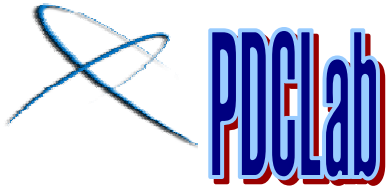


Lecture 4

Naming

Prof. Wilson Rivera

University of Puerto Rico at Mayaguez
Electrical and Computer Engineering Department

Outline

- Names, identifiers, addresses
- Flat naming
- Structured naming
- Attribute based naming

Naming Entities (1)

- Entity → resources, processes, users, network connections, web pages, newsgroups, etc
- Name → string of bits or characters used to refer an entity
- Access point → an entity to operate another entity
- Address → name of an access point
  - An entity may offer more than one access point
  - An entity may change its access point in the course of time

- **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**.

Naming Entities (2)

- **Identifier** → A name used uniquely identify an entity
  - 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)

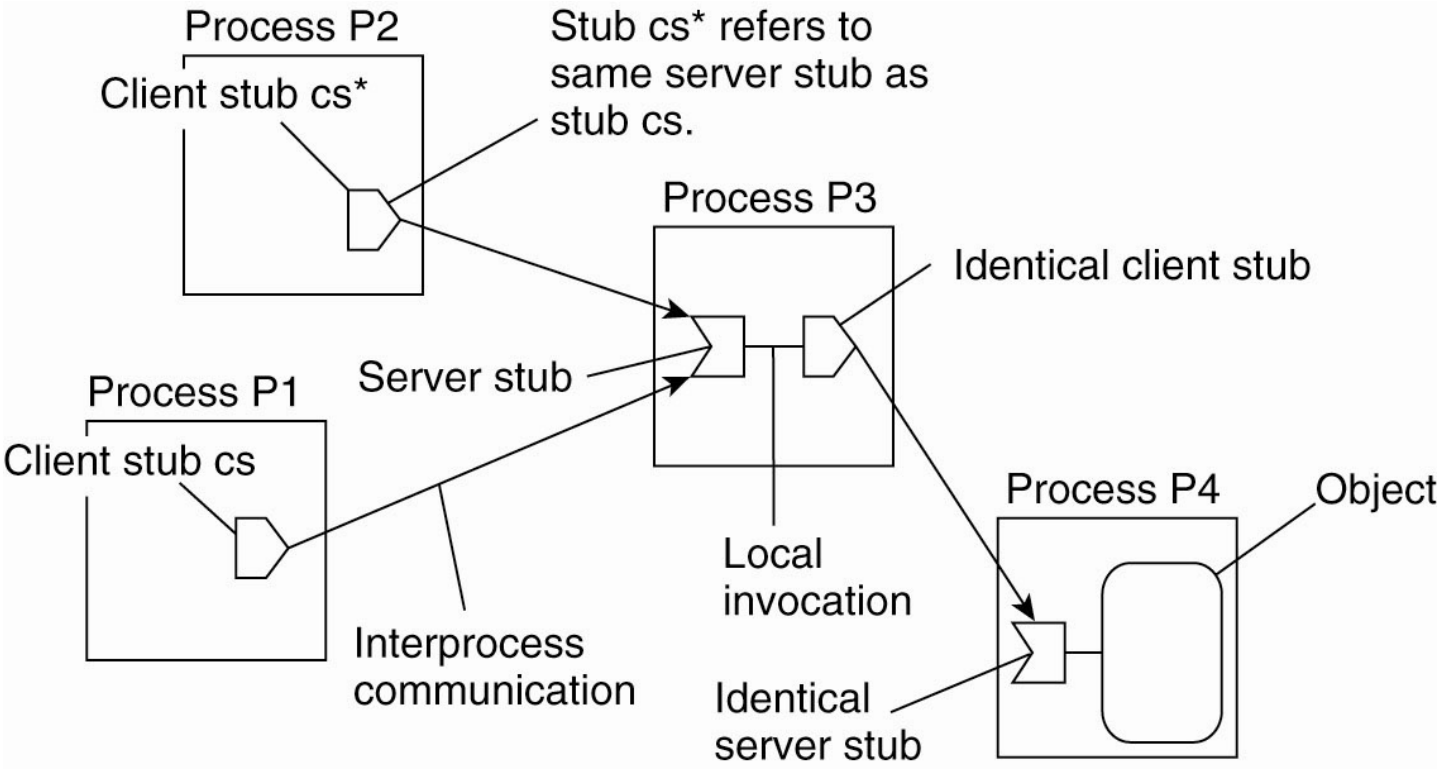
Flat Naming

- How to locate an entity given only its identifier
- Locating mobile entities
- Several approaches
  - Broadcasting
  - Forwarding pointers
  - Home based approaches
  - Distributed hash tables
  - Hierarchical approaches

Broadcasting

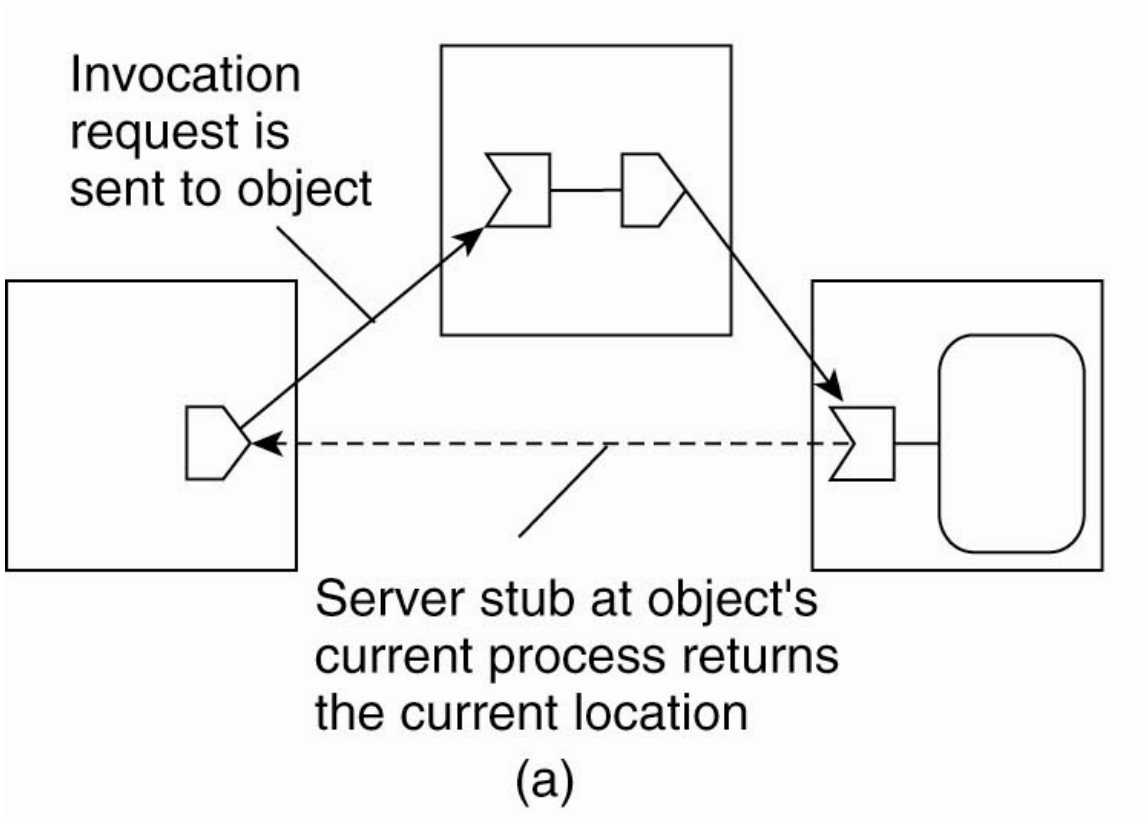
- Simply broadcast the ID, requesting the entity to return its current address.
  - e.g. Address Resolution Protocol (ARP)
- Broadcasting requires all processes to listen to incoming location requests
  - Multicasting to mitigate
- Can never scale beyond local-area networks
  - Bandwidth
  - Interrupted hosts

Forwarding Pointers (1)



