Distributed Systems Design
COMP 6231

Naming: Chapter 5 Part 2

Lecture 6

Essam Mansour

Today…

- Last Session:
  - Naming- Part I
- Today’s Session:
  - Naming- Part II

Classes of Naming

- Flat naming
- Structured naming
- Attribute-based naming

Classes of Naming

- Flat naming
- Structured naming
- Attribute-based naming

Structured Naming

- Structured names are composed of simple human-readable names
  - Names are arranged in a specific structure

- Examples:
  - File-systems utilize structured names to identify files
    - /home/userid/work/dist-systems/naming.txt
  - Websites can be accessed through structured names
    - www.concordia.ca

Name Spaces

- Structured names are organized into *name spaces*

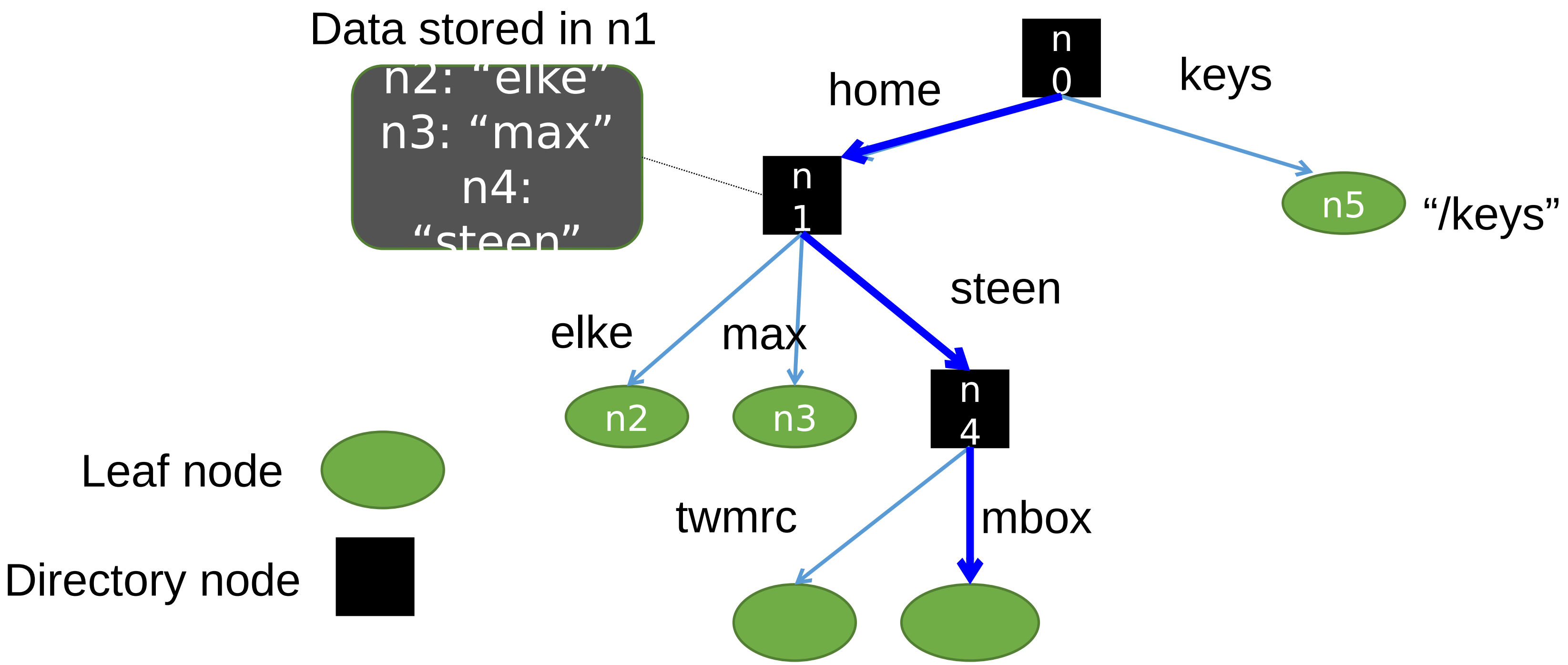
- A name space is a *directed graph* consisting of:
  - Leaf nodes
    - Each leaf node represents an entity
    - A leaf node generally stores the *address* of an entity (e.g., in DNS), or the *state* of (or the *path* to) an entity (e.g., in file systems)

  - Directory nodes
    - Directory node refers to other leaf or directory nodes
    - Each outgoing edge is represented by (*edge label, node identifier*)

- Each node can store any type of data
  - I.e., State and/or address (*e.g., to a different machine*) and/or path

Name Spaces: An Example

Looking up for the entity with name "/home/steen/mbox"



Name Resolution

- The process of looking up a name is called *name resolution*

- Closure mechanism:
  - Name resolution cannot be accomplished without an *initial directory node*

  - The *closure mechanism* selects the implicit context from which to start name resolution

