Distributed Systems
Naming

[2] **Names - Introduction**

Usage of names:

- to share resources,
- to uniquely identify entities,
- to refer to locations.

A name can be resolved to the entity it refers to.

Name – a string of bits or characters that is used to refer to an entity.

- typical entities: hosts, printers, disks, files, processes, users, mailboxes, newsgroups, Web pages, messages, network connections,
- **an access point** – special entity to access another entity,
- **an address** – the name of an access point,
- the address of an entity access point simply called an address of the entity.

[3] **Names, Identifiers, Addresses**

- if an entity offers more than one access point not clear which address to use as a reference,
- **location independent** names,

An identifier – a name that is used to uniquely identify an entity.
An identifier has the following properties:

1. An identifier refers to at most one entity.
2. Each entity is referred to by at most one identifier.
3. An identifier always refers to the same entity (it is never reused).

Remarks:

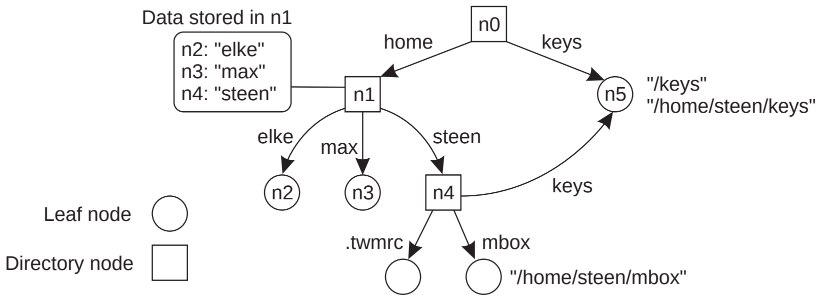
- by using identifiers much easier unambiguously reference to entities,

- **human-friendly names** in contrast to addresses and identifiers.

[4] Name Spaces (1)

- names in distributed systems organized into **name spaces**,
- name space may be represented as a labeled, directed graph,
  - a leaf node represents a named entity,
  - a directory node stores a table with outgoing edges representation as pairs (edge label, node identifier) – **directory table**,
- **root node** of the naming graph,
- **path name**: N:<label-1, label-2, ..., label-n>,
- **absolute path name** (starts with root) vs. **relative path name**,
- **global name** – denotes the same entity in the whole system,
- **local name** – with interpretation depending on where the name is being used.

[5] Name Spaces (2)



A general naming graph with a single root node

- n5 can be referred to by /home/steen/keys as well as /keys,
- the idea of directed acyclic graph,

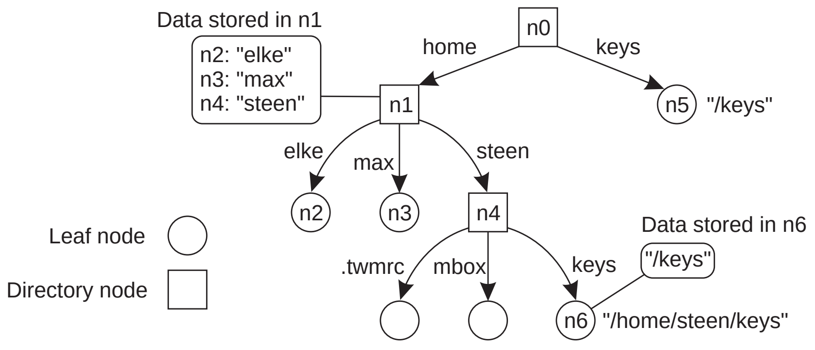
- in Plan9 all resources (processes, hosts, I/O devices, network interfaces) named as files – single naming graph for all resources in a distributed system,

[6] **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 name resolution process continues,
- **closure mechanism** – knowing **how** and **where** to start name resolution,
  - Unix file system: the inode of the root directory is the first inode in the logical disk,
  - ”000312345654” not recognizable as string, but recognizable as a phone number,
- **alias** another name for the same entity,
- **hard links** versus **symbolic links**.

