Distributed Systems

Naming and Location

Dynamic and Distributed
Information Systems

Today's Agenda

- Introduction to Naming
- Hierarchical Naming
- Flat Naming
- Attribute-based Naming
- Location Service

Why do we need Naming?

Accessing entities

- We need to operate on entities, for instance:
  - Hosts
  - Printers
  - Web pages
  - Files (e.g., on an FTP server)
  - Web Services
  - Users
- To operate on an entity, we need to access it
  - Each entity should have an access point, usually called address
- Various types of addresses exist, for instance:
  - IP address + port
  - Printer name
  - URL
  - User ID

And how about names?

- If we just need an address to access an entity, what do we need names for?
- Various reasons:
  - An entity may change its address due to relocation (e.g., new IP address)
  - An entity may change its address due to reorganization (e.g., a company's web server may be moved to a new host)
  - An entity may have more than one access points (replication)
- Naming provides an abstraction useful for
  - Providing location independence
  - Allowing the relocation of entities
  - Allowing a single reference to a set of alternative access points
  - In some cases, for offering human-friendly names

Naming Types

- There are three general categories of Naming types
  - Hierarchical Naming
  - Flat Naming
  - Attribute-based Naming

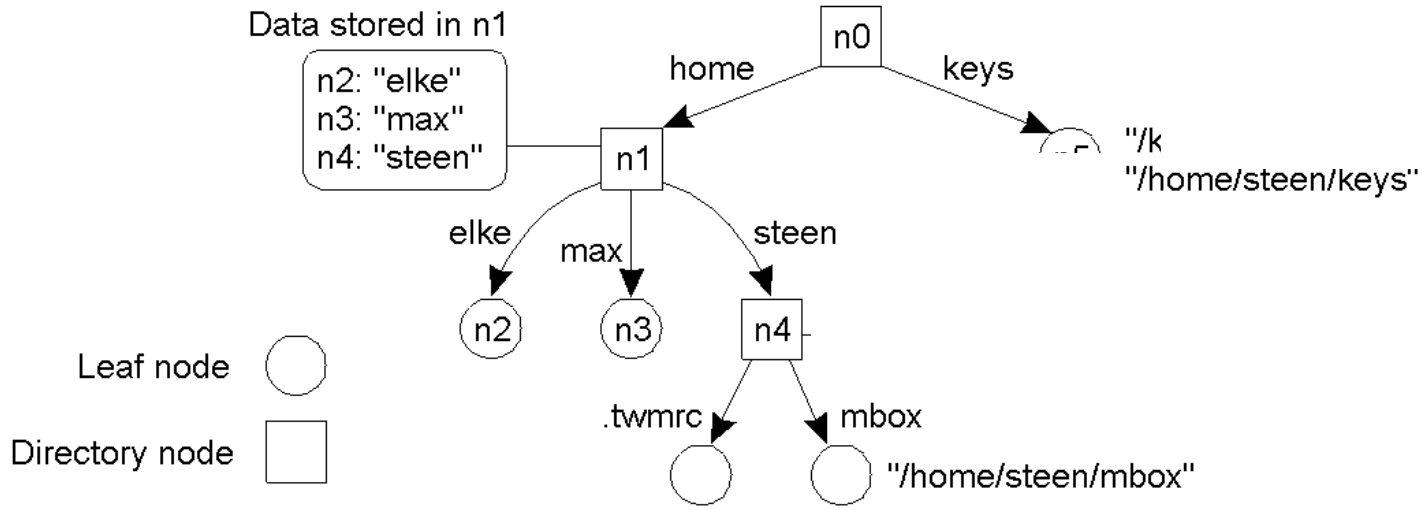
Hierarchical Naming

Dynamic and Distributed
Information Systems

Hierarchical Naming

- Based on the concept of Name Spaces
  - A graph of names
- Each name is a path in the naming graph
  - If starting from the root, we call it an absolute name
  - Otherwise, we call it a relative name
- Examples (of absolute names):
  - File systems: /home/spyros/Teaching/DS08/exam.pdf
  - DNS: www.ifi.uzh.ch
  - URLS: http://www.ifi.uzh.ch/ddis/teaching/distsystems070/

Name Spaces

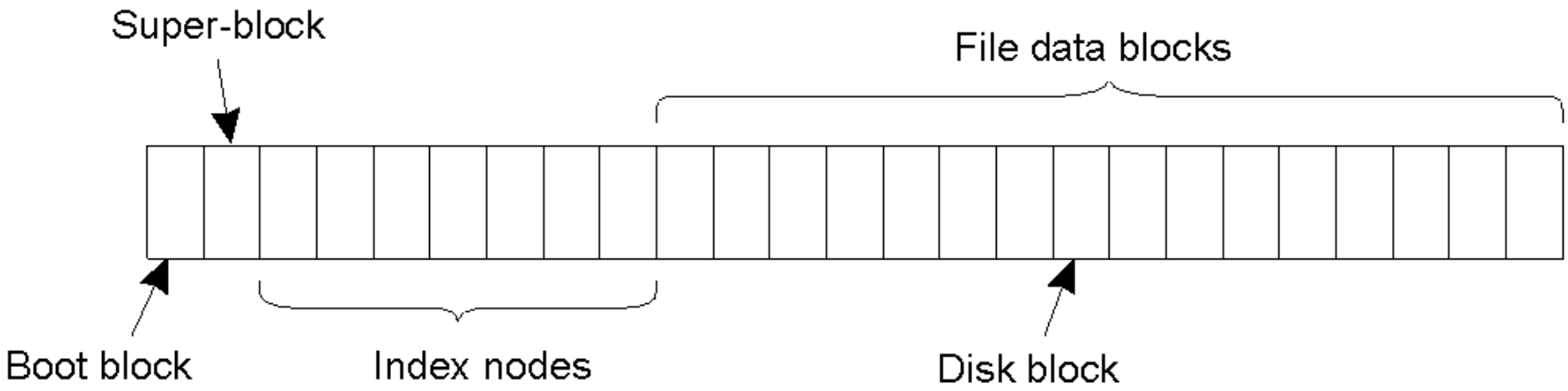


A general naming graph with a single root node.

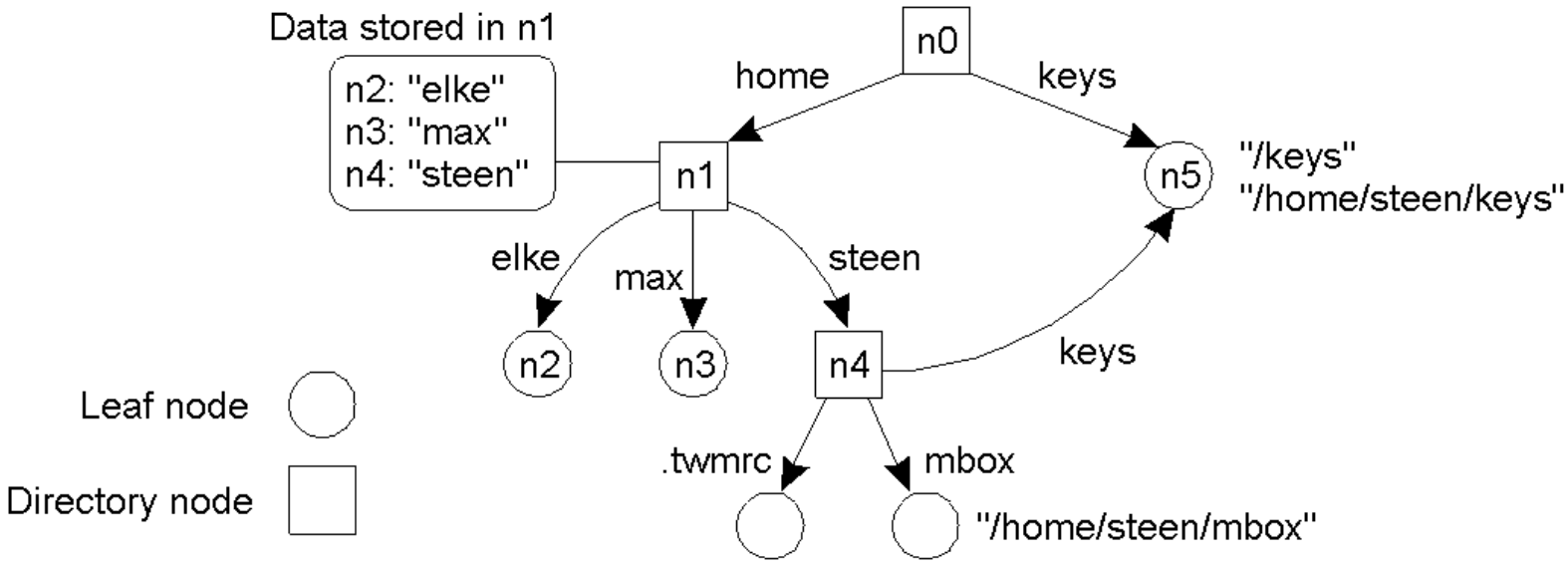
- A leaf node represents a named entity
  - no outgoing links
- A directory node stores links to other nodes (leafs or other directory nodes)

Unix File System outline

The general organization of the UNIX file system implementation on a logical disk of contiguous disk blocks.