  - Examples:
    - www.concordia.ca: start at the DNS Server
    - /home/steen/mbox: start at the root of the file-system

Name Linking

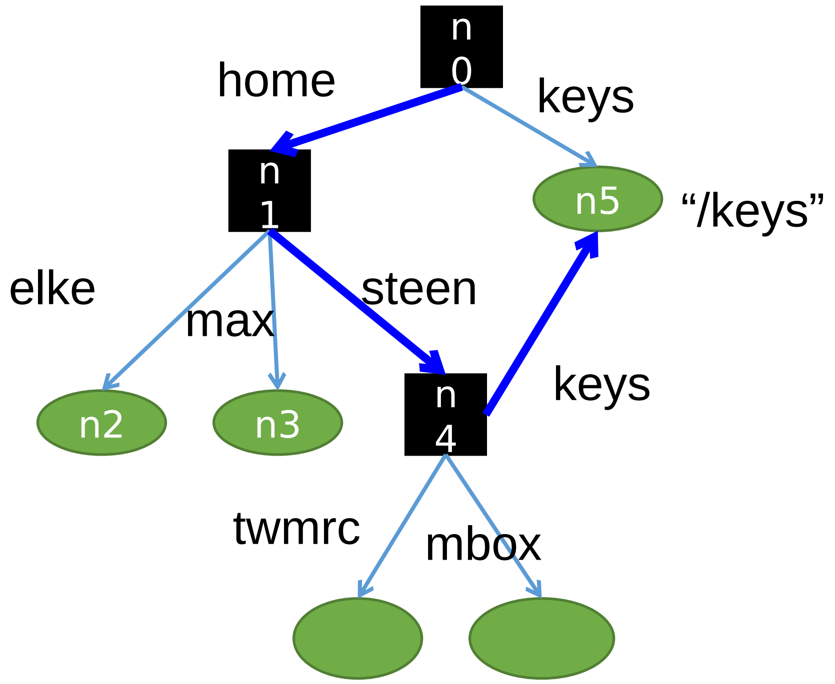
- The name space can be effectively used to link two different entities

- Two types of links can exist between the nodes:
    1. Hard Links
    2. Symbolic Links

1. Hard Links

- There is a directed link from the hard link to the actual node
- Name resolution:
  - Similar to the general name resolution
- Constraint:
  - There should be no cycles in the graph

"/home/steen/keys" is a hard link to "/keys"

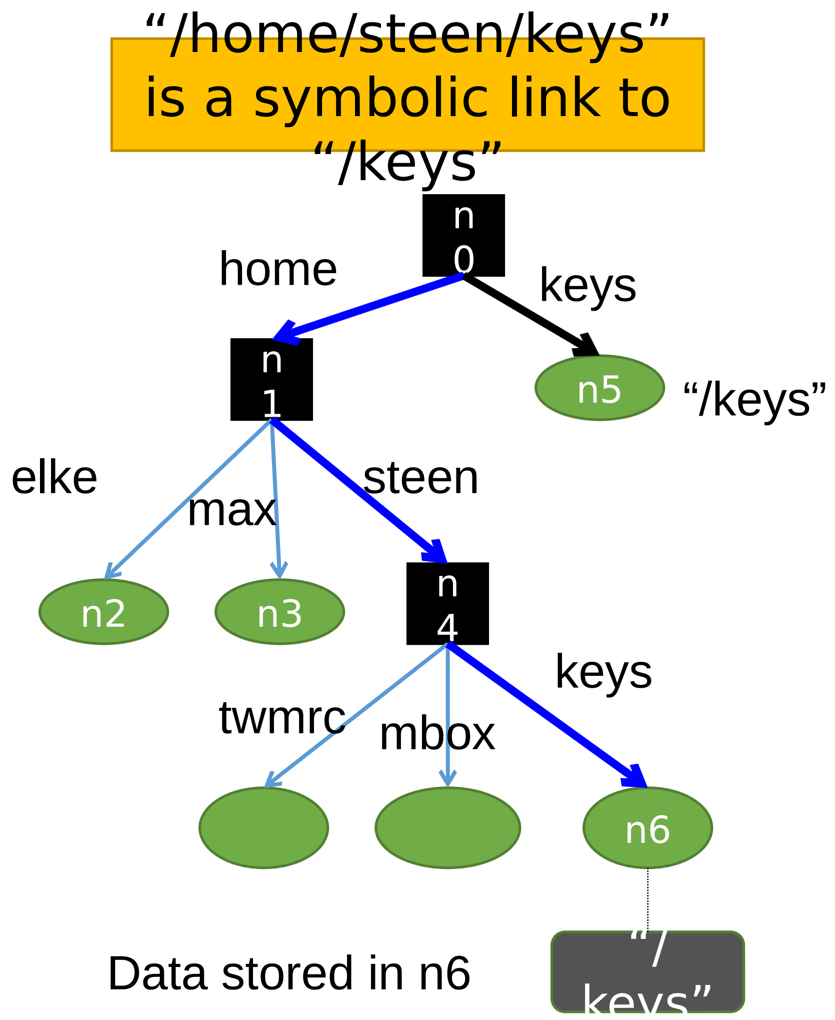


2. Symbolic Links

- Symbolic link stores the name of the original node as *data*

- Name resolution for a symbolic link SL
  - First resolve SL’s name
  - Read the content of SL
  - Name resolution continues with content of SL

