Distributed Operating Systems

Naming

dr inż. Adam Kozakiewicz

akozakie@ia.pw.edu.pl

Institute of Control and Information Engineering
Warsaw University of Technology

Naming – Introduction

Applications of names:

- √ sharing of resources,
- √ unique identification of entities (objects),
- √ resource location.

Names are *resolved*, resulting in the named entity.

Name – a sequence of bits or characters used to identify an entity.

- √ typical entities: hosts, printers, disks, files, processes, users, mailboxes, web pages, messages, network connections, …
- √ **access point** – a special entity providing access to another entity
- √ **address** – name of the access point,
- √ address of the access point for an entity is simply called the entity's address.

Names, Identifiers, Addresses

- √ if an entity offers more than one access point, it is not clear which one to use as a reference,
- √ location independent names are necessary,

Identifier – a name used to uniquely identify an entity.
Properties:

1. An identifier never refers to more than one entity.
2. No entity is referred to by more than one identifier.
3. Once assigned, an identifier always refers to the same entity – it cannot be reused.

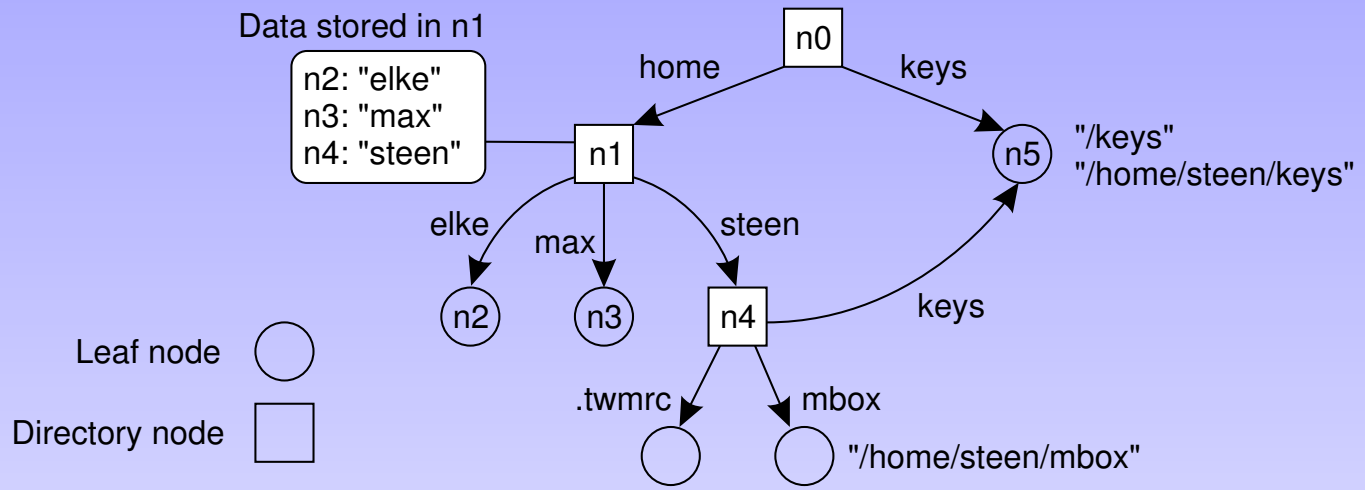
Remarks:

- √ References using identifiers are unambiguous.
- √ Identifiers and addresses are rarely good **human-friendly names**.

Name Spaces (1)

- √ names in distributed ssystems are usually organized into **name spaces**.
- √ a name space can be represented as a labeled, directed graph,
  - ⋆ leaf nodes represent named entities,
  - ⋆ directory nodes store **directory tables** with a pair (edge label, node identifier) for each outgoing edge,
- √ **root node** of a naming graph,
- √ **path name**: N:<label-1, label-2, ..., label-n>,
- √ **absolute path name** starts at the root node, others are **relative path names**,
- √ **global name** – refers to the same entity throughout the system,
- √ **local name** – interpretation depends on context

Name Spaces (2)



A general naming graph with a single root node

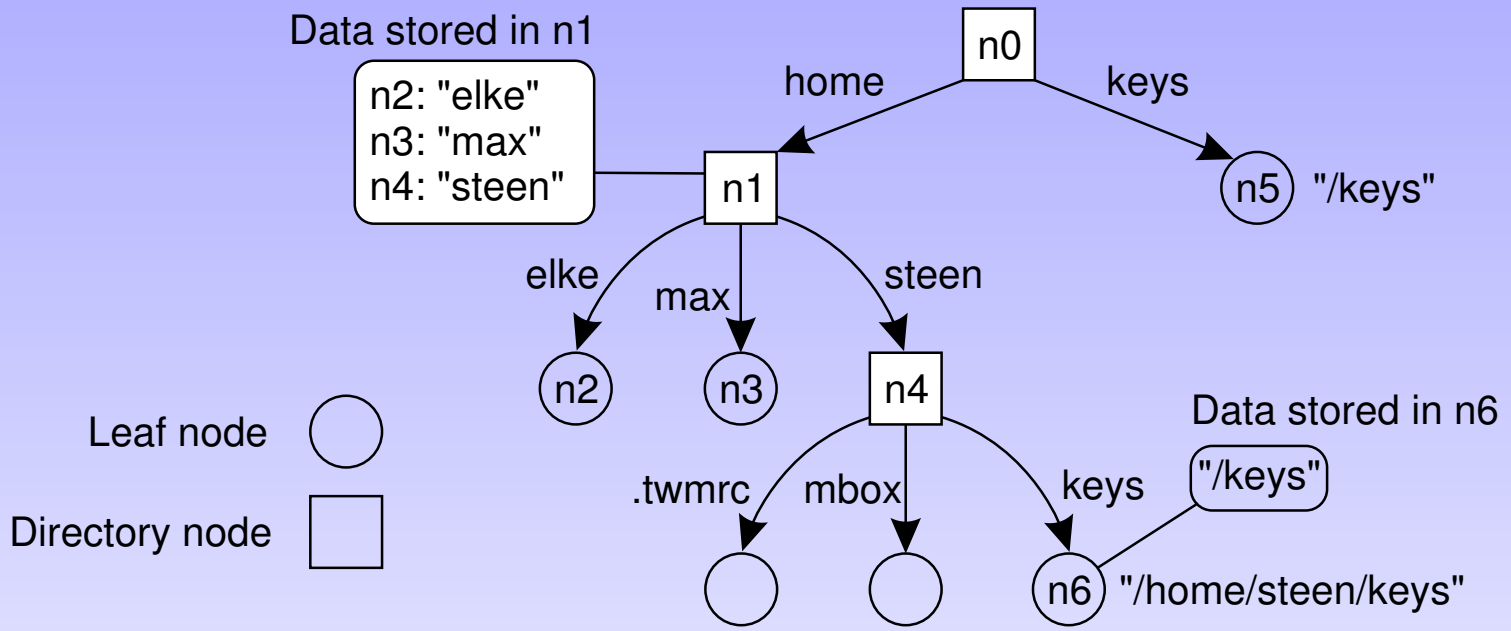
- √ n5 can be referred to by /home/steen/keys, but also /keys,
- √ directed **acyclic** graph,
- √ in Plan9 all resources, including processes, hosts, I/O devices, network interfaces, etc. are named as files – a single naming graph.