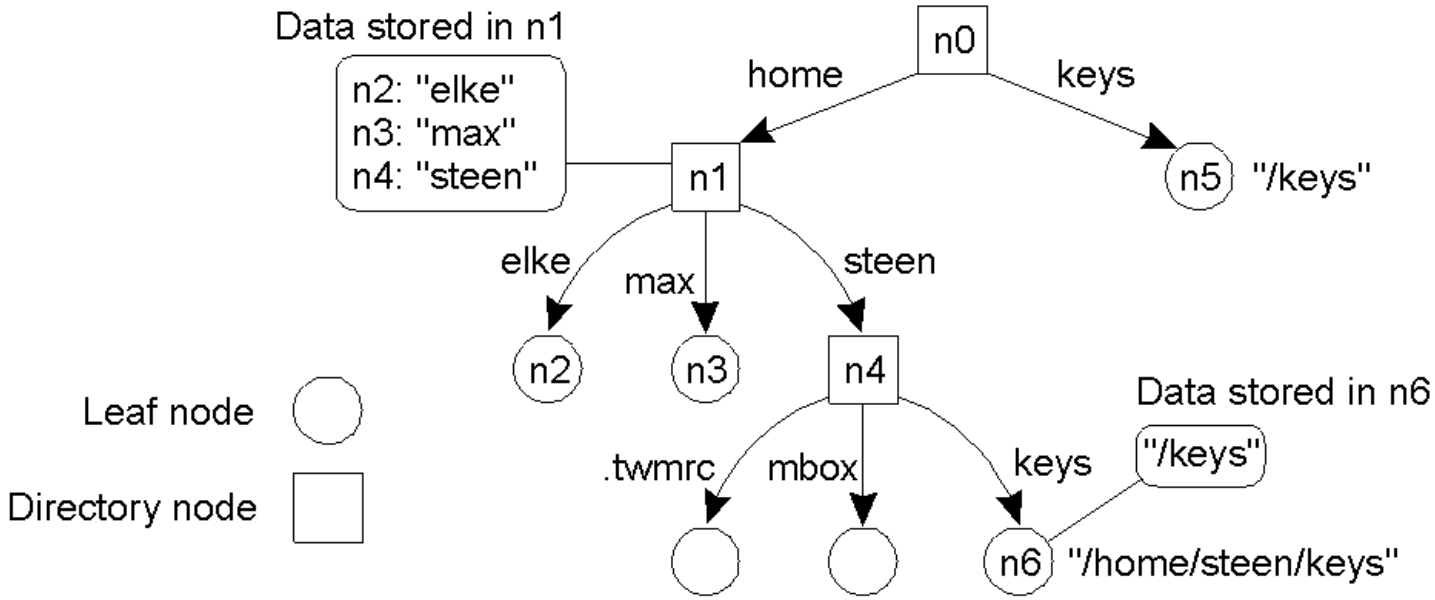


Multiple Names (1): Hard Links



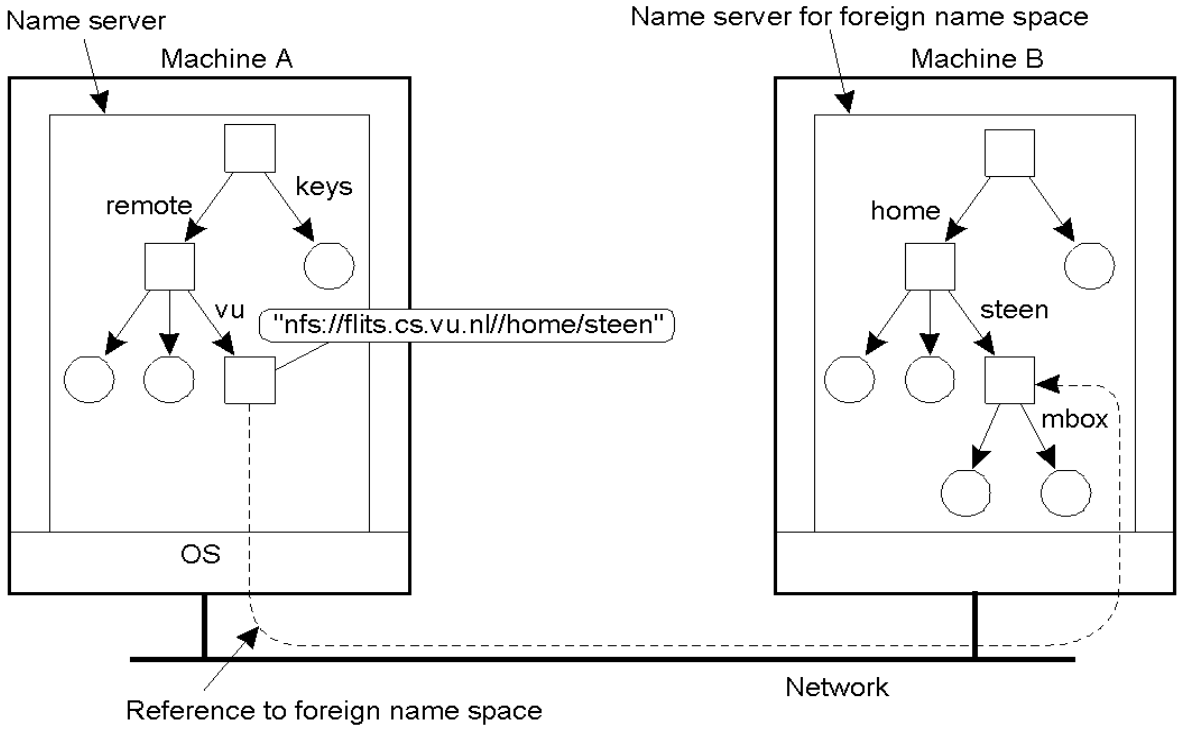
- An entity may have multiple names within a namespace
  - Multiple paths that lead to the same leaf node.
- In Unix, these are called hard links
  - Created by: **`ln /keys /home/steen/keys`**
  - Similar syntax to copy command (**`cp`**)

Multiple Names (2): Symbolic Links



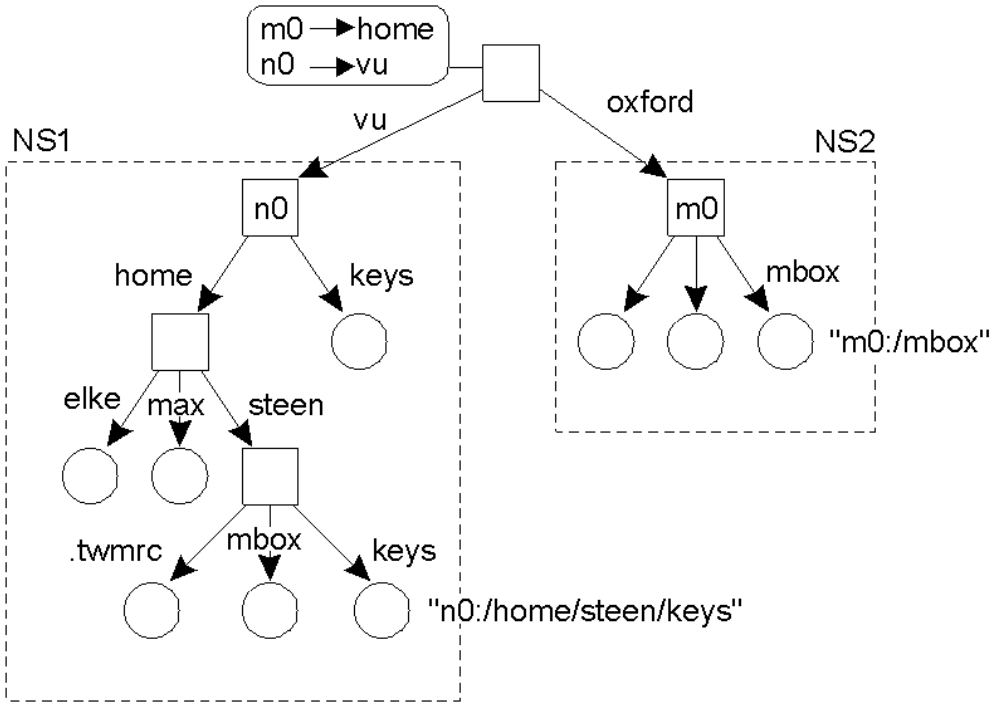
- A special node contains the absolute (or relative) name of another node
- In Unix, these are called soft links
  - Created by: **`ln -s /keys /home/steen/keys`**
  - Same syntax as for hard links, but using the **`-s`** switch

Mounting: Remote Linking



- A symbolic link may be referring to a remote Name Space, through a specific process protocol
- This is called mounting
  - In Unix use the command:
    `sudo mount nfs://flits.cs.vu.nl//home/steen /remote/vu`

Merging Name Spaces



- Adding a new root, and mounting two or more namespaces below it
  - Used in the DEC Global Name Service

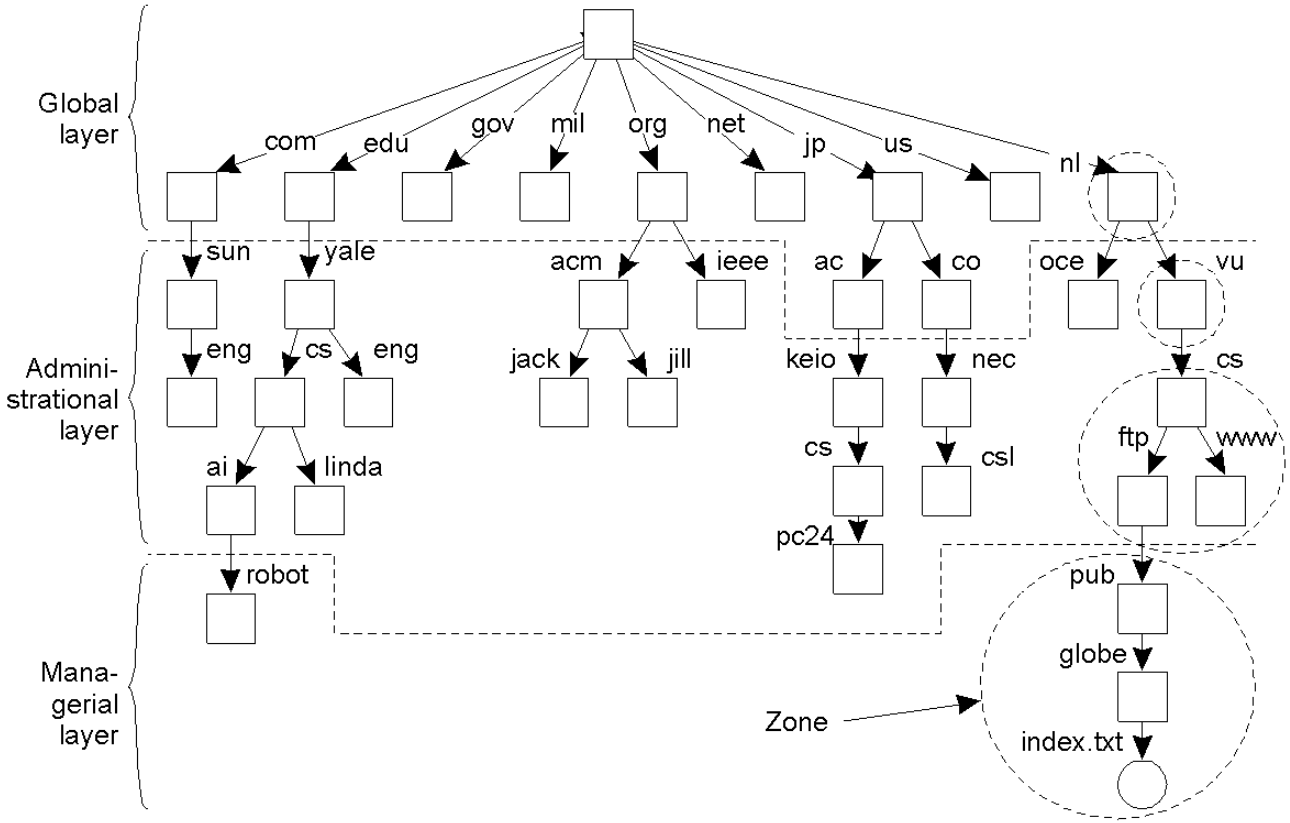
- Problem: Absolute names of all namespaces are changed
- Solution: At root node cache the original top-level names
  - E.g., the root remembers that **`home`**, **`keys`** map to **`/vu`**, and **`mbox`** maps to **`/oxford`**
  - Problem: Only filesystems with different top-level directories can be merged

(Much) Larger Scale

- Ok, so far, for small scale naming
  - e.g., a company-wide file system
- But what happens when the scale grows to …global?!
  - e.g., DNS

Name Space Distribution

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



Domain Name Systems (DNS)

- How can we map structured hostnames to IP addresses?
  - Old days: HOSTS.TXT file FTPed among hosts…
- Now we have DNS, a distributed directory service
  - Hierarchical name space
  - Each level separated by '.'
    - Analogous to '/' in file systems ('\' in Windows)
  - One global root
    - Replicated across 13 root servers
    - There have been Denial-of-Service (DoS) attack on these root servers, but none of them successful
    - Because of caching, queries to root servers are relatively rare
- DNS has proven to be the most longstanding and stable global directory service, if not the only one

DNS layers

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.