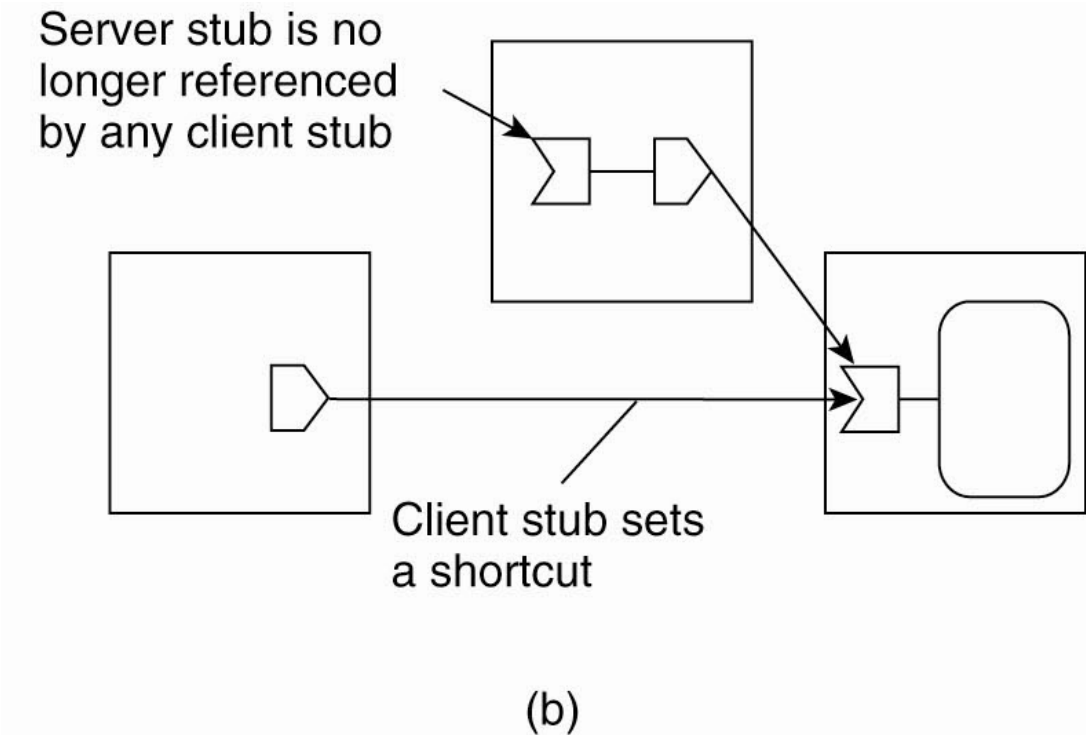
- The principle of forwarding pointers using (client stub, server stub) pairs.

Forwarding Pointers (2)



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

Forwarding Pointers (3)

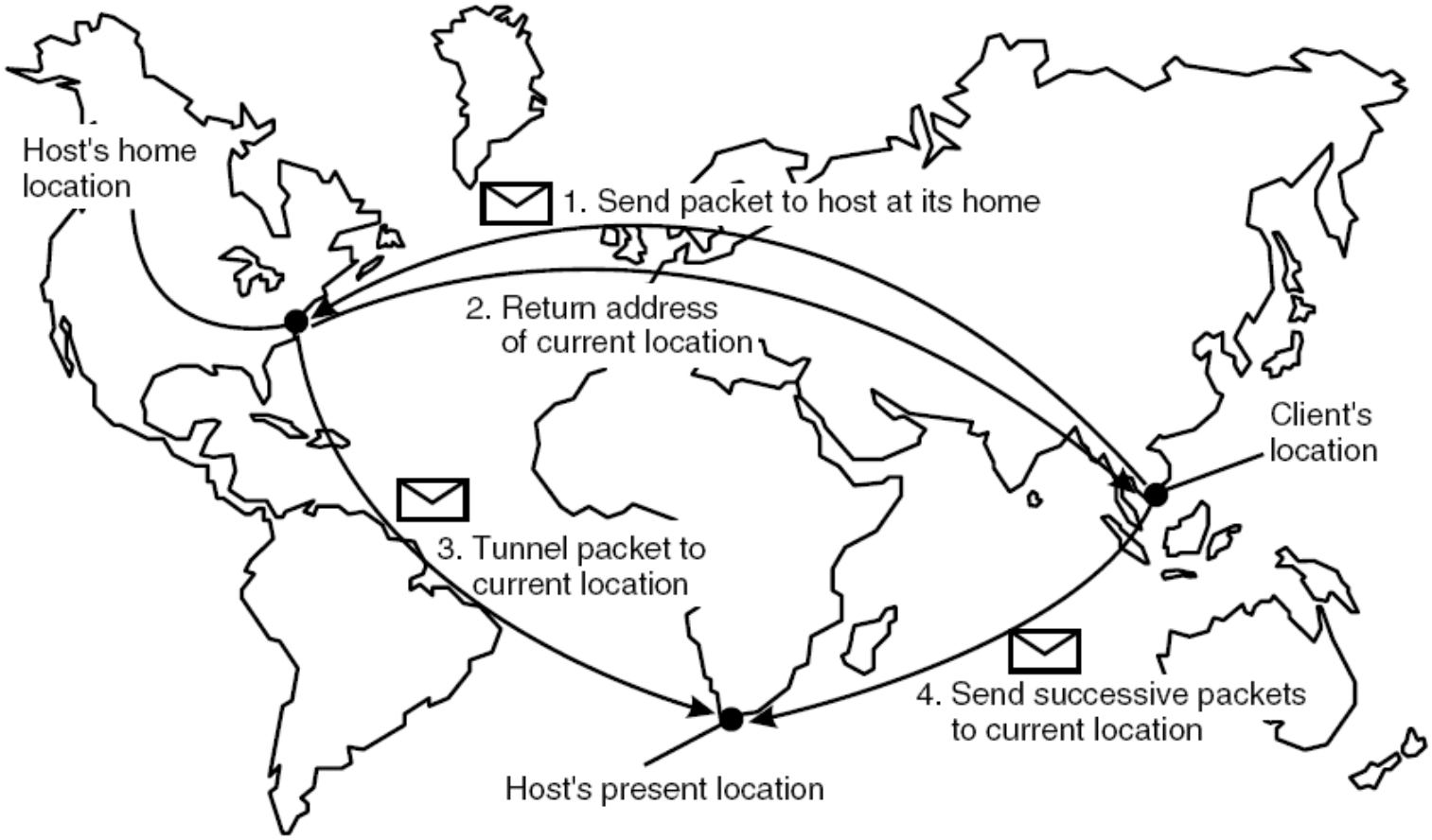


- Redirecting a forwarding pointer by storing a shortcut in a client stub.

Forwarding Pointers (3)

- 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
  - Sending the response directly to the initiating client
  - Along the reverse path of forwarding points (distributed garbage collection)
- Geographical scalability problems:
  - Long chains are vulnerable
  - Intermediate location management
  - Increased network latency at dereferencing

Home-Based Approaches (1)



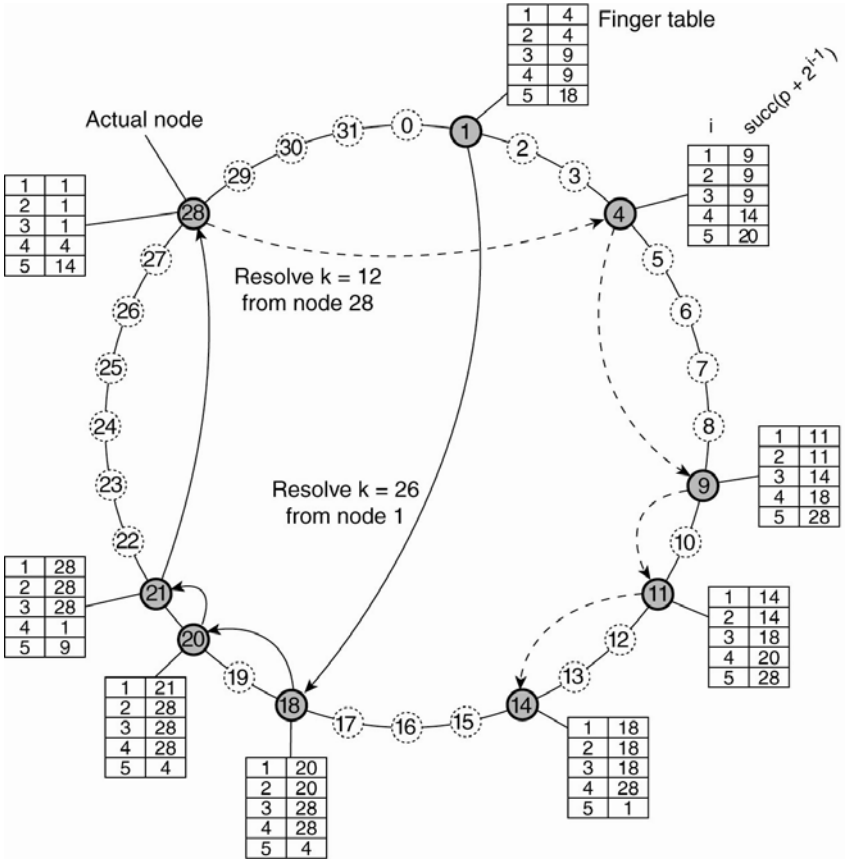
- The principle of Mobile IP.

