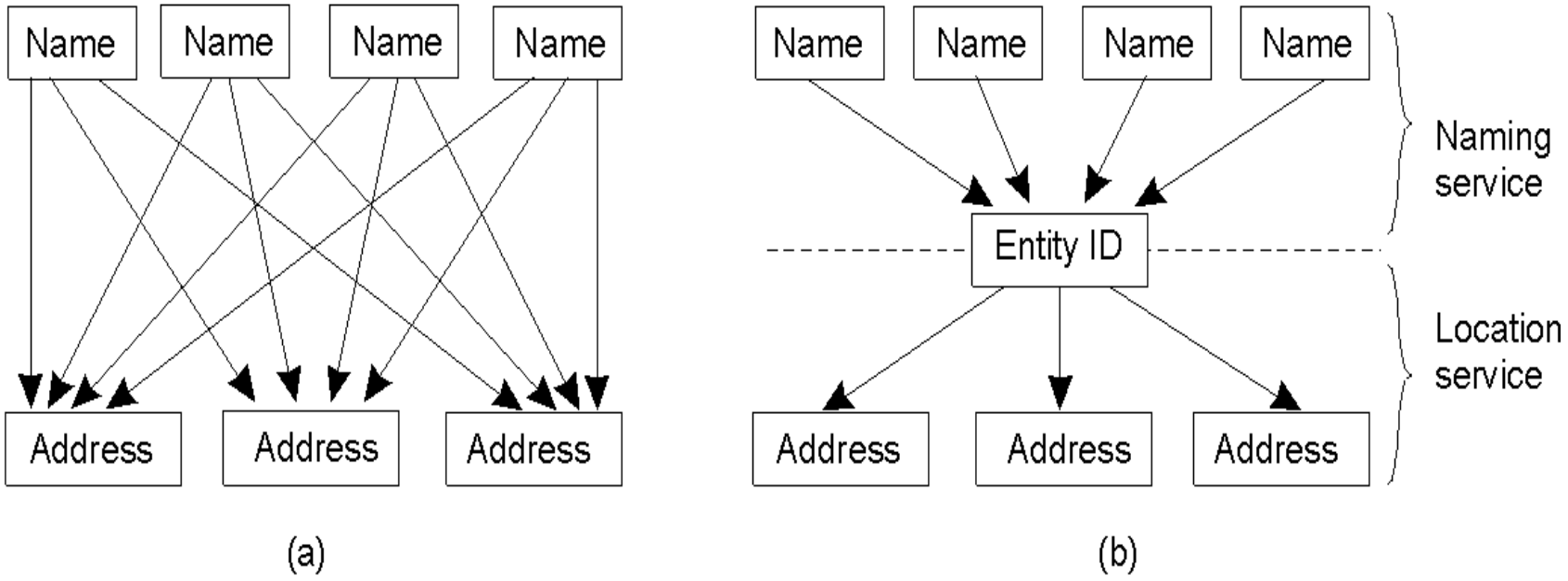
The X.500 Name Space (3)

•**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

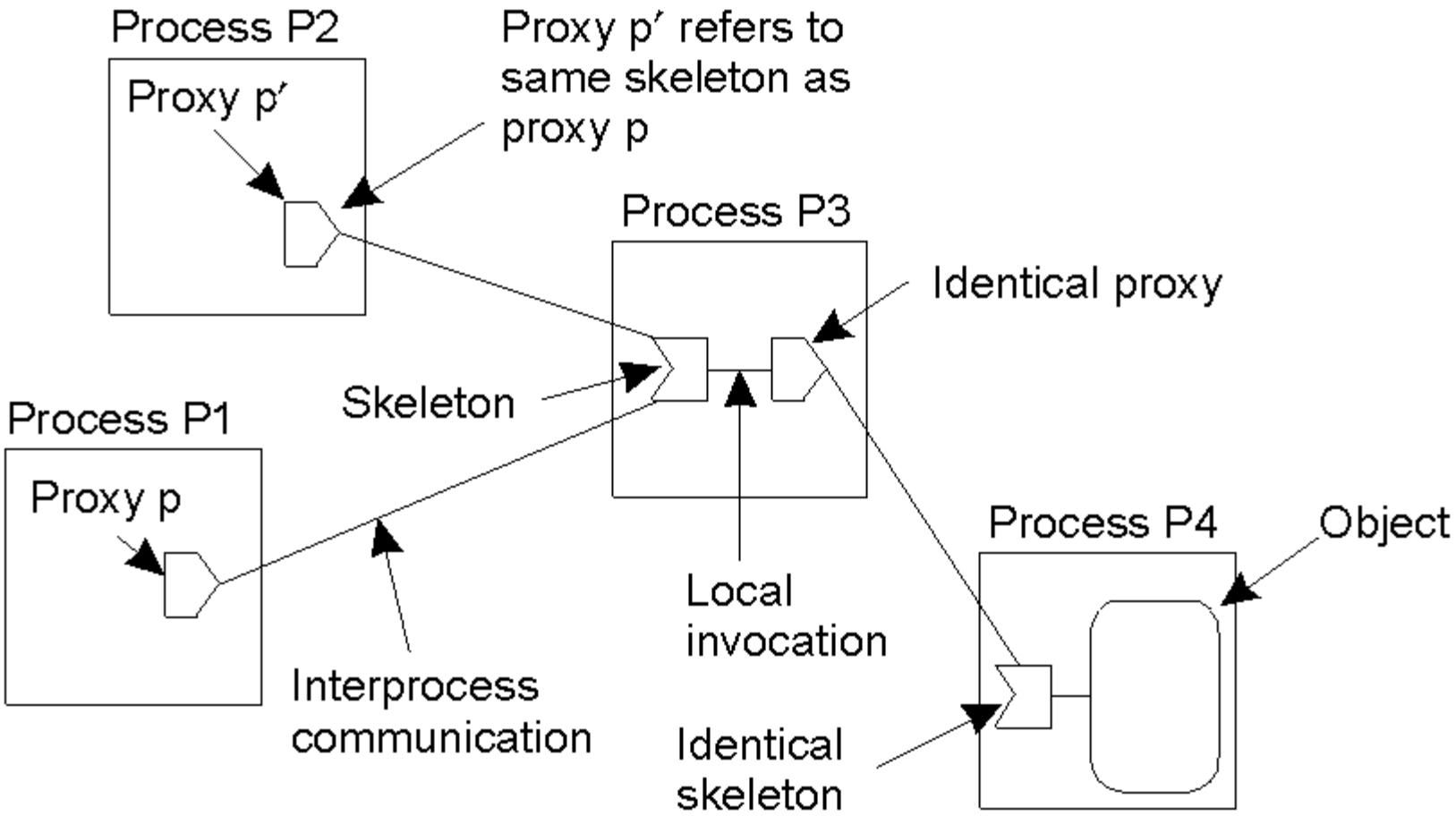
- Two directory entries having *Host_Name* as RDN.