- Constraint:
  - No cyclic references should be present



Mounting of Name Spaces

- Two or more name spaces can be merged transparently by a technique known as *mounting*

- With mounting, a directory node in one name space will store the identifier of the directory node of another name space

- Network File System (NFS) is an example where different name spaces are mounted
    - NFS enables *transparent* access to remote files

Mounting of Name Spaces

Issue

Name resolution can also be used to merge different name spaces in a transparent way through mounting: associating a node identifier of another name space with a node in a current name space.

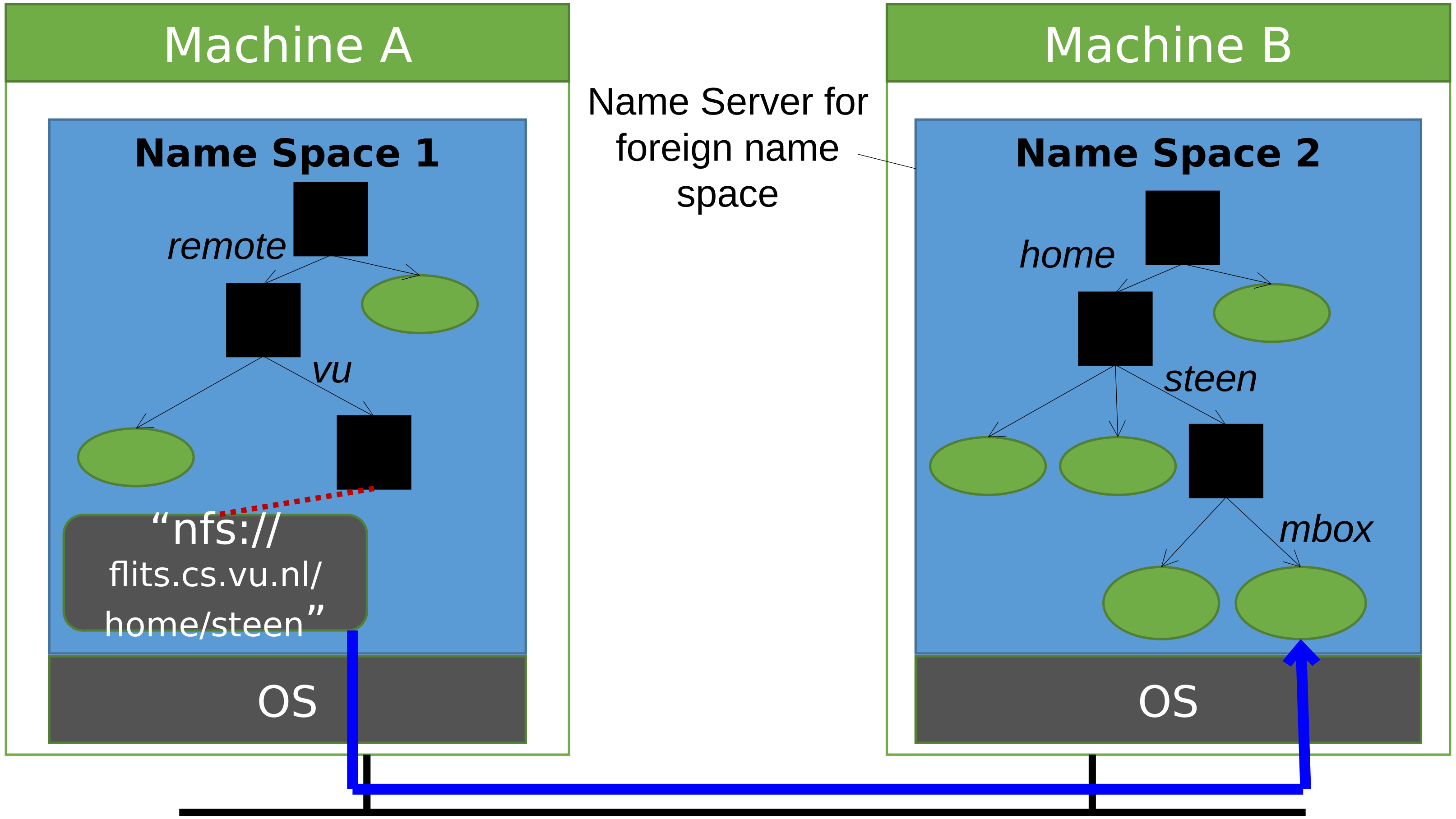
Terminology

- Foreign name space: the name space that needs to be accessed
- Mount point: the node in the current name space containing the node identifier of the foreign name space
- Mounting point: the node in the foreign name space where to continue name resolution