[7] **Linking and Mounting (1)**



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

[8] **Linking and Mounting (2)**

- mount point – the directory node storing the node identifier,

– mounting point – the directory node in the foreign name space.

To mount a foreign name space in a distributed system the following information is required:

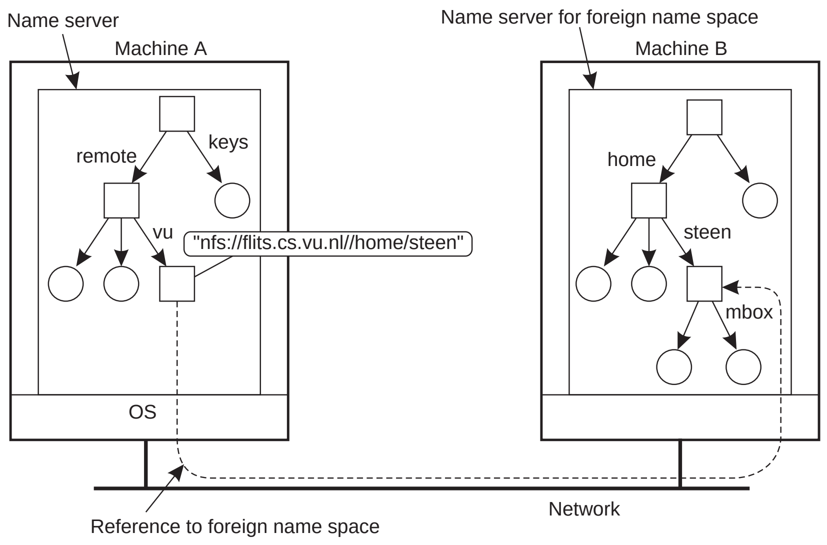
1. The name of an access protocol.
2. The name of the server.
3. The name of the mounting point in the foreign name space.

Remarks:

– each of these names has to be resolved,

– NFS as an example.

[9] **Linking and Mounting (3)**

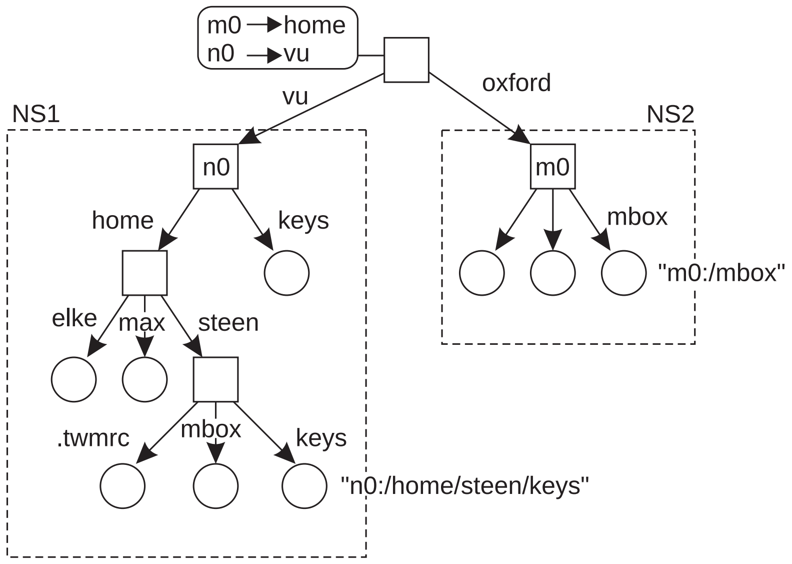

Name server
Machine A
remote
keys
vu
"nfs://flits.cs.vu.nl//home/steen"
OS
Name server for foreign name space
Machine B
home
steen
mbox
Network
Reference to foreign name space

Mounting remote name spaces through a specific process protocol.

[10] **Linking and Mounting (4)**

- in DECs **GNS** (**Global Name Service**) new root is added making all existing root nodes its children,
- names in GNS always (implicitly) include the identifier of the node from where resolution should normally start,
- /home/steen/keys in NS1 expanded to n0:/home/steen/keys,
- hidden expansion,
- assumed that node identifiers universally unique,
- therefore: all nodes have different identifiers.

