Naming

Distributed Systems
Santa Clara University 2016

Naming

- Names are needed
  - to share resources
  - to uniquely identify entities
  - to refer to locations
- Name resolution:
  - Passing from name to entity
- Naming in distributed systems:
  - Efficiency
  - Scalability
  - Support for mobility
  - Support for dynamism:
    - Creation and deletion of names

Naming Entities

- Distinguish
  - Names (bit string)
  - Entities (things)
  - Access point to an entity = address of an entity

Naming Entities

- Location independent
  - Name for an entity that is independent of the address
  - Example: File names

Naming Entities

- Human friendly names
  - Usually represented as character strings
    - Examples: File names, DNS

Naming Entities

- Identifier
  - An 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
    - It is never reused:
      - With any number of exceptions

Fundamental Problem of Naming

- **Name resolution**: How to resolve names and identifiers to addresses?

1. Use a (distributed table of <*name, address*> entries
2. *Information-Centric Networking*: Use routing to approach the entity in several steps
   - Blurs network layer and name resolution

Flat Naming

- Name contains no hint for its address
  - Example: IP addresses in a LAN

Flat Naming: Broadcasting

- Broadcast the ID, requesting the entity to return its current address
  - Can never scale beyond local-area networks
  - Requires all processes to listen to incoming location requests

- Address Resolution Protocol (ARP)
  - To find out which MAC address is associated with an IP address, broadcast the query “who has this IP address”?

Flat Naming: Broadcasting

- Multicasting
  - E.g.: Typically, router hardware has “network addresses”
  - Useful if there are several replicas to find the *nearest* replica

Flat Naming with Mobility: Forwarding Pointers

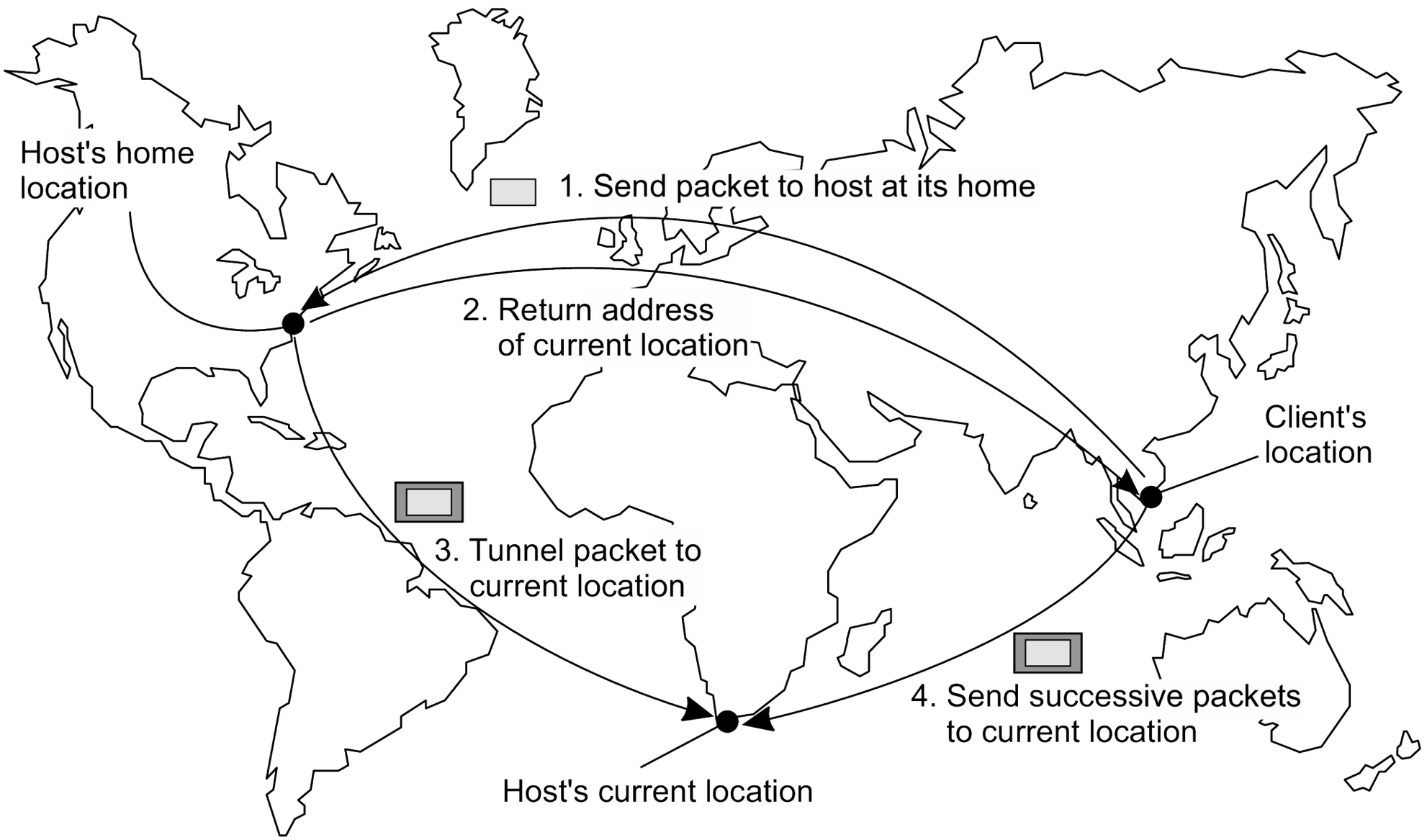
- When an entity moves from *A* to *B*:
  - Leave a forwarding reference at *A*
- Vulnerabilities:
  - Forwarding chains can become long
  - All intermediate nodes need to maintain a link
  - Creates a multitude of points of failures

Flat Naming with Mobility: Home Based Approaches

- Official address is the *home address*
  - Address visible to clients / interlocutors is the home address
  - Requests are forwarded to the current address

Flat Naming with Mobility: Home Based Approaches

- Official address is the *home address*

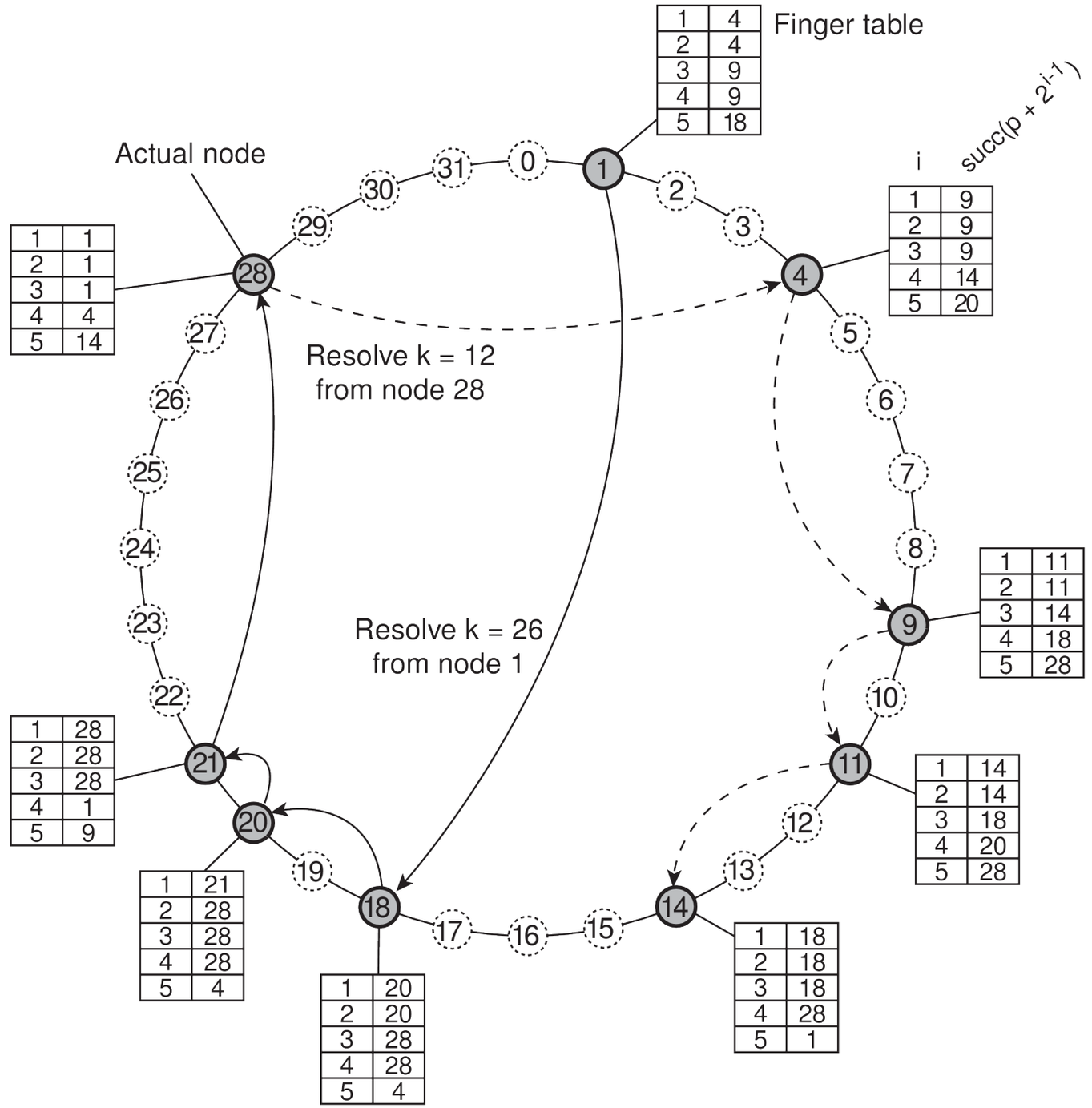


Flat Naming: Distributed Hash Tables

- Identifiers are unsigned integers
  - Find the address by routing using the identifier