Distributed Hash Tables (1)

- Finger Tables
  - e.g. Chord System; Stoica et. al. (2003)
- To look up a key ***k***, node ***p*** will then immediately forward the request to node ***q*** with index ***j*** in ***p***'s finger table where

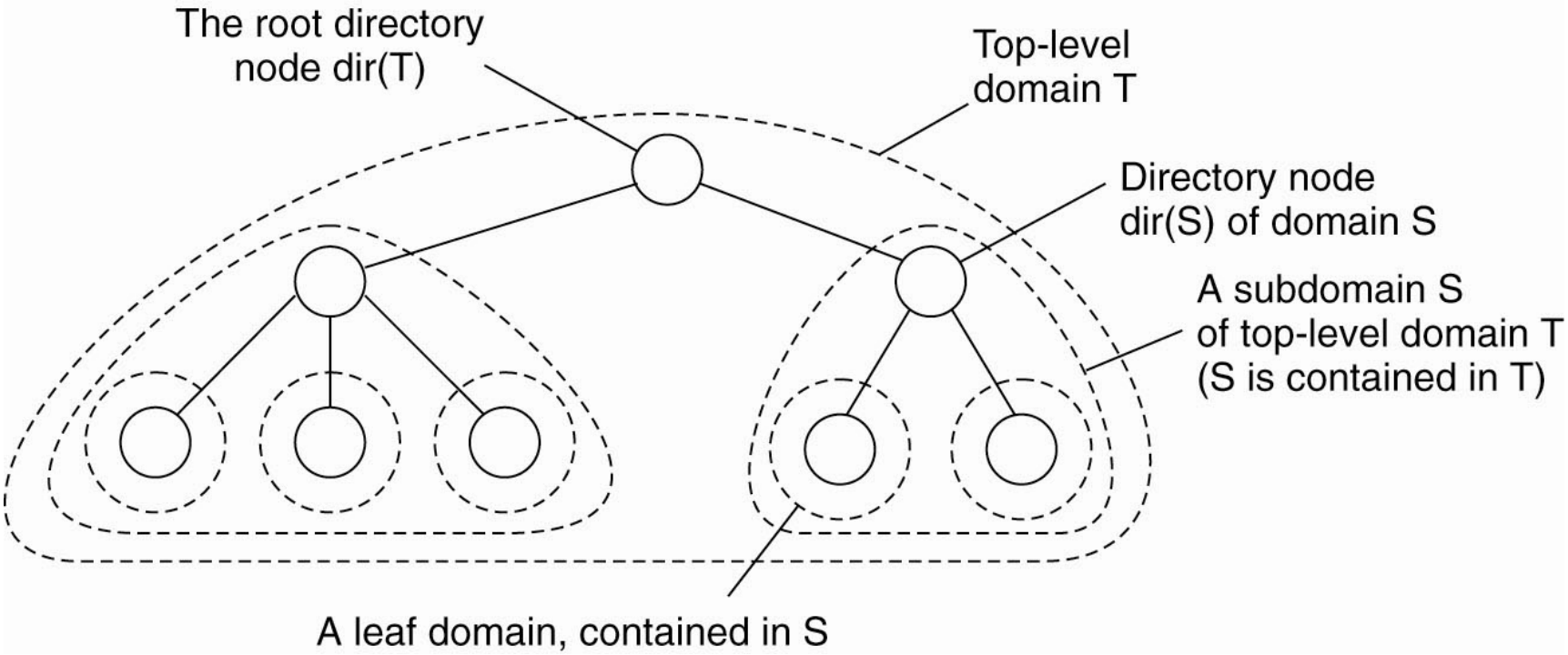
$$q = FT_p(j) \leq k \leq FT_p(j+1)$$

Distributed Hash Tables (2)



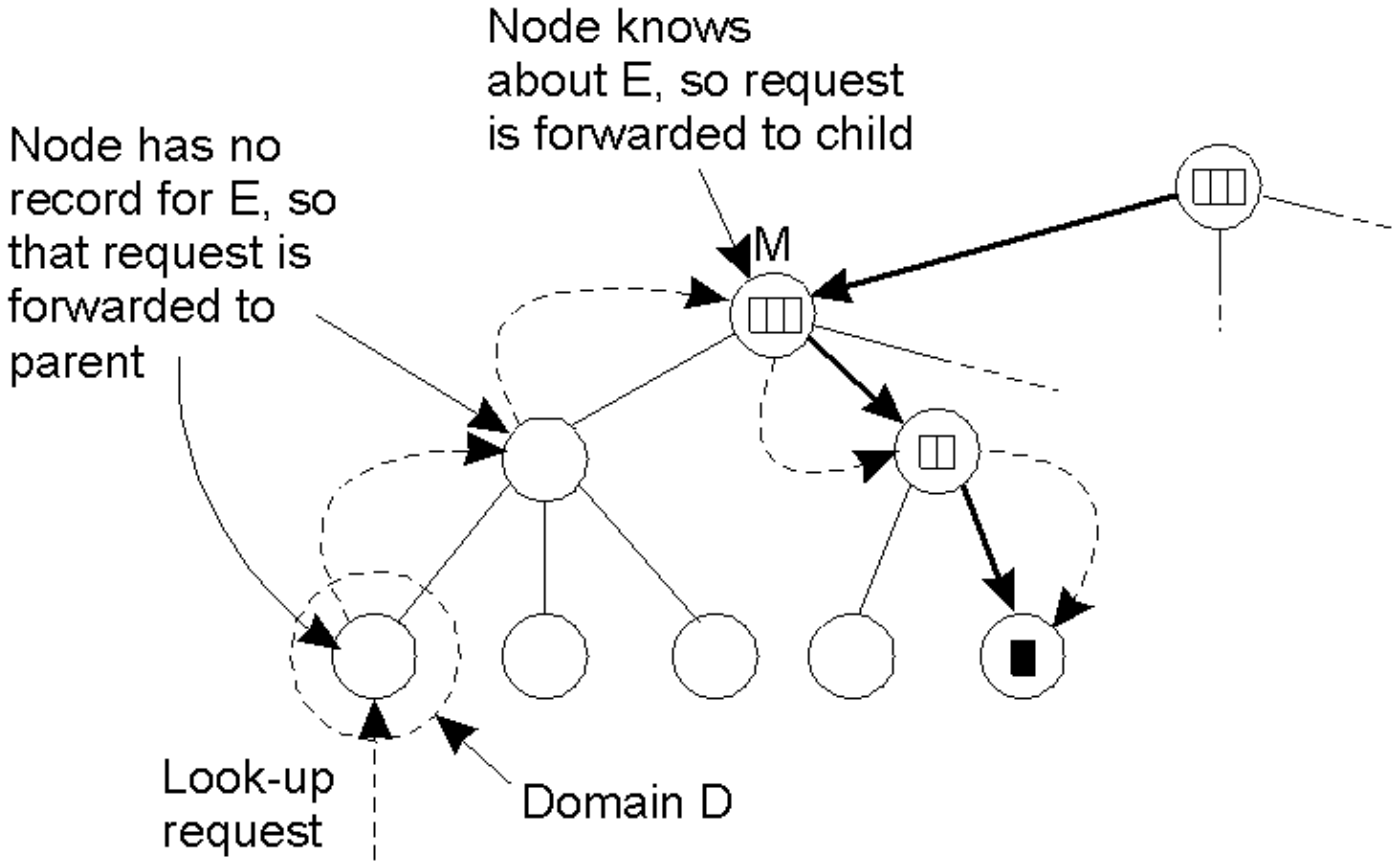
- Resolving key 26 from node 1 and key 12 from node 28 in a Chord system.