Name Resolution

Name resolution – the process of looking up a name given a path name.

- √ a name lookup returns the identifier of a node from where the resolution process continues,
- √ **closure mechanism** – defines **how** and **where** to start name resolution,
  - ⋆ Unix file system: the first inode in the logical disk is the inode of the root directory,
  - ⋆ “004921211555555” seen as a string makes no sense, but as a phone number it is recognizable,
- √ **alias** – another name for the same entity,
- √ **hard links** vs. **symbolic links**
  - ⋆ if the naming graph is a tree, only one hard link per entity is possible

Linking and Mounting (1)



The concept of a symbolic link in a naming graph.

Linking and Mounting (2)

- √ **mount point** – the directory node storing the node identifier,
- √ **mounting point** – the directory node in the foreign name space.

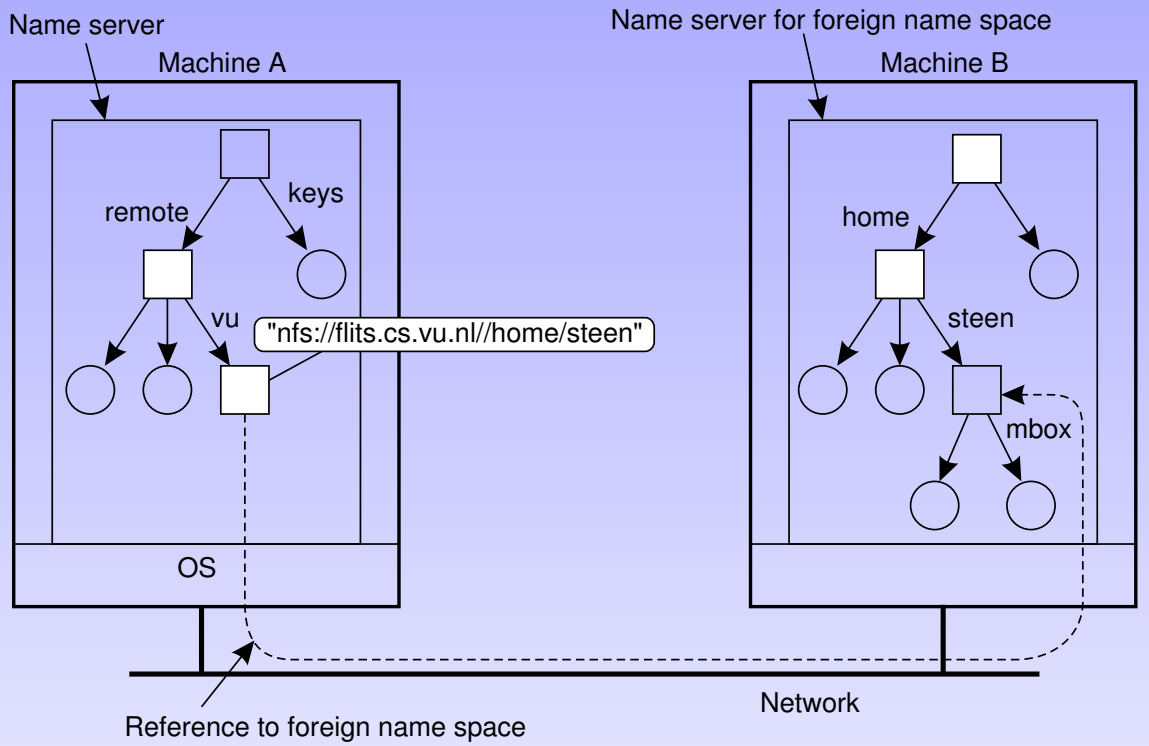
Information required to mount a foreign name space in a distributed system:

1. Name of an access protocol.
2. Name of the server.
3. Name of the mounting point in the foreign name space.

Remarks:

- √ each of these names has to be resolved,
- √ each of these names is required, but default values may hide that fact,
- √ NFS as an example.

Linking and Mounting (3)

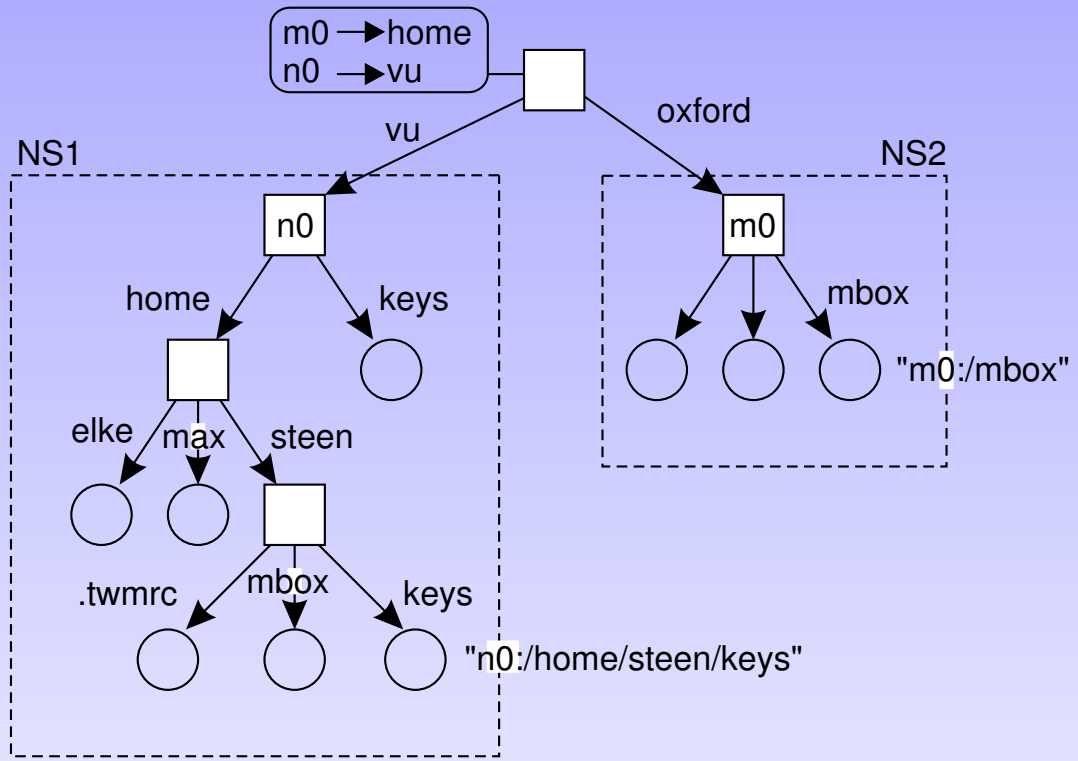


Mounting of a remote name space using the NFS protocol.