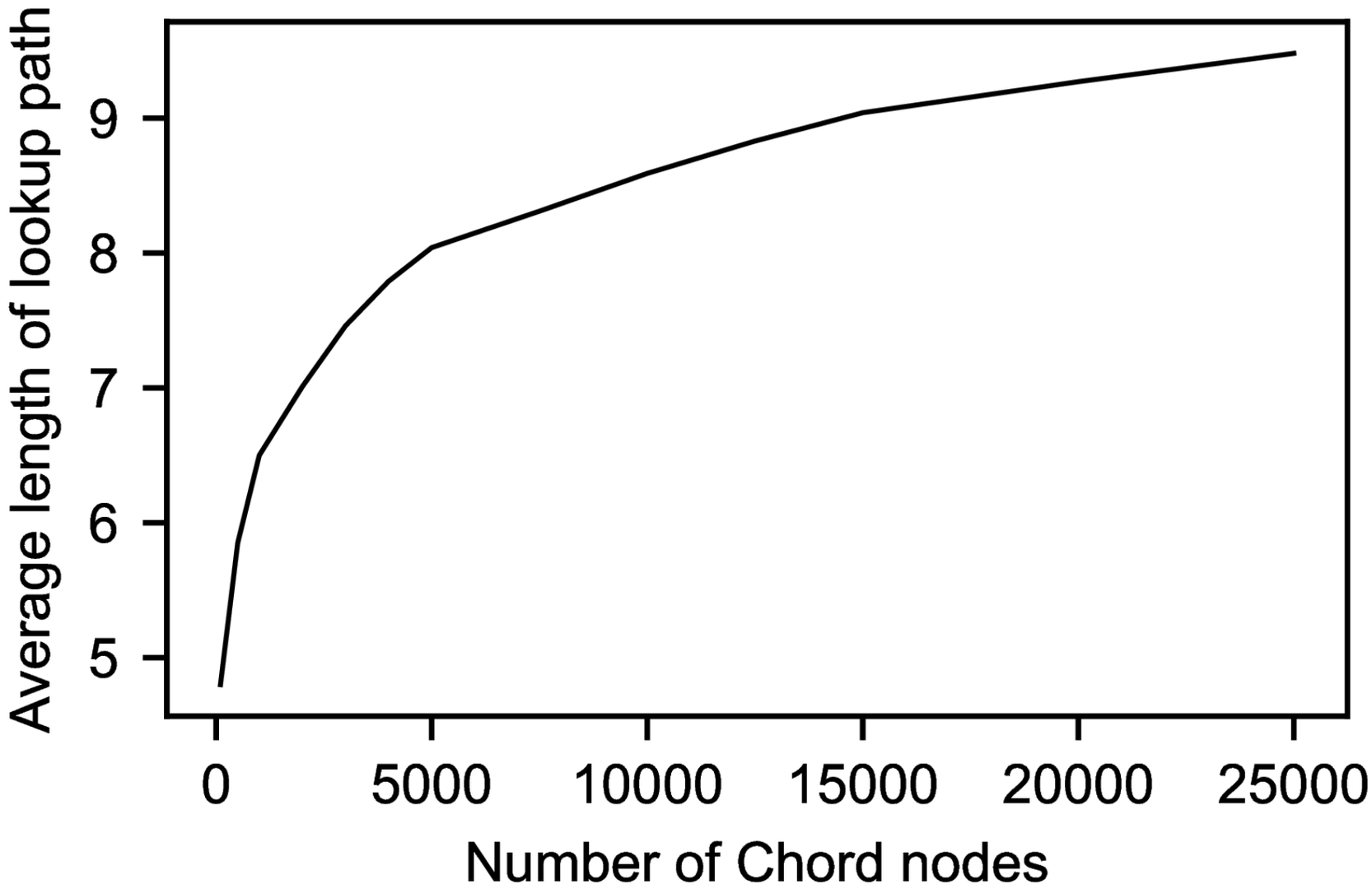
Flat Naming: Distributed Hash Tables

- Chord



Flat Naming: Distributed Hash Tables

- Chord

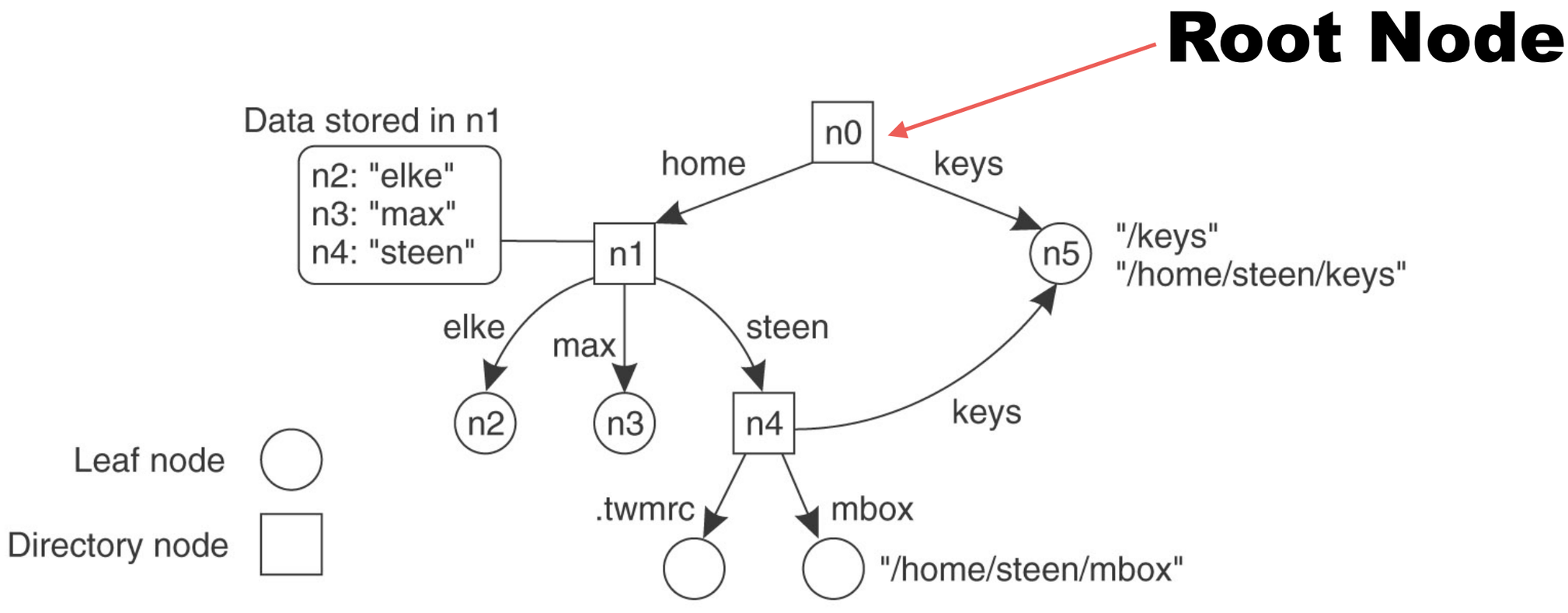


Flat Naming: Distributed Hash Tables

- Chord works in an overlay network with possibly inefficient routing
  - **Topology-based assignment of node identifiers**
  - **Proximity Routing:** In Chord, have more than one successor and choose the one physically closer to forward
  - **Proximity neighbor selection**

Structured Naming: Naming Entities

- Name space
  - Used to organize names in a distributed system
  - Name space represented as a tree
  - Leaf node represents named entity



Structured Naming: Naming Entities

- Each leaf node given by a path from the / a root
  - **Path name**
  - absolute path name: path starts with a root
  - relative path name: path starts with an intermediate node
- Example:
  - File Systems

Structured Naming: Naming Entities

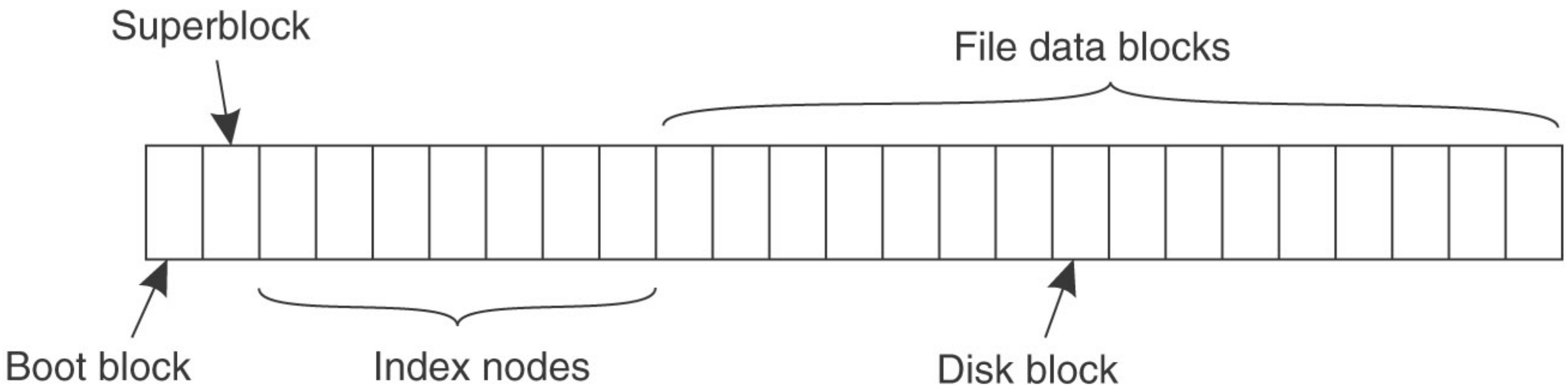
- Restrictions on naming graph
  - Can allow to have nodes with more than one incoming link
    - Mounting
  - But graph is acyclic

Structured Naming: Naming Entities

- Unix file system:
  - Interior node represents a file directory
  - Leaf node represents a file
  - Single root directory

- File system
  - Contiguous series of blocks
  - divided into boot block
  - superblock
  - index nodes (inodes)
  - file data blocks

Structured Naming: Naming Entities



Structured Naming: Naming Entities

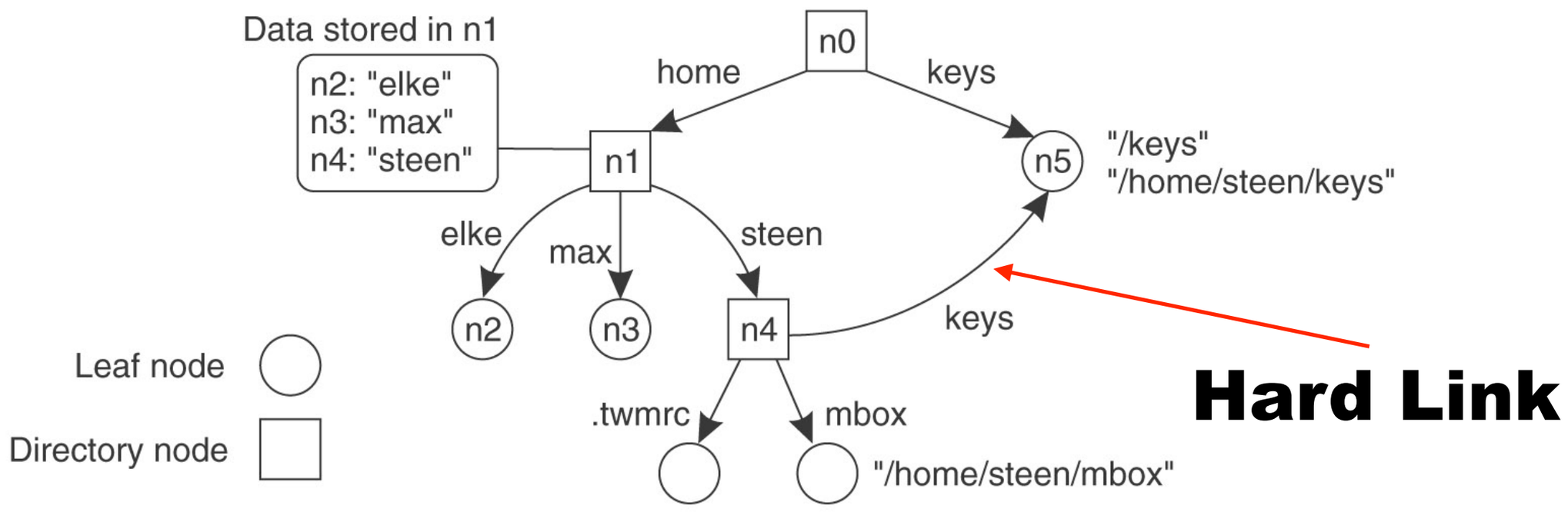
- Boot block:
  - Used to load the OS into memory
- Superblock:
  - Information on the file system
- Inode:
  - Exact information of where data for the associated file can be found

Structured Naming: Naming Entities

- Name resolution
  - Closure mechanism:
    - Selects the starting node
    - UNIX file system: Inode of the root directory is the very first inode
      - Resolution of `/usr/thomas/classes/distributed`:
        - Need access to the root note
        - Then follow information
- Difficult with another directory as root
  - Need prior access to the root
- HOME directory
  - Gives home directory of user
  - File names resolve from the home directory
  - Each user has its own copy of the variable HOME in the runtime

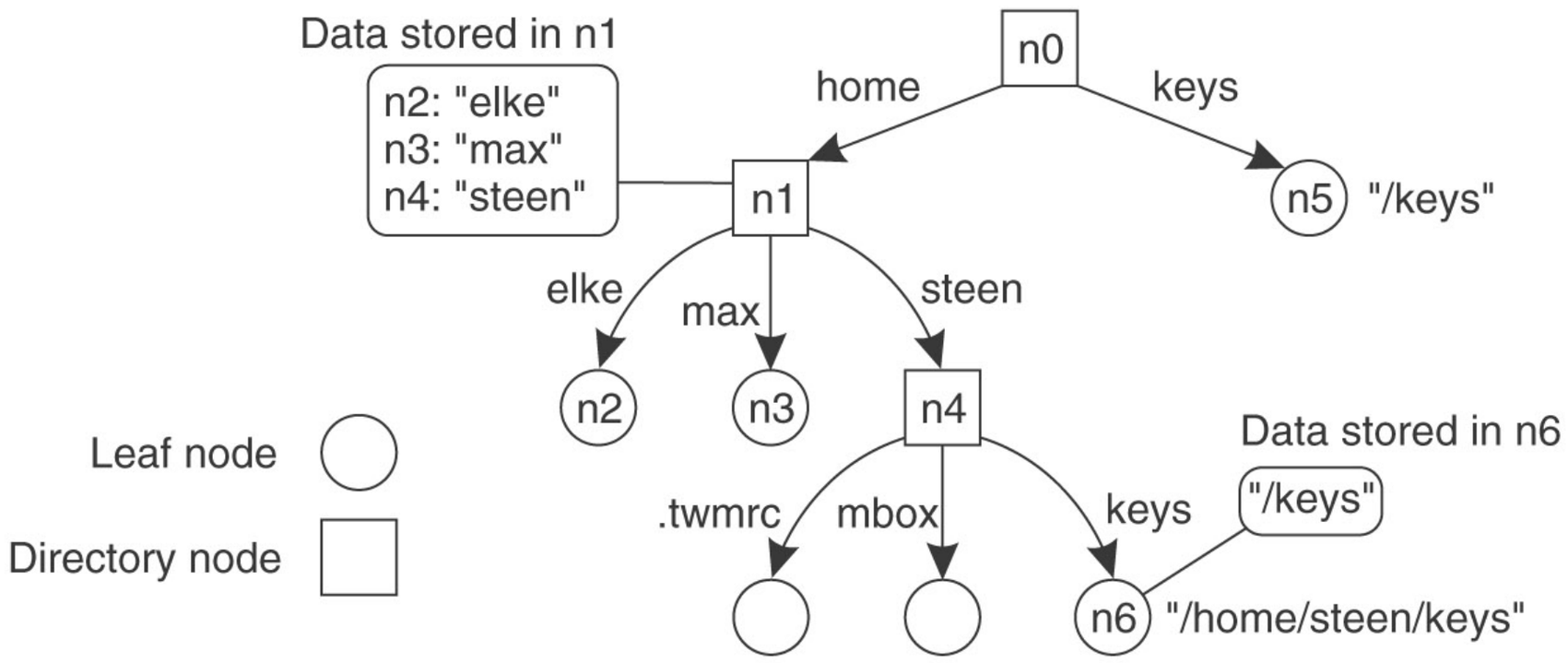
Naming Entities

- Alias:
  - Alternative name
  - Can use multiple absolute paths to refer to the same node
  - UNIX hard link