Hierarchical Approaches (1)



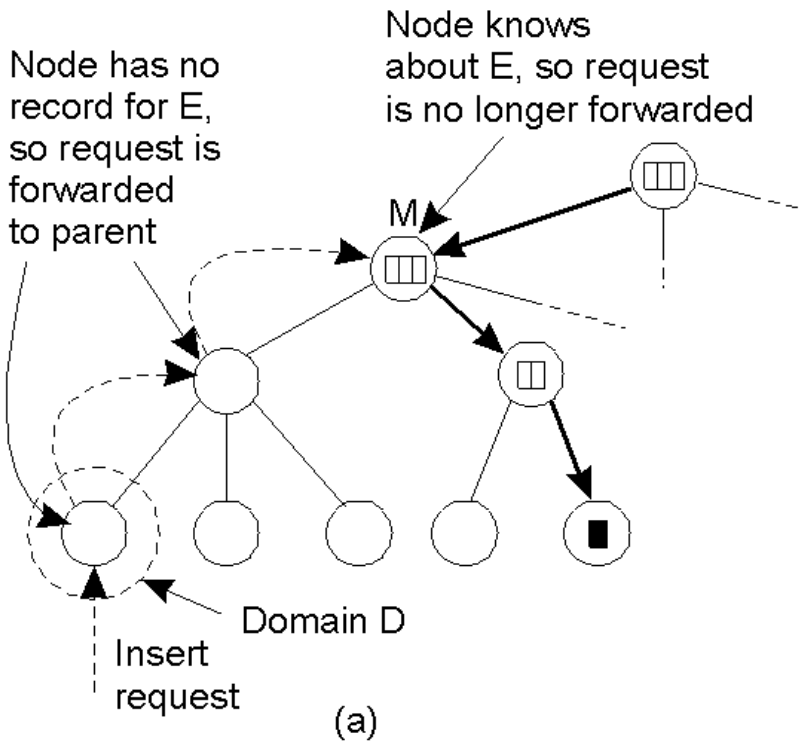
- Hierarchical organization of a location service into domains, each having an associated directory node.

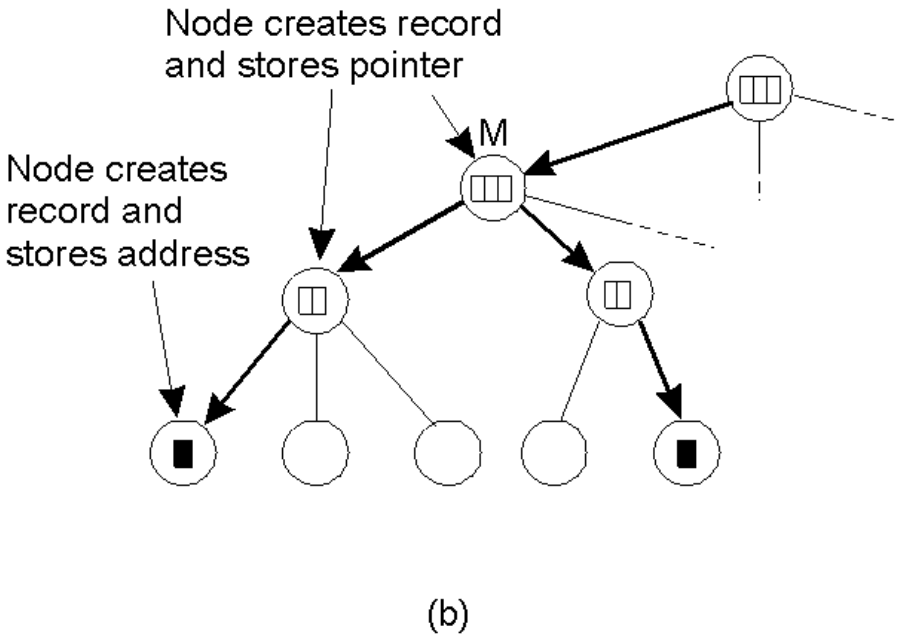
Hierarchical Approaches (2)



- Looking up a location in a hierarchically organized location service.

Hierarchical Approaches (3)

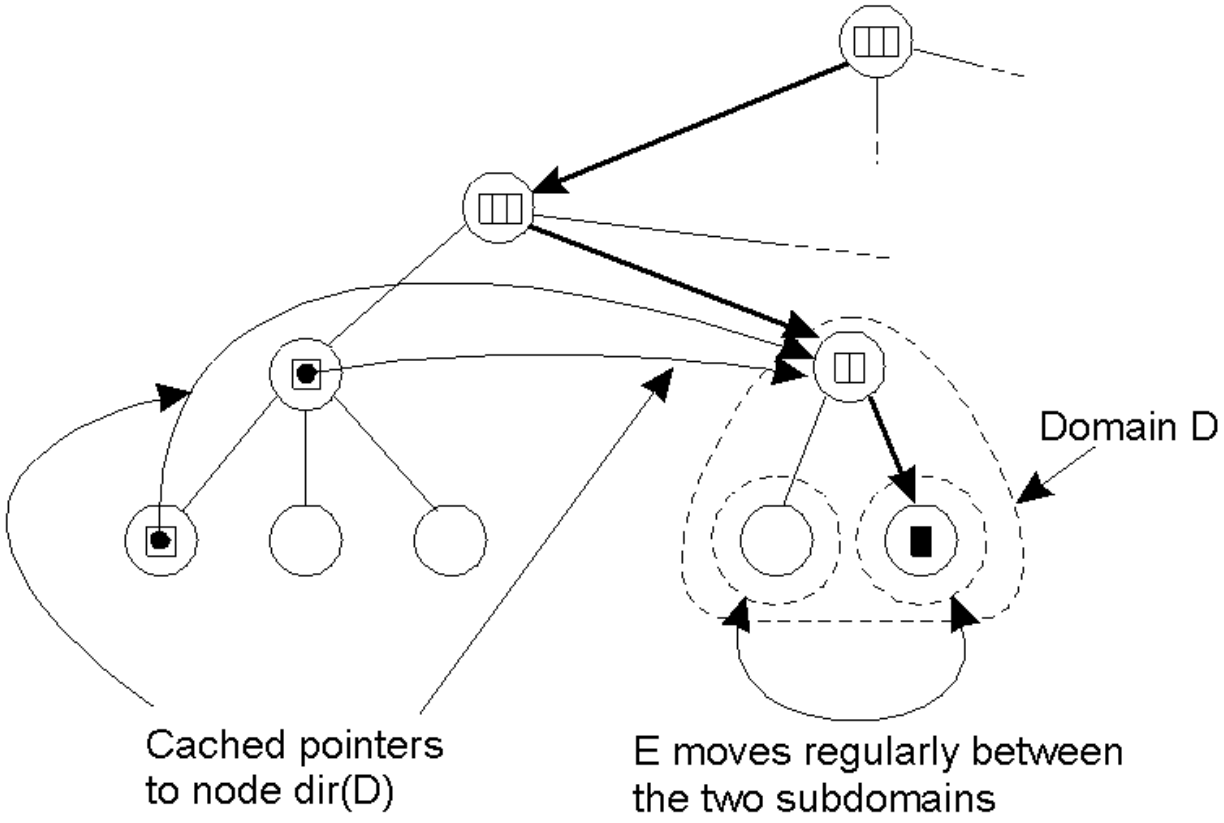




a) An insert request is forwarded to the first node that knows about entity *E*.

b) A chain of forwarding pointers to the leaf node is created.

Hierarchical Approaches (4)



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

Hierarchical Approaches (5)

- **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,

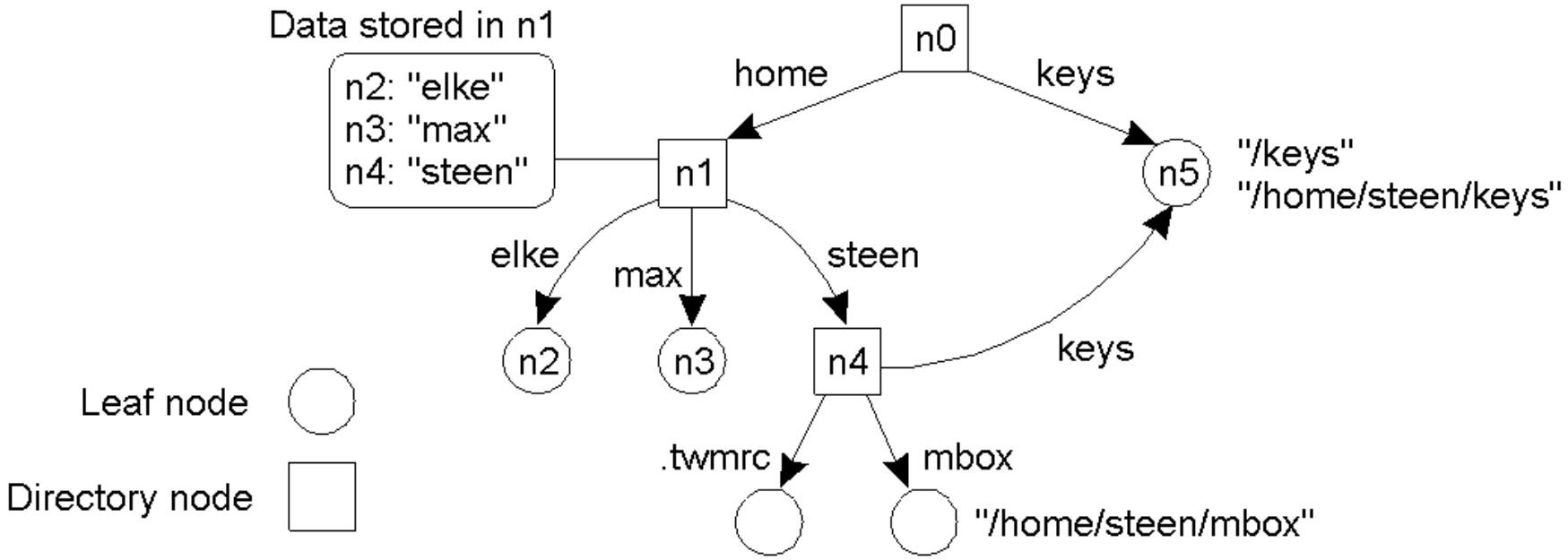
Structured Naming

- Human readable names
  - Name spaces
  - Name resolution
  - Name space distribution
  - Domain Name System (DNS)