Mounting across a network

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

Example of Mounting Name Spaces in NFS



Name resolution for “/remote/vu/home/steen/mbox” in a distributed file system

Distributed Name Spaces

- In large-scale distributed systems, it is essential to distribute name spaces over multiple name servers
  - Distribute the nodes of the naming graph
  - Distribute the name space management
  - Distribute the name resolution mechanisms

Layers in Distributed Name Spaces

- Distributed name spaces can be divided into three *layers*

Global Layer

- Consists of high-level directory nodes
- Directory nodes are jointly managed by different administrations

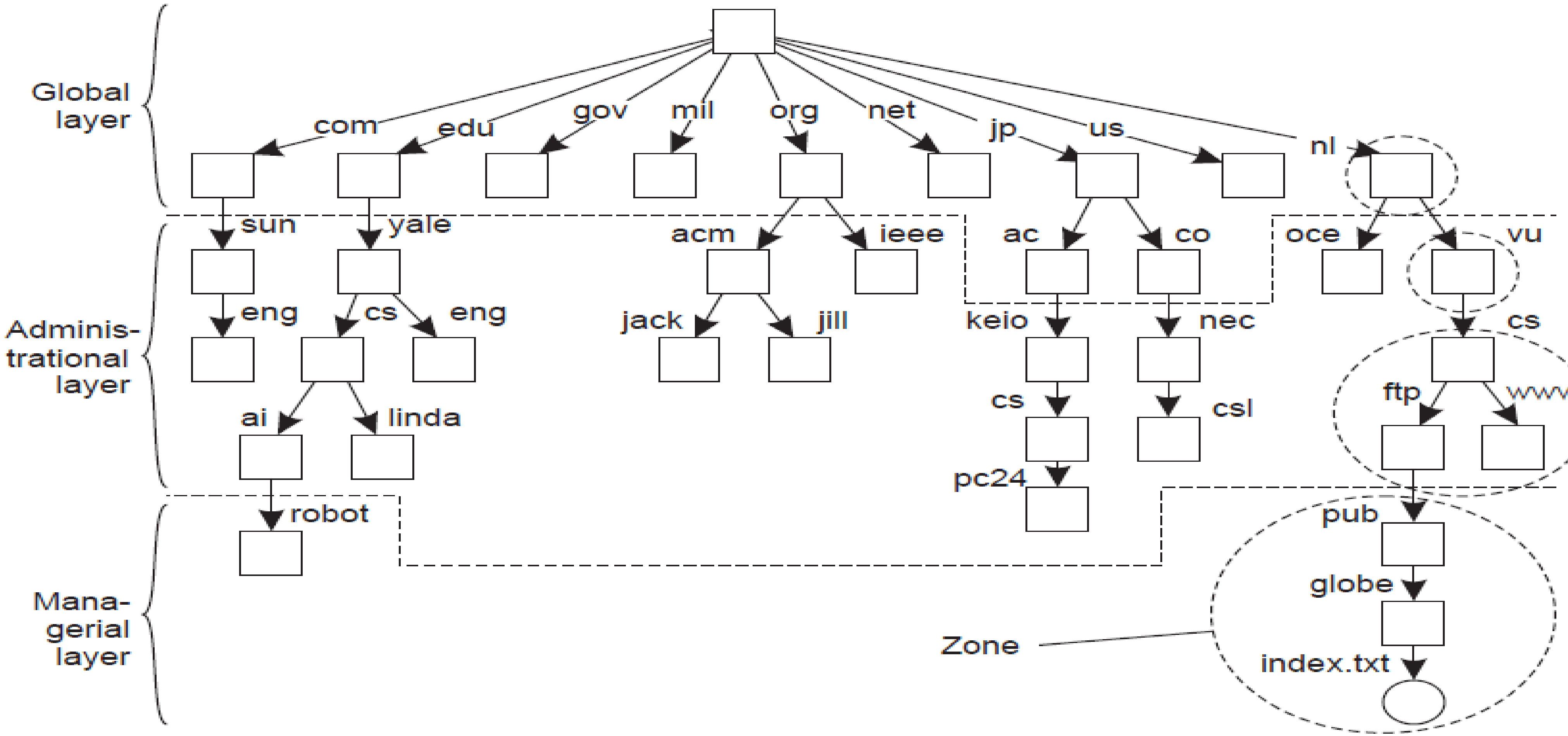
Administr at-ional Layer

- Contains mid-level directory nodes
- Directory nodes grouped together in such a way that each group is managed by an administration

Manageri al Layer

- Contains low-level directory nodes within a single administration
- The main issue is to efficiently map directory nodes to local name servers