[11] **Linking and Mounting (5)**



Organization of the DEC Global Name Service.

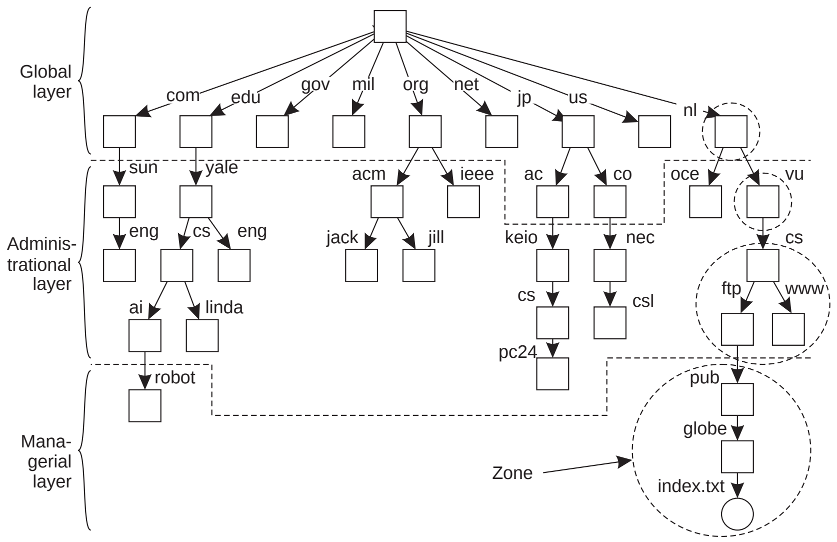
[12] **Name Space Distribution (1)**

- name space distribution,
- large scale name spaces partitioned into logical layers

    - global layer,
    - administrational layer,
    - managerial layer.

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

[13] **Name Space Distribution (2)**



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

[14] **Name Space Distribution (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 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.

[15] **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>

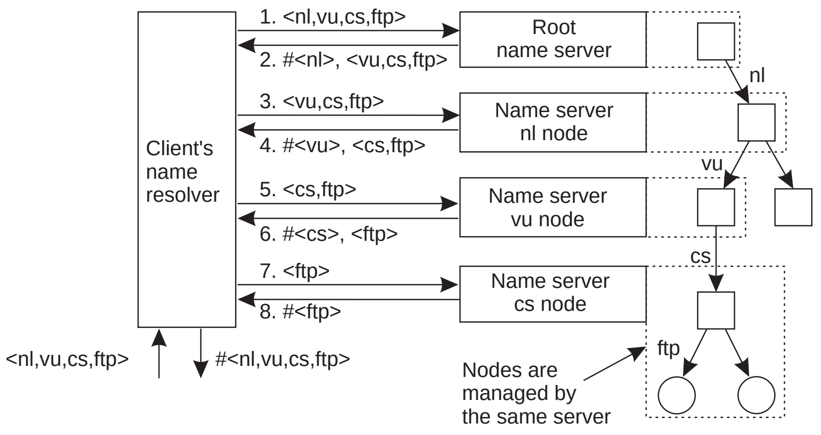
Iterative name resolution:

- a name resolver hands over the complete name to the root name server, but root resolves only nl and returns address of the associated name server
- caching restricted to the clients name resolver as a compromise intermediate name server shared by all clients.

Recursive name resolution:

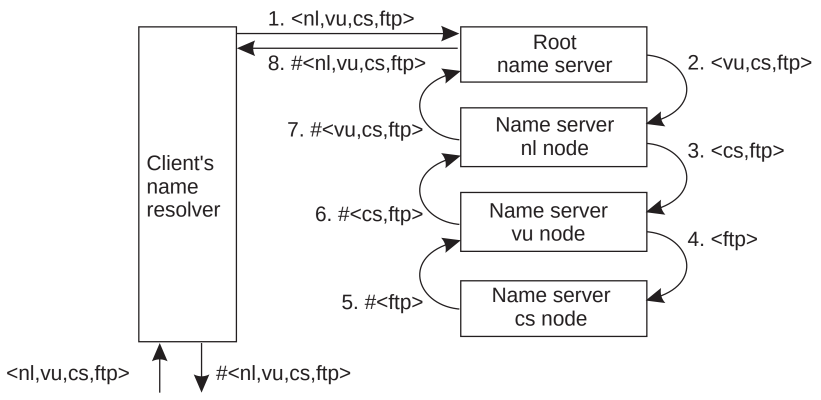
- a name server passes the result to the next name server it finds,
- drawback: puts a higher performance demand,
- caching results more effective comparing to iterative name resolution,
- communication costs may be reduced.