Linking and Mounting (4)

- **GNS** (*Global Name Service*, developed by DEC) introduces a new, global root node, making all existing ones its children.
- names in GNS always (implicitly) include the identifier of the node where resolution should normally start,
- e.g. /home/steen/keys in NS1 is expanded to n0:/home/steen/keys,
- hidden expansion,
- GFS assumes that node identifiers are **universally** unique.

Linking and mounting (5)



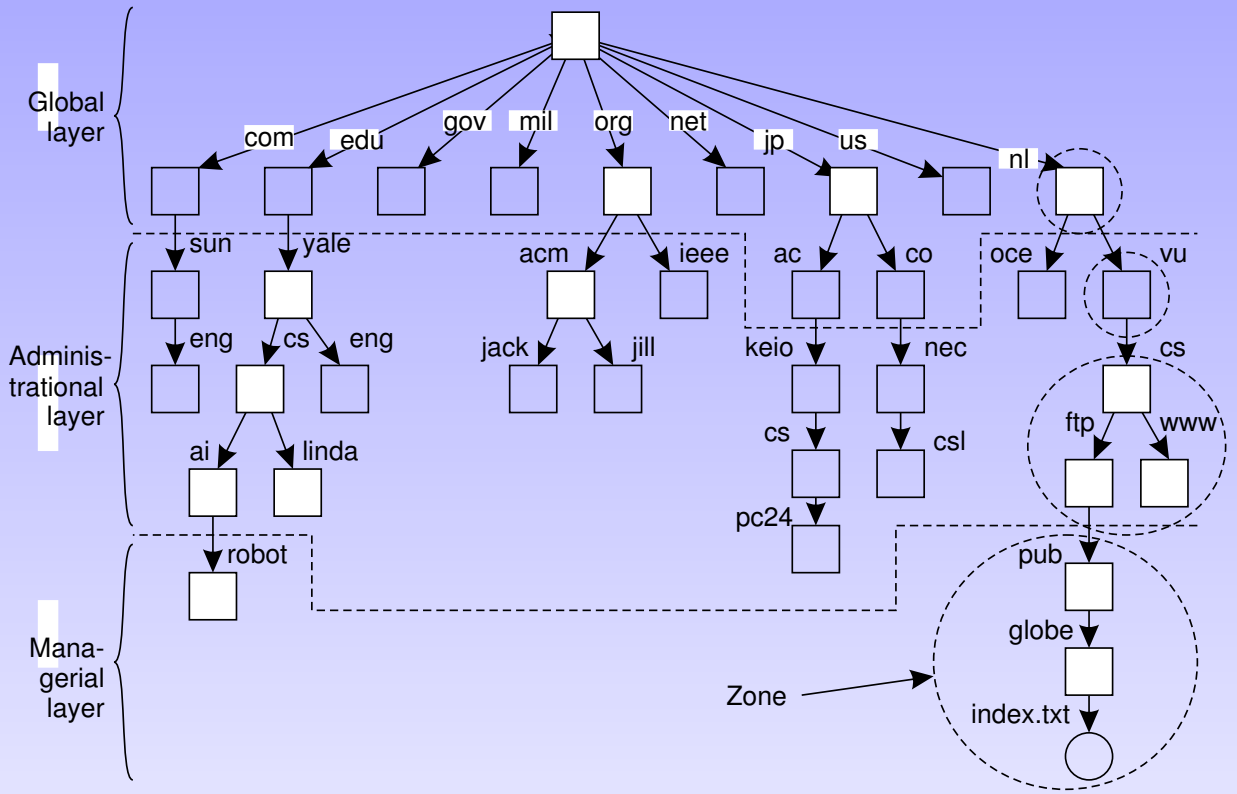
Organization of DEC Global Name Service.

Distributed Name Spaces (1)

- √ large scale name spaces are partitioned into logical layers
  - ⋆ global layer,
  - ⋆ administrational layer,
  - ⋆ managerial layer.

- √ a name space may be divided into zones,
- √ **zone** – a part of the name space that is implemented by a separate name server,
- √ reasons: availability, performance, caching, replication.

Distributed Name Spaces (2)



An example of DNS name space partitioning into three layers, zones indicated on one branch.

Distributed Name Spaces (3)

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 of name servers implementing nodes in a large-scale name space partitioned into three layers.

Implementation of Name Resolution (1)

Each client has access to a local **name resolver**.

- √ ftp://ftp.cs.vu.nl/pub/globe/index.txt
- √ root:<nl, vu, cs, ftp, pub, globe, index.txt>

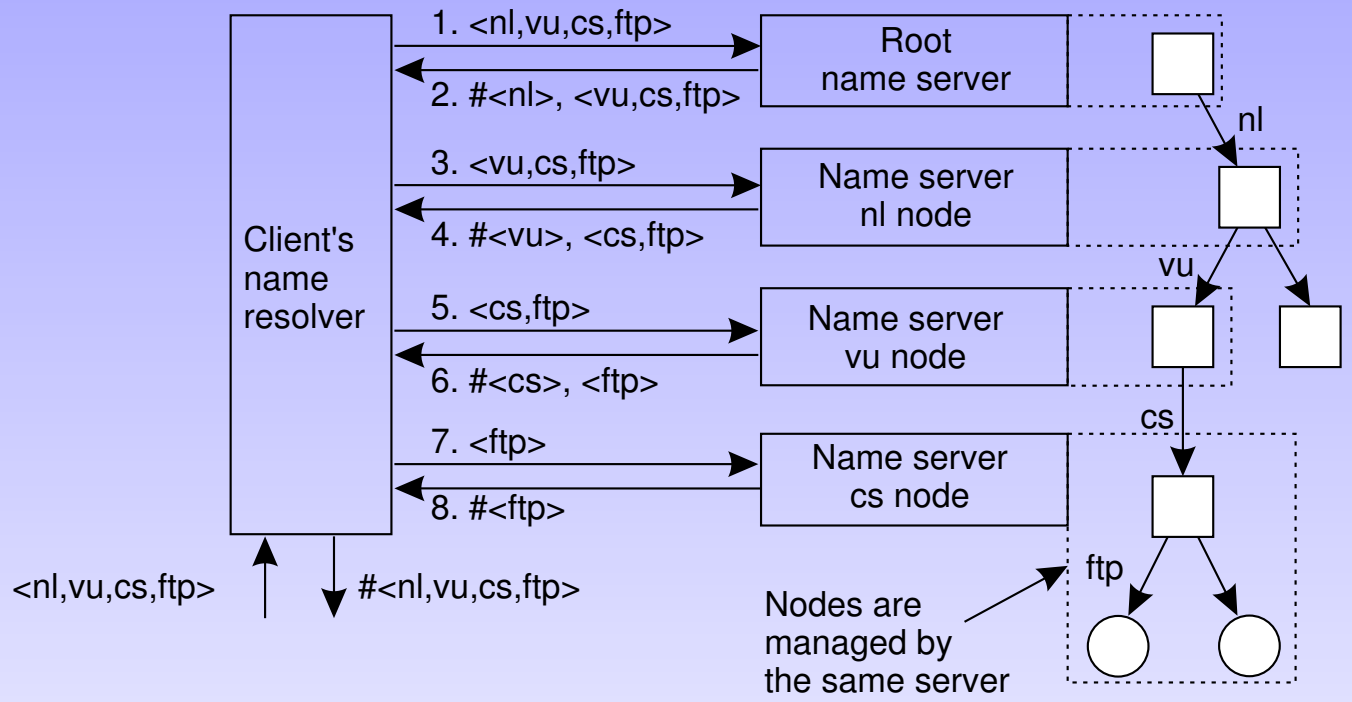
Iterative name resolution:

- √ name resolver hands over the complete name to the root name server, which resolves “nl” and returns the address of the associated name server; resolver then contacts that server and so on.
- √ buffering takes place only at the clients name resolver; possible compromise – intermediate name server shared by all clients.