Naming versus Locating Entities

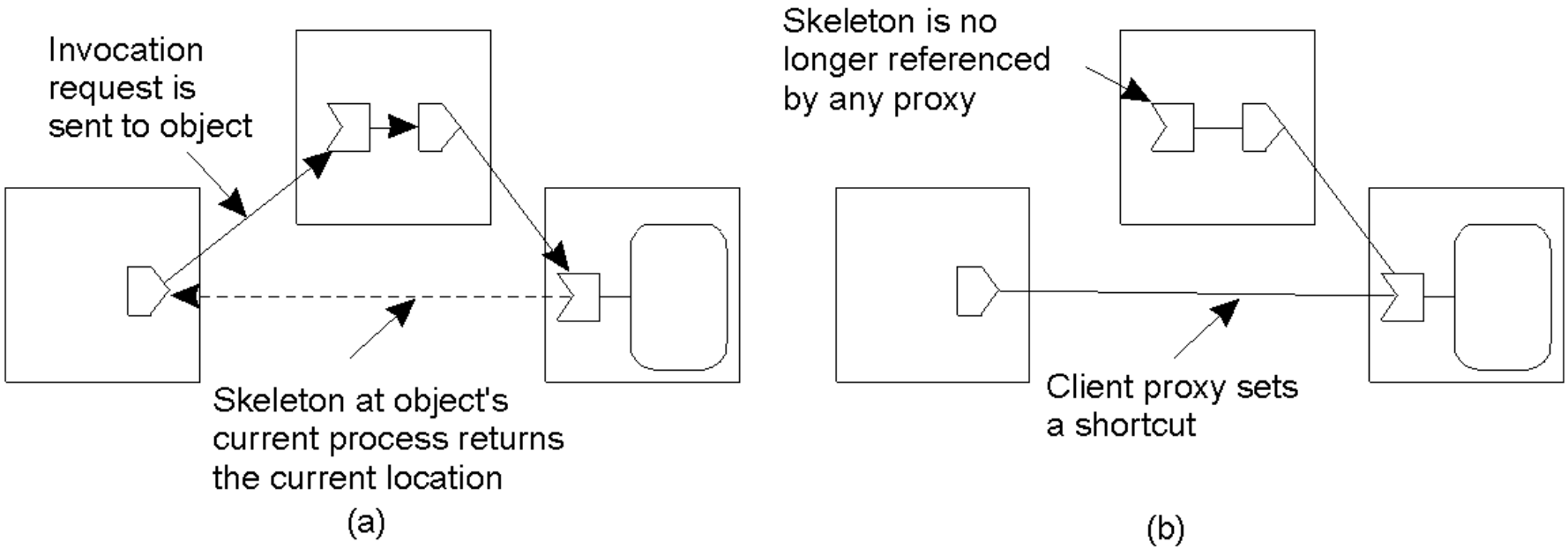


- Direct, single level mapping between names and addresses.
- Two-level mapping using identities.