Name Spaces (1)

- A labeled, directed graph with two types of nodes
  - Leaf node
    - » Represents a named entity and stores information about the entity
  - Directory node
    - » Has a number of outgoing edges each labeled with a name
    - » Stores a table (directory table) in which an outgoing edge is represented as a pair (edge label, node identifier)
- Used to organize names in a distributed system

Name Spaces (2)



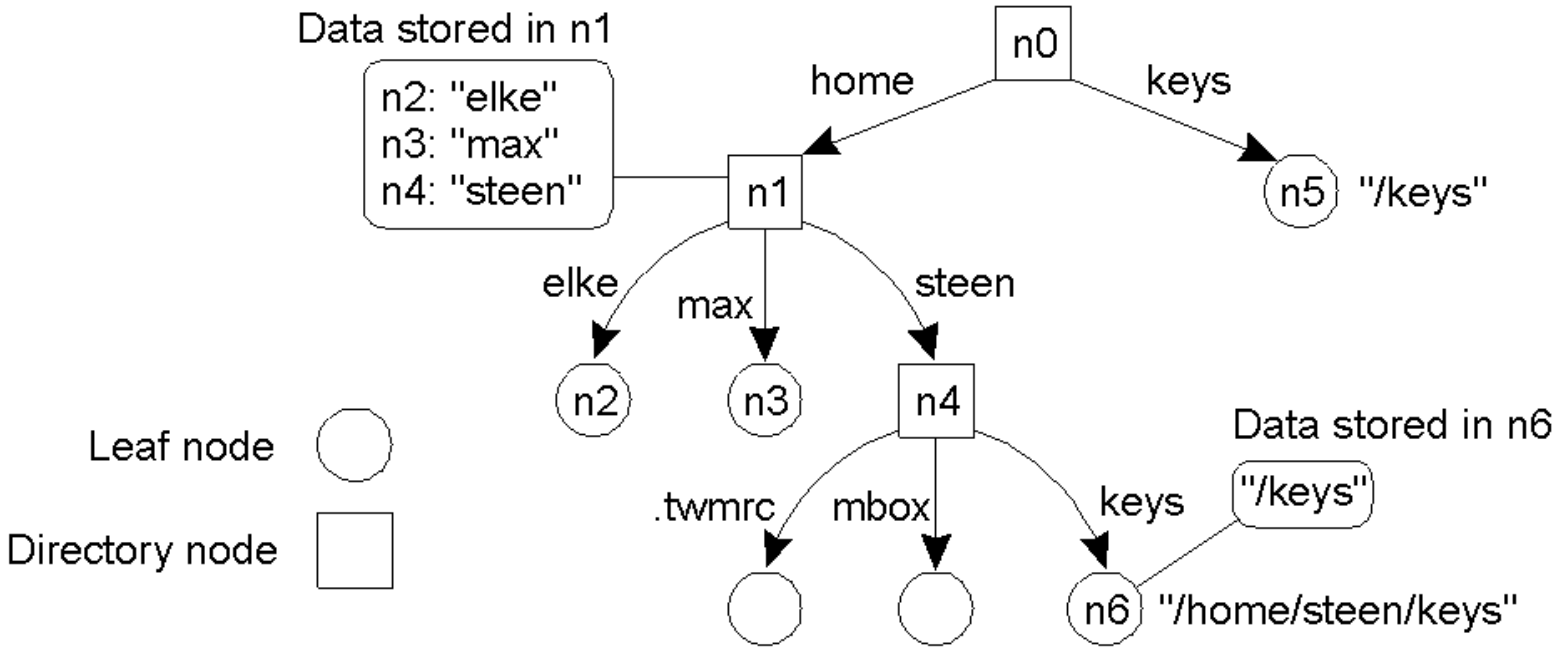
Name Resolution (1)

- The process of looking up a name
  - N:<label-1, label-2, ……., label-n>
    - » Starts at node N of the naming space
    - » The name label-1 is looked up in the directory table
    - » Returns the identifier of the node to which label-1 refers
    - » Continues looking up labels
  - How do we actually find that (initial) node?
- **Closure mechanism** → The mechanism to select the implicit context from which to start name resolution.

Name Resolution (2)

- **Name Linking (Aliases)**
  - **Hard link**
    - » A name is resolved by following a specific path in a naming graph from one node to another.
    - » Allows multiple absolute path names
  - **Symbolic link**
    - » Allows a node *O* to contain a *name* of another node
    - » A node stores an absolute path name instead of replicating information

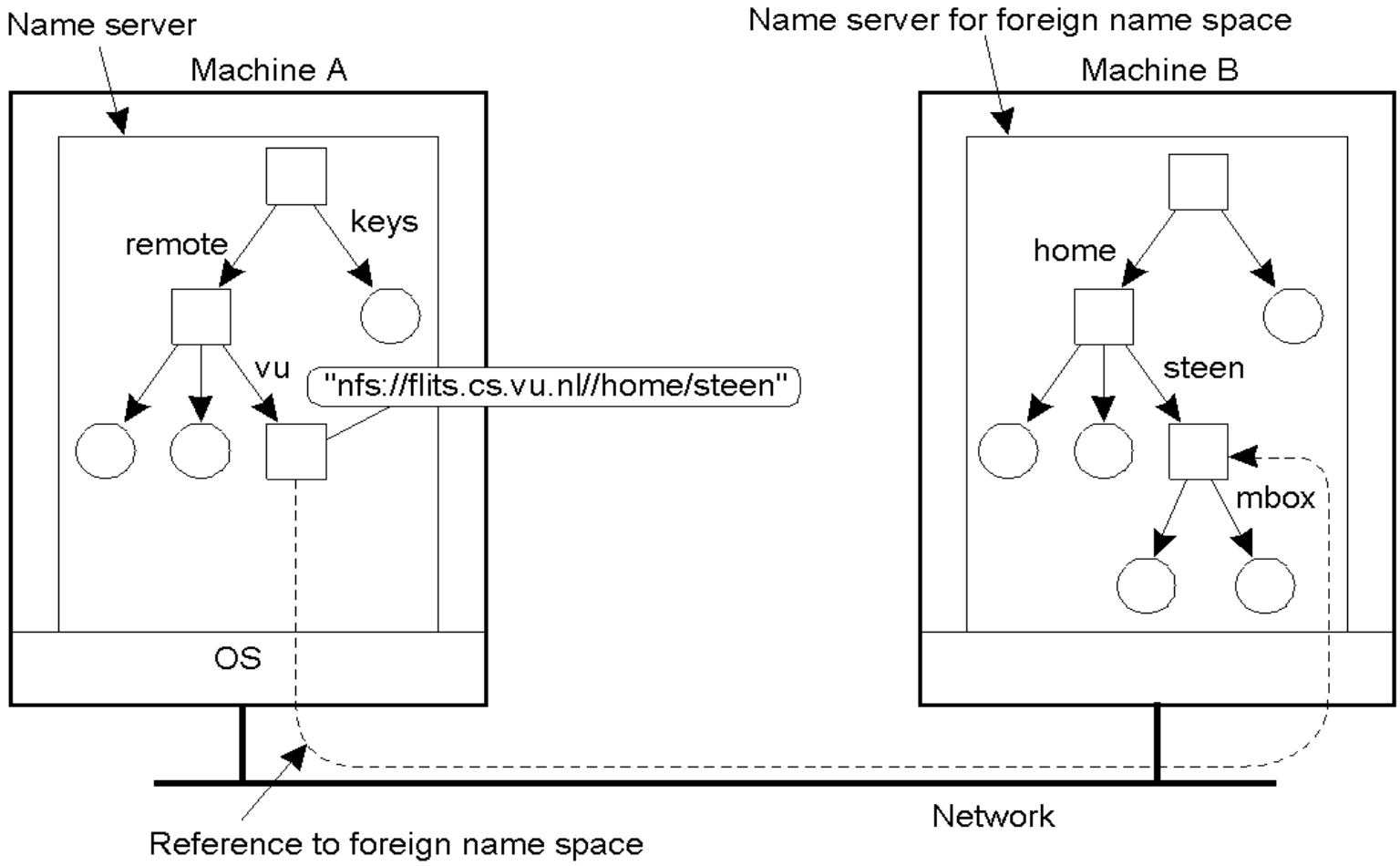
Name Resolution (3)