Naming Entities

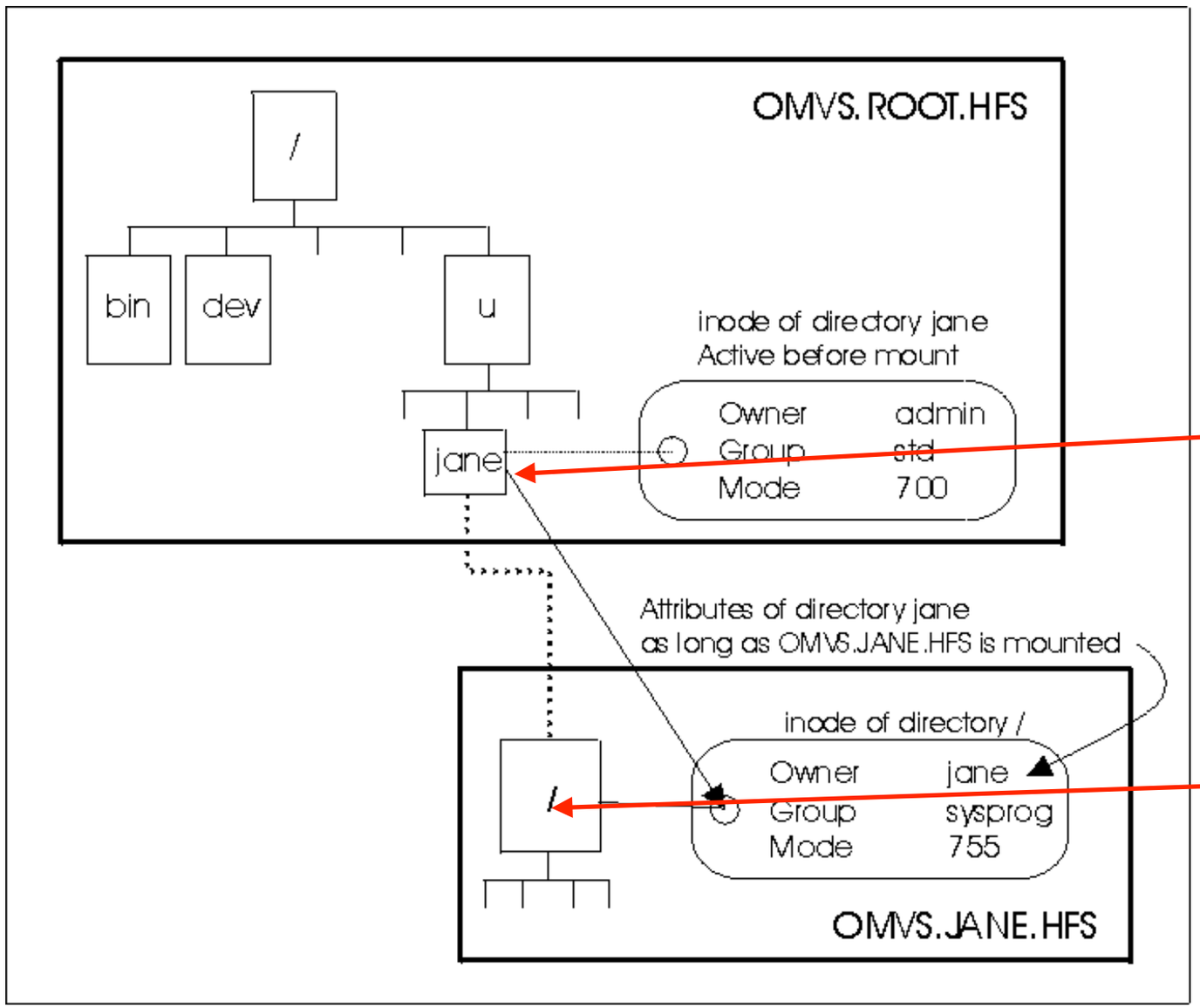
- Alias
  - Second mechanism:
    - Alias is another leaf node
    - Which contains the absolute pathname of the entity
  - UNIX symbolic links



Naming Entities

- Merging name spaces:
  - Done by mounting one namespace in another
    - Directory node stores identifier of a directory from a different name space
    - This is the *mount point*
    - Directory node in the foreign name space is the *mounting point*

Naming Entities



Mount point

Mounting point

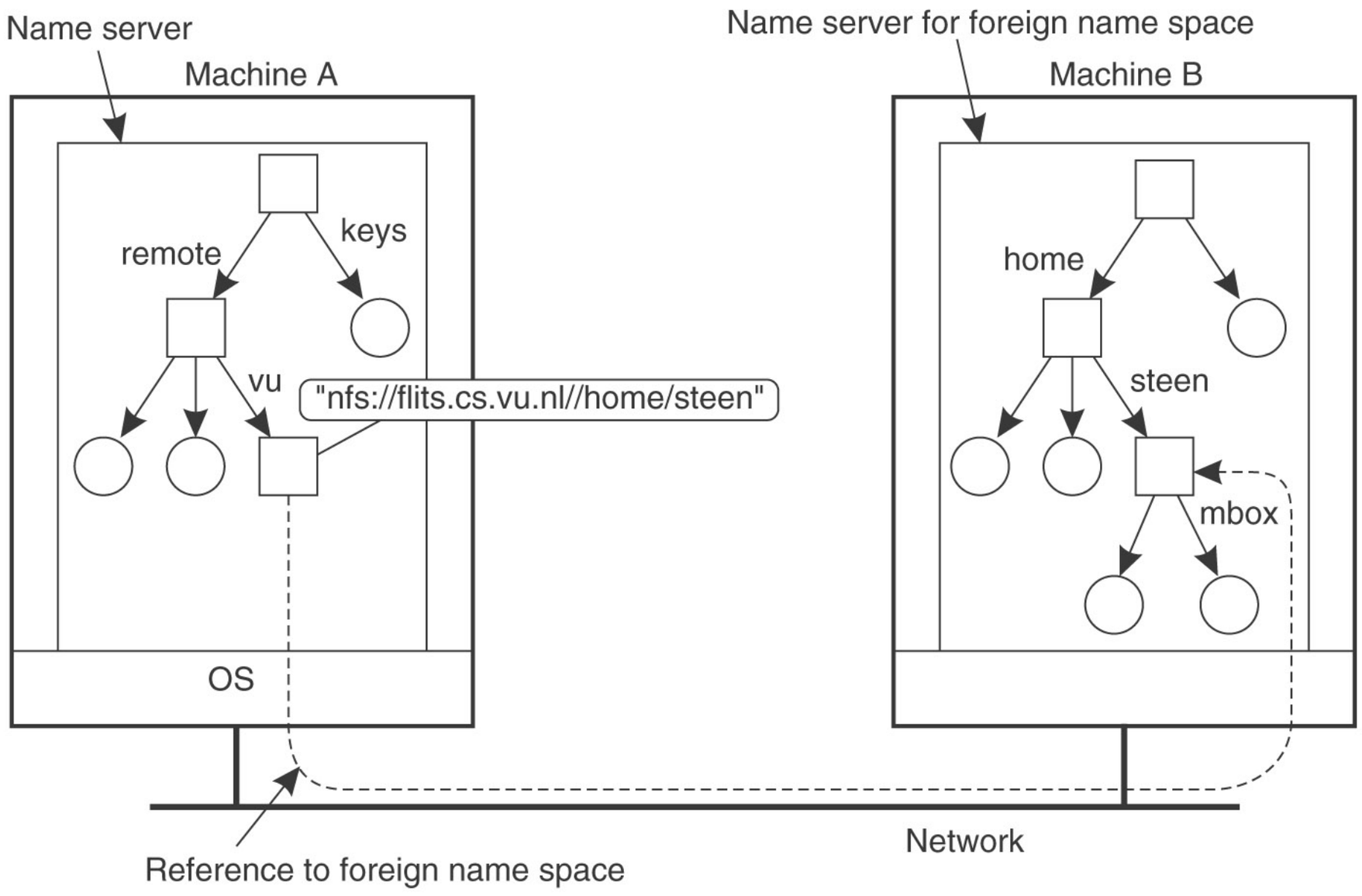
Naming Entities

- The process of mounting in a file system can be generalized to other name spaces
- Directory node that acts as mount point
  - Needs to have all information to access the mounting point in the foreign name space

Naming Entities

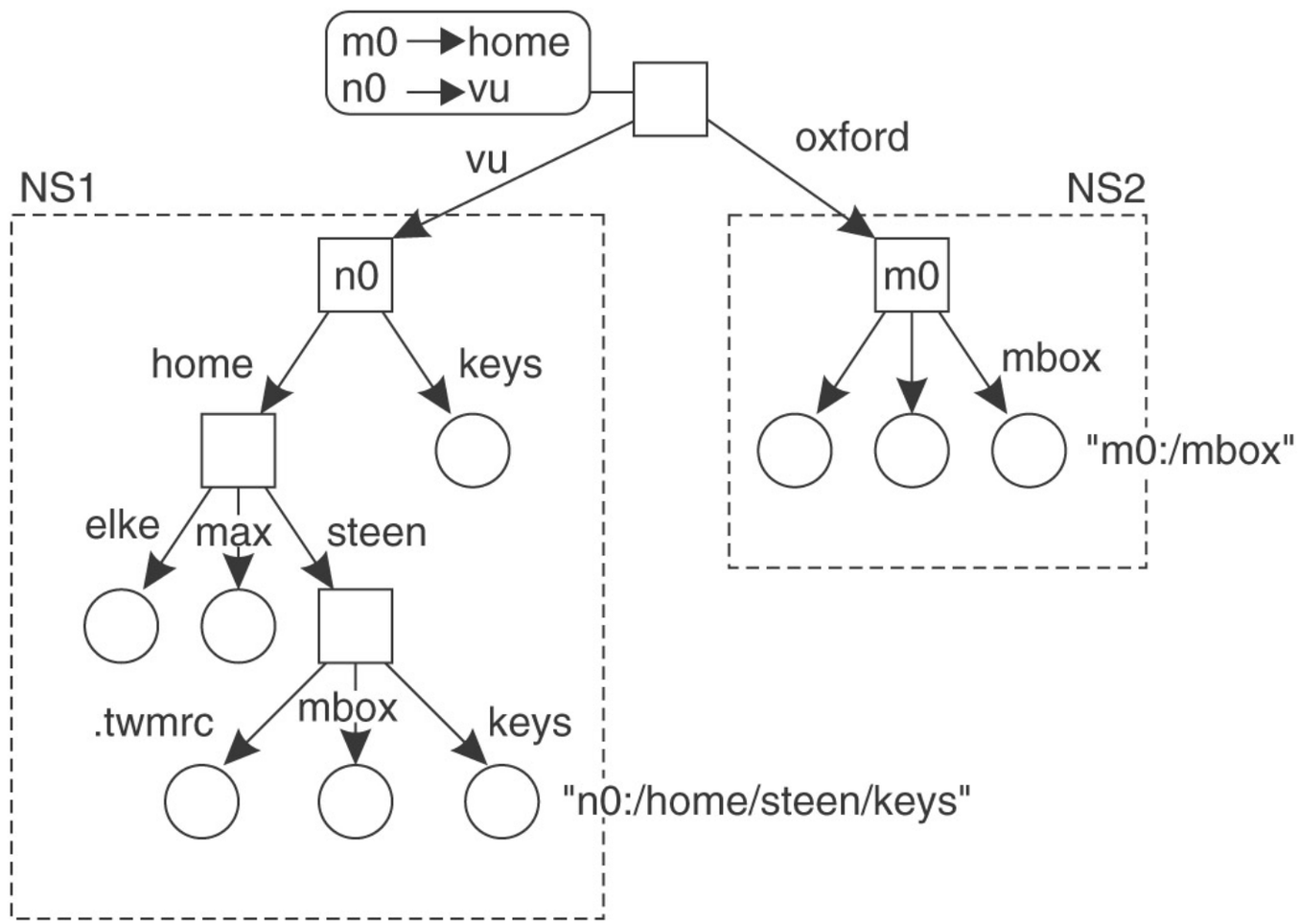
- SUN network file system
  - *nfs* in an URL specifies an nfs file
  - *nfs://beowulf.cse.ucsd.edu//home/tschwarz*
    - specifies file (which is a directory) called home/tschwarz on server beowulf.cse.ucsd.edu