Recursive name resolution:

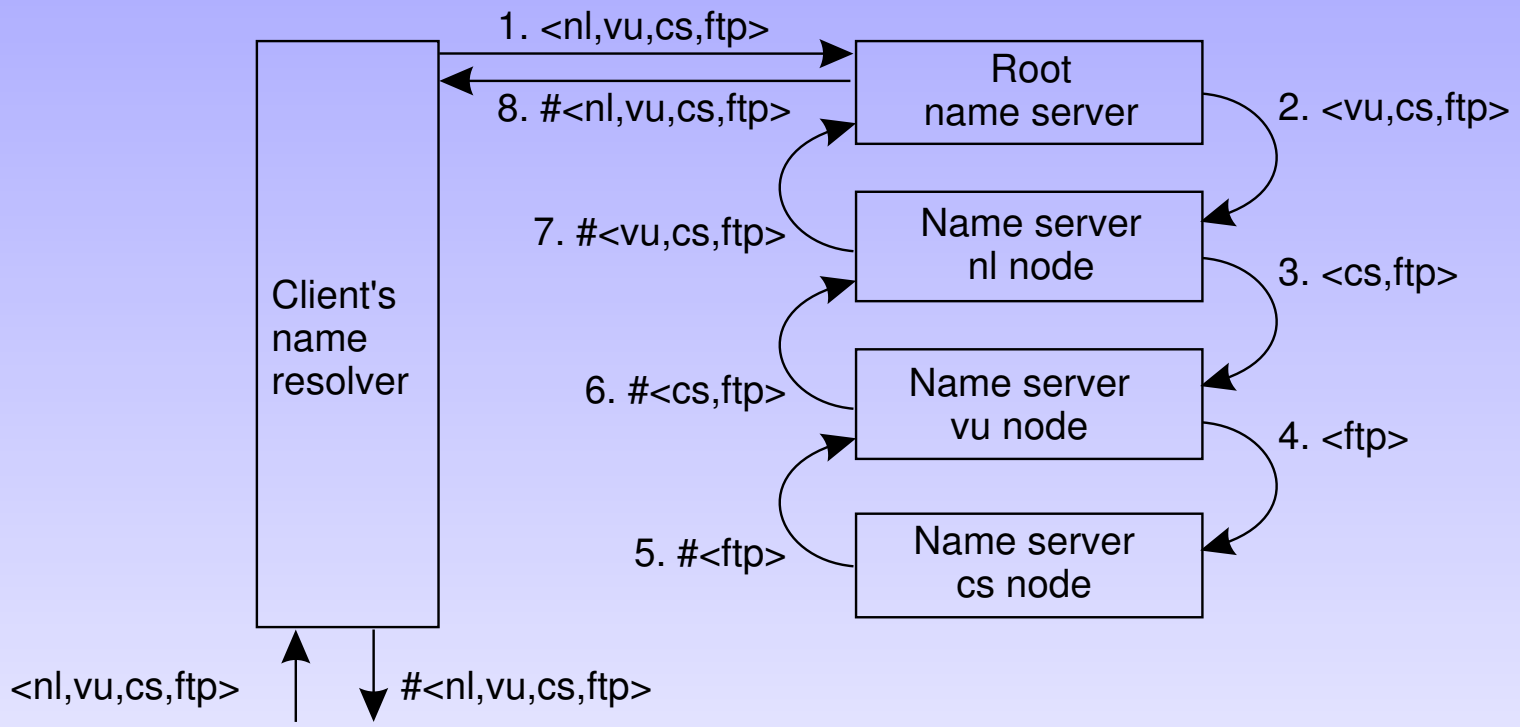
- √ each name server passes the (partial) request to the next name server it finds
- √ high performance demands
- √ simple, effective caching and reduced communication costs

Implementation of Name Resolution (2)



Iterative name resolution.

Implementation of Name Resolution (3)



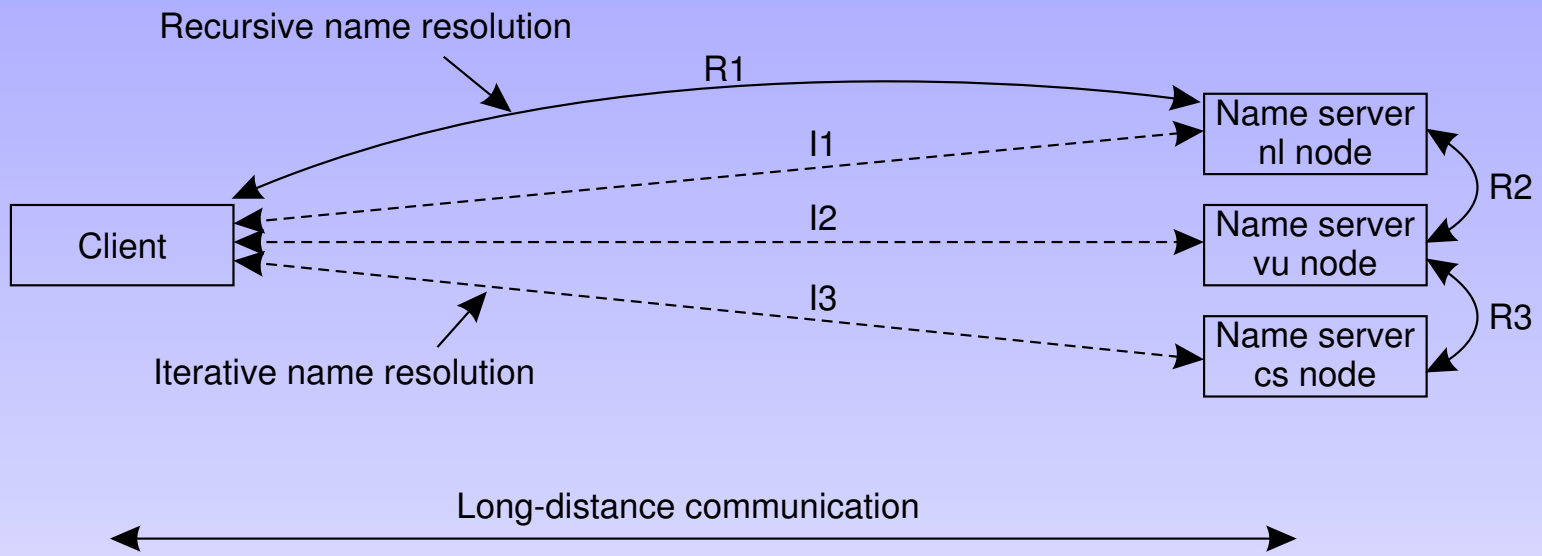
Recursive name resolution.