Distributed Name Spaces – An Example



Comparison of Name Servers at Different Layers

	Global	Administrational	Managerial
Geographical scale of the network	Worldwide	Organization	Department
Total number of nodes	Few	Many	Vast numbers
Number of replicas	Many	None or few	None
Update propagation	Lazy	Immediate	Immediate
Is client side caching applied?	Yes	Yes	Sometimes
Responsiveness to lookups	Seconds	Milliseconds	Immediate

Distributed Name Resolution

- Distributed name resolution is responsible for mapping names to addresses in a system where:
  - Name servers are distributed among participating nodes
  - Each name server has a local *name resolver*

- We will study two distributed name resolution algorithms:
  1. Iterative Name Resolution
  2. Recursive Name Resolution

1. Iterative Name Resolution

1. Client hands over the complete name to *root name server*

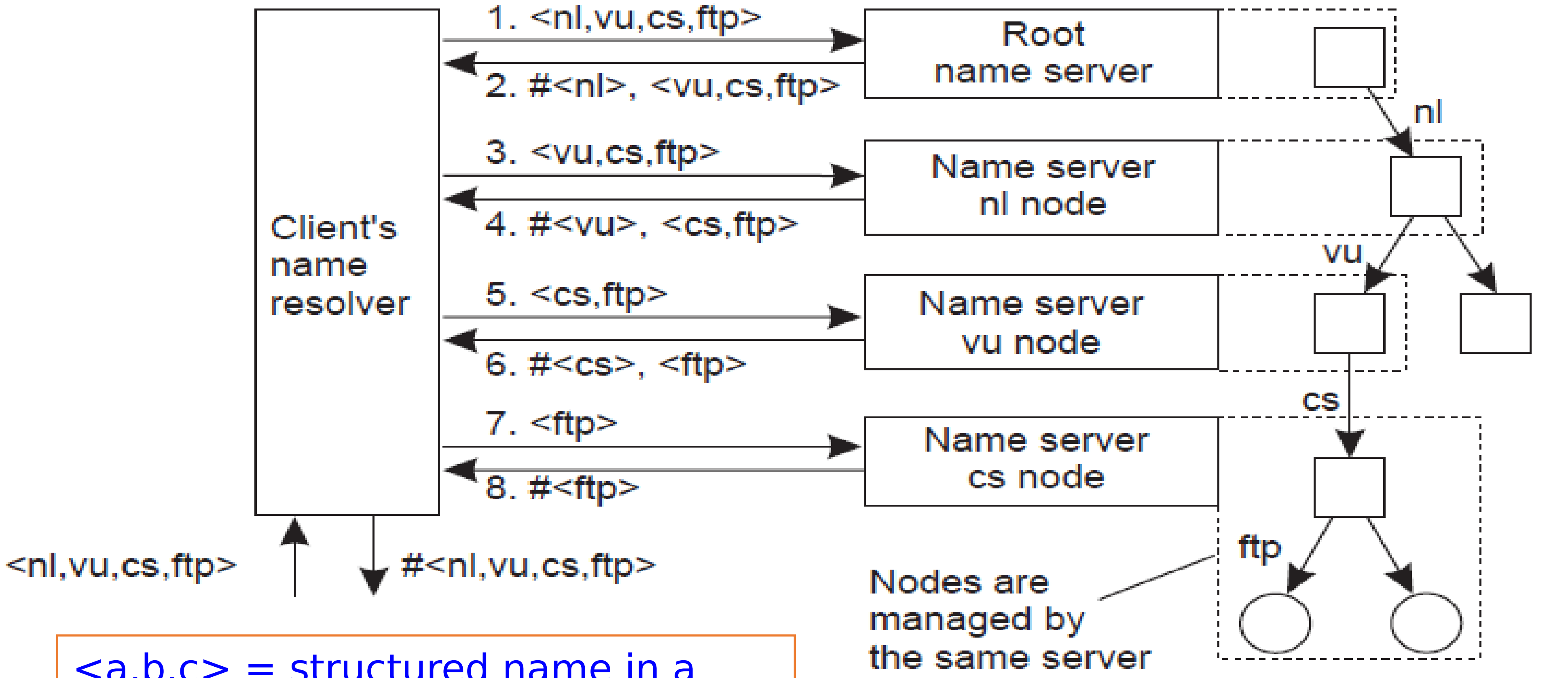
2. Root name server resolves the name as far as it can, and returns the result to the client
    - The root name server returns the address of the next-level name server (say, NLNS) if address is not completely resolved

3. Client passes the unresolved part of the name to the NLNS

4. NLNS resolves the name as far as it can, and returns the result to the client (and probably its next-level name server)

5. The process continues untill the full name is resolved

1. Iterative Name Resolution – An Example



<a,b,c> = structured name in a sequence
#<a> = address of node with name “a”