Naming Entities



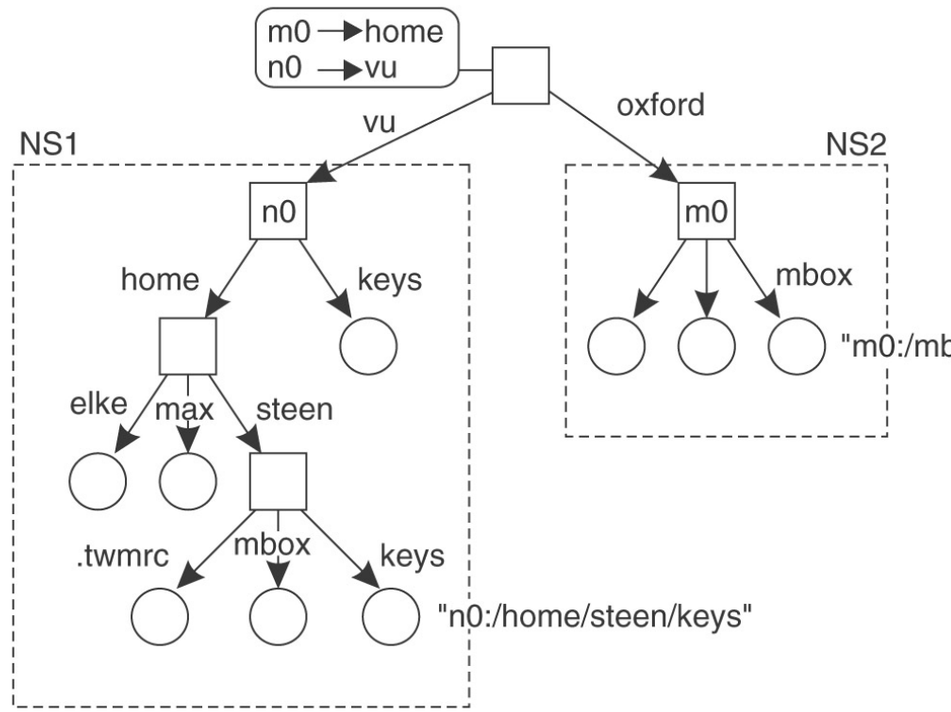
Naming Entities

- DEC's Global Name Service (GNS)
  - Merges namespaces by adding a new root and making the existing roots its children



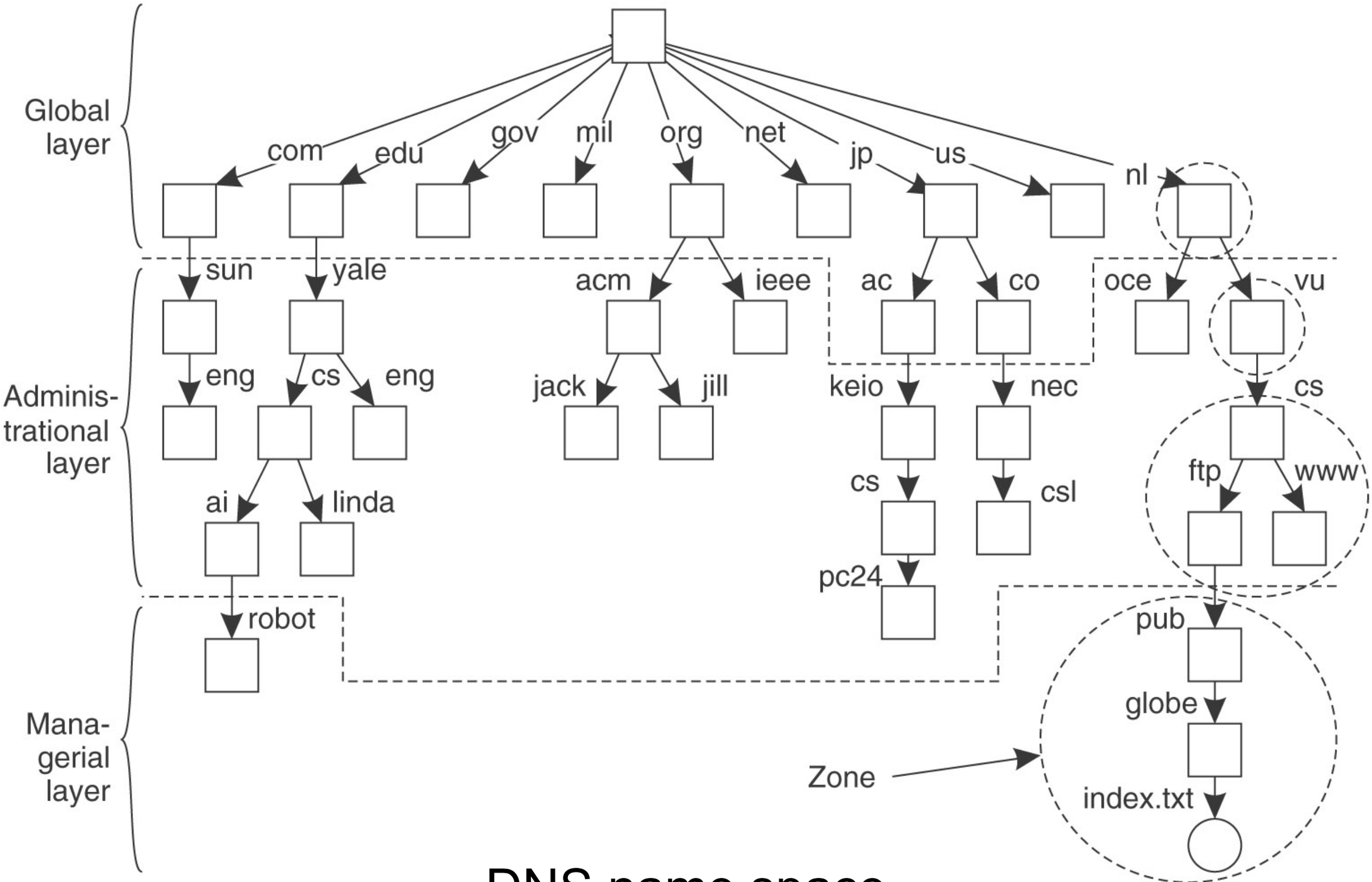
Naming Entities

- Can merge two namespaces without changing either one
- New root needs to be known in either namespace
- Resolution starts from there
  - Example:
    - `n0:/home/steen`
    - translated to `vu/home/steen`
    - Resolved in the new namespace



Naming Entities

- Implementing a name space
  - Name spaces for large-scale systems are hierarchical
  - But partitioned into logical layer
  - Global layer:
    - Highest level nodes
- Administrative layer:
  - Directory nodes managed within a single organization
- Managerial layer:
  - Nodes that change regularly



DNS name space

Naming Entities

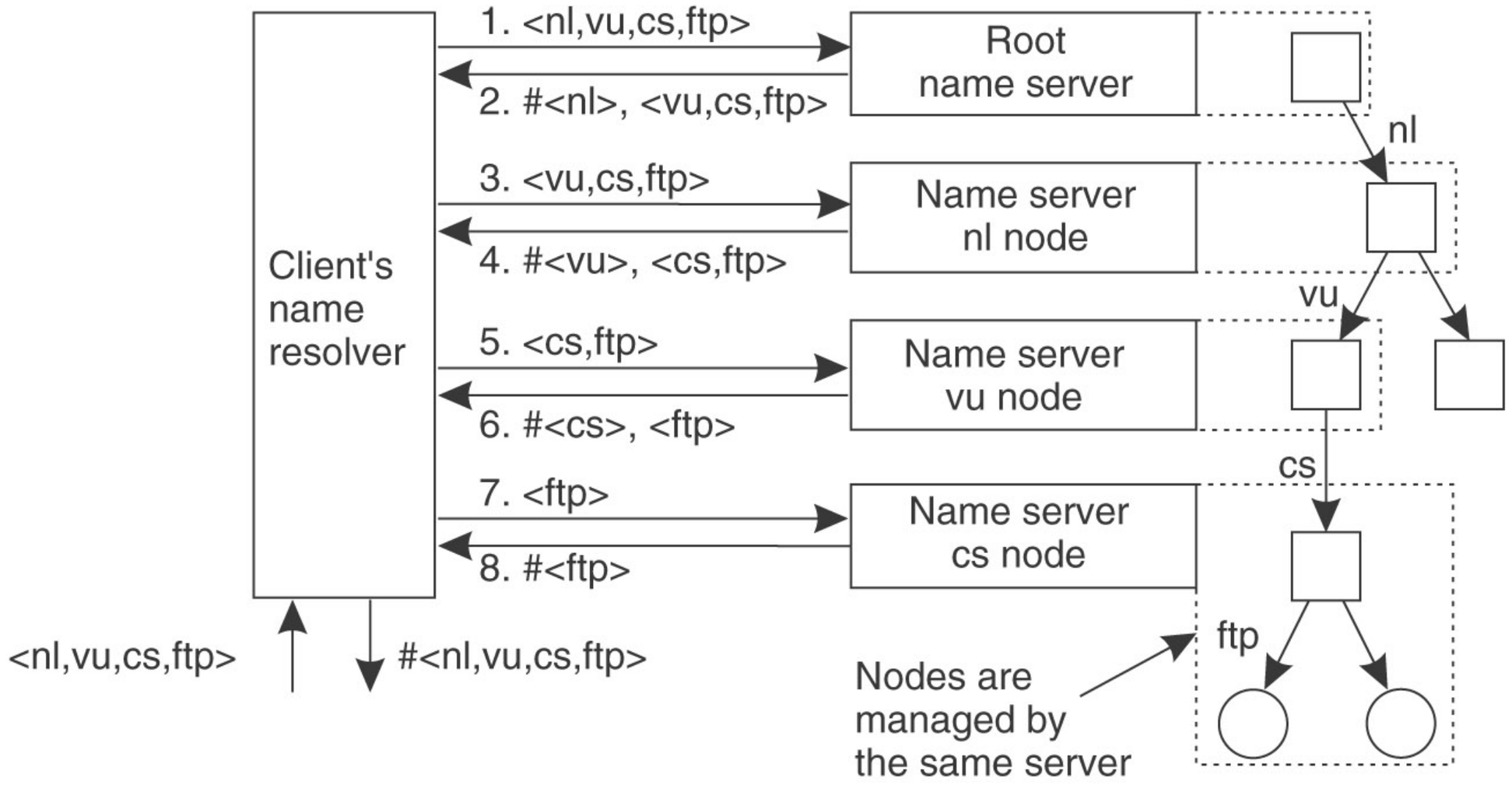
- Name space servers:
  - Availability more critical at the organizational layer
  - Performance
    - Administrative and global layer have few changes
    - Caching is very effective
    - Administrative layer needs faster responses

Naming Entities

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

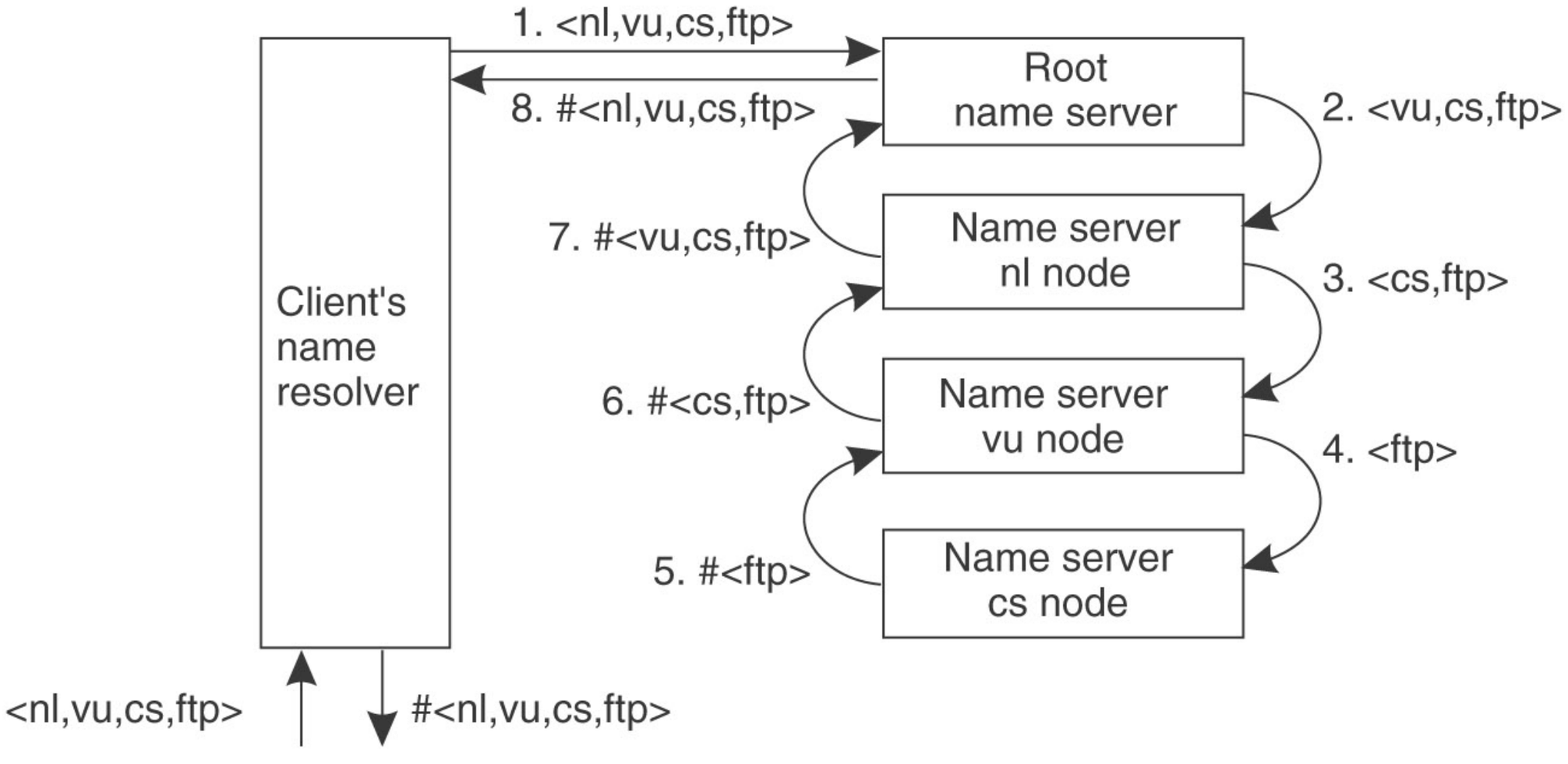
Naming Entities

- Implementation of name resolution
  - Iterative name resolution for nl:vu:cs:ftp