Implementation of Name Resolution (4)

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 resolution of the name <nl, vu, cs, ftp> with caching.

Implementation of Name Resolution (5)



Comparison of communication costs for recursive and iterative name resolution.

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 in the DNS name space nodes.

DNS Implementation (1)

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

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

DNS Implementation (2)

Name	Record type	Record value
cs.vu.nl	NS	solo.cs.vu.nl
solo.cs.vu.nl	A	130.37.24.1

Part of the description for the vu.nl domain containing the cs.vu.nl domain.

X.500

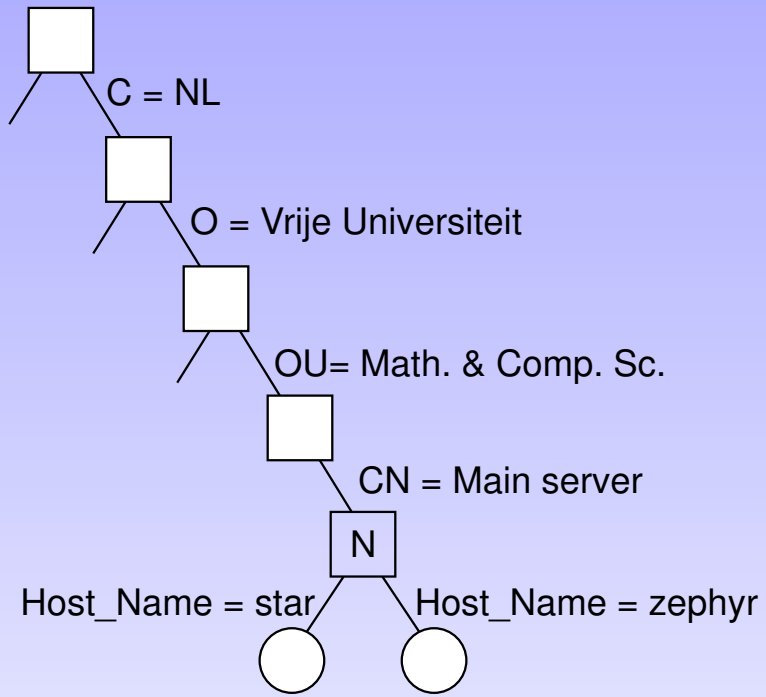
- **directory service** – special kind of naming service in which a client can look for an entity based on a description of properties instead of a full name,
- OSI X.500 directory service,
- **Directory Information Base (DIB)** – the collection of all directory entries in an X.500 directory service,
- each record in DIB is uniquely named, using a sequence of naming attributes,
- each attribute is a **Relative Distinguished Name (RDN)**,
- **Directory Information Tree (DIT)** – a hierarchy of the collection of directory entries,
- DIT is a graph with directory entries as nodes,
- each node in DIT may act as a directory in the traditional sense.

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, 192.31.231.42, 192.31.231.66
FTP_Server	—	130.37.24.11
WWW_Server	—	130.37.24.11

A simple directory entry using X.500 naming conventions.

The X.500 Name Space (2)



Part of the directory information tree.

X.500 Implementation

- DIT is usually partitioned and distributed across several servers, known as **Directory Service Agents (DSA)**,
- each DSA implements advanced search operations,
- clients are represented by **Directory User Agents (DUA)**,
- example: retrieving a list of all main servers at VU:
  - answer=search(&(C=NL)(O=Vrije Universiteit)(OU=*)(CN=Main server))
- searching is expensive,
- access to the service using **Directory Access Protocol (DAP)**.

GNS Names and X.500

(/... or /.:) + (X.500 or DNS name) + (local name)

- √ /.../ENG.IBM.COM.US/nancy/letters/to/lucy
- √ /.../Country=US/OrgType=COM/OrgName=IBM/Dept=ENG/nancy/letters/to/lucy
- √ /.:/nancy/letters/to/lucy

LDAP

- √ **Lightweight Directory Access Protocol (LDAP)**,
- √ application-level protocol on top of the TCP/IP stack,
- √ LDAP servers were created as specialized gateways to X.500 servers,
- √ parameters for lookups and updates are passed as strings,

- √ LDAP offers:
  - ⋆ SSL-based security model,
  - ⋆ defined API,
  - ⋆ universal data exchange format, LDIF,

- √ http://www.openldap.org

Naming vs. Locating Entities (1)

Three types of names:

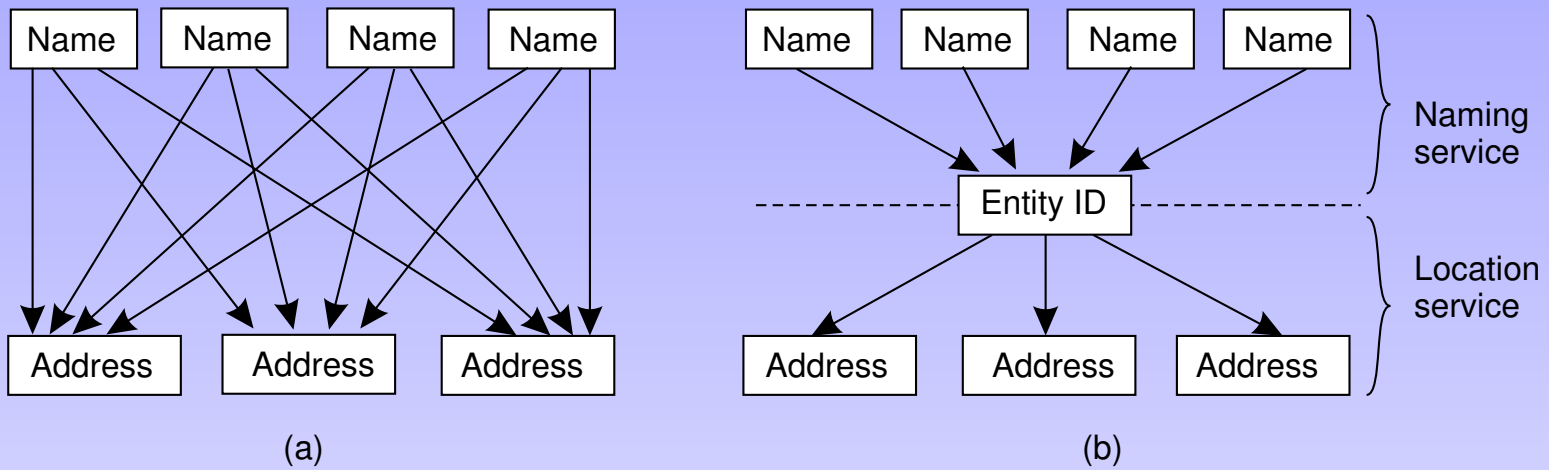
- √ human-friendly names (a.k.a. aliases)
- √ identifiers,
- √ addresses.