[16] **Implementation of Name Resolution (2)**



The principle of iterative name resolution

[17] **Implementation of Name Resolution (3)**



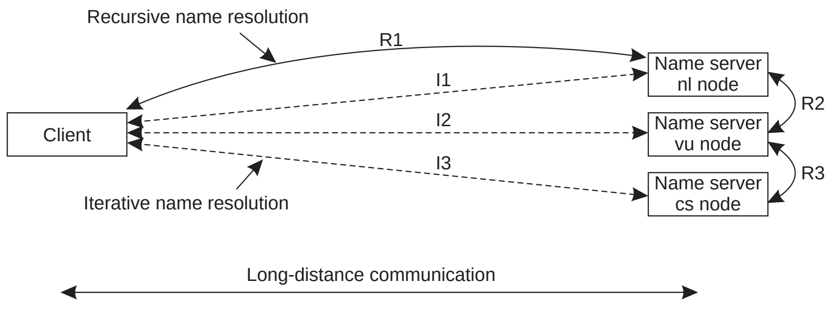
The principle of recursive name resolution.

[18] **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 name resolution of <nl, vu, cs, ftp>. Name servers cache intermediate results for subsequent lookups.

[19] **Implementation of Name Resolution (5)**


Recursive name resolution
R1
I1
I2
I3
Client
Name server nl node
Name server vu node
Name server cs node
R2
R3
Iterative name resolution
Long-distance communication

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

[20] **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.

[21] **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.

[22] **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 which contains the cs.vu.nl domain.

[23] **X.500**

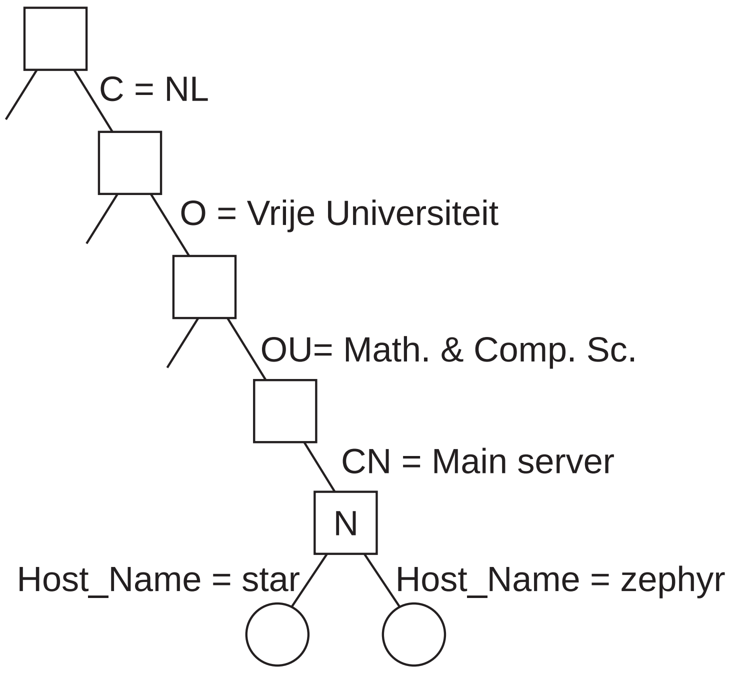
- **directory service** – special kind of naming service in which a client can look for an entity based on a description of properties instead of a full name.
- OSI X.500 directory service,
- **Directory Information Base (DIB)** – the collection of all directory entries in an X.500 directory service,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

[24] **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 example of a X.500 directory entry using X.500 naming conventions.