Naming Entities

- Implementation of name resolution
  - Recursive name resolution for nl:vu:cs:ftp



Naming Entities

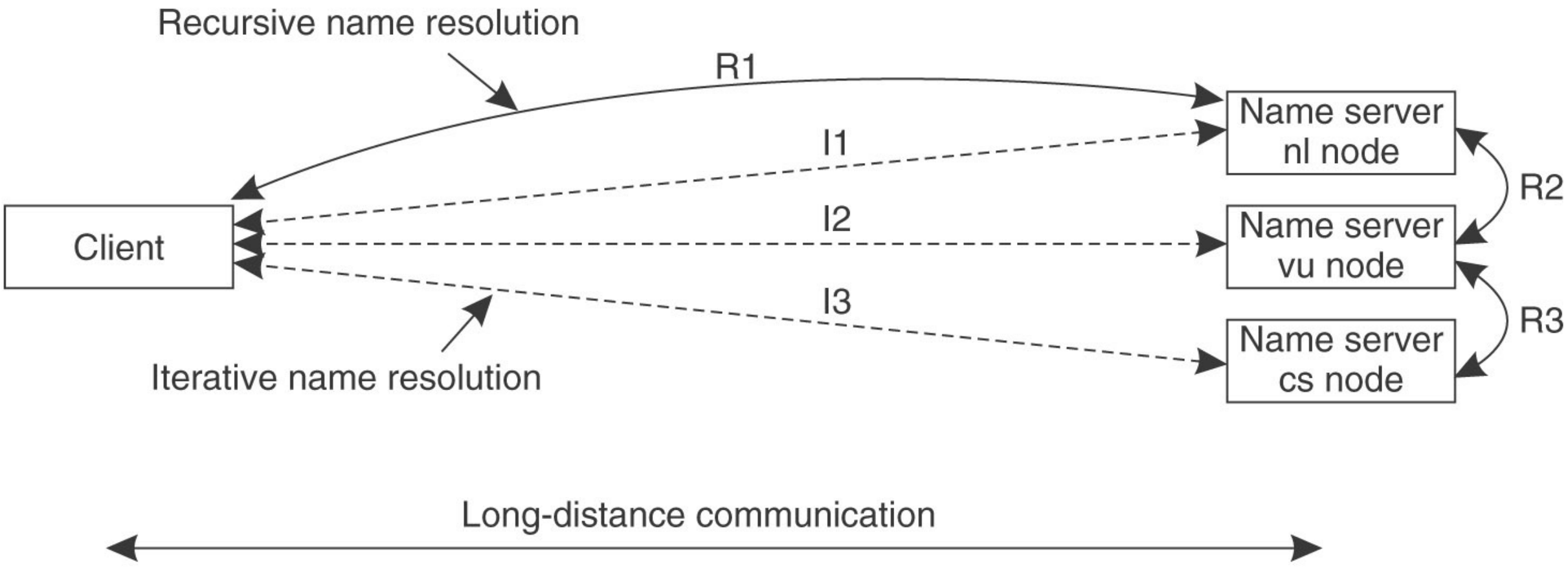
- Recursive name resolution
  - Caching results is more effective
  - Less communication costs
- Iterative name resolution
  - Caching restricted to client's name resolver

Naming Entities

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 with caching

Naming Entities



Naming Entities

- Directory Service:
  - Allows look up based on a description of properties instead of a full name

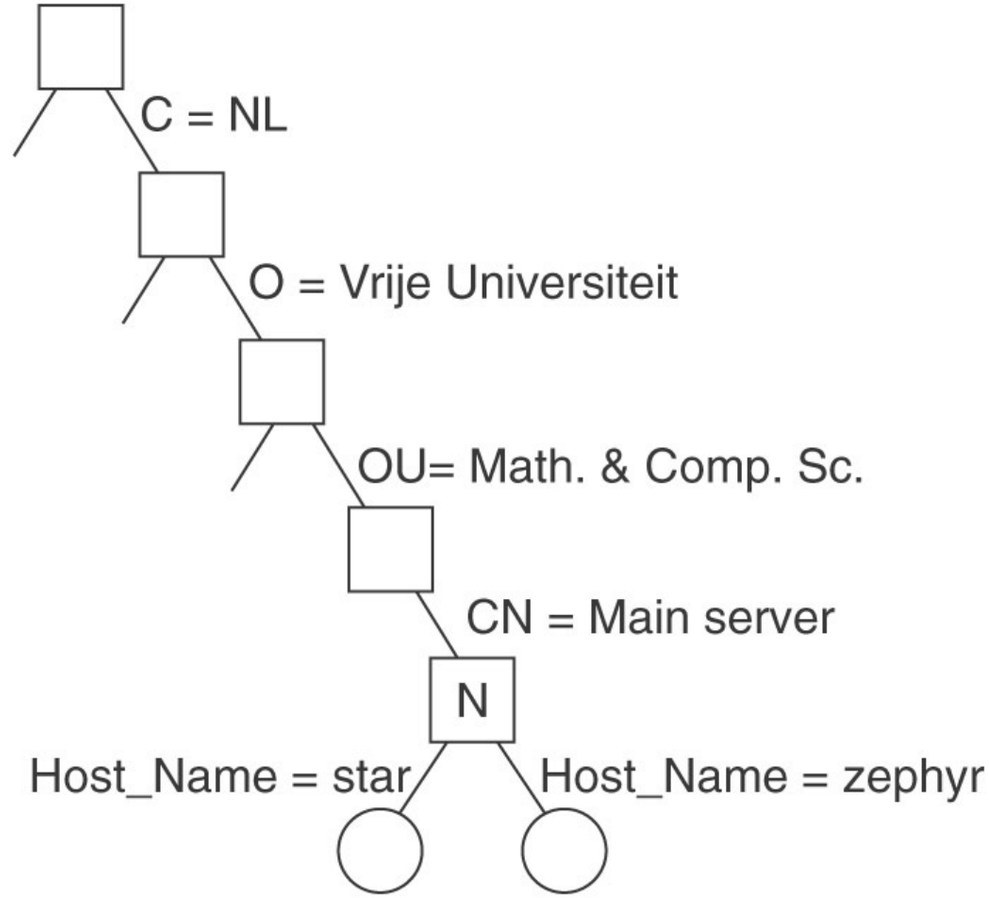
Naming Entities

- X.500
  - Directory entries consists of a list of attribute value pairs
    - /C=NL/O=Vrije Universiteit/OU= Math. & Comp. Sc.

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, 192.31.231.42, 192.31.231.66
FTP_Server	—	130.37.24.11
WWW_Server	—	130.37.24.11

Naming Entities

- X.500
  - Directory Information Tree



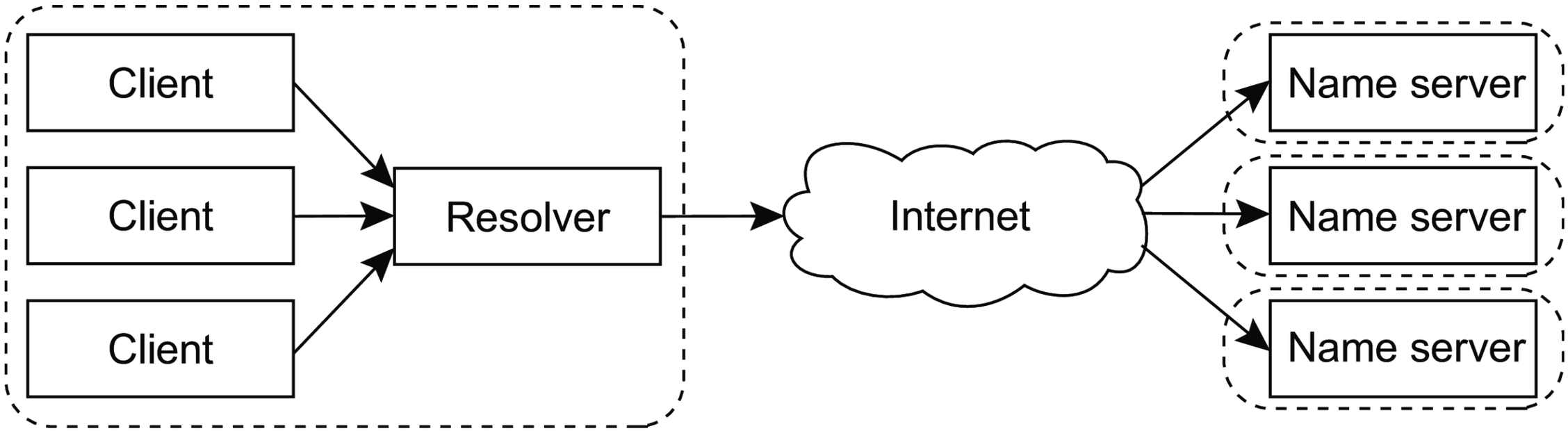
Naming Entities

- X.500 Implementation
  - Partition Directory Information Tree across several servers, the **Directory Service Agents**
  - **Directory User Agents** responsible for lookup
    - Have more lookup options than DNS
- **Lightweight Directory Access Protocol (LDAP)**
  - Simplified protocol to accommodate X.500 searches
  - Simpler than official X.500 protocol

DNS Implementation

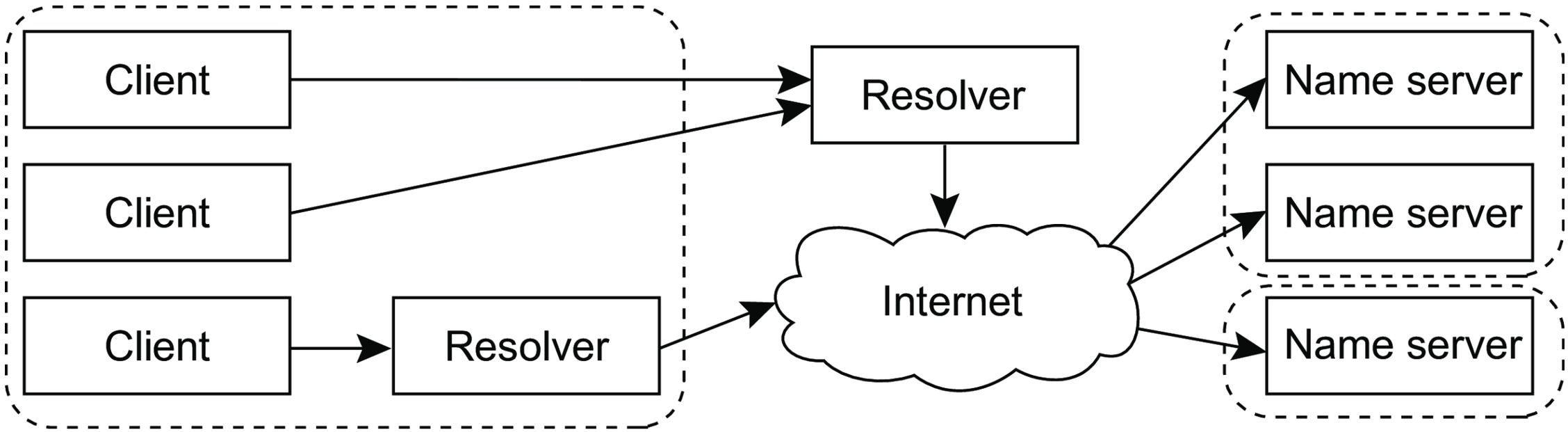
- “Old” DNS is becoming obsolete
  - Many organization use an external DNS server
    - CDS use the DNS server address to select the nearest replica
  - Browsers often have their own choice of DNS servers
  - DNS is often outsourced
    - and so is less centralized