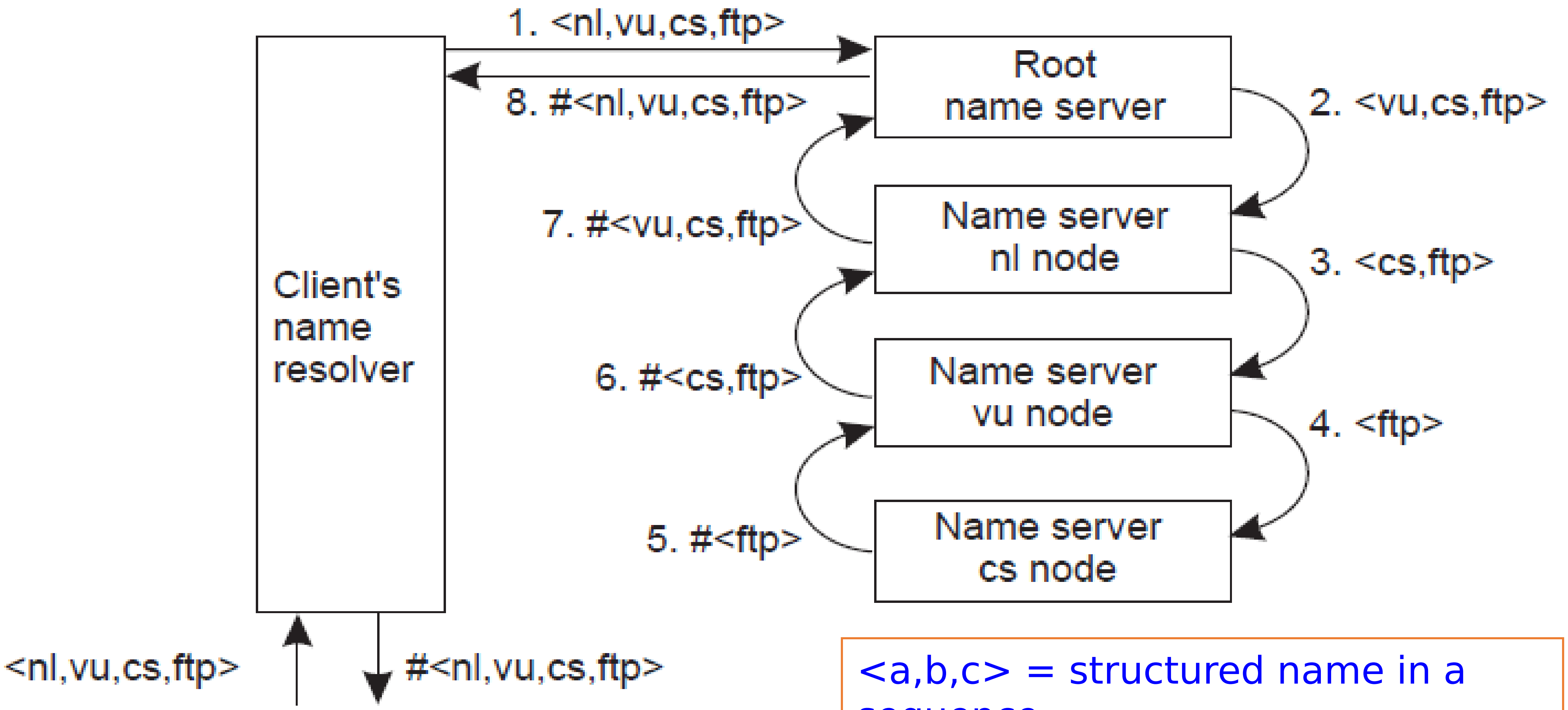
e name “*ftp.cs.vu.nl*”

2. Recursive Name Resolution

- Approach:
  - Client provides the name to the root name server
  - The root name server passes the result to the next name server it finds
  - The process continues till the name is fully resolved

- Drawback:
  - Large overhead at name servers (especially, at the high-level name servers)

2. Recursive Name Resolution – An Example



<a,b,c> = structured name in a sequence
#<a> = address of node with name “a”

Resolving the na

Scalability issues

Size scalability

We need to ensure that servers can handle a large number of requests per time unit
⇒ high-level servers are in big trouble.