What to do if ftp.cs.vu.nl is moved to ftp.cs.unisa.edu.au?

- √ the new *address* can be stored in the cs.vu.nl DNS, **but**
- √ if it moves again, cs.vu.nl will have to be updated!
- √ the new *name* can be stored in the cs.vu.nl DNS, **but**
- √ lookup operation becomes less efficient every time the machine is moved...

Solution – separate naming and locating entities using identifiers!

Naming vs. Locating Entities (2)



a. direct, single level mapping between names and addresses,

b. two-level mapping using identifiers.

√ locating an entity is a separate service – *location service*.

Replication System in EDG Grids

- The replication system is responsible for providing efficient storage and access to large, immutable files in the grid.
- Every file has three types of names:

  GUID *Global Unique IDentifier* – automatically generated random identifier, 128 bits, e.g.:
  `guid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6`

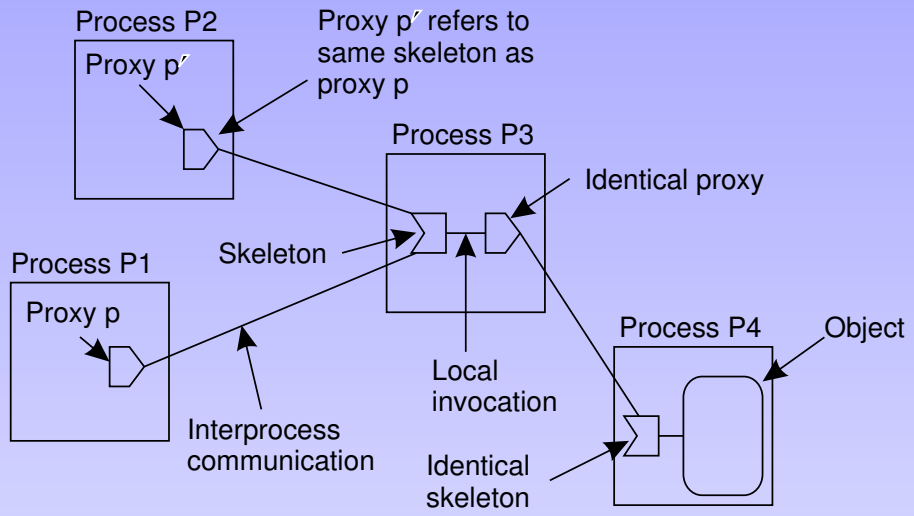
  LFN *Logical File Name*, or simply alias, a human-created and human-friendly name, e.g.:
  `lfn:cms/20030203/run2/track1`

  PFN/SFN/SURL *Physical/Storage File Name, Site URL*, address of the file, e.g.:
  `srm://pcrd24.cern.ch/flatfiles/cms/output10_1`

Location Service Implementation

1. simple solutions:
   - √ broadcasting and multicasting,
     - ⋆ ARP, used to find data-link layer address given an IP address
   - √ forwarding pointers,
     - ⋆ when an entity moves, it leaves behind a reference to its new location
2. home-based approaches,
3. hierarchical approaches.

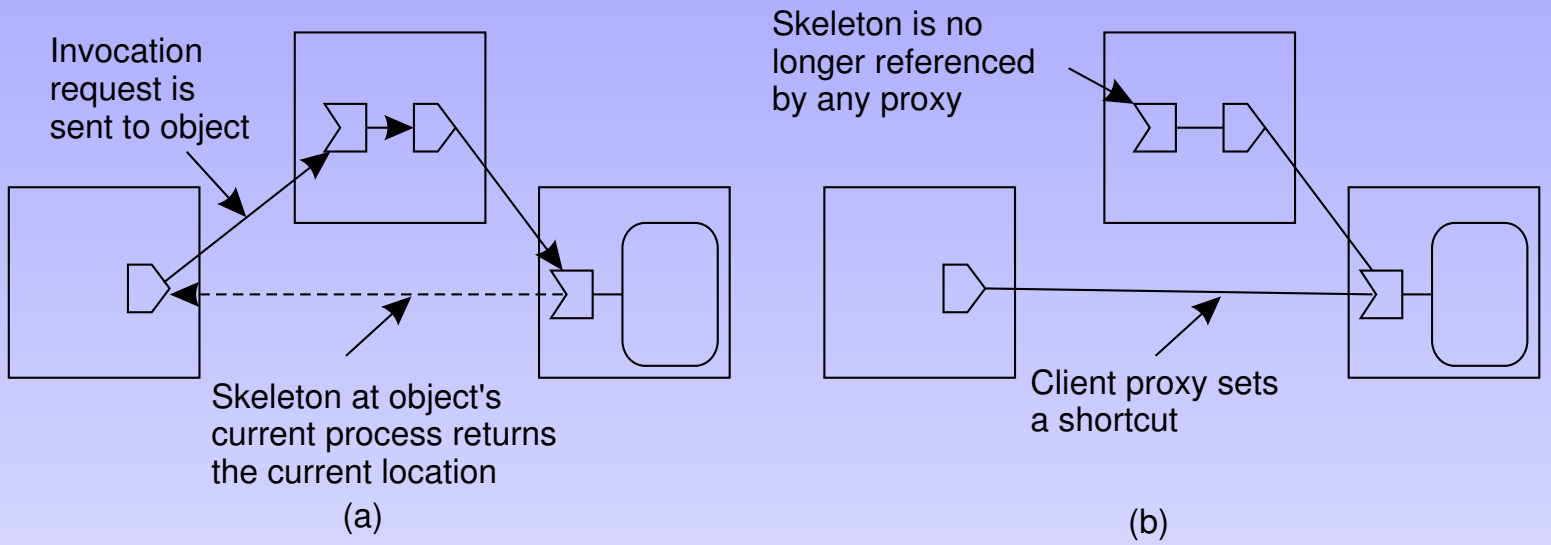
Forwarding Pointers (1)



Forwarding pointers using (proxy, skeleton) pairs:

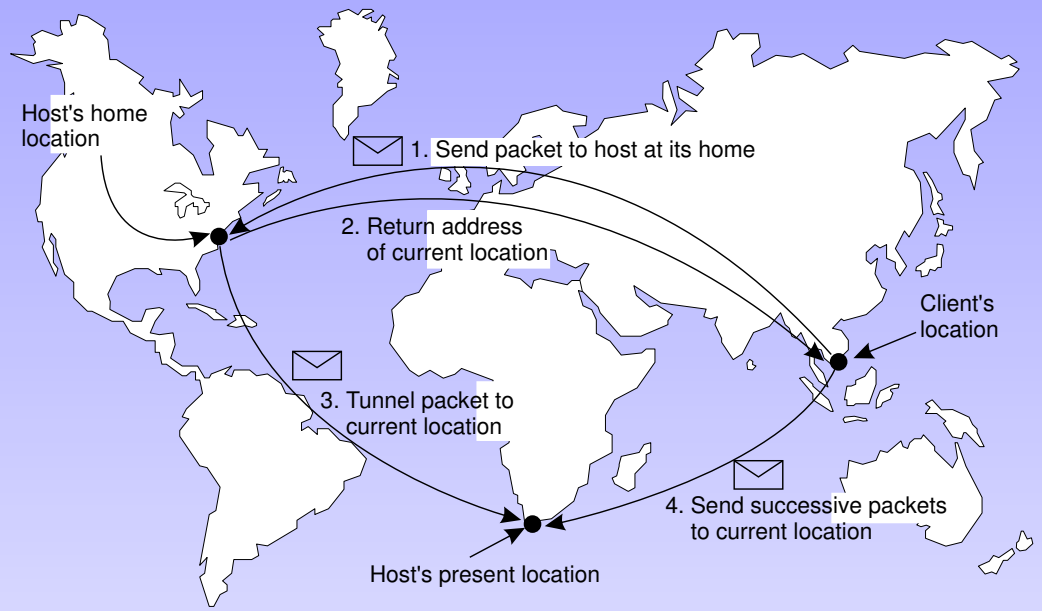
- √ **skeletons** act as entry items for remote references, **proxies** as exit items,
- √ moving an entity from A to B installs a proxy on A and a skeleton on B.

Forwarding pointers (2)



Redirection of forwarding pointers – storing a shortcut in a proxy.
Problem of dead skeletons – hard to solve for rarely referenced entities.

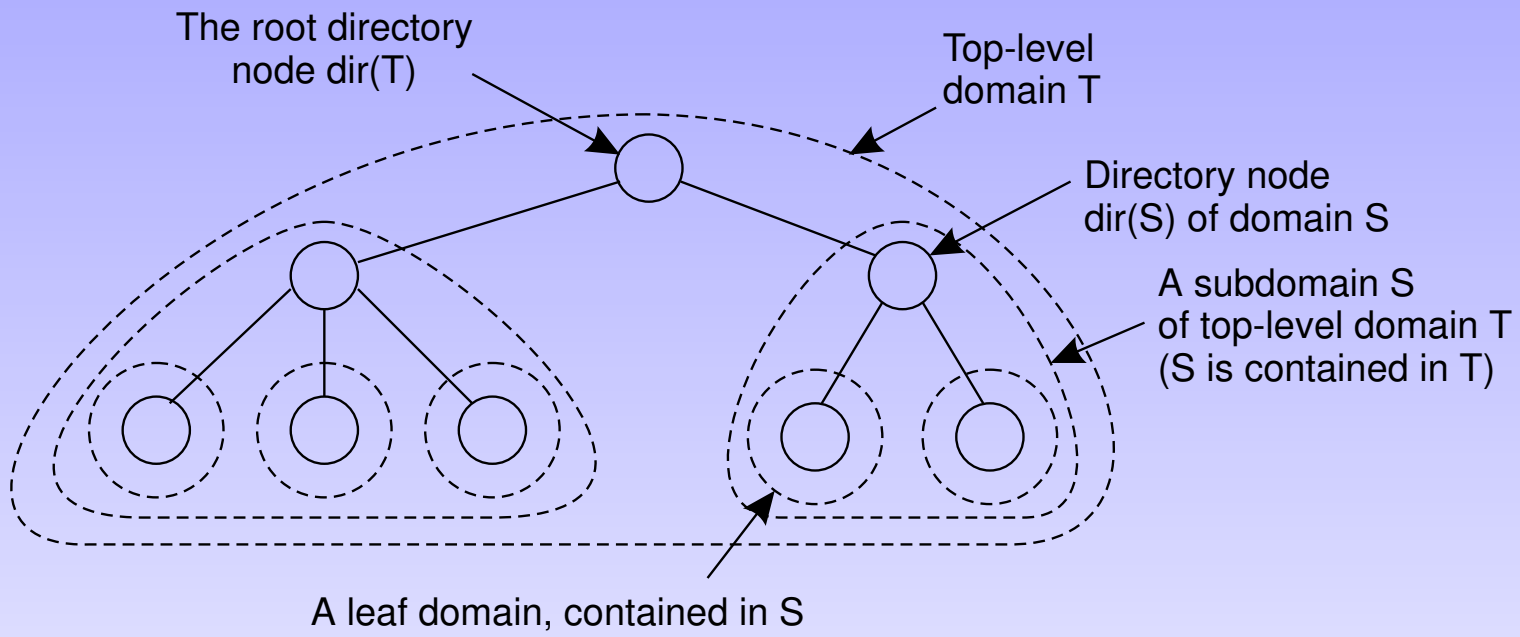
Home-Based Approaches



The principle of Mobile IP.

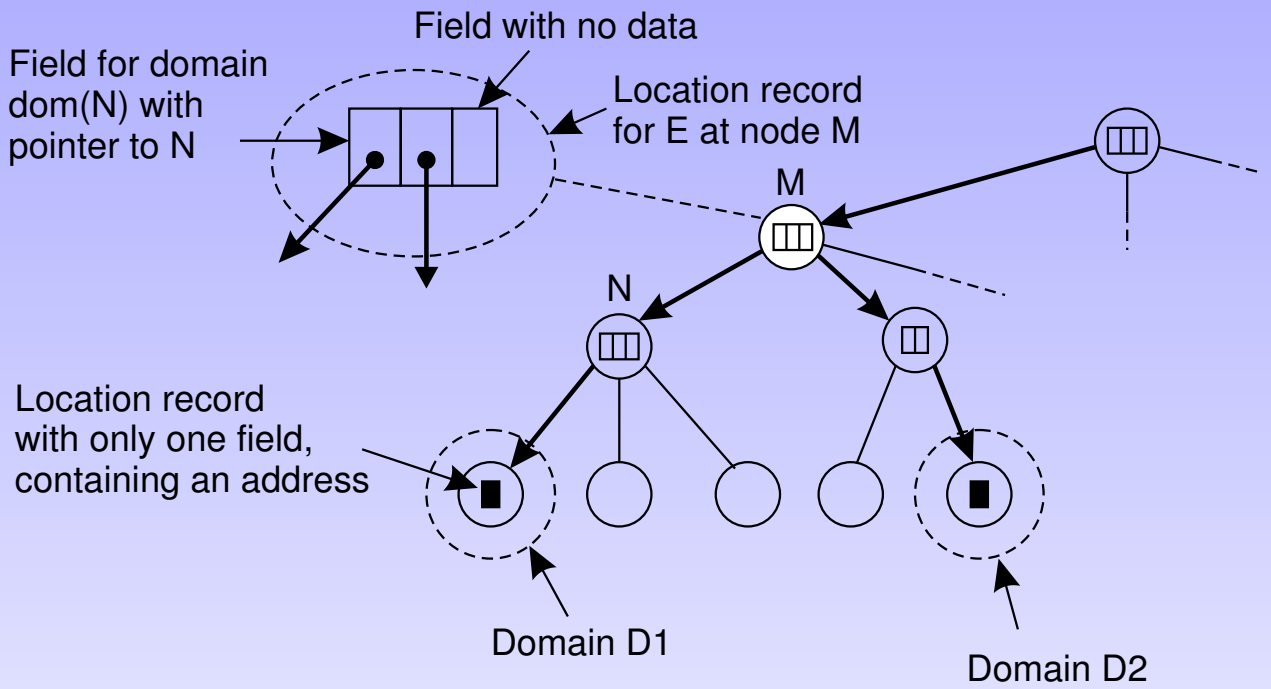
- √ predefined home location, home agent in home LAN, fixed IP address,
- √ mobile host requesting a temporary care-of-address in another network is registered afterwards at the home agent.