DNS is simple but powerful

- Three major components
  - Domain Name Space and Resource Records
    - Specification of a tree-structured name space, and data associated with names
  - Name Servers
    - Hold information about the above
    - Have information about a name space subset (zone), and have pointers to other name servers
    - May be authority for a zone (have full information about it)
  - Resolvers
    - Client programs that extract information from name servers

The DNS Name Space

- In the DNS namespace, information is organized in Resource Records (RRs)
- The most important RRs for DNS are:

RR type	Entity	Description
SOA	Zone	“Start of Authority”: Holds information on the represented zone
A	Host	Contains an IP address (4 bytes) of the host this node represents
AAAA	Host	Same, but for IPv6 address (16 bytes)
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 human-readable information for an entity

DNS configuration

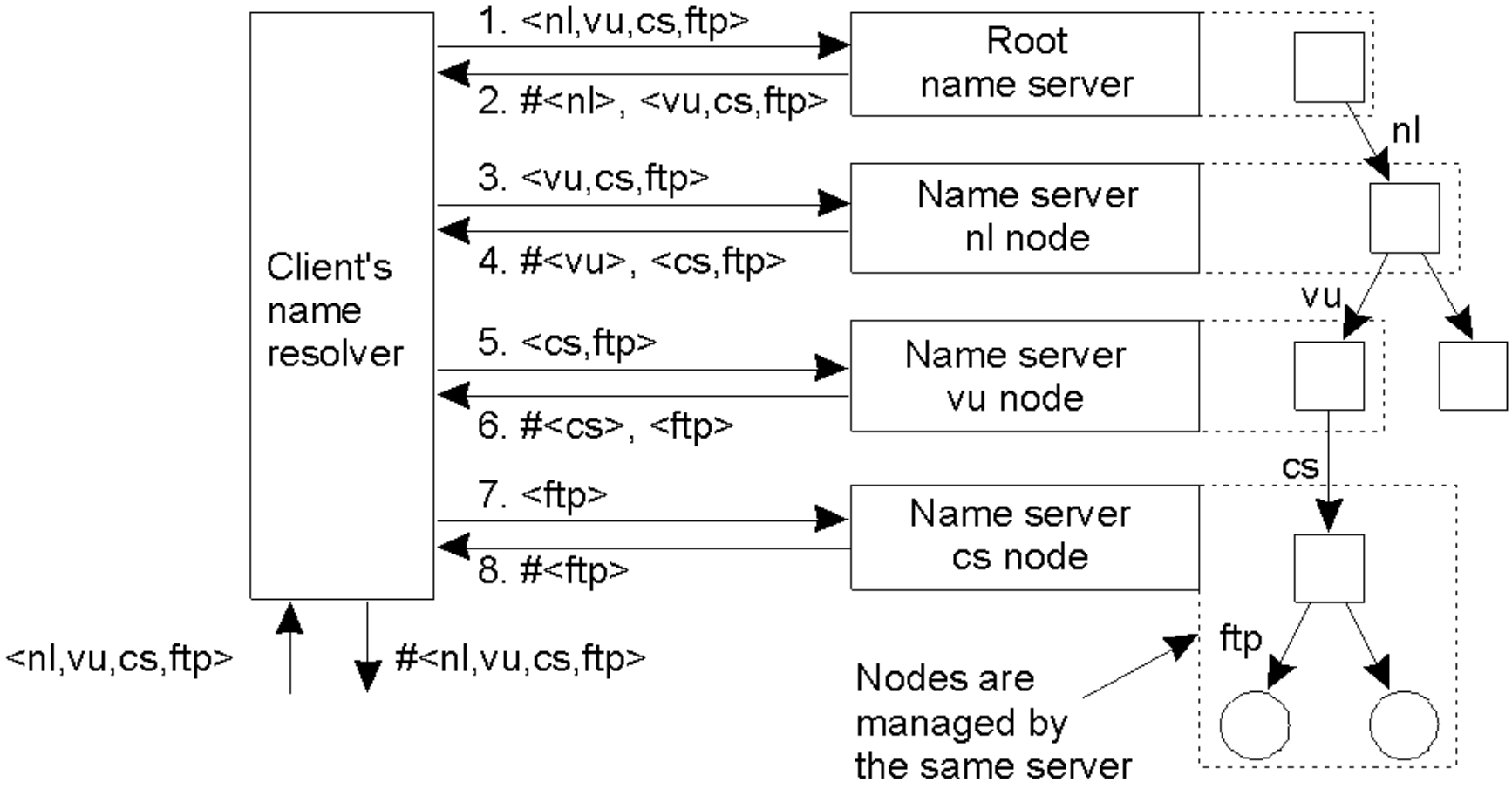
- An excerpt from the DNS database for the zone *cs.vu.nl*.

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

How does it work?

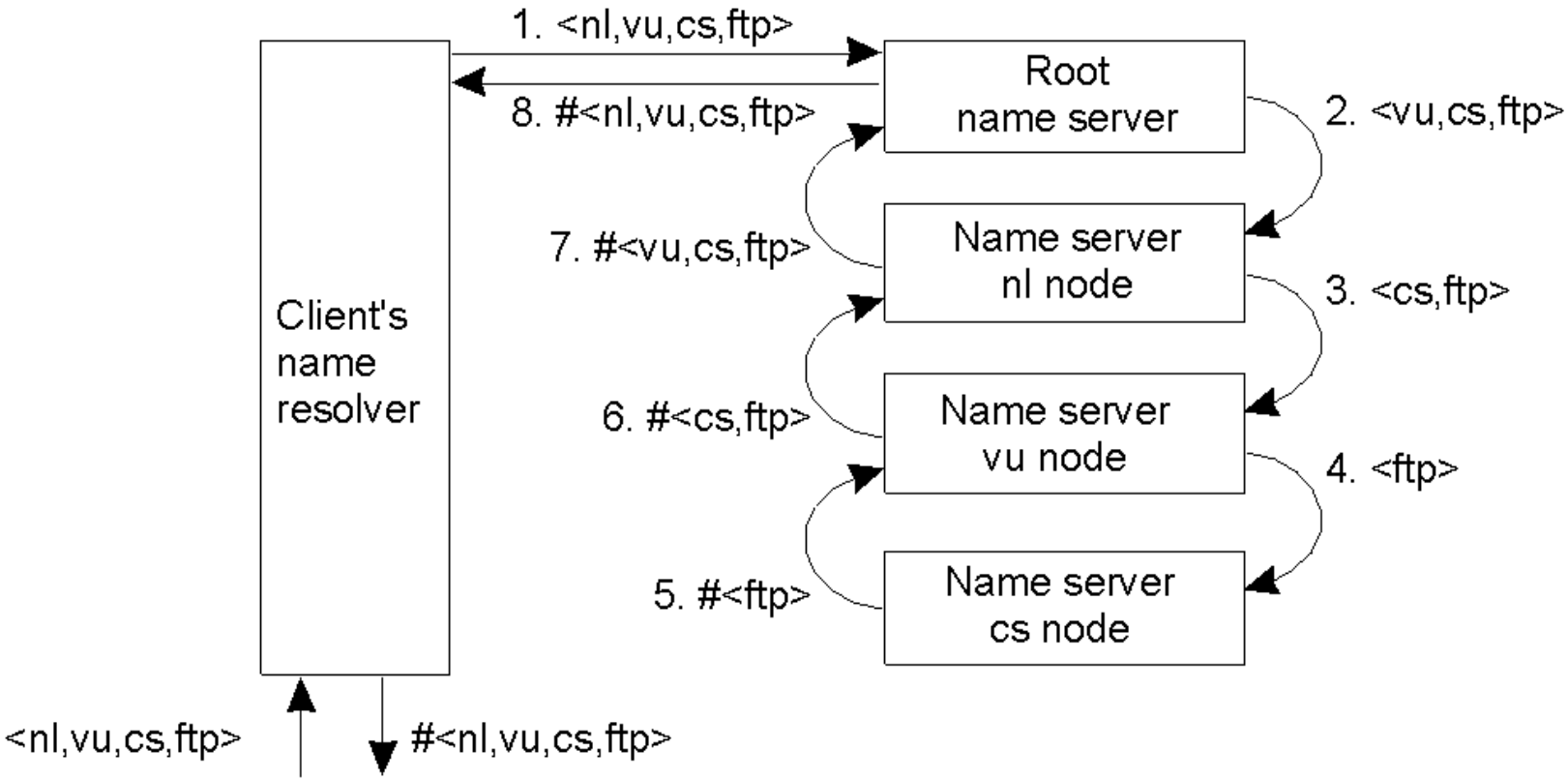
Iterative Name Resolution

The principle of iterative name resolution.



Recursive Name Resolution

The principle of recursive name resolution.

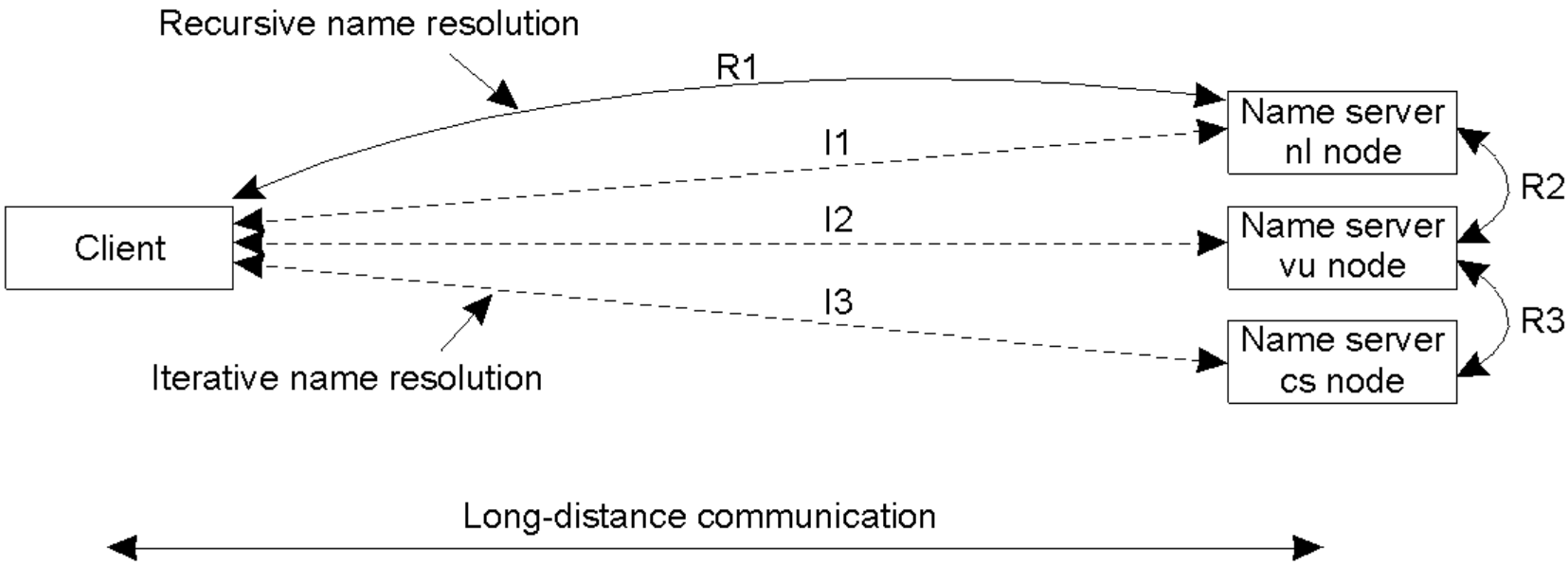


Name Resolution Implementation

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.