Name Resolution (4)

- We have different name spaces that we wish to access from any given name space (merging name spaces)
- Solution: **Mounting**
  - **Mount point** → Directory node storing the identifier of a directory node form a different name space
  - **Mounting point** → Directory node in the other name space that is referred to.
  - To mount a foreign name space in a distributed system it is required
    - The name of an access protocol (protocol implementation)
    - The name of the server (address)
    - The name of the mounting point in the foreign space (identifier)
  - e.g. Network File System (NFS)

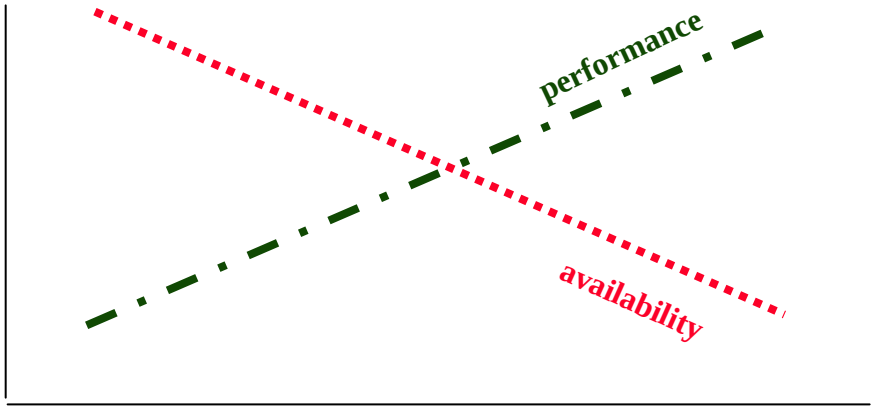
Name Resolution (5)



Name Space Distribution (1)

- Distribute the name resolution process as well as name space management across multiple machines, by distributing nodes of the naming graph.
- Consider a three level hierarchical naming graph and distinguish
  - **Global level** (Consists of the high-level directory nodes)
    - Main aspect is that these directory nodes have to be jointly managed by different administrations
    - Stability (directory tables are rarely changed)
  - **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
    - Relatively stable
  - **Managerial level** (Consists of low-level directory nodes)
    - within a single administration. Main issue is effectively mapping directory nodes to local name servers.
    - Nodes may change regularly

Name Space Distribution (2)



Item	Global	Administrational	Managerial
1	Worldwide	Organization	Department
2	Few	Many	Vast numbers
3	Seconds	Milliseconds	Immediate
4	Lazy	Immediate	Immediate
5	Many	None or few	None
6	Yes	Yes	Sometimes

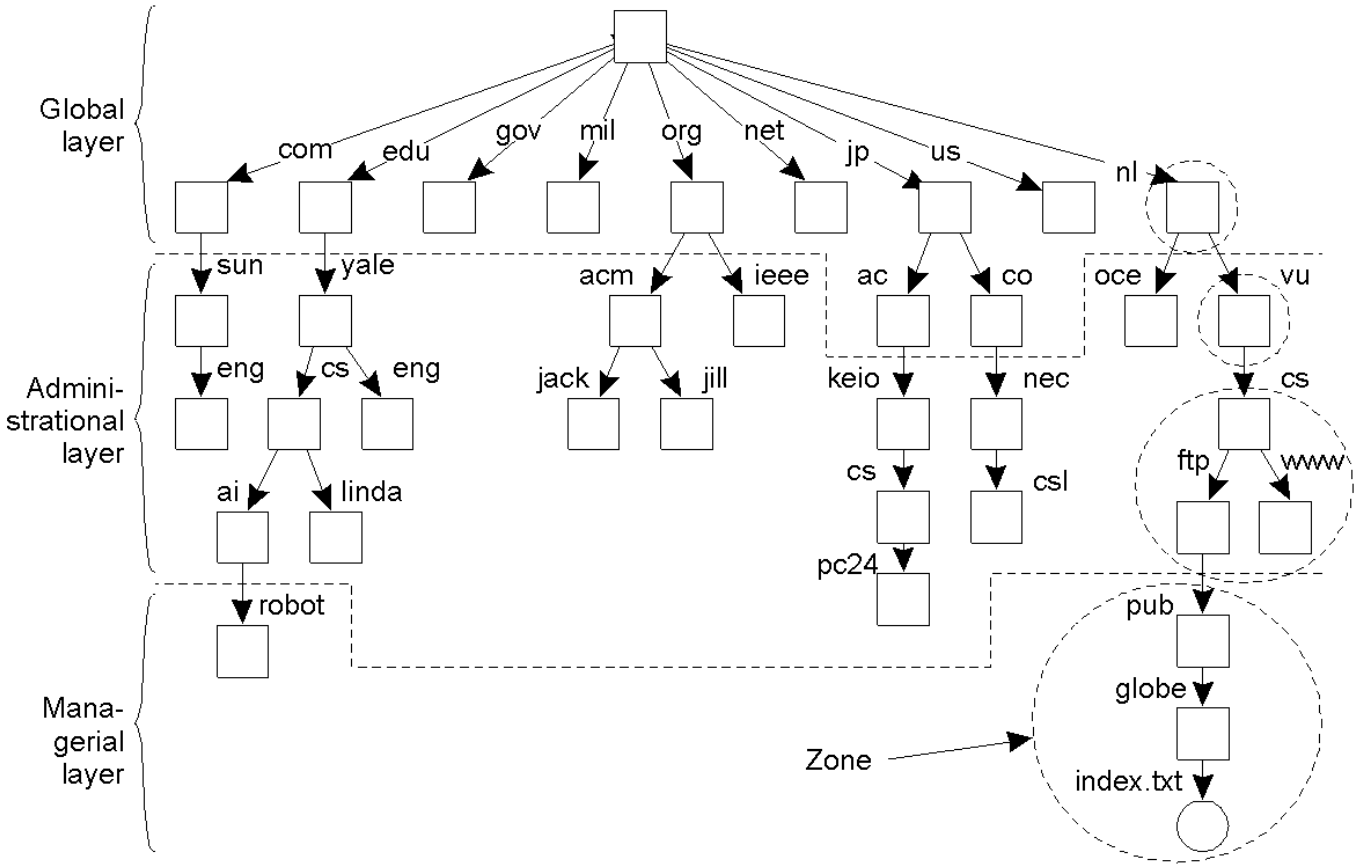
1: Geographical scale
2: # Nodes
3: Responsiveness

4: Update propagation
5: # Replicas
6: Client-side caching?