DNS Implementation



Old DNS

DNS Implementation



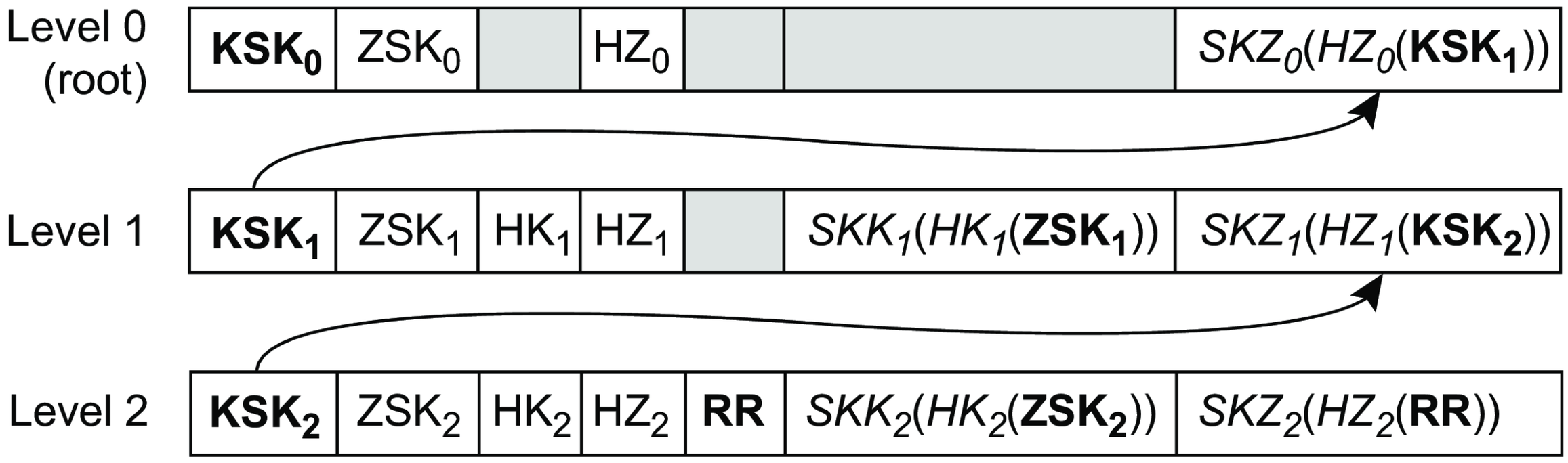
Modern DNS
(partially implemented)

DNS Implementation

- DNS is becoming more centralized
  - DNS records need to be authenticated
  - Top-level DNS already uses DNSSEC with signing of DNS records
  - DNSSEC allows for longer packets
  - Uses a trust chain from the root

DNS Implementation

- DNSSEC trust chain from root server



DNS Implementation

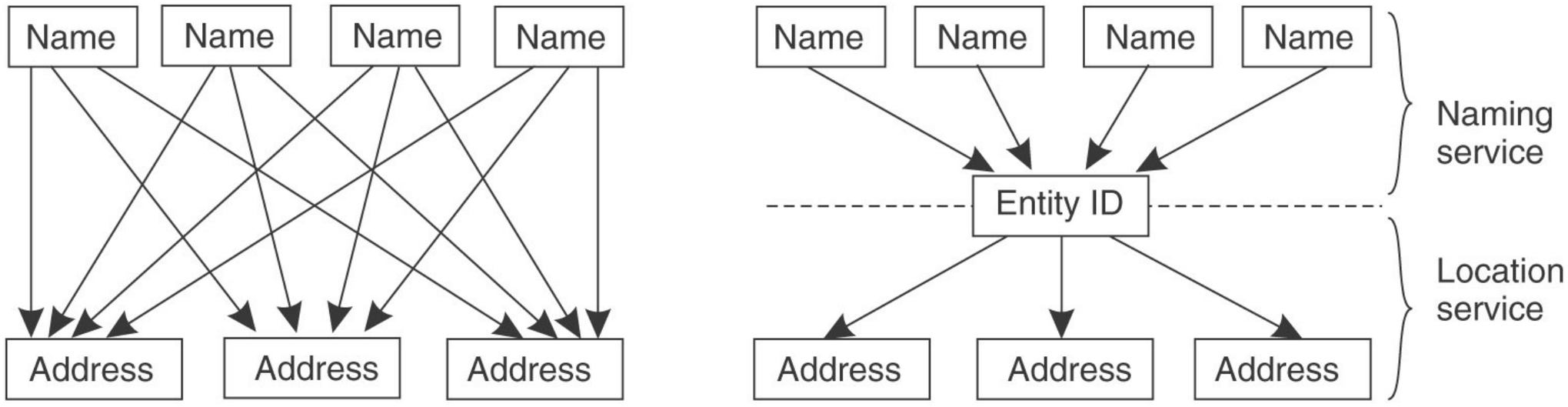
- Protecting privacy of DNS users
  - To make DNS queries private:
    - Use TLS
    - Use HTTPS

DNS Implementation

- Future DNS:
  - Completely decentralized??

Location Mobile Entities

- Traditional systems (DNS) have difficulties if entities move between domains
- To enable location:

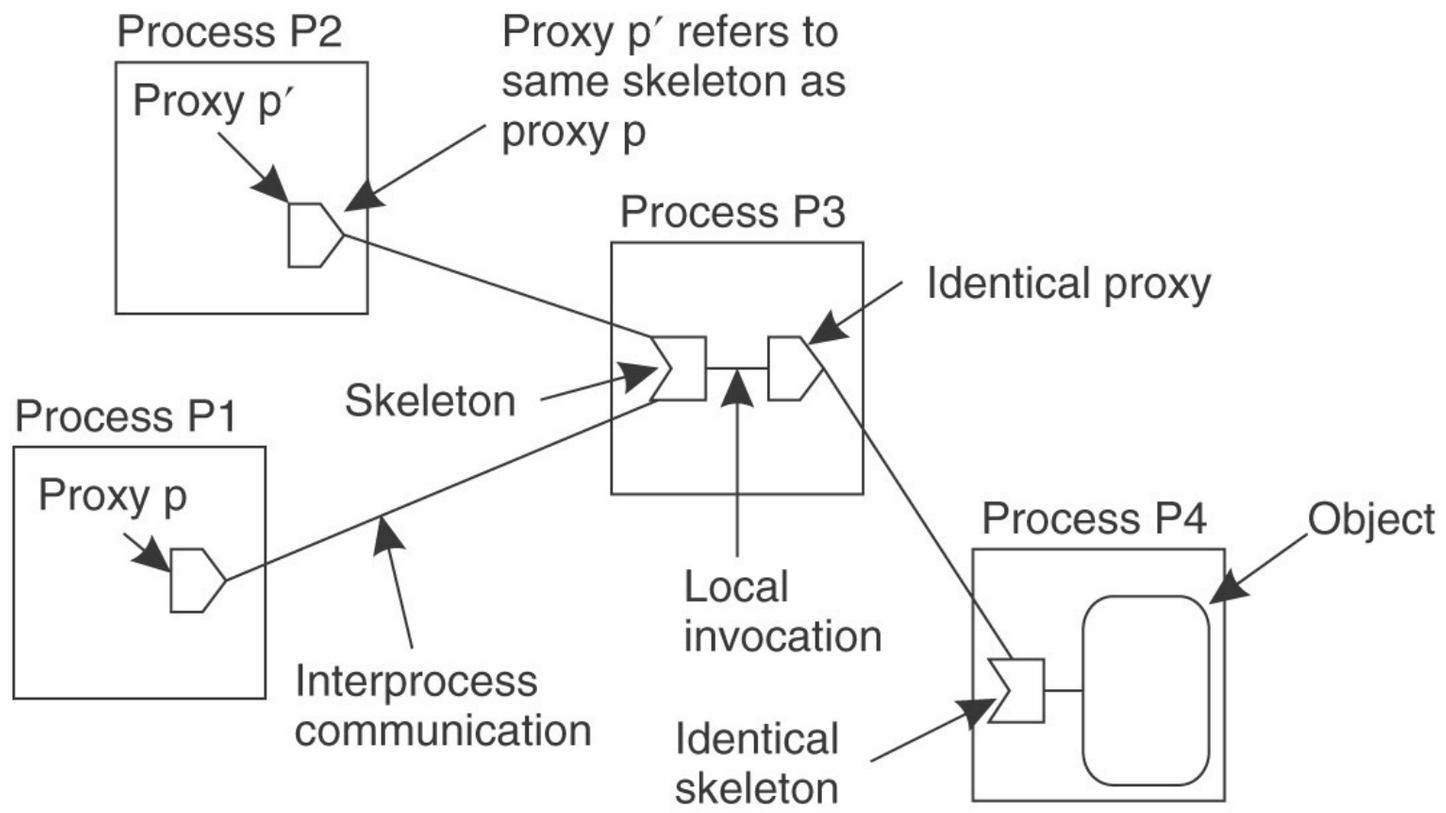


Locating Mobile Entities

- Simple solutions (LAN)
  - Broadcasting and Multicasting
    - Example: ARP
  - Forwarding Pointers
    - Entity leaves pointer behind when moving

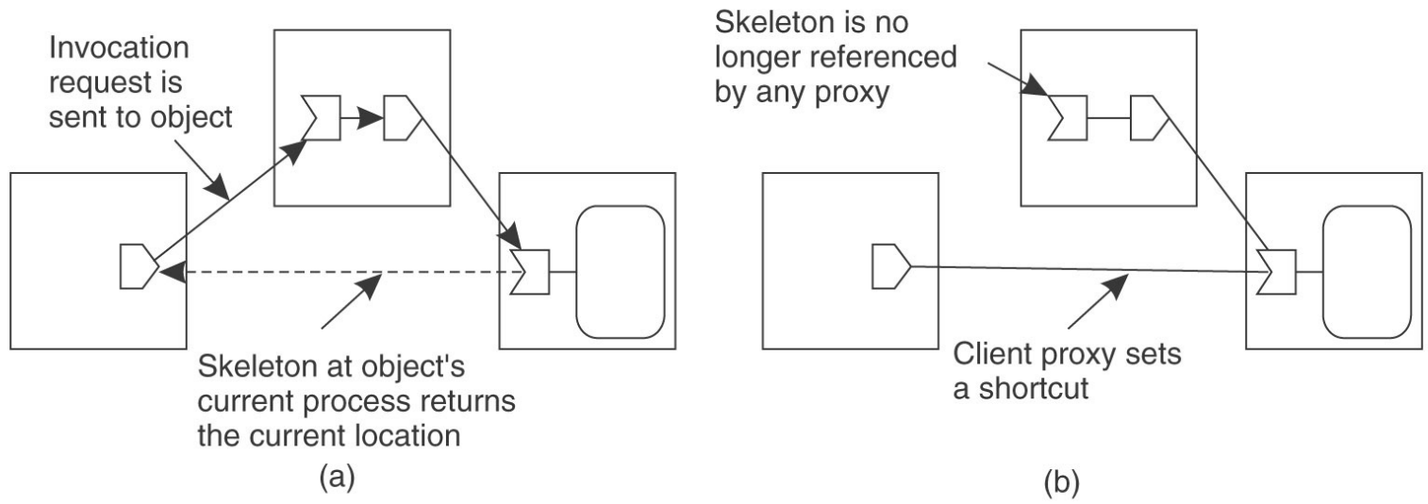
Locating Mobile Entities

- Example for forwarding pointers:
  - SSP chains for distributed objects
  - Skeletons are entry items
  - Proxies are exit items
  - Whenever an object moves, it leaves behind a proxy