Solution

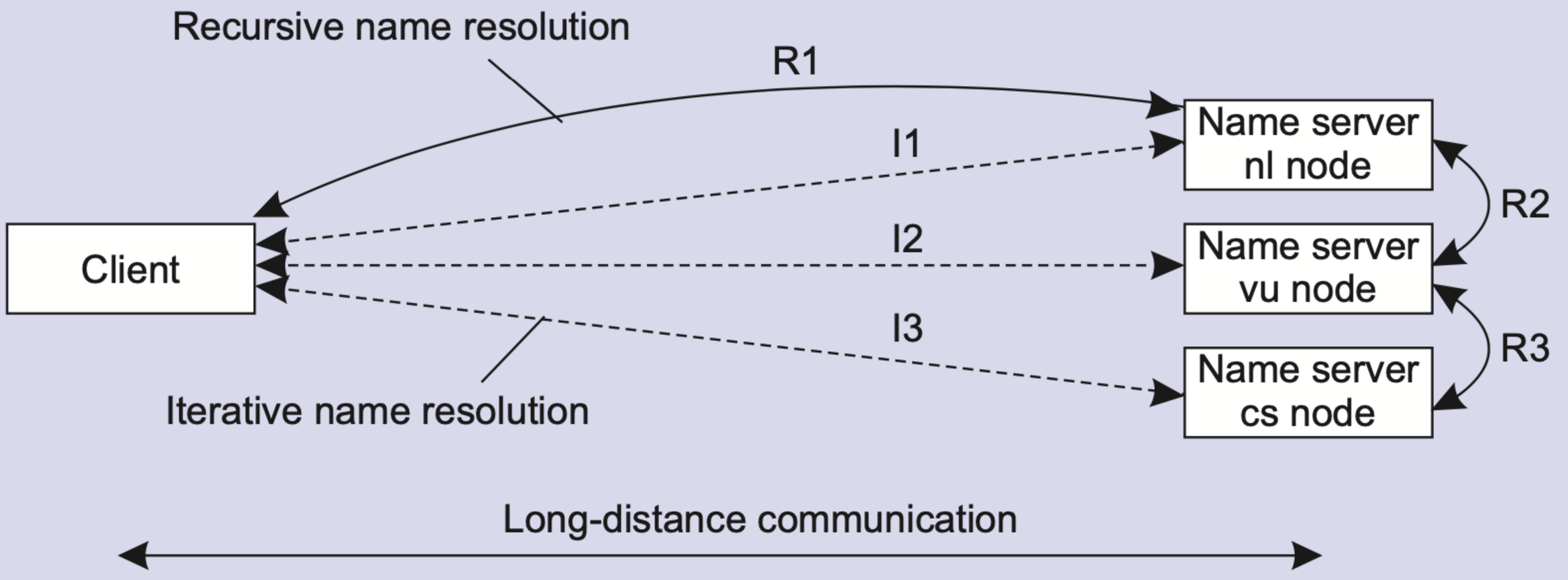
Assume (at least at global and administrational level) that content of nodes hardly ever changes. We can then apply extensive replication by mapping nodes to multiple servers, and start name resolution at the nearest server.

Observation

An important attribute of many nodes is the address where the represented entity can be contacted. Replicating nodes makes large-scale traditional name servers unsuitable for locating mobile entities.

Scalability issues

We need to ensure that the name resolution process scales across large geographical distances



Problem

We need to ensure that servers can handle a large number of requests per time unit ⇒ high-level servers are in big trouble.

Domain Name System (DNS)

Essence

- Hierarchically organized name space with each node having exactly one incoming edge ⇒ edge label = node label.
- domain: a subtree
- domain name: a path name to a domain's root node.

Information in a node

Type	Refers to	Description
SOA	Zone	Holds info on the represented zone
A	Host	IP addr. of host this node represents
MX	Domain	Mail server to handle mail for this node
SRV	Domain	Server handling a specific service
NS	Zone	Name server for the represented zone
CNAME	Node	Symbolic link
PTR	Host	Canonical name of a host
HINFO	Host	Info on this host
TXT	Any kind	Any info considered useful

Classes of Naming

- Flat naming
- Structured naming
- Attribute-based naming

Attribute-based Naming

- In many cases, it is much more convenient to name, and look up entities by means of their attributes
  - Similar to traditional directory services (e.g., yellow pages)

- However, the lookup operations can be extremely expensive
  - They require to match requested attribute values, against actual attribute values, which might require inspecting all entities

- Solution: Implement basic directory service as a database, and combine it with traditional structured naming system

- We will study Light-weight Directory Access Protocol (LDAP); an example system that uses attribute-based naming

Implementing directory services

Solution for scalable searching

Implement basic directory service as database, and combine with traditional structured naming system.

Lightweight Directory Access Protocol (LDAP)

Each directory entry consists of (*attribute, value*) pairs, and is uniquely named to ease lookups.

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

Light-weight Directory Access Protocol (LDAP)

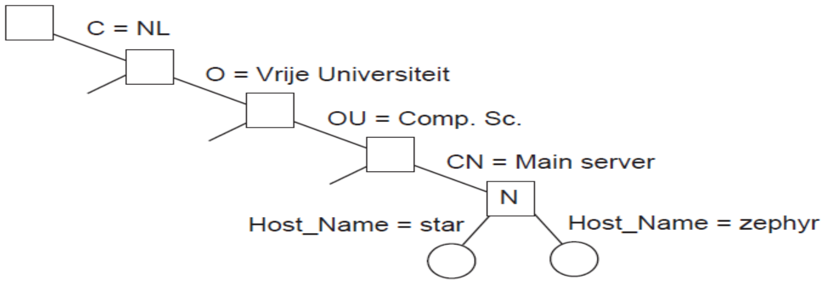
- LDAP directory service consists of a number of records called "directory entries"
  - Each record is made of (attribute, value) pairs
  - LDAP standard specifies five attributes for each record
- Directory Information Base (DIB) is a collection of all directory entries
  - Each record in a DIB is unique
  - Each record is represented by a distinguished name

```
E.g., /C=NL/O=Vrije Universiteit/OU=Comp. Sc.
```