Iterative vs. Recursive



- The comparison between recursive and iterative name resolution with respect to communication costs.
- If there are N levels to be resolved:
  - Recursive takes 2*N messages
  - Iterative *could* take N+1 messages (the last DNS server sends reply back to client)
    - But Iterative takes 2*N messages too. Why?

Flat Naming

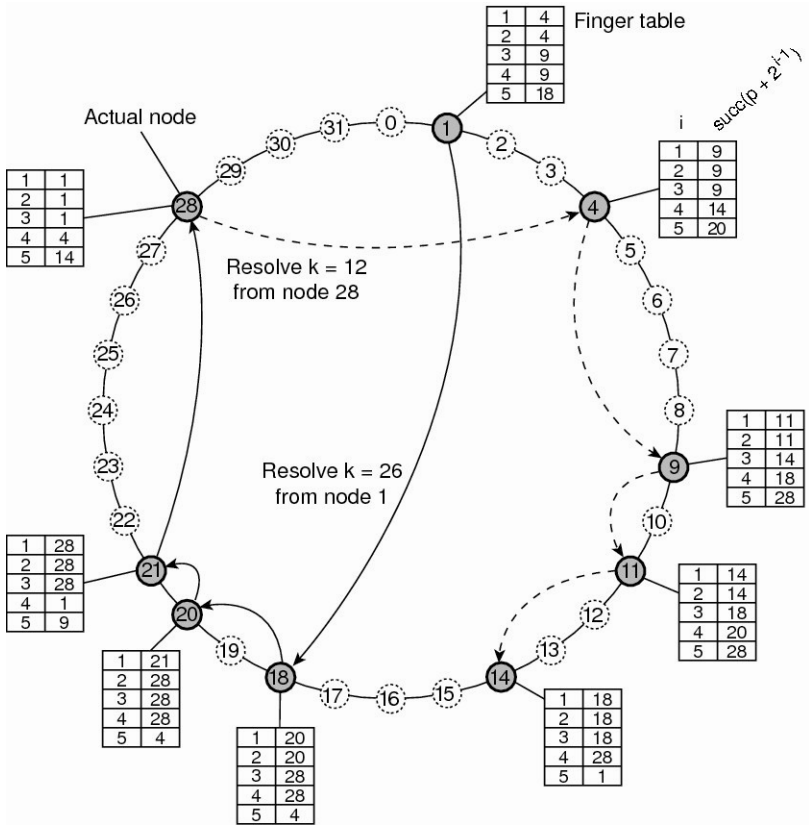
Dynamic and Distributed
Information Systems

Flat Naming

- Useful when we want to address a space in a homogeneous way
  - E.g., memory addressing
- Very common in centralized systems (memory, low-level disk access, etc.)
  - In decentralized systems very complicated
- Naïve approach to resolve a name:
  - Flood all network asking who has the name in question
  - The node that has that name replies
- Apparently this won't work in large scale systems

Distributed Hash Tables

- Each node
  - Has a unique ID
  - Maintains a small routing table (pointers to other nodes)
- Objects are stored in the node with the next higher ID
  - E.g., item with ID 26 will be stored at node 28 (because no node has ID 26 or 27)
- Can support thousands or millions of nodes
- This example is from the Chord DHT



Attributed-based Naming

Dynamic and Distributed
Information Systems

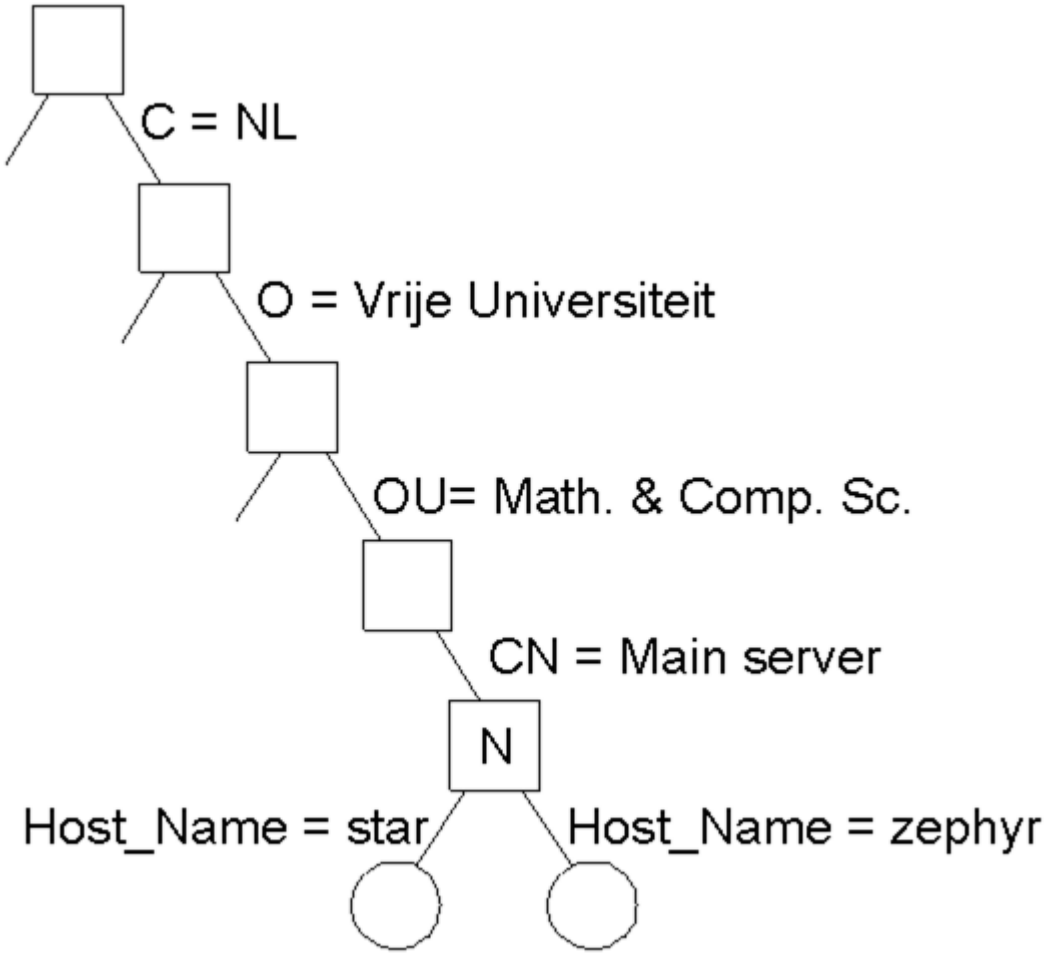
The X.500 Name Space

Attribute	Abbr.	Value
Country	C	NL
Locality	L	Amsterdam
Organization	L	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

- Part of the directory information tree.



The X.500 Name Space

- Two directory entries having *Host_Name* as RDN.

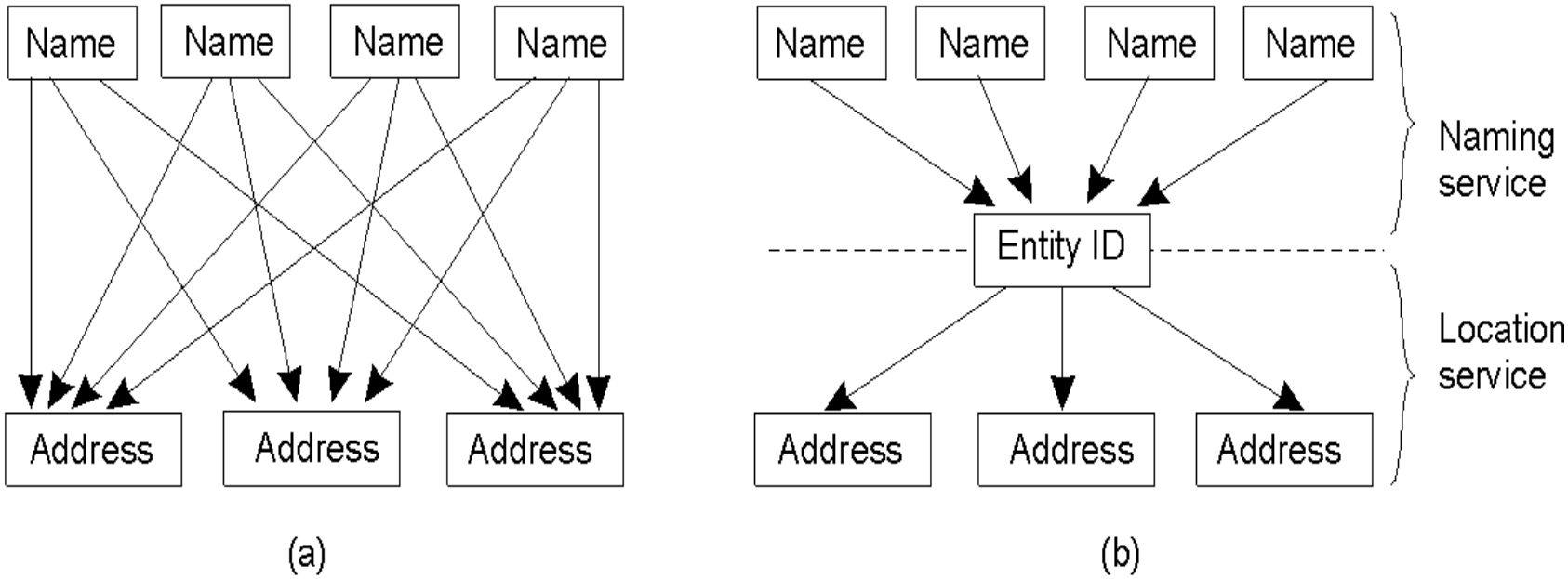
Attribute	Value
Country	NL
Locality	Amsterdam
Organization	Vrije Universiteit
OrganizationalUnit	Math. & Comp. Sc.
CommonName	Main server
Host_Name	star
Host_Address	192.31.231.42