Forwarding Pointers (1)

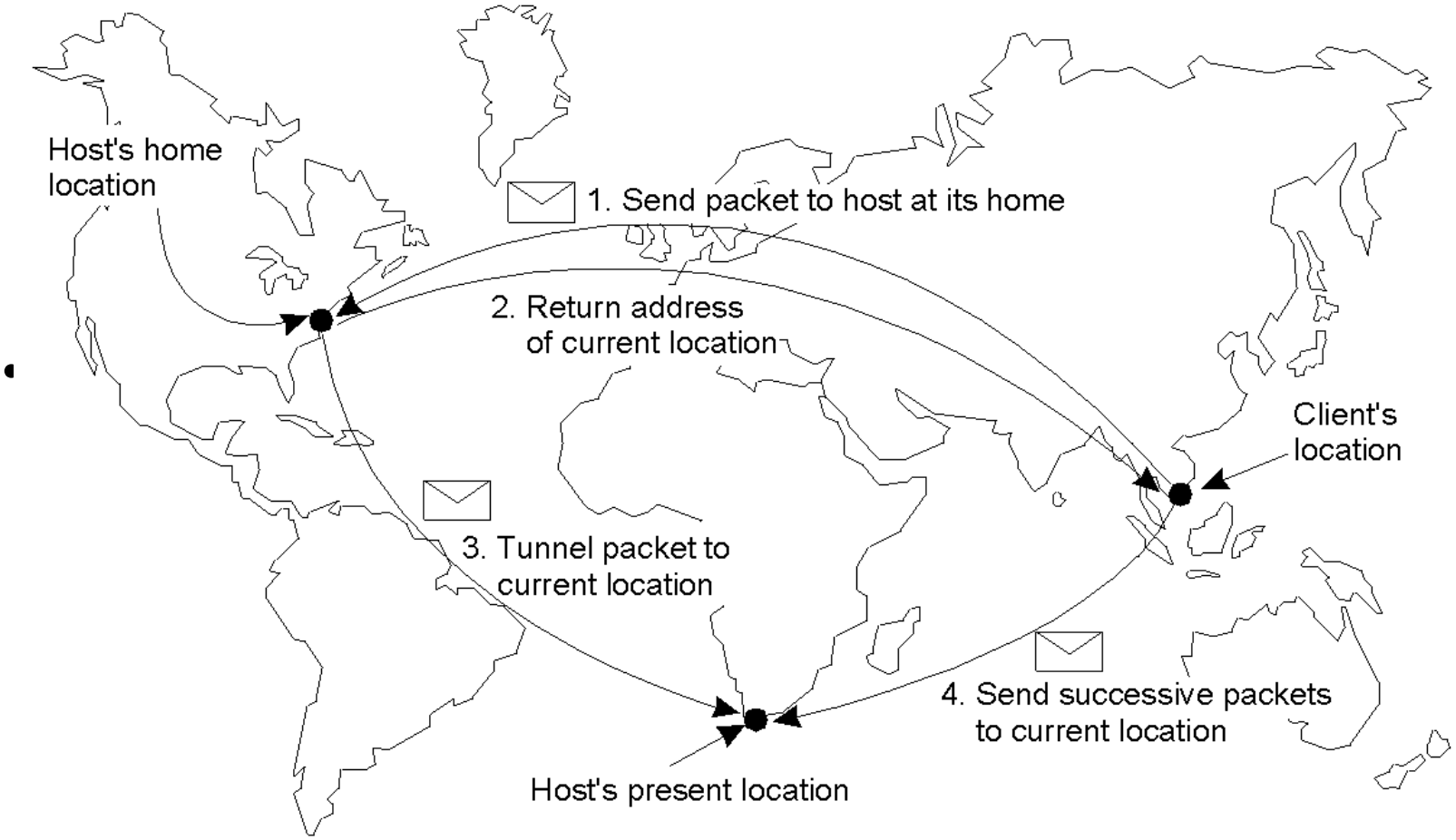


Forwarding Pointers (2)