High Availability → Replication Servers → Consistency

High Performance → Caching → Synchronization

Name Space Distribution (3)

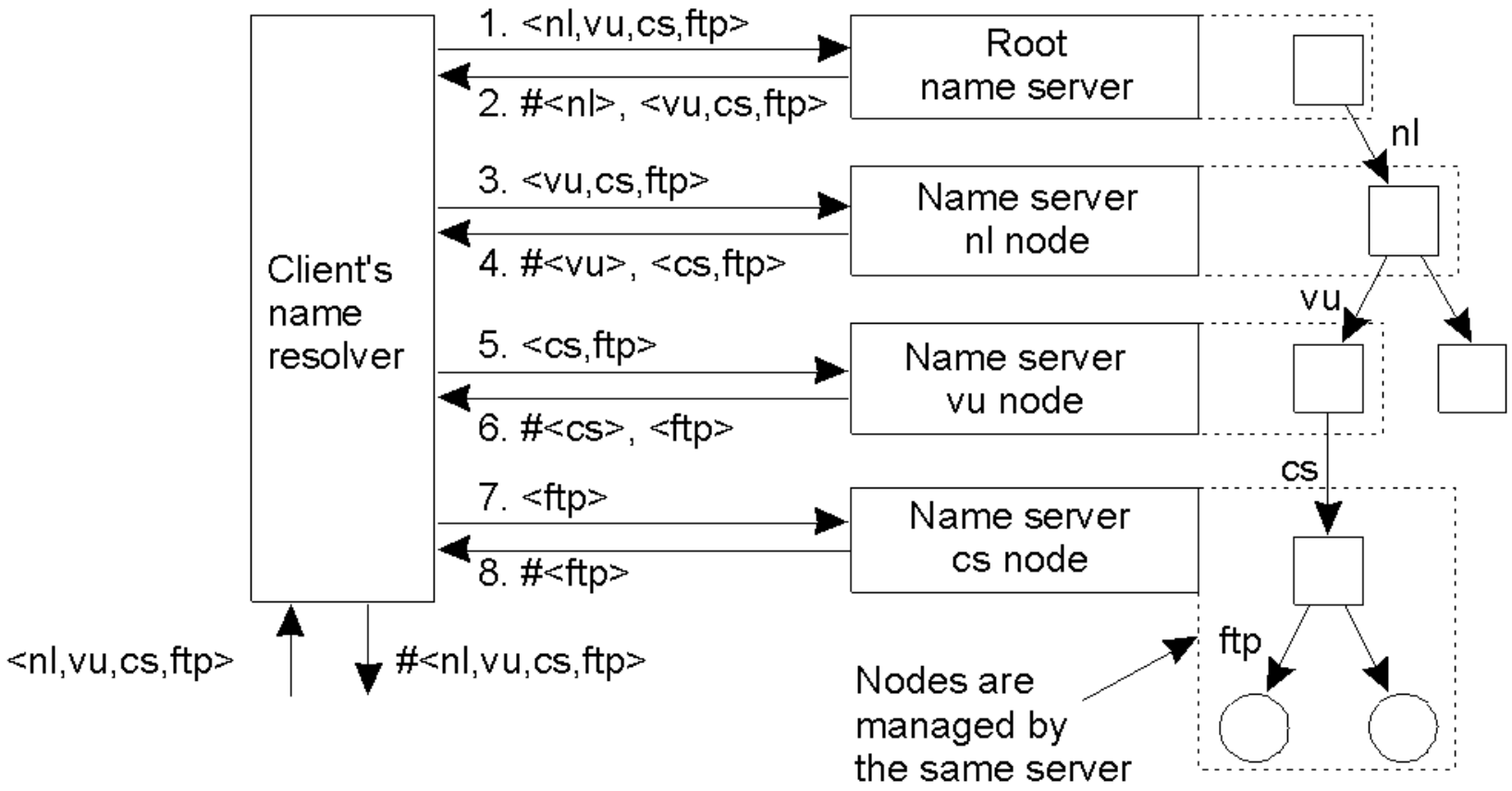


- An example partitioning of the DNS name space, including Internet-accessible files, into three layers.

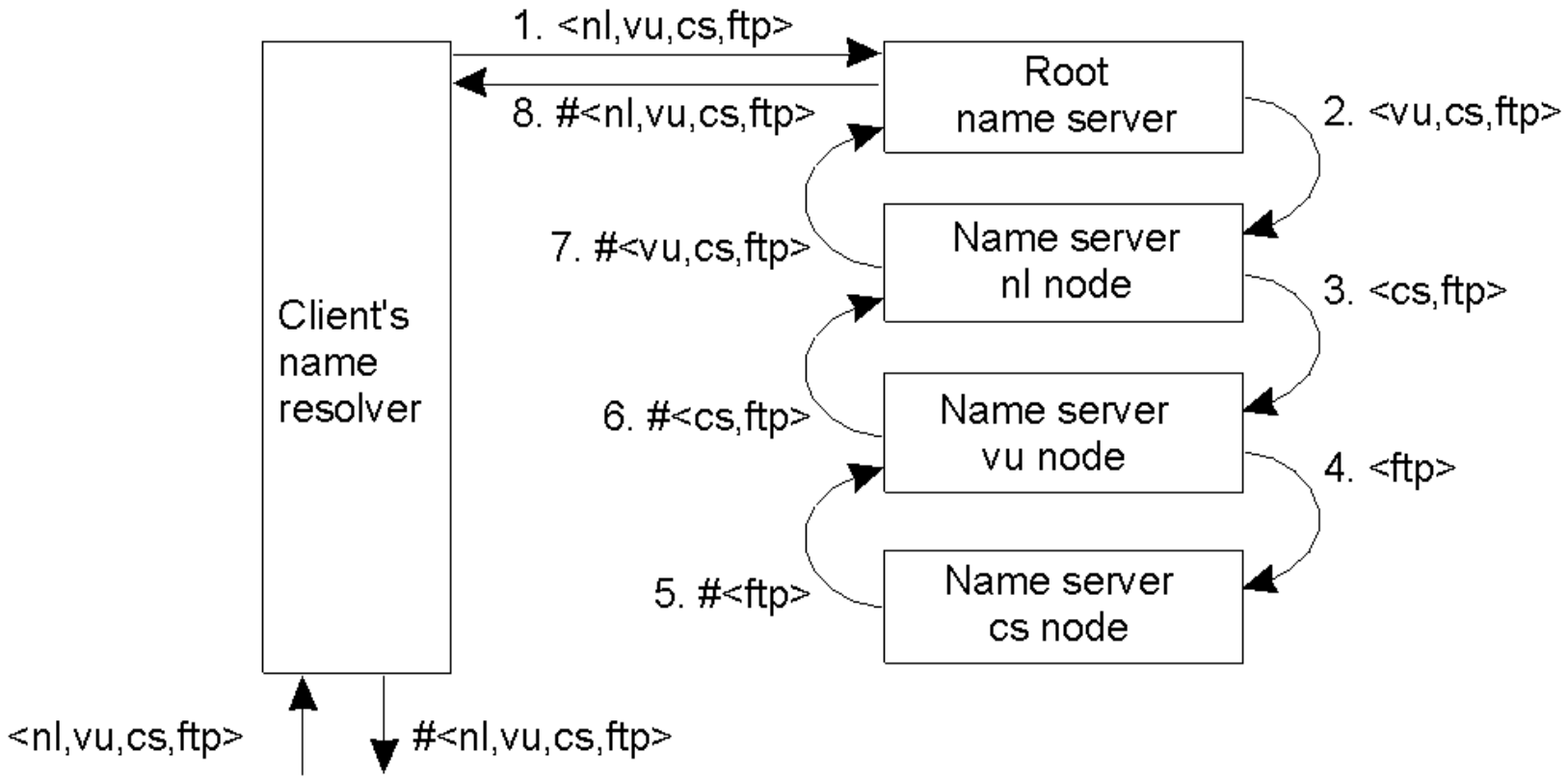
Name Space Distribution (4)

- Two ways to implement name resolution in distributed systems
  - Iterative Name Resolution
    - resolve(dir, [name1, …., nameK]) is sent to server0 responsible for dir
    - Server0 resolves (dir, name1) → dir1 returning the address of server1 which stores dir1
    - Client sends resolve(dir1, [name2, …., nameK]) to server1
    - continues
  - Recursive Name Resolution
    - resolve(dir, [name1, …., nameK]) is sent to server0 responsible for dir
    - Server0 resolves (dir, name1) → dir1 and sends resolve(dir1, [name2, …., nameK]) to server1 which stores dir1
    - continues
    - Server0 waits for the result from server1 and returns it to the client