[25] **The X.500 Name Space (2)**



Part of the directory information tree.

[26] **X.500 Implementation**

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

[27] **GNS Names**

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

- (/... 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

[28] **LDAP**

- **Lightweight Directory Access Protocol (LDAP)**,
- an application-level protocol implemented on top of TCP,
- LDAP servers as specialized gateways to X.500 servers,
- parameters of lookup and update simply passed as strings,

- LDAP contains:
    - defined security model based on SSL,
    - defined API,
    - universal data exchange format, LDIF,

- http://www.openldap.org

[29] **Naming versus Locating Entities (1)**

Three types of names distinguished:

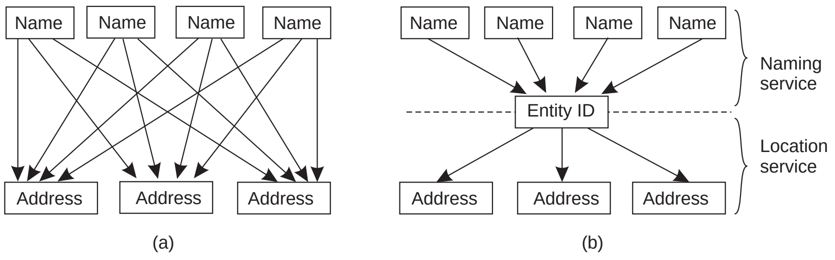
- human-friendly names,
- identifiers,
- addresses.

What happens if a machine ftp.cs.vu.nl is to move to ftp.cs.unisa.edu.au?

- recording the address of the new machine in the DNS for cs.vu.nl **but**,
- whenever it moves again, its entry in DNS in cs.vu.nl has to be updated as well,
- recording the name of the new machine in the DNS for cs.vu.nl **but**,
- lookup operation becomes less efficient.

Better solution: naming from locating entities separation by introduction of identifiers.

[30] **Naming versus Locating Entities (2)**



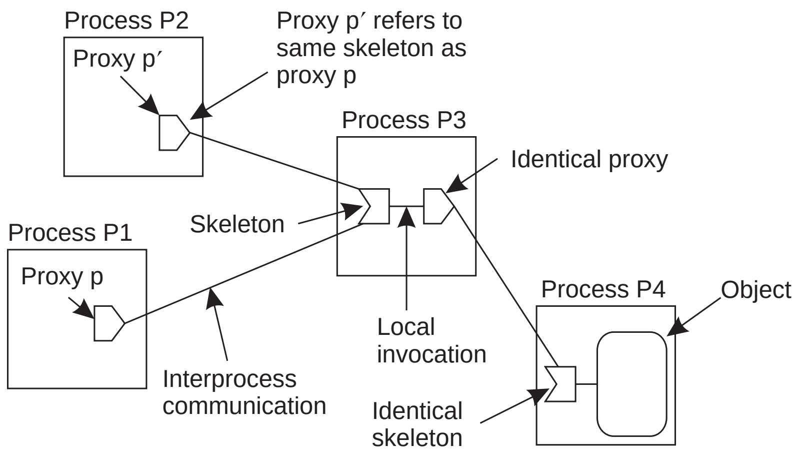
a. direct, single level mapping between names and addresses,
b. two-level mapping using identifiers.

- locating an entity is handled by means of a separate **location service**.

[31] **Location service implementations**

1. simple solutions
   - broadcasting and multicasting,
     - ARP to find the data-link address given only 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.

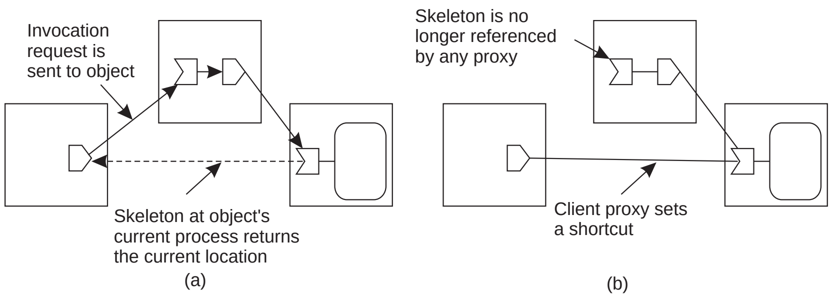
[32] **Forwarding Pointers (1)**



The principle of forwarding pointers using (proxy, skeleton) pairs.