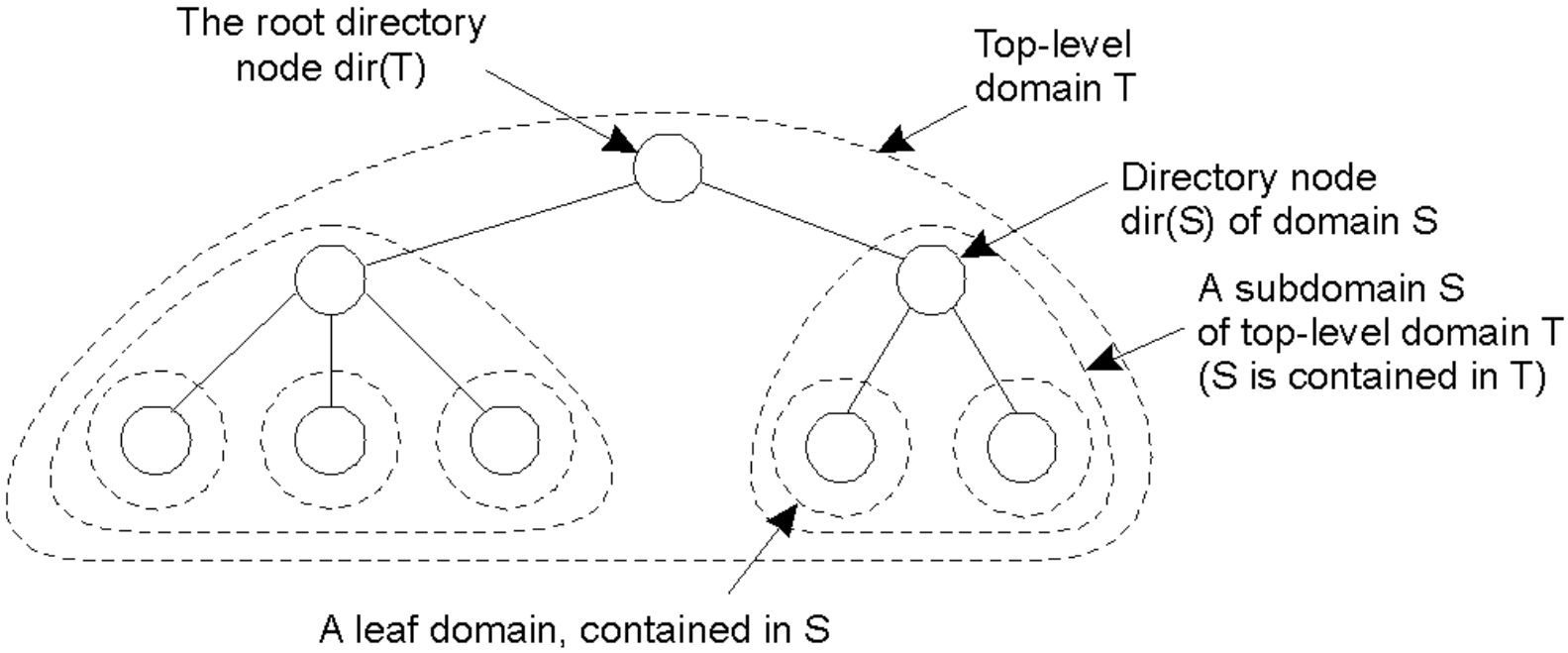


- Redirecting a forwarding pointer, by storing a shortcut in a proxy.