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

Directory Information Tree in LDAP

- All the records in the DIB can be organized into a hierarchical tree called *Directory Information Tree (DIT)*



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

- LDAP provides advanced search mechanisms based on attributes by traversing the DIT
- Example syntax for searching all Main_Servers in Vrije Universiteit:

```
search("&(C = NL) (O = Vrije Universiteit) (OU = *) (CN = Main server)")
```

Drawbacks of distributed index

Quite a few

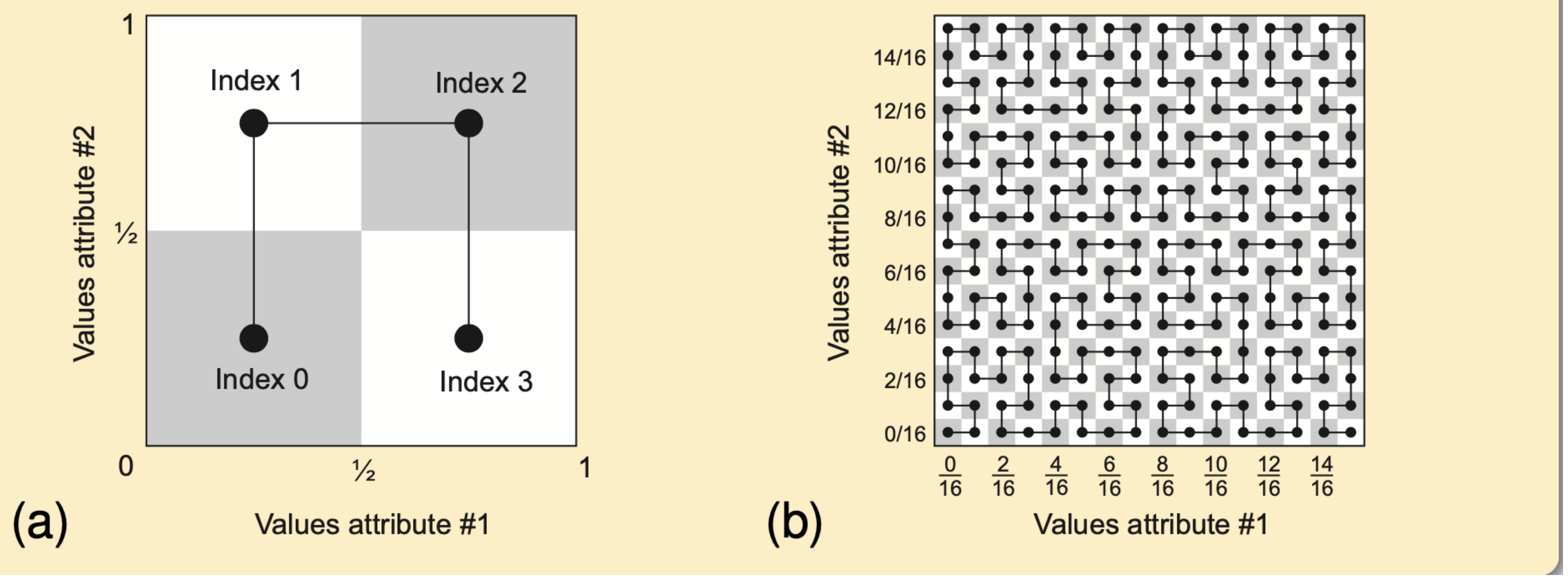
- A query involving *k* attributes requires contacting *k* servers
- Imagine looking up "*lastName = Smith* ∧ *firstName = Pheriby* ": the client may need to process many files as there are so many people named "Smith."
- No (easy) support for range queries, such as "*price* = [*1000 − 2500*]."

Alternative: map all attributes to 1 dimension and then index

Space-filling curves: principle

- Map the *N*-dimensional space covered by the *N* attributes {*a1*,...,*aN*} into a single dimension
- Hashing values in order to distribute the 1-dimensional space among index servers.

Hilbert space-filling curve of (a) order 1, and (b) order 4



Summary

- Naming and name resolutions enable accessing entities in a distributed system

- Three types of naming:
  - Flat Naming
    - Broadcasting, forward pointers, home-based approaches, Distributed Hash Tables (DHTs)
  - Structured Naming
    - Organizes names into Name Spaces
    - Distributed Name Spaces
  - Attribute-based Naming
    - Entities are looked up using their attributes

Next Class

- Concurrency and Synchronization
  - Explain the need for synchronization

  - Analyze how computers synchronize their clocks and concurrent accesses to resources
    - Clock Synchronization Algorithms
    - Mutual Exclusion Algorithms