Locating Mobile Entities

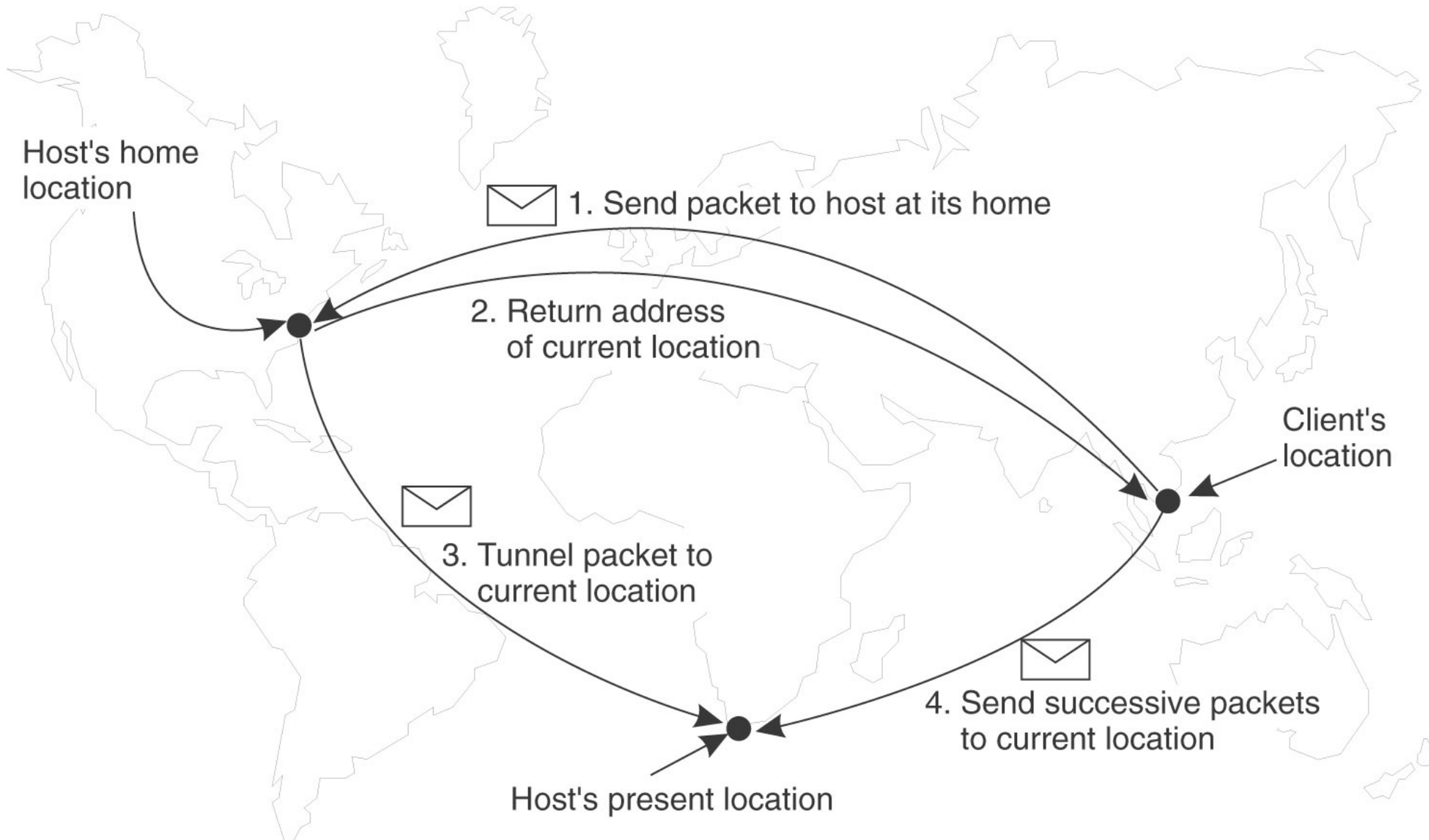
- Example for forwarding pointers:
  - SSP chains for distributed objects
    - Whenever an object moves
      - it leaves behind a proxy
    - To avoid following long chains
      - Create a short-cut whenever you are following a cha...
      - Send answer via proxies to invoking skeleton
        - Avoids orphan skeleton-proxy pairs
      - Send answer to original client directly
        - Better communication
- To avoid problems with crashed proxy, skeletons:
  - Object's home location
    - where object was created
    - keeps the address of object wherever it is now



Locating Mobile Entities

- Home based approaches
  - Maintain home location
    - **Home agent**
  - Mobile entity creates **care-of address**
  - Example: Mobile IP

Locating Mobile Entities

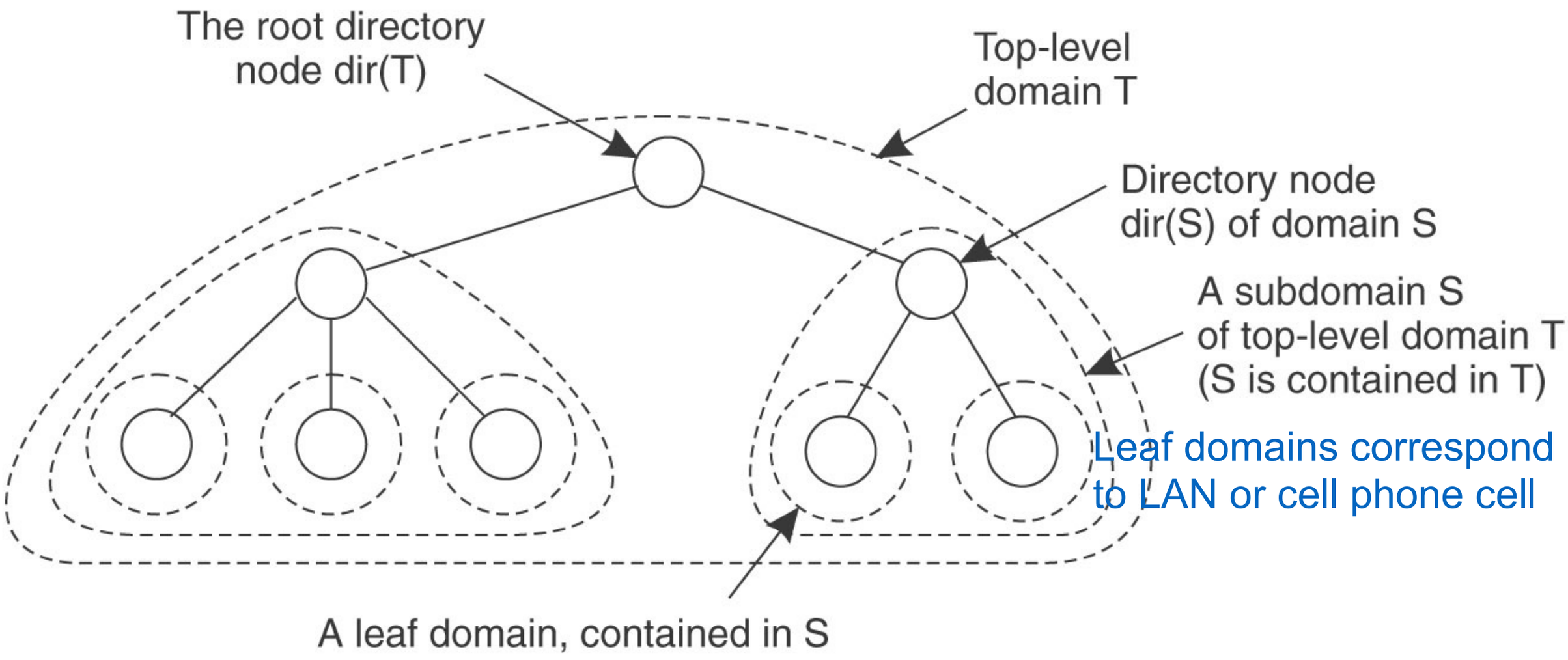


Locating Mobile Entities

- Mobile Telephony
  - Client first checks local registry
    - whether client is available locally
  - If not: contact entity’s home location to find current location

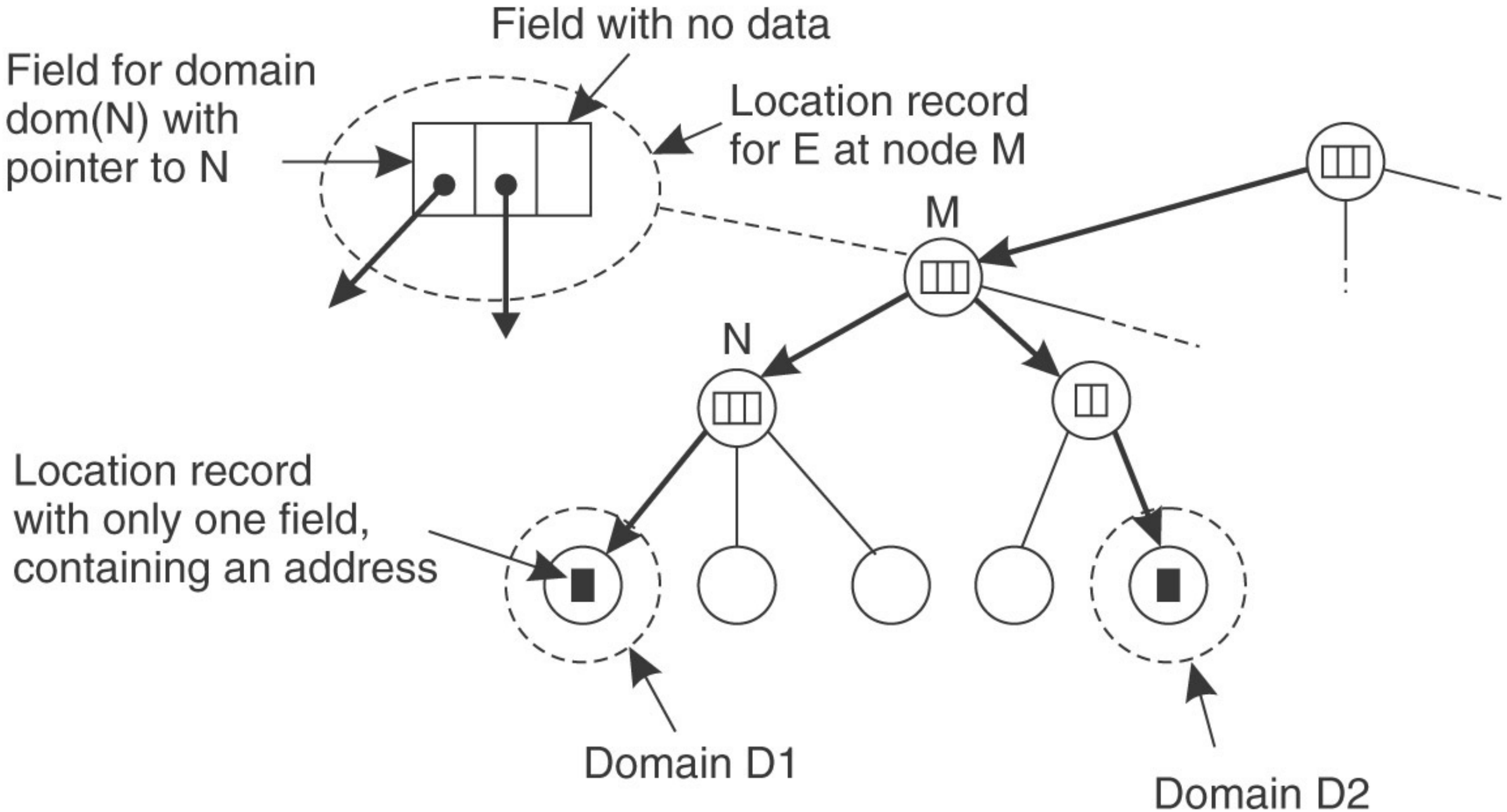
Locating Mobile Entities

- Can be extended to hierarchical approaches



Locating Mobile Entities

Entity can have two addresses



Attribute Based Naming

- Entities have several <attribute: value> tuples
- Attribute based naming is also known as ***directory services***
  - ***Resource Description Framework (RDF)***
    - Describe resources as triples:
      - Subject — Predicate — Object
    - Example: “New York has the postal abbreviation NY”
    - `<urn:x-states:New%20York> <http://purl.org/dc/terms/alternative> "NY"`

Attribute Based Naming
LDAP

- Lightweight Directory Access Protocol (LDAP)
  - LDAP directory service:
    - Number of directory entries as attribute-value pairs

Attribute	Abbr.	Value
Country	C	NL
Locality	L	Amsterdam
Organization	O	VU University
OrganizationalUnit	OU	Computer Science
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

Attribute Based Naming LDAP

- ***Directory Information Base*** (DIB): the collection of all directory entries
  - Each record is uniquely named: **Relative Distinguished Name** (RDN)