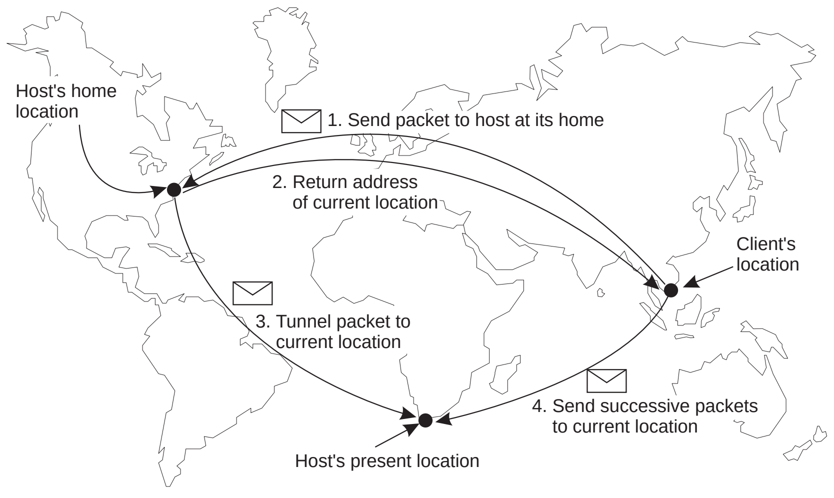
- **skeletons** acts as entry items for remote references, **proxies** as exit items,
- whenever move from A to B, proxy installed on A and referring skeleton on B.

[33] **Forwarding Pointers (2)**


Invocation request is sent to object
Skeleton at object's current process returns the current location
(a)
Skeleton is no longer referenced by any proxy
Client proxy sets a shortcut
(b)

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

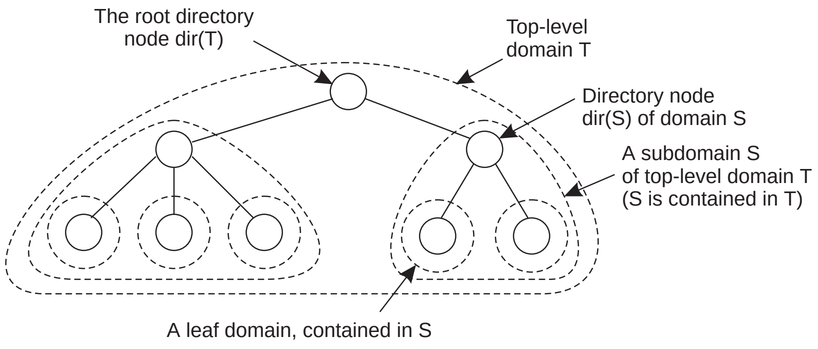
[34] **Home-Based Approaches**


Host's home location
1. Send packet to host at its home
2. Return address of current location
Client's location
3. Tunnel packet to current location
4. Send successive packets to current location
Host's present location

The principle of Mobile IP.