Hierarchical Approaches (1)



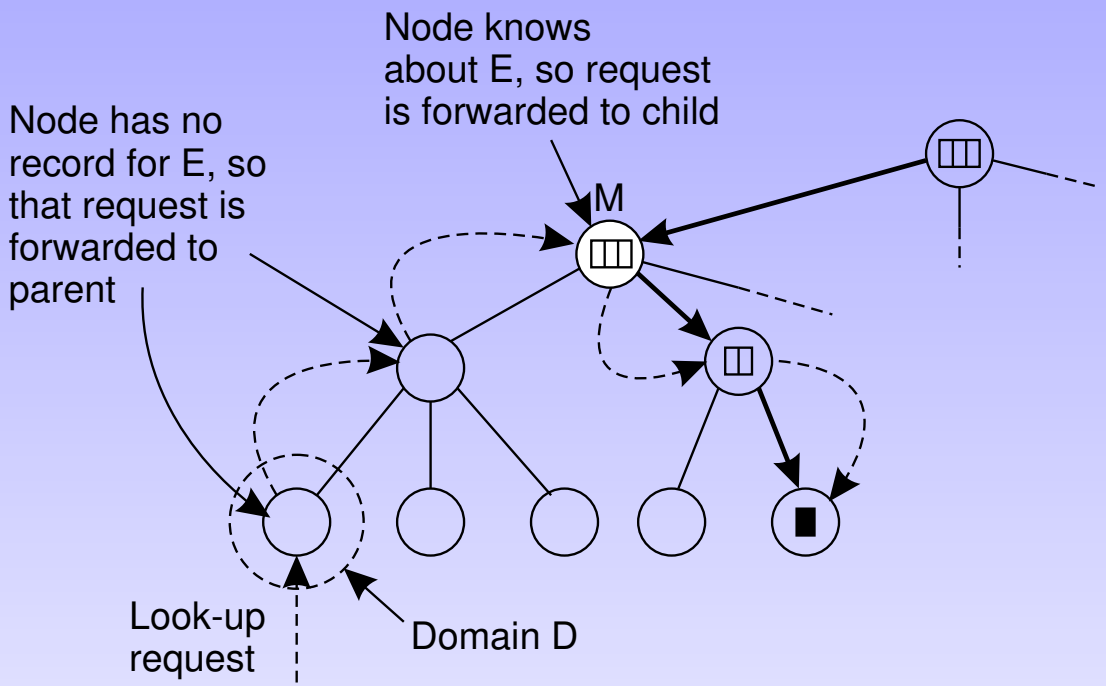
Hierarchical organization of a location service into domains, each having an associated directory node.

Hierarchical Approaches (2)



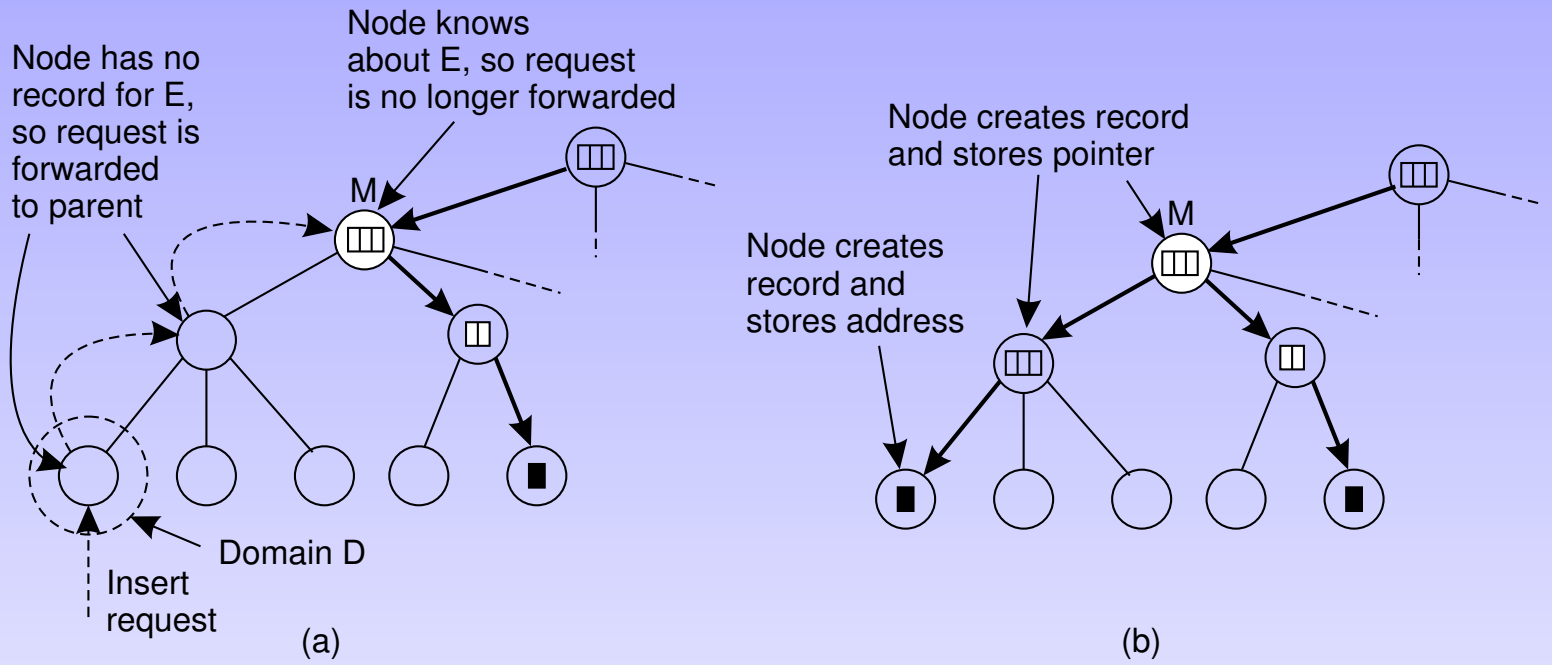
An example of storing the address of an entity having two addresses in different leaf domains.

Hierarchical Approaches (3)



Looking up a location in a hierarchical location service.

Hierarchical Approaches (4)



1. Insert requests are forwarded towards the root to the first node that knows about entity E.
2. A chain of forwarding pointers to the leaf node is created.