Home-Based Approaches

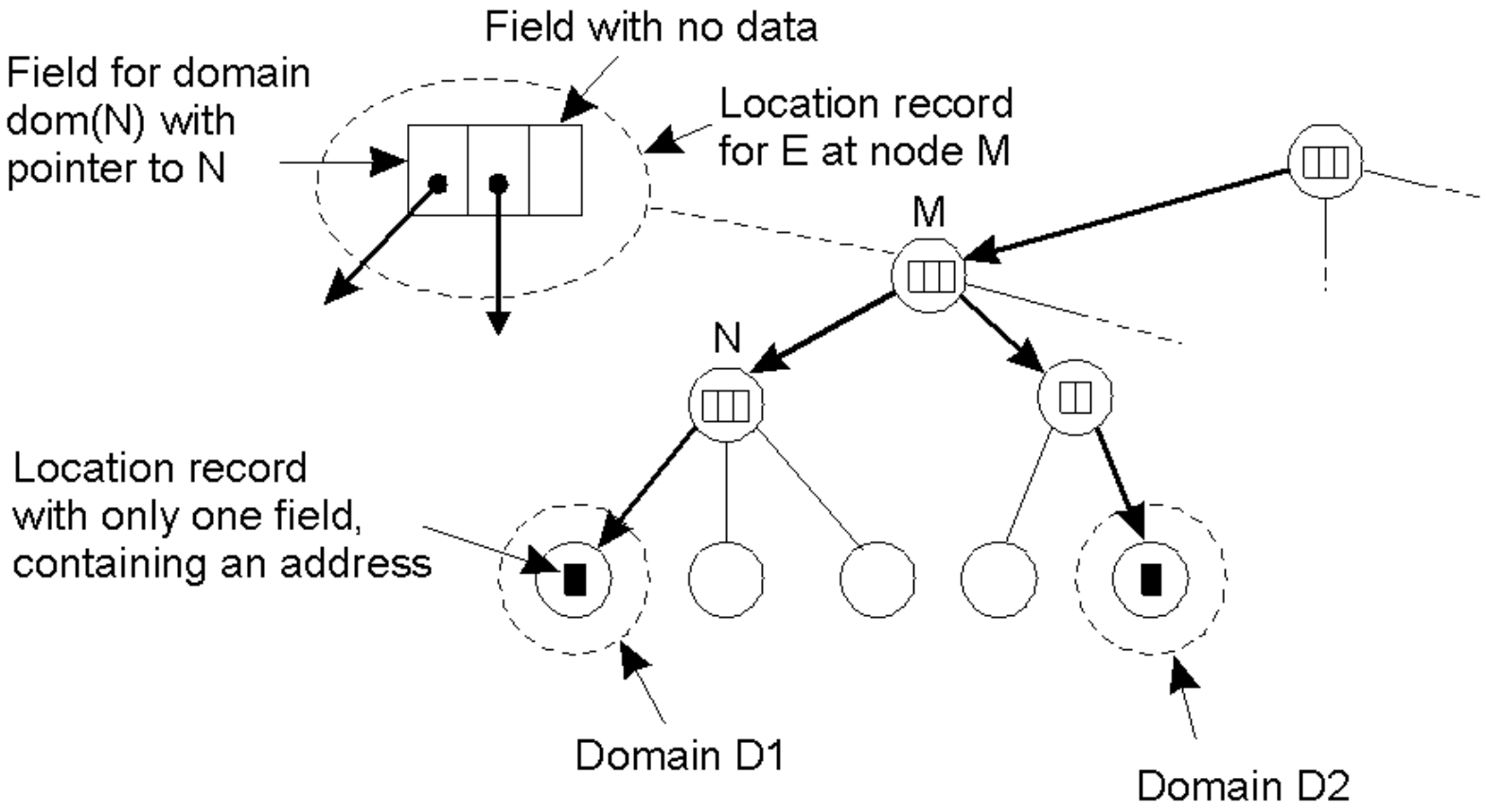


Hierarchical Approaches (1)



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

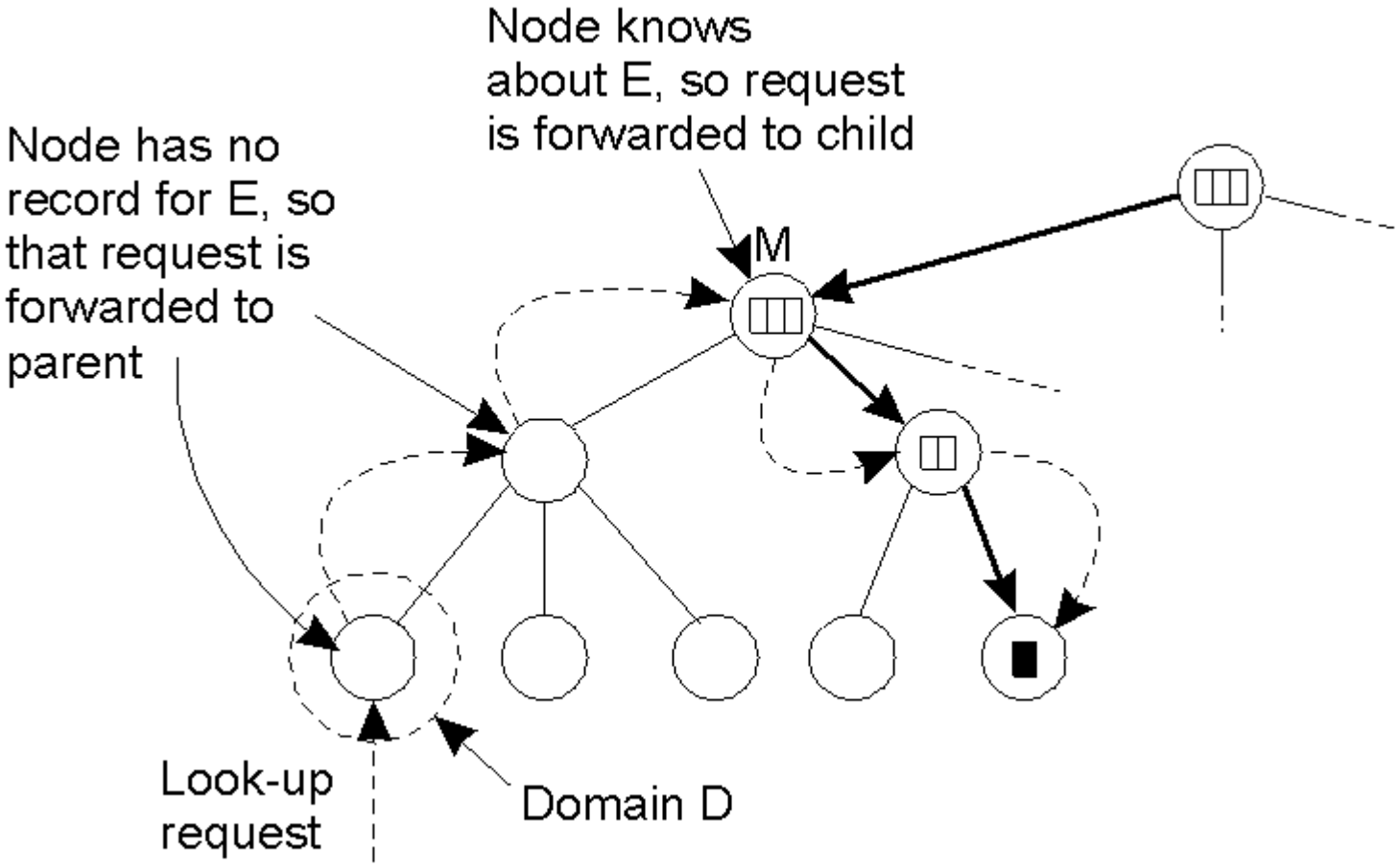
Hierarchical Approaches (2)