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

Location Service

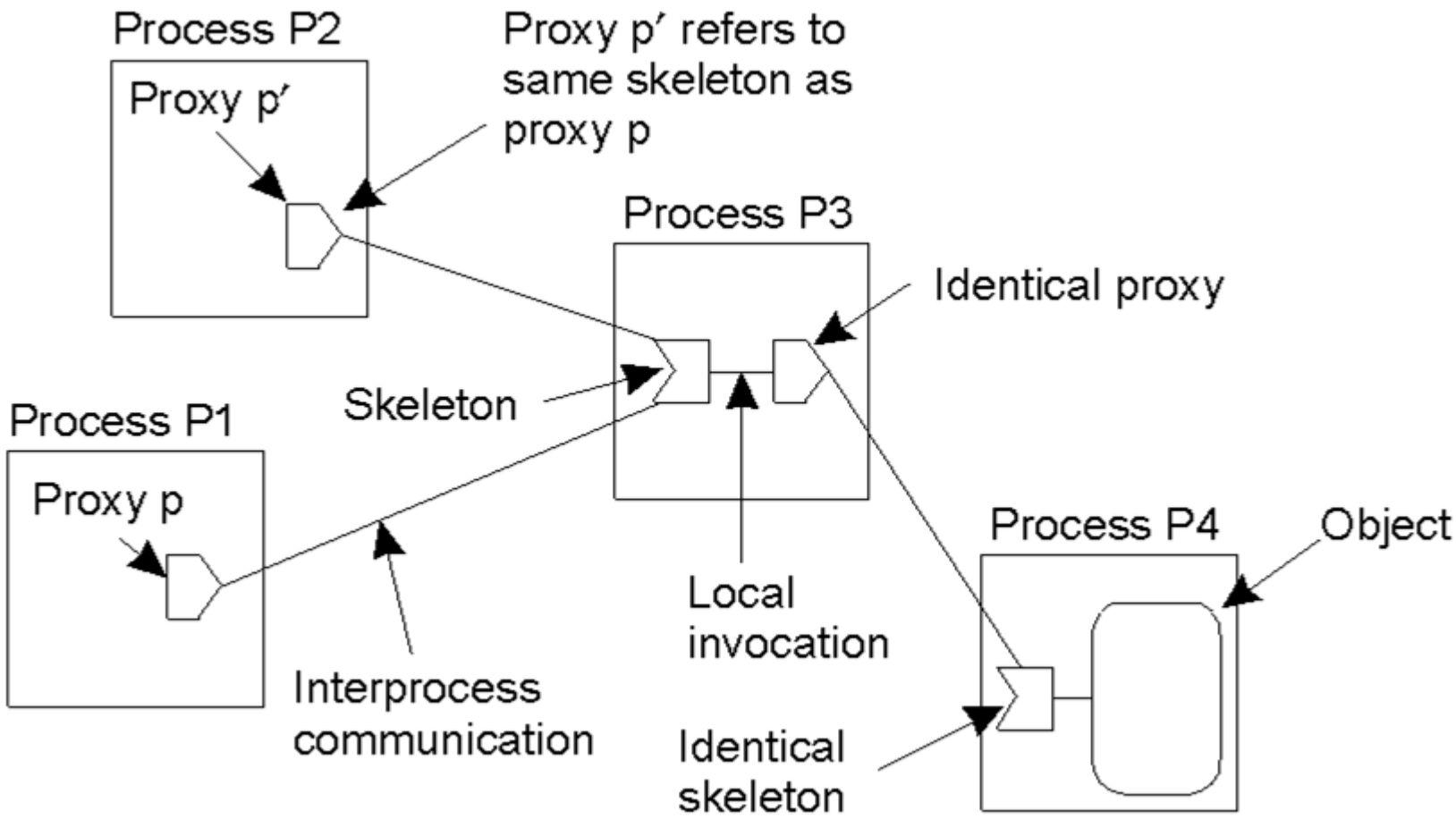
Dynamic and Distributed
Information Systems

Naming versus Locating Entities



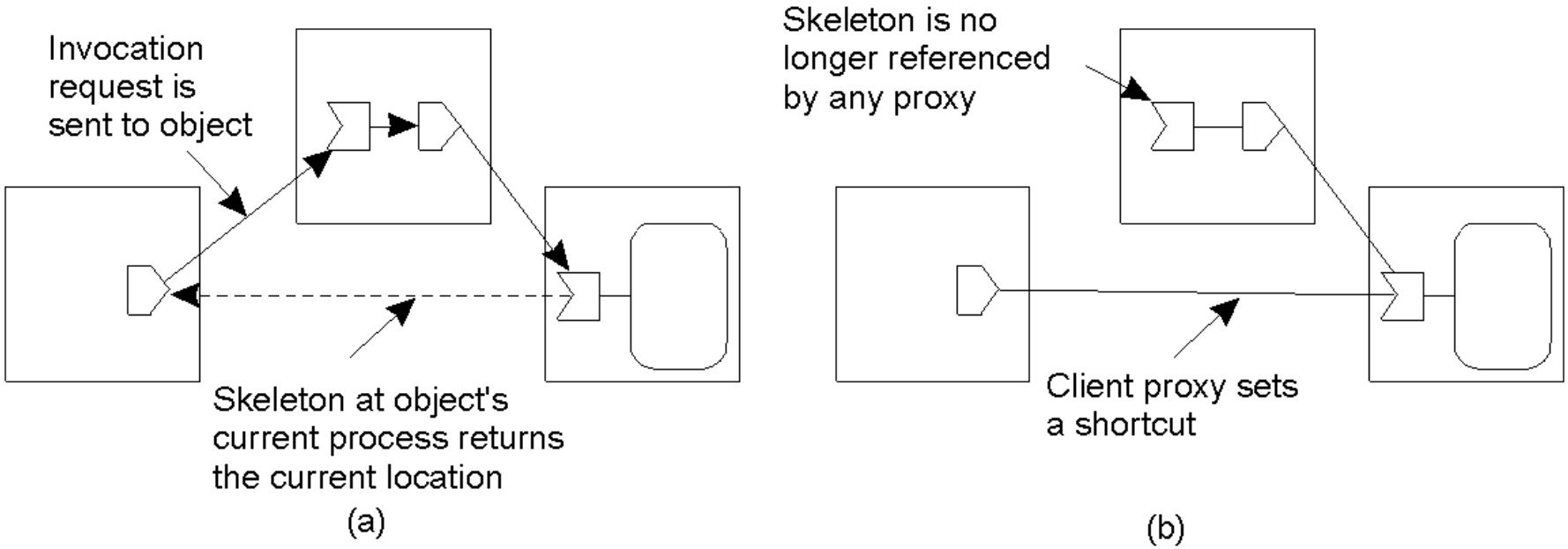
a) Direct, single level mapping between names and addresses.
b) T-level mapping using identities.