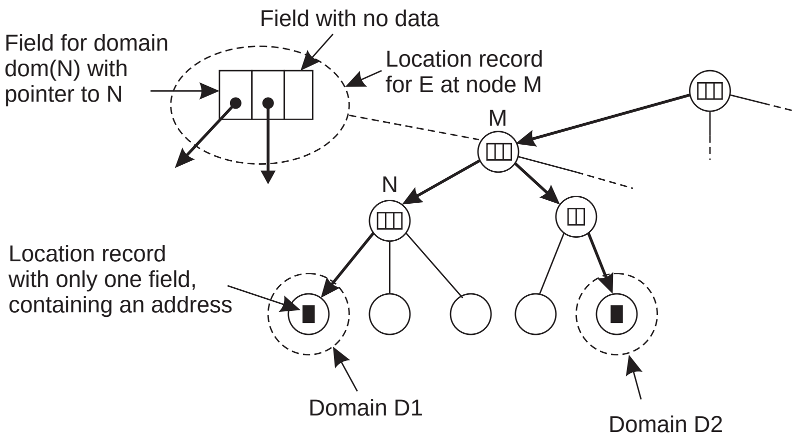
- home location, home agent in home LAN, fixed IP address,
- whenever the mobile host in another network requests a temporary care-of-address, registered afterwards at the home agent.

[35] **Hierarchical Approaches (1)**


The root directory node dir(T)
Top-level domain T
Directory node dir(S) of domain S
A subdomain S of top-level domain T (S is contained in T)
A leaf domain, contained in S

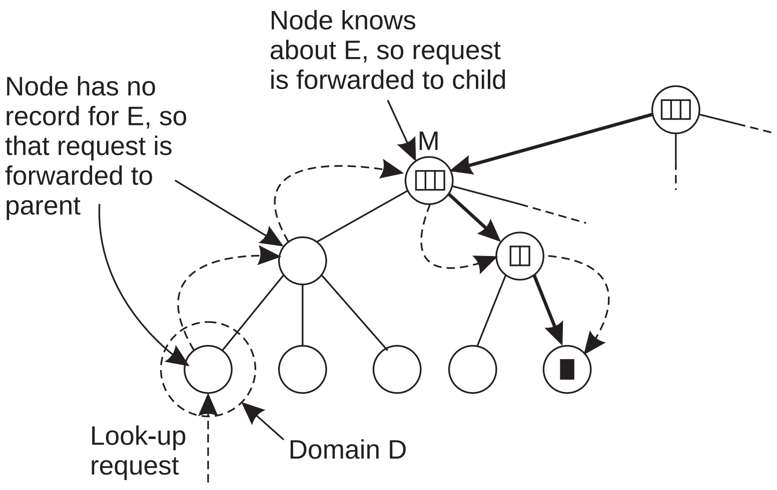
Hierarchical organization of a location service into domains, each having an associated directory node.

[36] **Hierarchical Approaches (2)**


Field with no data
Field for domain dom(N) with pointer to N
Location record for E at node M
M
N
Location record with only one field, containing an address
Domain D1
Domain D2

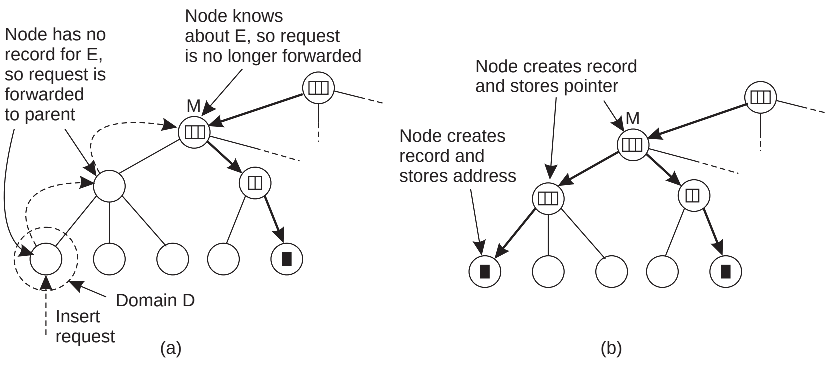
An example of storing information of an entity having two addresses in different leaf domains.

[37] **Hierarchical Approaches (3)**


Node knows about E, so request is forwarded to child
Node has no record for E, so that request is forwarded to parent
M
Look-up request
Domain D

Looking up a location in a hierarchically organized location service.

[38] **Hierarchical Approaches (4)**


Node has no record for E, so request is forwarded to parent
Node knows about E, so request is no longer forwarded
M
Domain D
Insert request
(a)
Node creates record and stores pointer
Node creates record and stores address
M
(b)

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