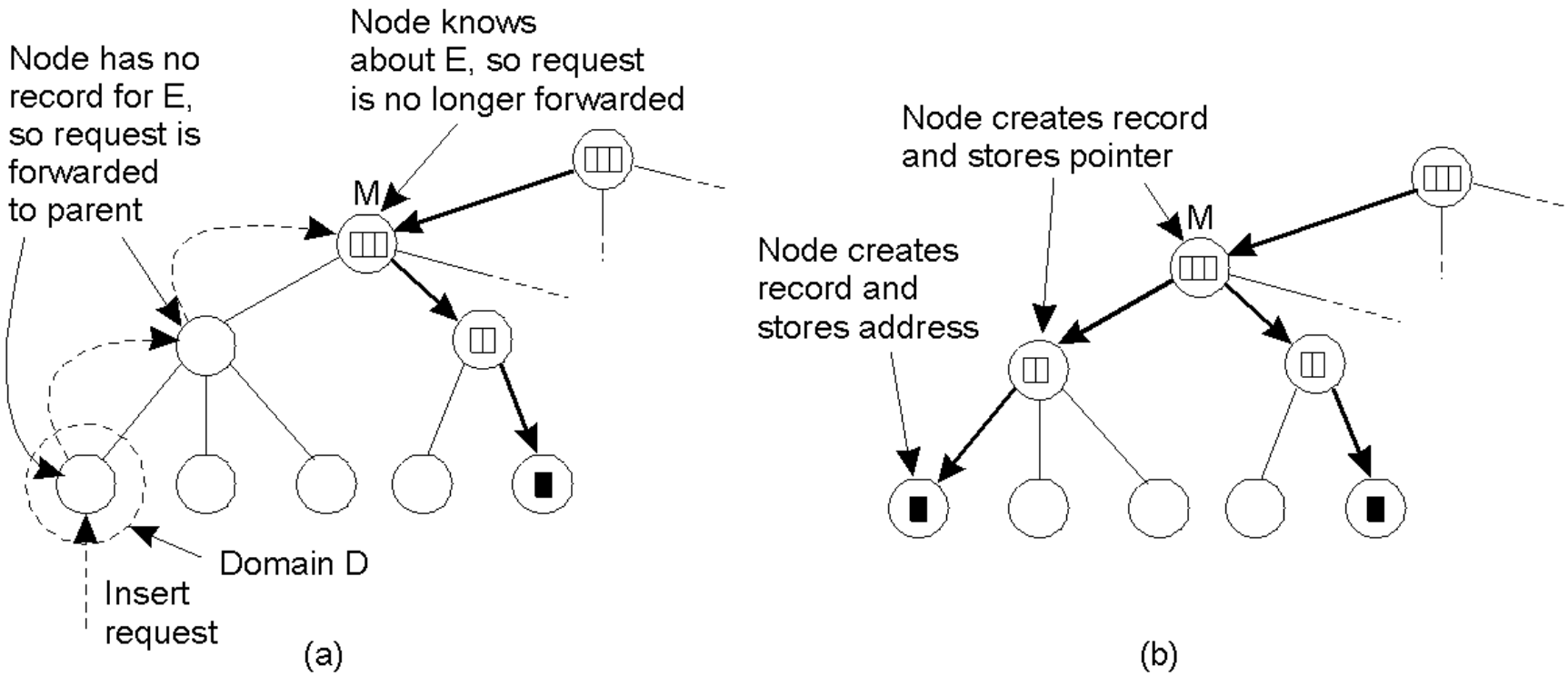
- An example of storing information of an entity having two addresses in different leaf domains.

Hierarchical Approaches (3)