Forwarding Pointers



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

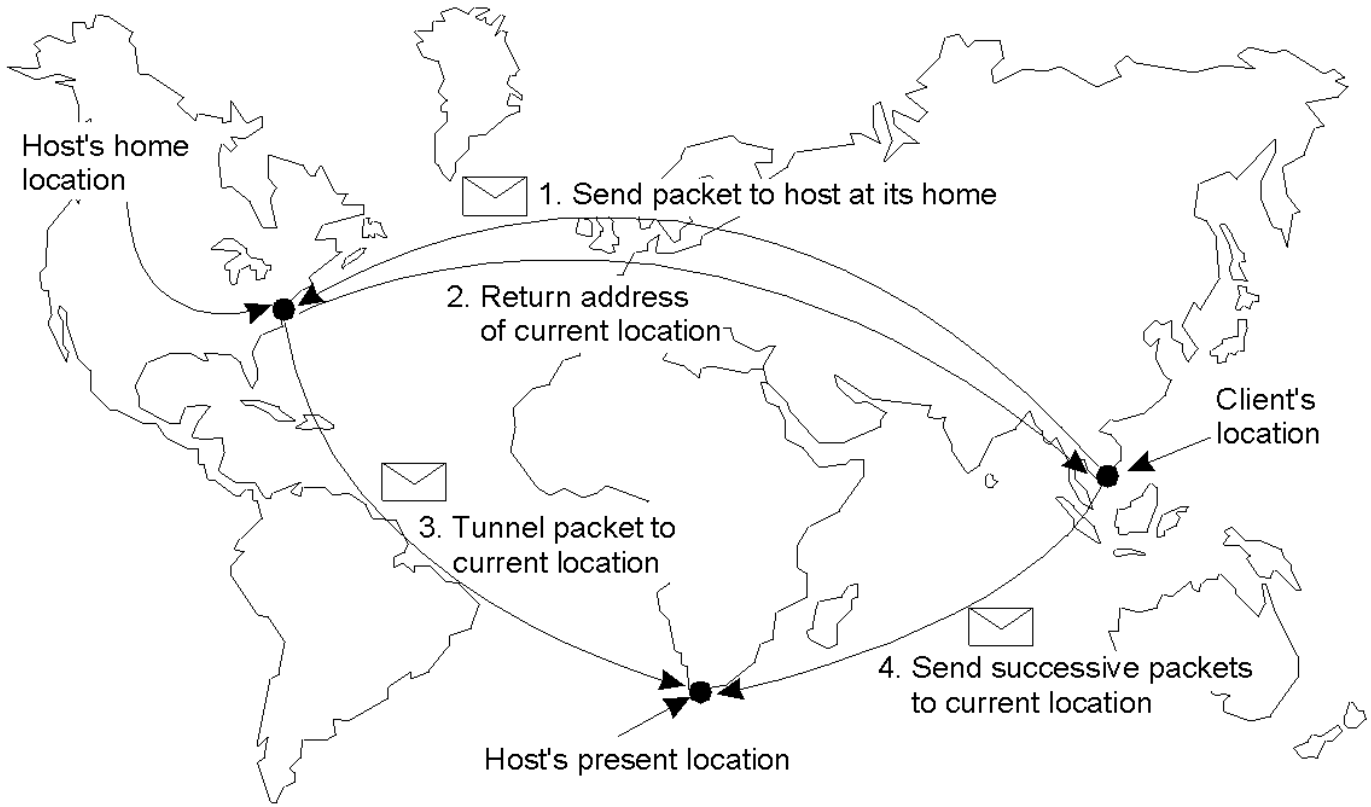
Forwarding Pointers



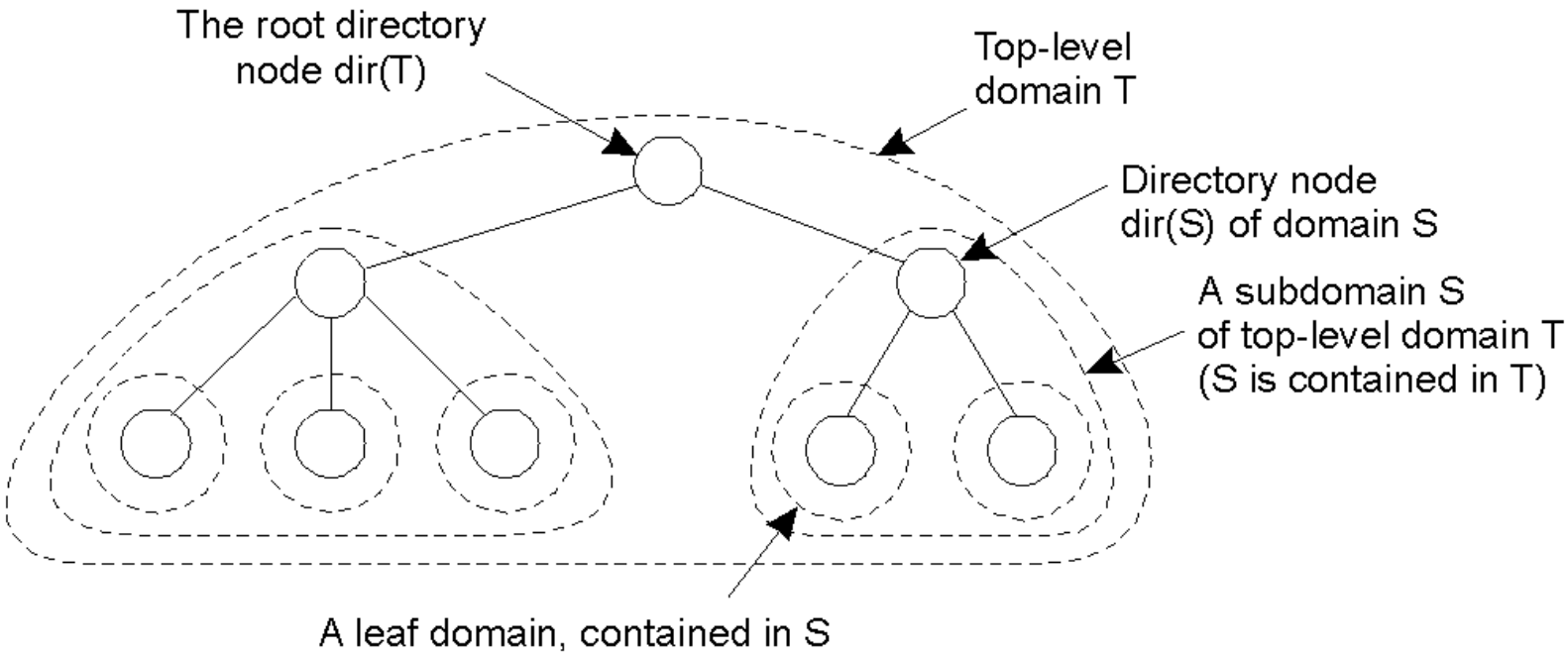
- Redirecting a forwarding pointer, by storing a shortcut in a proxy.

Home-Based Approaches

- The principle of Mobile IP.

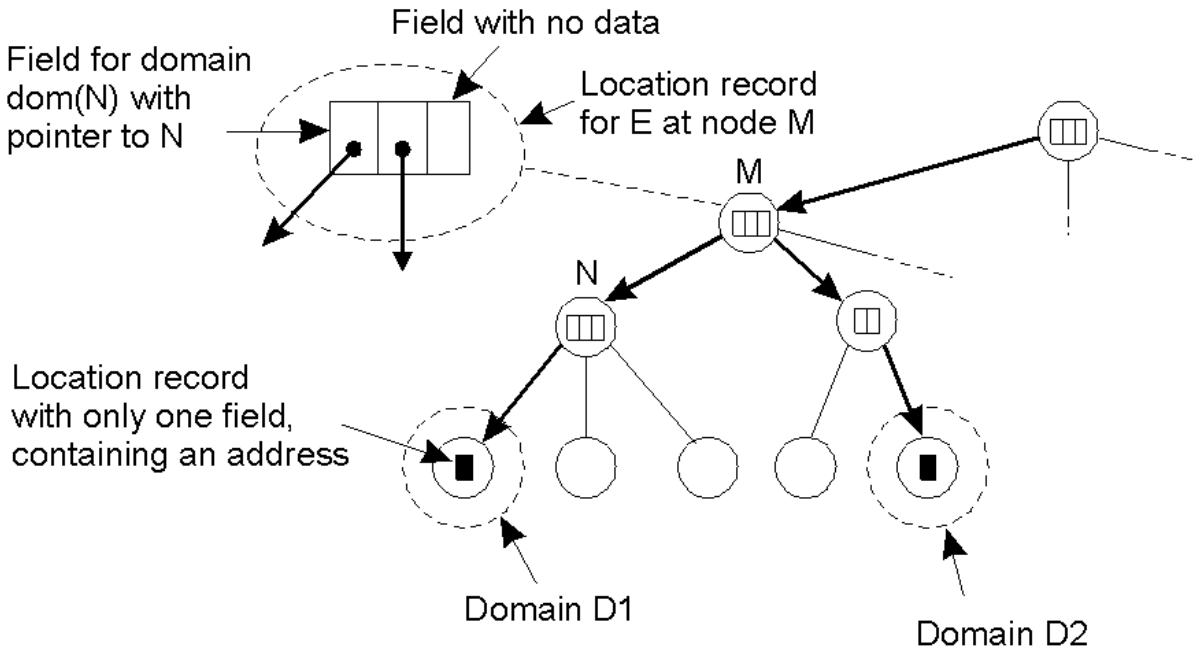


Hierarchical Location Service



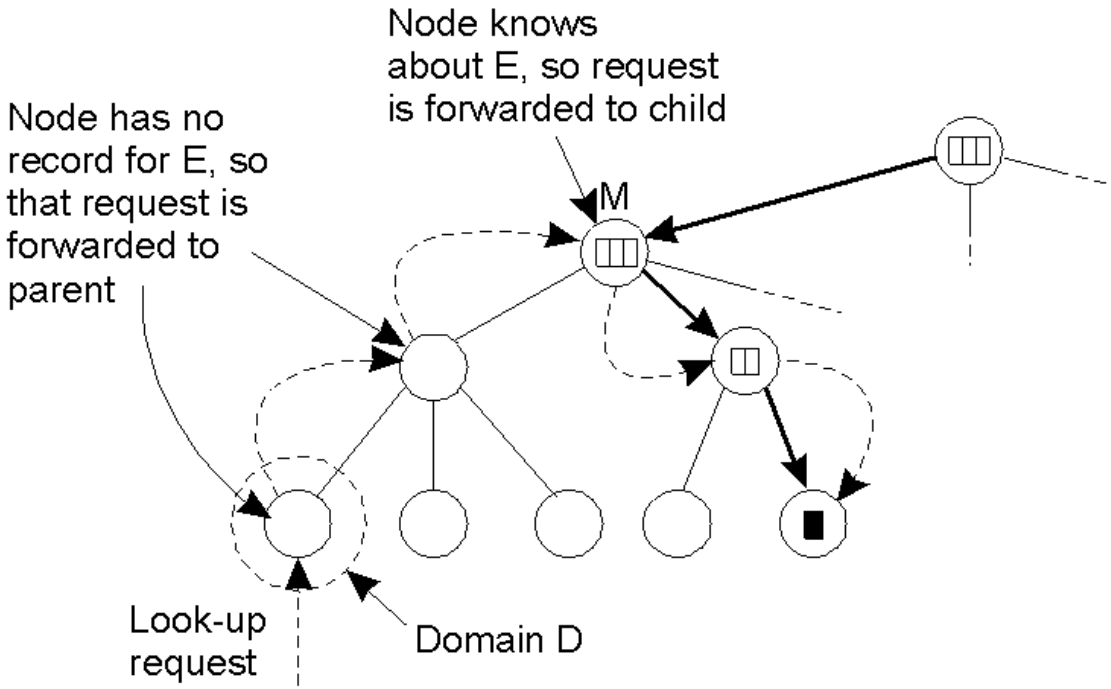
- Hierarchical organization of a location service into domains, each having an associated directory node.

Multiple addresses for an entity



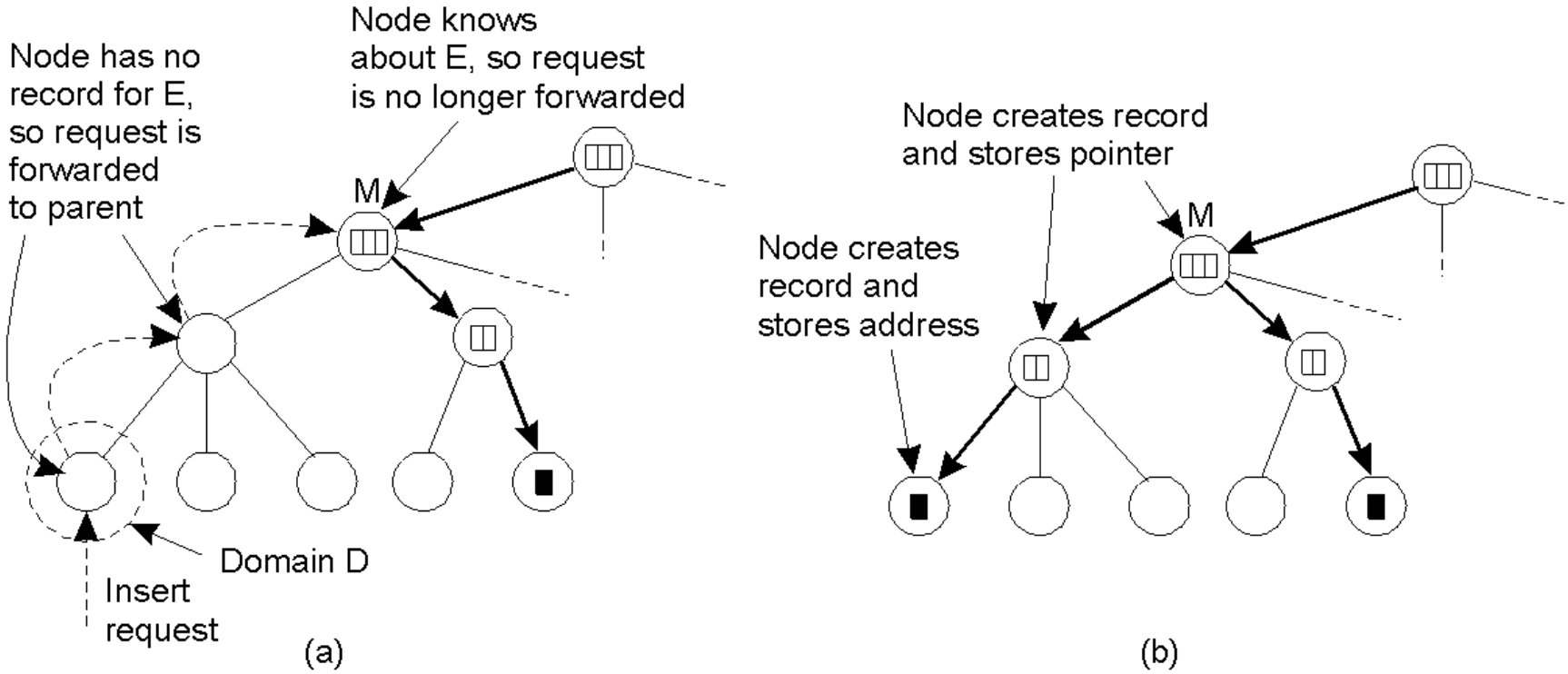
- An example of storing information of an entity having two addresses in different leaf domains.