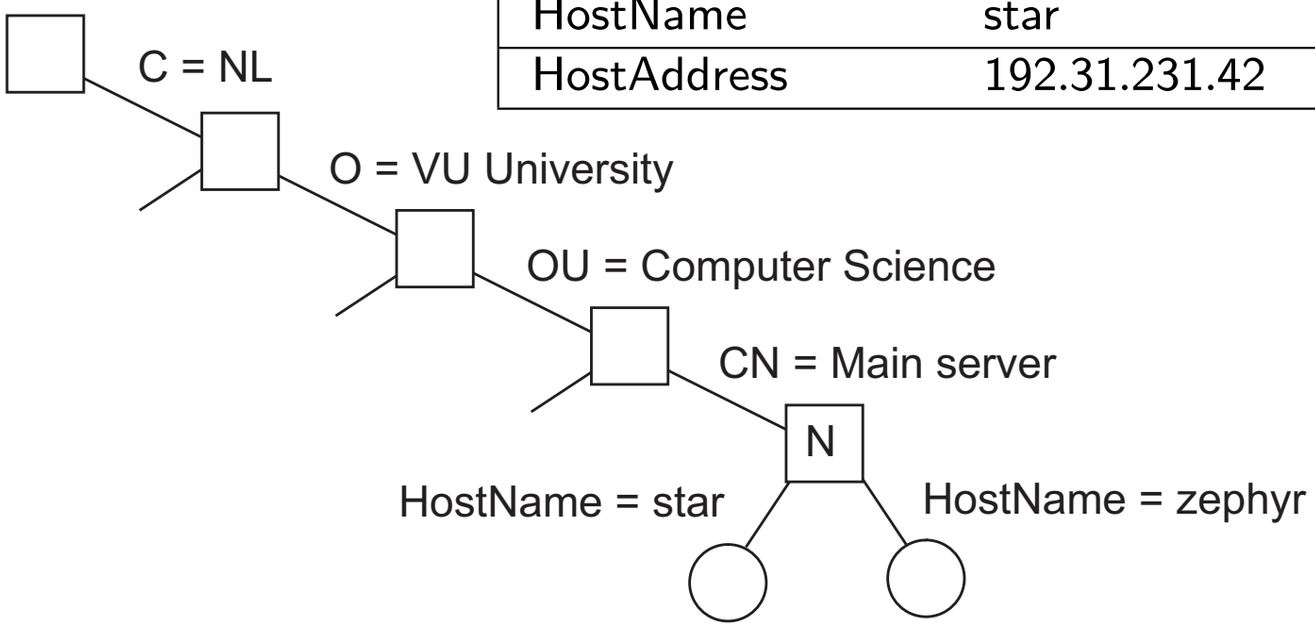
Attribute	Abbr.	Value
Country	C	NL
Locality	L	Amsterdam
Organization	O	VU University
OrganizationalUnit	OU	Computer Science
CommonName	CN	Main server
Mail_Servers	–	137.37.20.3, 130.37.24.6, 137.37.20.10
FTP_Server	–	130.37.20.20
WWW_Server	–	130.37.20.20

```
/C= NL/O= VU University/OU= Computer Science
```

Attribute Based Naming LDAP

- This gives a hierarchy of the collection of directory entries: **Directory Information Tree**

Attribute	**Value**	**Attribute**	**Value**
Locality	Amsterdam	Locality	Amsterdam
Organization	VUUniversity	Organization	VUUniversity
OrganizationalUnit	ComputerScience	OrganizationalUnit	ComputerScience
CommonName	Mainserver	CommonName	Mainserver
HostName	star	HostName	zephyr
HostAddress	192.31.231.42	HostAddress	137.37.20.10



Attribute Based Naming
LDAP

- LDAP look-up is similar to DNS
  - Usually, DIT is partitioned and distributed across several servers, the **directory service agents** (DSA)
  - Clients are represented by **directory user agents** (DUA)

Attribute Based Naming LDAP

- LDAP lookup has more facilities
  - `search("(C=NL)(O=VU University)(OU=*)(CN=Main server)”)`
- Lookups usually involve various leaf nodes of a DIT
  - Usually, several DSA need to be accessed

Attribute Based Naming LDAP

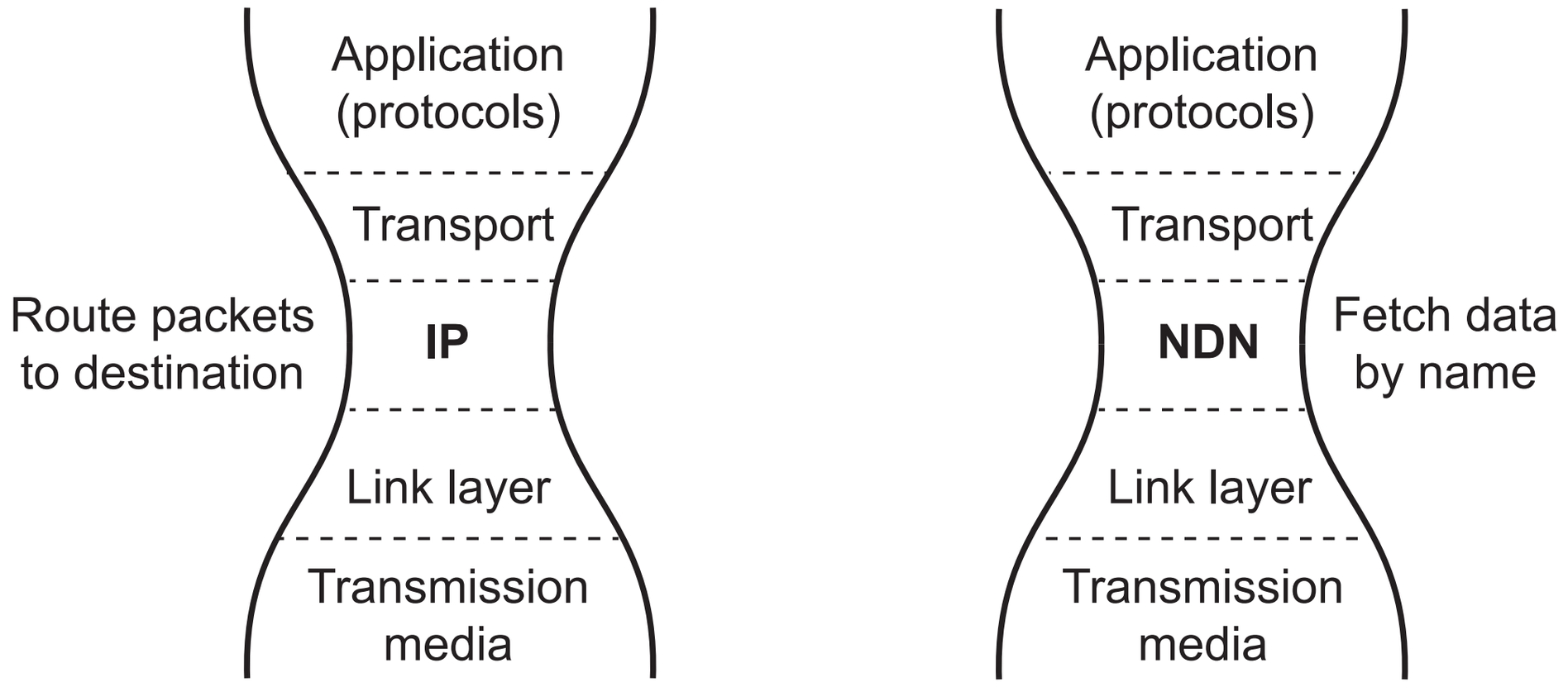
- Microsoft's Active Directory has a forest of LDAP domains
- Active Directory usually assumes there is a global index server (called a global catalog)

Attribute Based Naming Decentralized Implementations

- Building a distributed index
  - Naïve implementation
    - Use a different server for each attributes
    - Queries like
      - `search("(Country=NL)(Organization=VU University)(OrganizationalUnit=*)(CommonName=Main server)”)`
    - sent to the servers for Country, Organization, and CommonName

Named Data Networking

- **Information-centric networking**
  - **Named-Data Networking (NDN)**



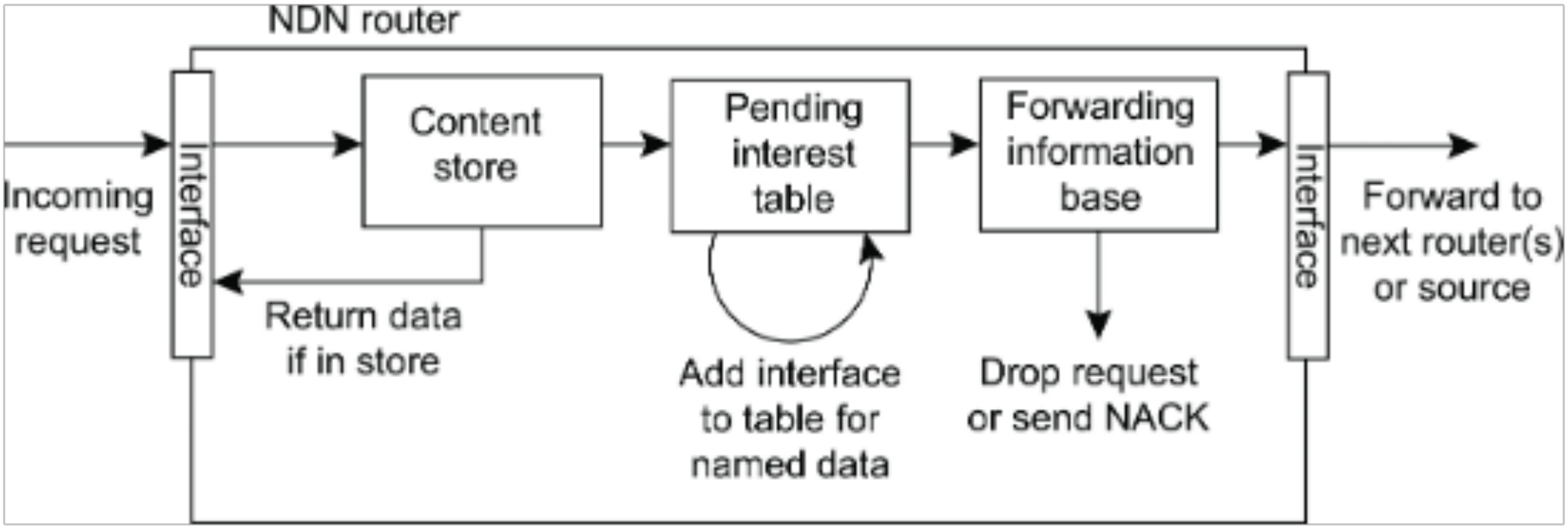
Named Data Networking

- Basics
  - Retrieve an entity from the network by using that entity's name and not address.
  - The network takes that name as input, and routes a request to a location where the entity is stored.
  - NDN takes over the role of IP in a future architecture of the Internet.

Named Data Networking

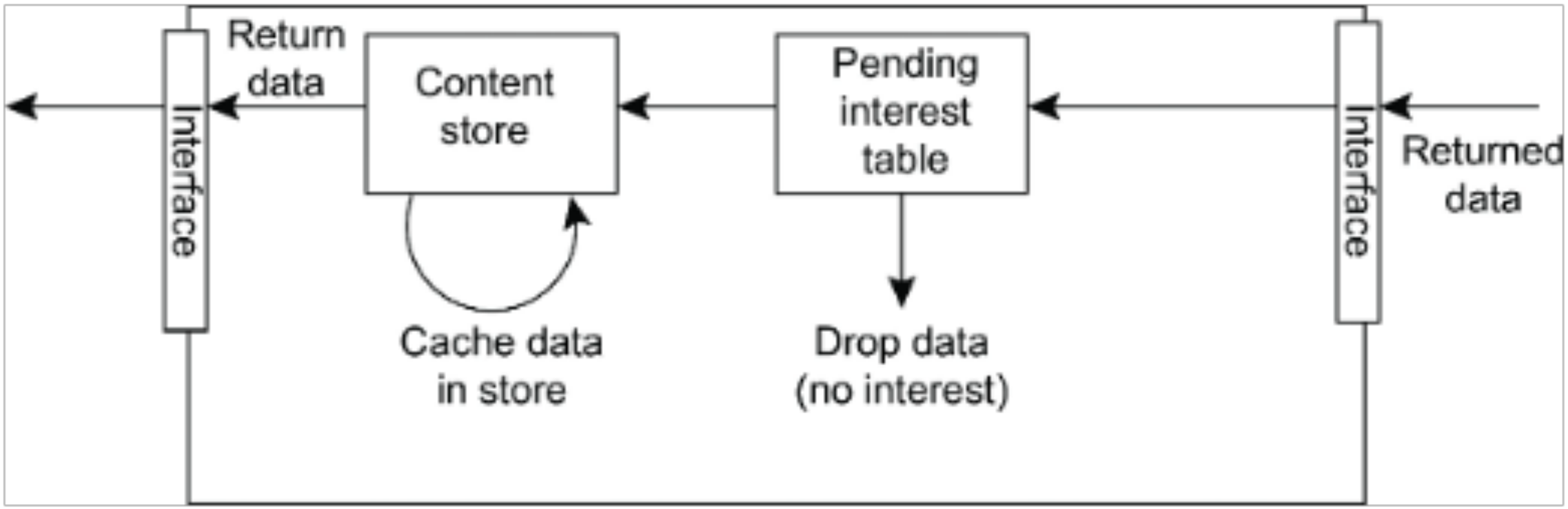
- NDN
  - A content store is essentially a cache for keeping data associated with a name
  - A pending interest table keeps track of a (name,interface) pair: if a request for data named N arrived through interface I of the router, then (N,I) is stored.
  - A forwarding information base tells the router what to do when it cannot serve a request

Named Data Networking



Forwarding a request to (a next router on the way to) its destination

Named Data Networking



Returning the request (to a router) on the path toward requester