- I



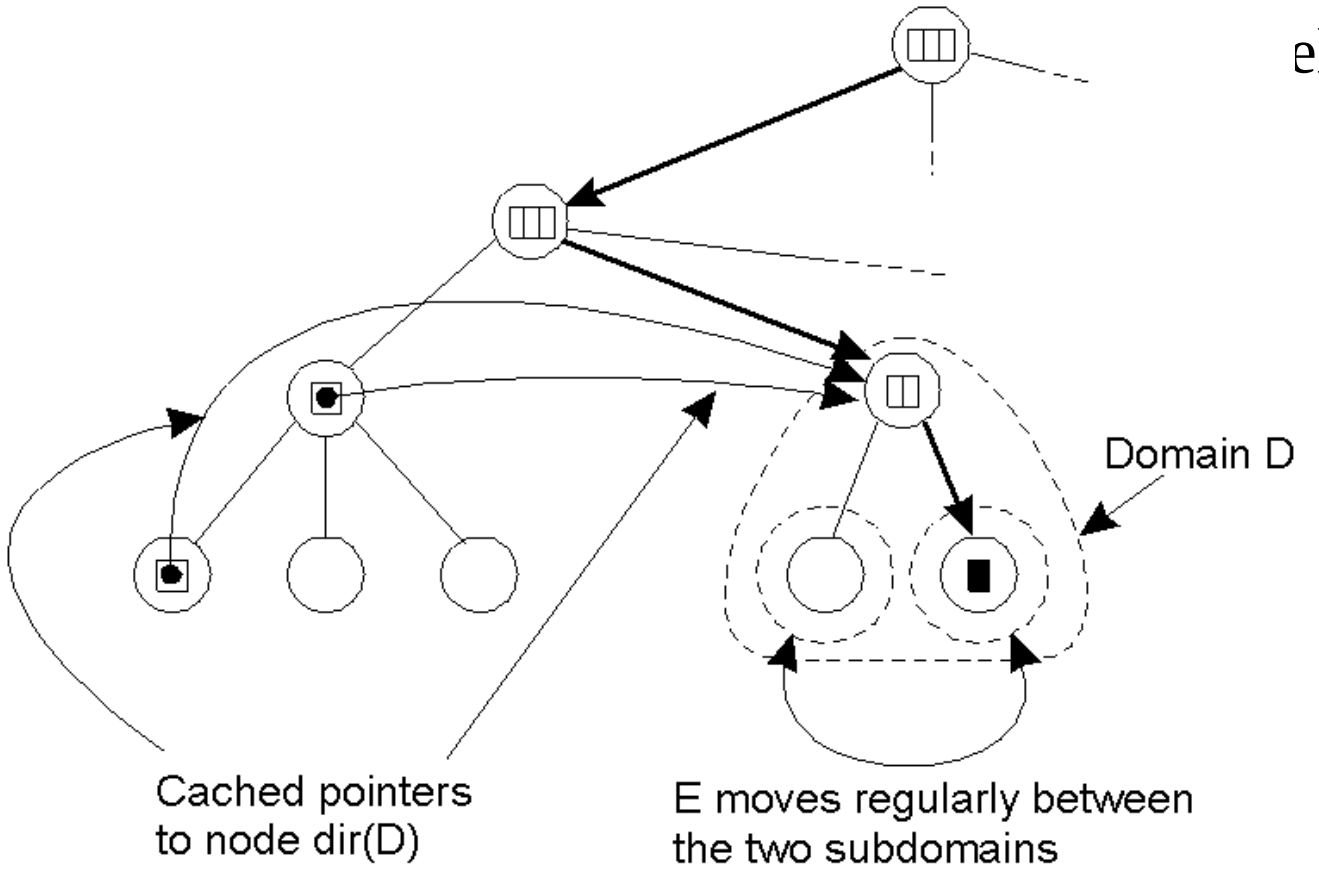
Hierarchical Approaches (4)



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

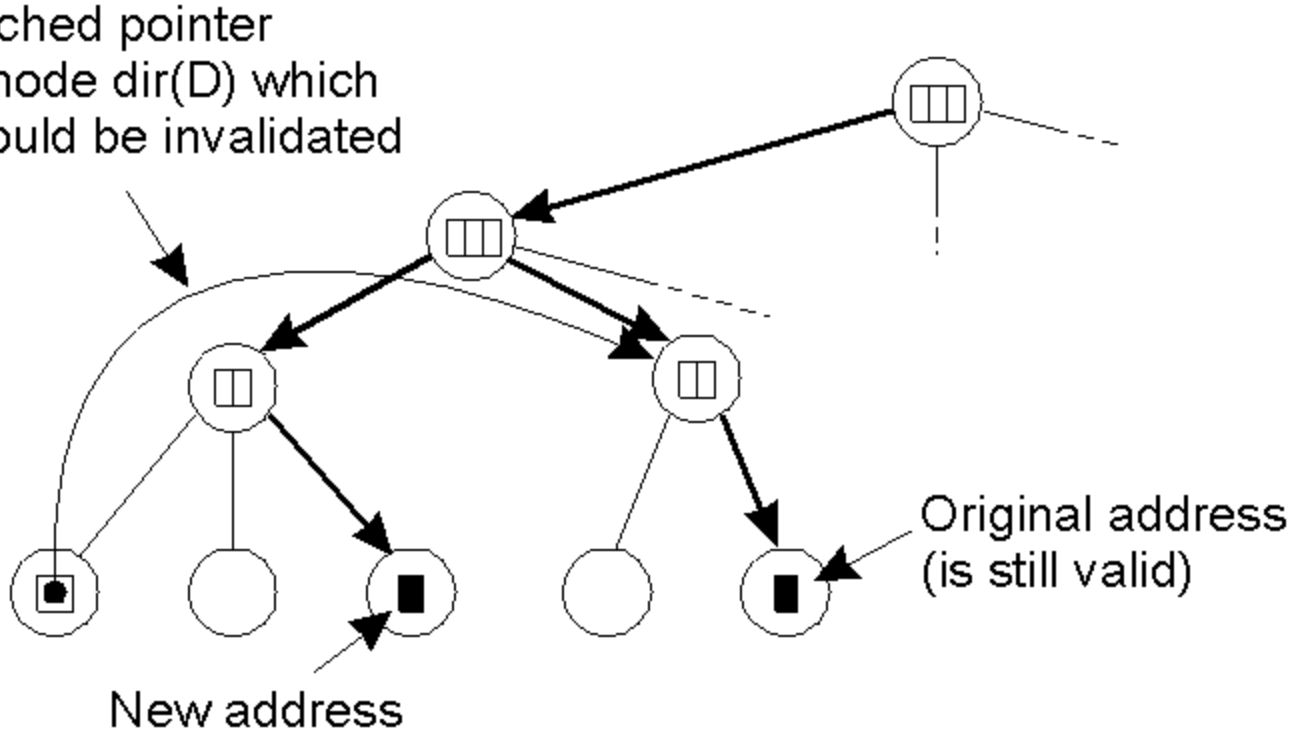
Pointer Caches (1)