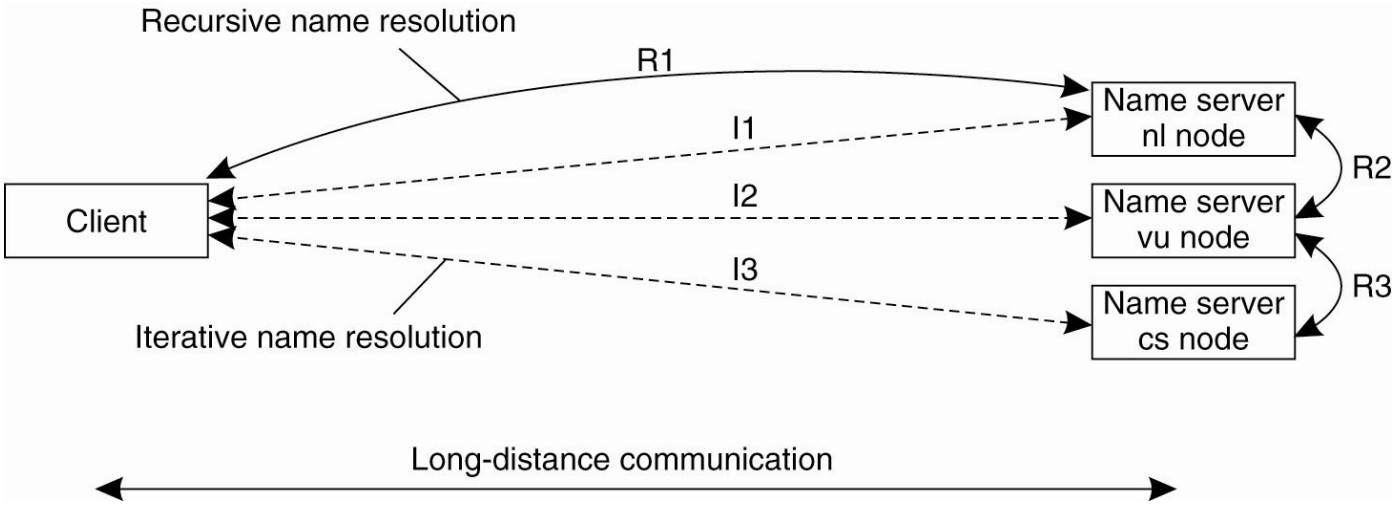
Iterative Name Resolution



Recursive Name Resolution



Recursive vs. Iterative



- Recursive Name Resolution
  - Drawback
    - » high performance demand on each name server (e.g. global layer servers may support only iterative name resolution)
  - Advantages
    - » Caching may be more effective
    - » Communication cost can be reduced

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.

DNS Implementation (1)

Name	Record type	Record value
cs.vu.nl.	SOA	star.cs.vu.nl. hostmaster.cs.vu.nl. 2005092900 7200 3600 2419200 3600
cs.vu.nl.	TXT	"Vrije Universiteit - Math. & Comp. Sc."
cs.vu.nl.	MX	1 mail.few.vu.nl.
cs.vu.nl.	NS	ns.vu.nl.
cs.vu.nl.	NS	top.cs.vu.nl.
cs.vu.nl.	NS	solo.cs.vu.nl.
cs.vu.nl.	NS	star.cs.vu.nl.
star.cs.vu.nl.	A	130.37.24.6
star.cs.vu.nl.	A	192.31.231.42
star.cs.vu.nl.	MX	1 star.cs.vu.nl.
star.cs.vu.nl.	MX	666 zephyr.cs.vu.nl.
star.cs.vu.nl.	HINFO	"Sun" "Unix"
zephyr.cs.vu.nl.	A	130.37.20.10
zephyr.cs.vu.nl.	MX	1 zephyr.cs.vu.nl.
zephyr.cs.vu.nl.	MX	2 tornado.cs.vu.nl.
zephyr.cs.vu.nl.	HINFO	"Sun" "Unix"

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

DNS Implementation (2)

ftp.cs.vu.nl.	CNAME	soling.cs.vu.nl.
www.cs.vu.nl.	CNAME	soling.cs.vu.nl.
soling.cs.vu.nl.	A	130.37.20.20
soling.cs.vu.nl.	MX	1 soling.cs.vu.nl.
soling.cs.vu.nl.	MX	666 zephyr.cs.vu.nl.
soling.cs.vu.nl.	HINFO	"Sun" "Unix"
vucs-das1.cs.vu.nl.	PTR	0.198.37.130.in-addr.arpa.
vucs-das1.cs.vu.nl.	A	130.37.198.0
inkt.cs.vu.nl.	HINFO	"OCE" "Proprietary"
inkt.cs.vu.nl.	A	192.168.4.3
pen.cs.vu.nl.	HINFO	"OCE" "Proprietary"
pen.cs.vu.nl.	A	192.168.4.2
localhost.cs.vu.nl.	A	127.0.0.1

Attributed-Based Naming

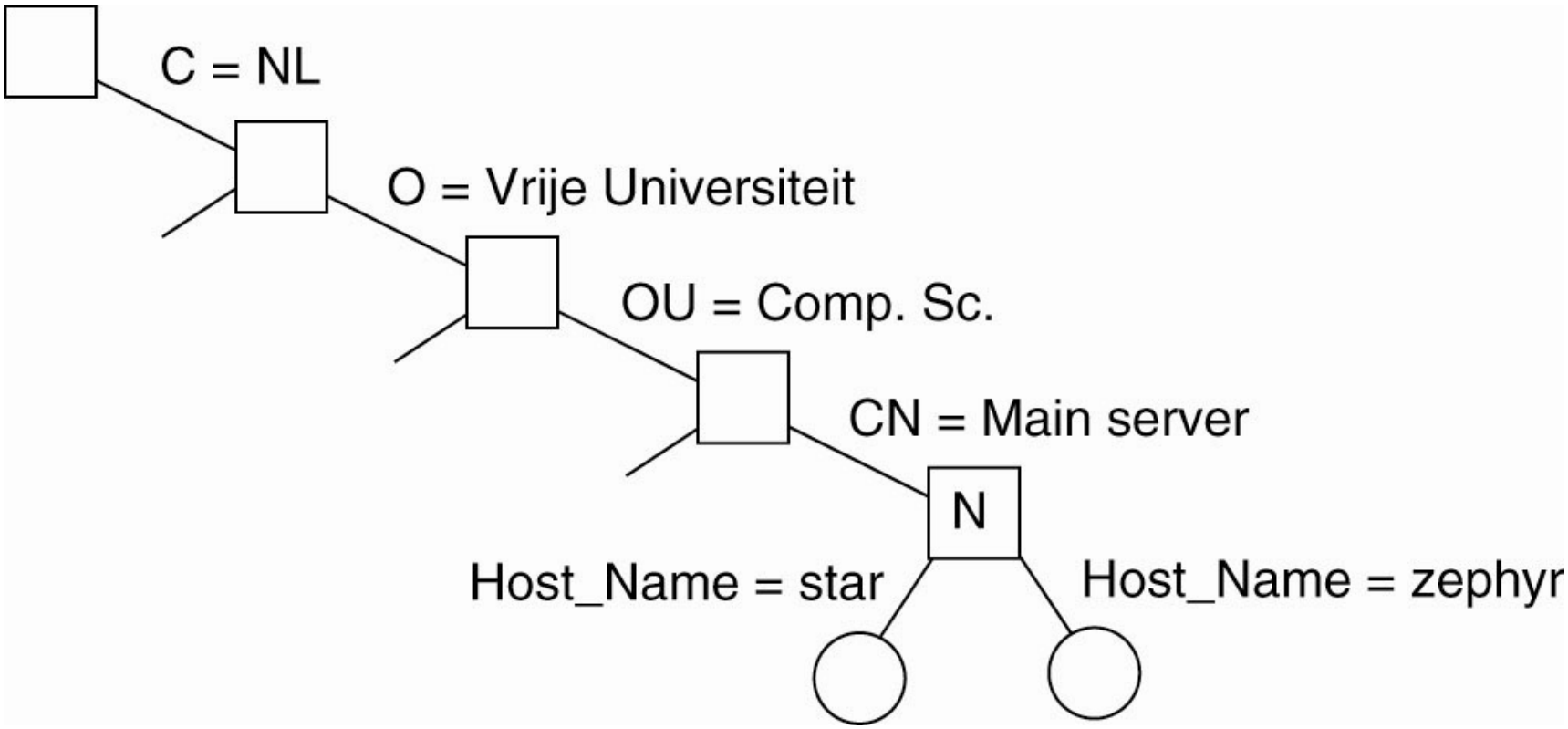
- A user can provide merely a description of what she is looking for
  - Describe an entity in terms of (*attribute, value*) pairs
  - an entity is assumed to have an associate collection of attributes
  - By specifying which values a specific attribute should have, a user constraints the set of entities that she is interested in
  - e.g. RDF: Resource Description Framework

Hierarchical Implementations: LDAP (1)

Attribute	Abbr.	Value
Country	C	NL
Locality	L	Amsterdam
Organization	O	Vrije Universiteit
OrganizationalUnit	OU	Comp. Sc.
CommonName	CN	Main server
Mail_Servers	—	137.37.20.3, 130.37.24.6, 137.37.20.10
FTP_Server	—	130.37.20.20
WWW_Server	—	130.37.20.20

LDAP = Lightweight Directory Access Protocol

Hierarchical Implementations: LDAP (2)



Hierarchical Implementations: LDAP (3)

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

Attribute	Value
Country	NL
Locality	Amsterdam
Organization	Vrije Universiteit
OrganizationalUnit	Comp. Sc.
CommonName	Main server
Host_Name	zephyr
Host_Address	137.37.20.10

(b)

- Two directory entries having *Host_Name* as RDN.