Looking up an entity



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

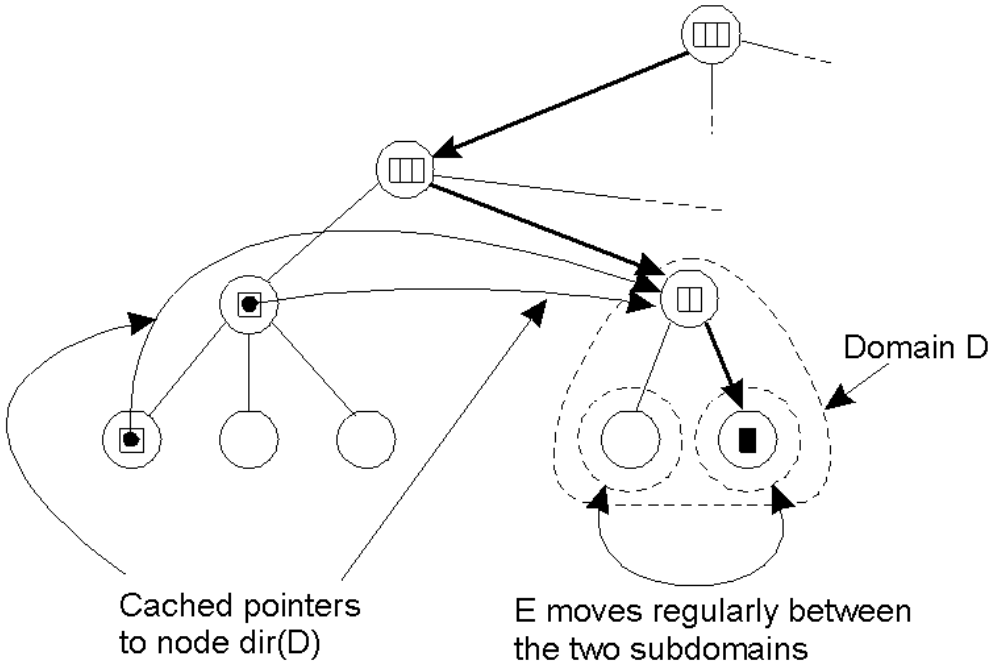
Inserting new Location Record for an entity



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.

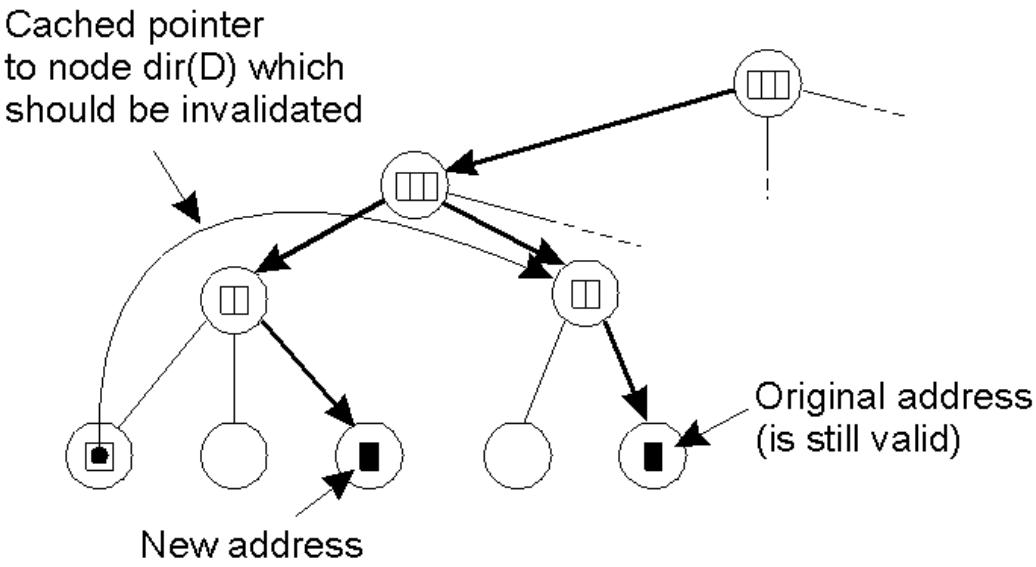
Pointer Caches (1)

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



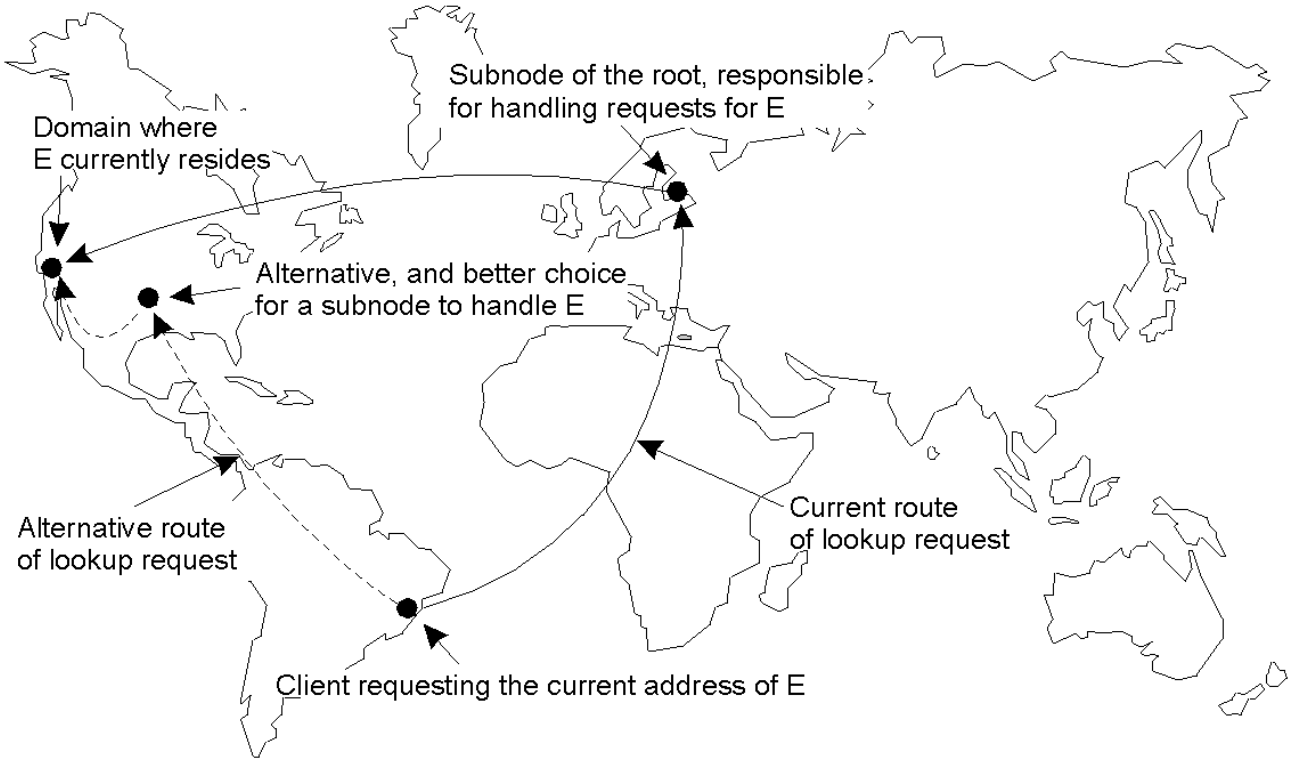
Open question

- A cache entry that needs to be invalidated because it returns a nonlocal address, while such an address is available.