- Cachi el
  doma

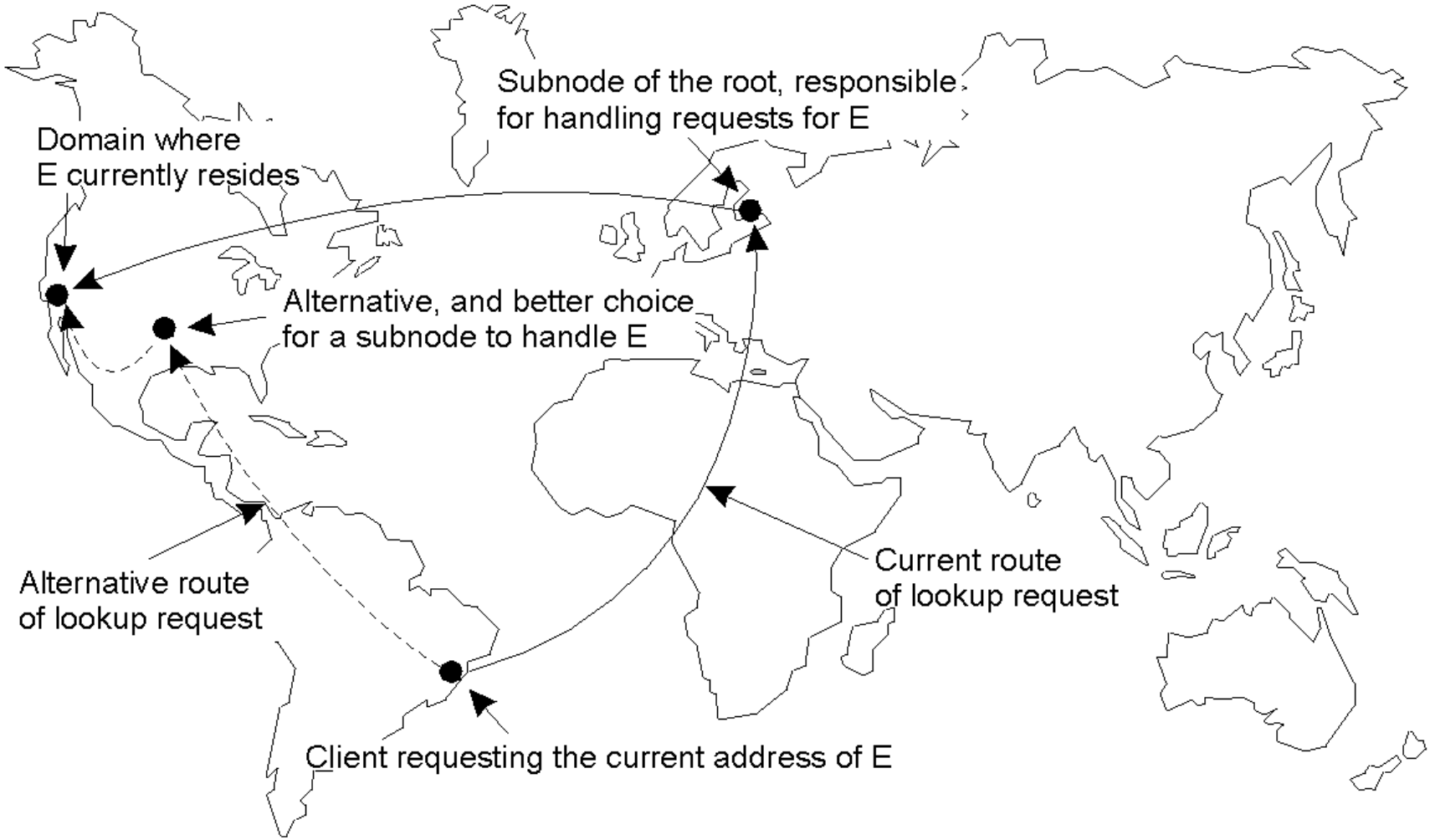


Pointer Caches (2)

- A cache entry that needs to be invalidated because it returns a non



Scalability Issues



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