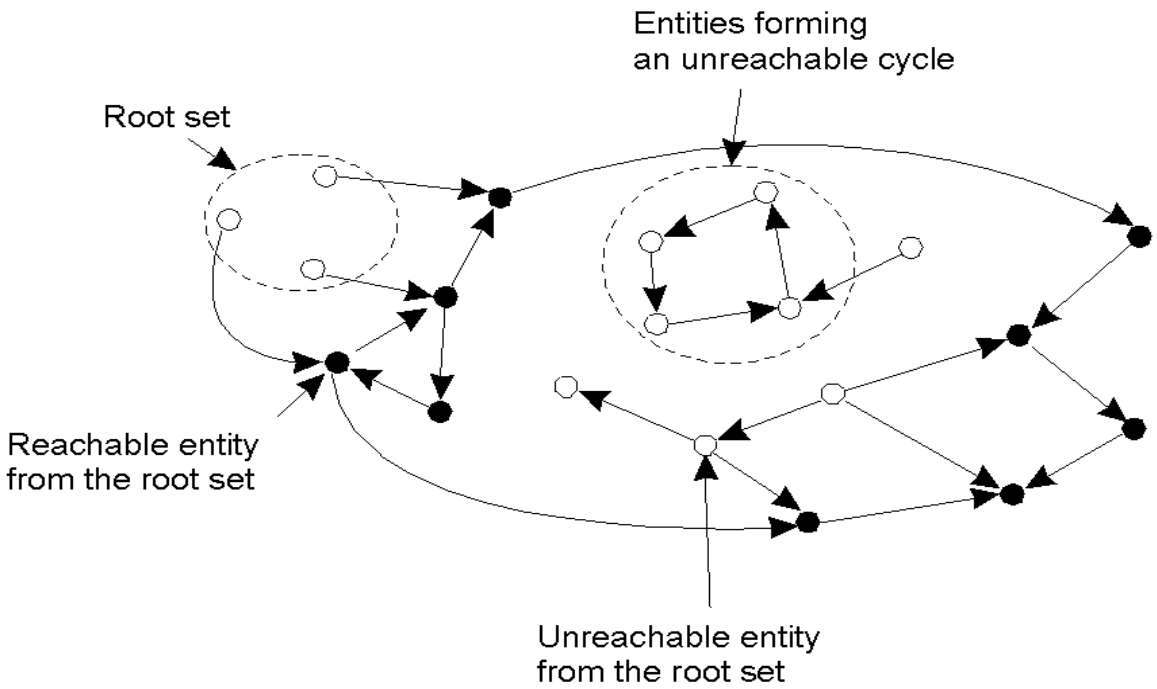
Scalability Issues

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



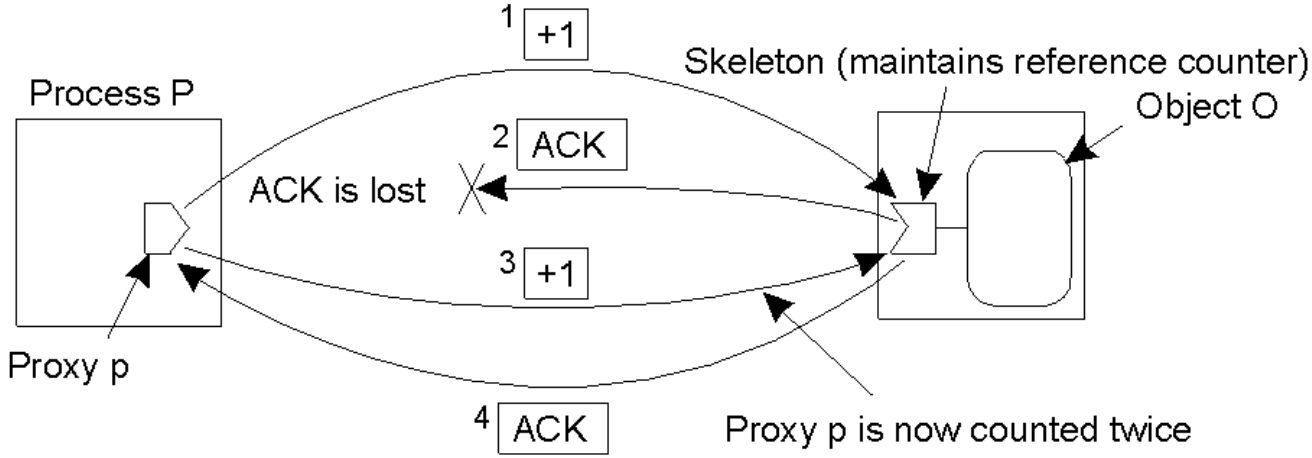
Unreferenced Objects

An example of a graph representing objects containing references to each other.

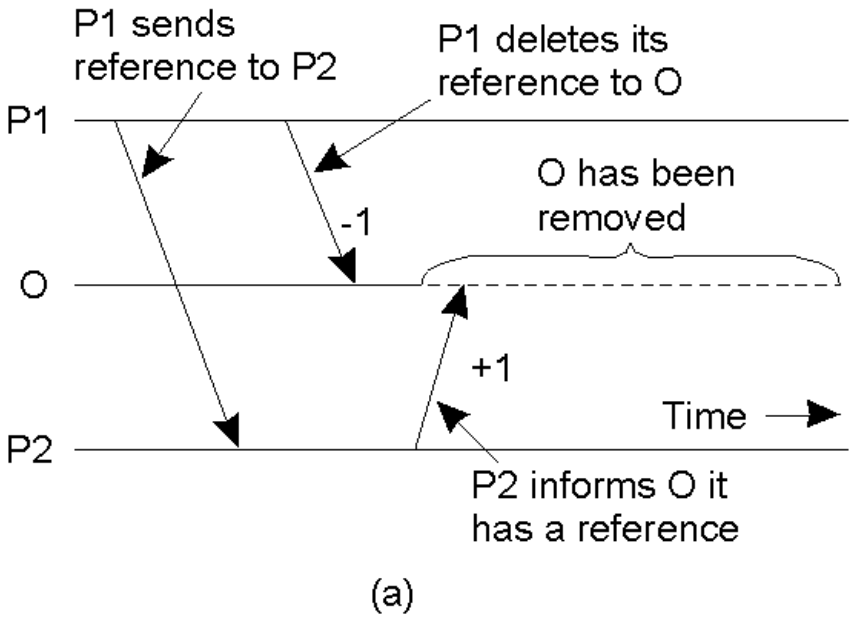


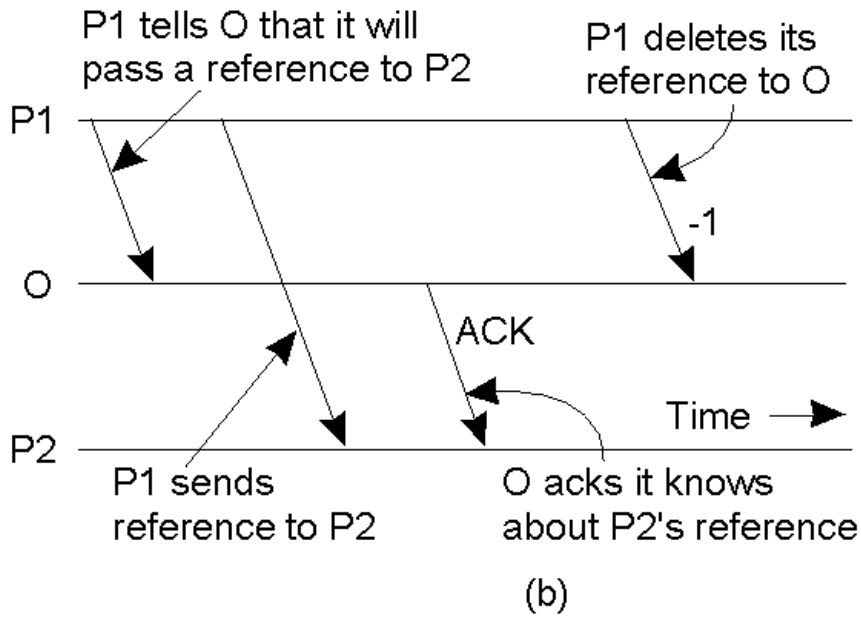
Reference Counting (1)

- The problem of maintaining a proper reference count in the presence of unreliable communication.



Reference Counting (2)



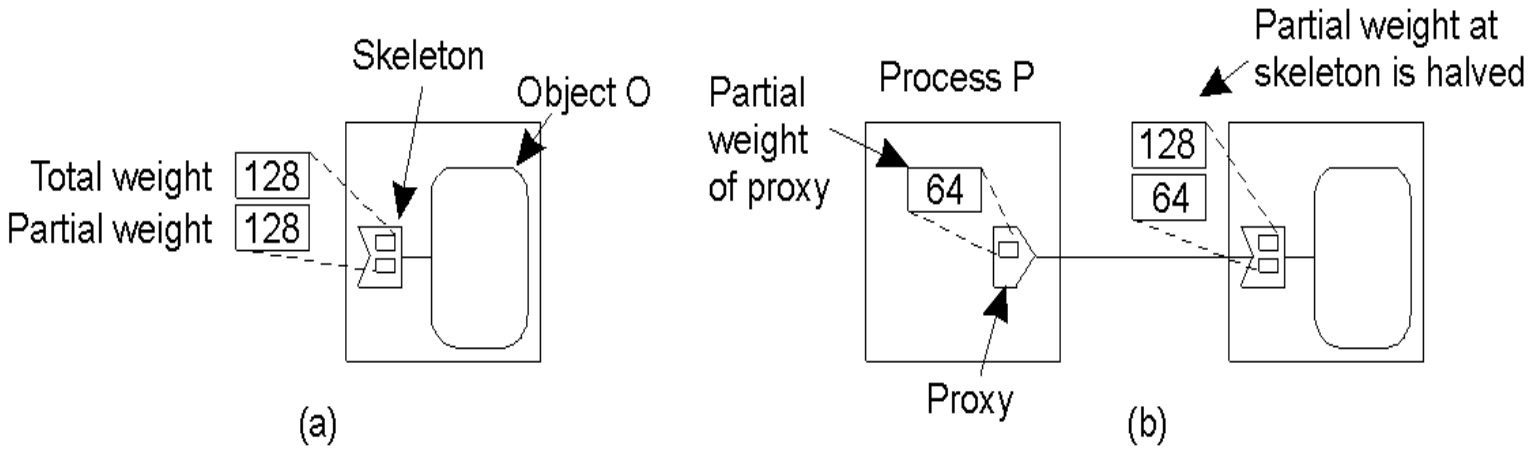


a) Copying a reference to another process and incrementing the counter too late

b) A solution.

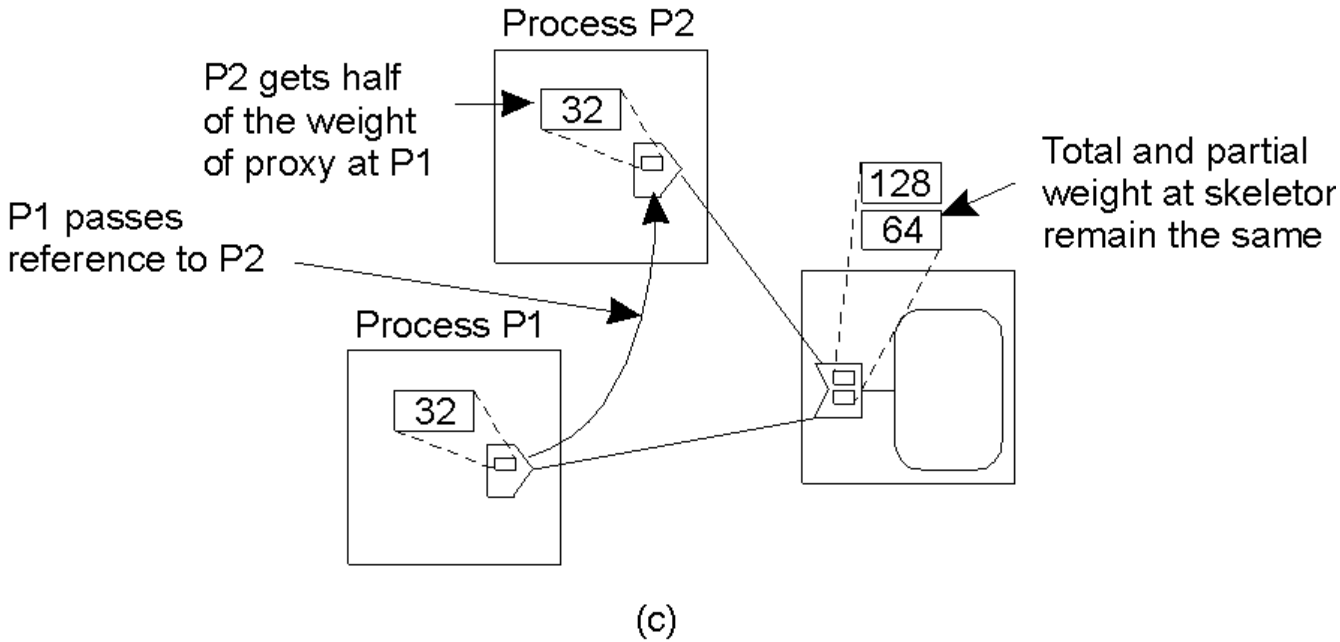
Advanced Referencing Counting (1)

- The initial assignment of weights in weighted reference counting
- Weight assignment when creating a new reference.



Advanced Referencing Counting (2)

- Weight assignment when copying a reference.



(c)

Advanced Referencing Counting (3)

- Creating an indirection when the partial weight of a reference has reached 1.