CSE 5306
Distributed Systems

Naming

Jia Rao

http://ranger.uta.edu/~jrao/

Naming

- Names play a critical role in all computer systems
  - To access resources, uniquely identify entities, or refer to locations
- To access an entity, you have to resolve the name and find the entity
  - Name resolution
- In a distributed system, the naming system itself is implemented across multiple machines
  - Efficiency and scalability are the keys

Addresses

- To access an entity, we need the access point, which is a special entity
  - ✓ The name of an access point is an address
- An entity may have multiple access points, and its access point may change
  - ✓ The address of an access point should not be used to name the entity
  - ✓ E.g., each person has multiple phone numbers to reach him/her, and these numbers may be re-assigned to another person
- Therefore, what we need is a name for an entity that is independent from its addresses
  - ✓ i.e., a location-independent name

True Identifiers

- Are the names that are used to uniquely identify an entity in a distributed system
- True identifiers have the following property
  - ✓ Each identifier refers to at most one entity
  - ✓ Each entity referred to by at most one identifier
  - ✓ An identifier always refers to the same entity (no identifier reuse)
- A simple comparison of two identifiers is sufficient to test if they refer to the same entity

Issues of Naming

- How to resolve names and identifiers to addresses
- A naming system maintains a name-to-address binding in the form of mapping table
    - ✓ A centralized table in a large network is not scalable
- The name resolution as well as the table is often distributed across multiple machines

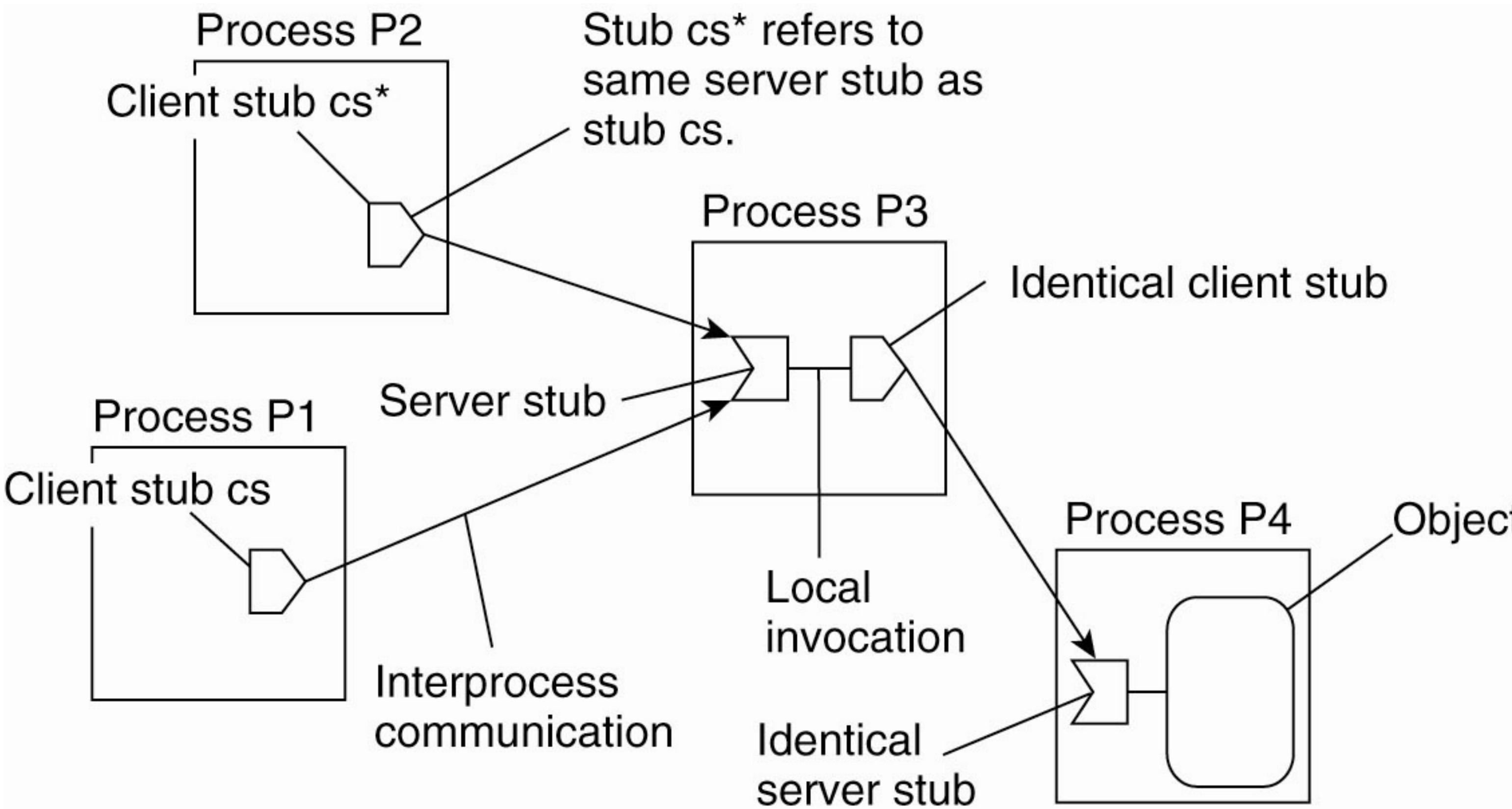
Flat Names

- An identifier is often a string of random bits
  - ✓ Does not contain any information on how to locate the access point of its associated entity
- Two simple solutions to locate the entity given an identifier
  - ✓ Broadcasting and multicasting (e.g., ARP)
    - Broadcasting is expensive, multicast is not well supported
  - ✓ Forwarding pointers
    - When an entity moves, it leaves a pointer to where it went
    - A popular approach to locate mobile entities

Forwarding Pointers

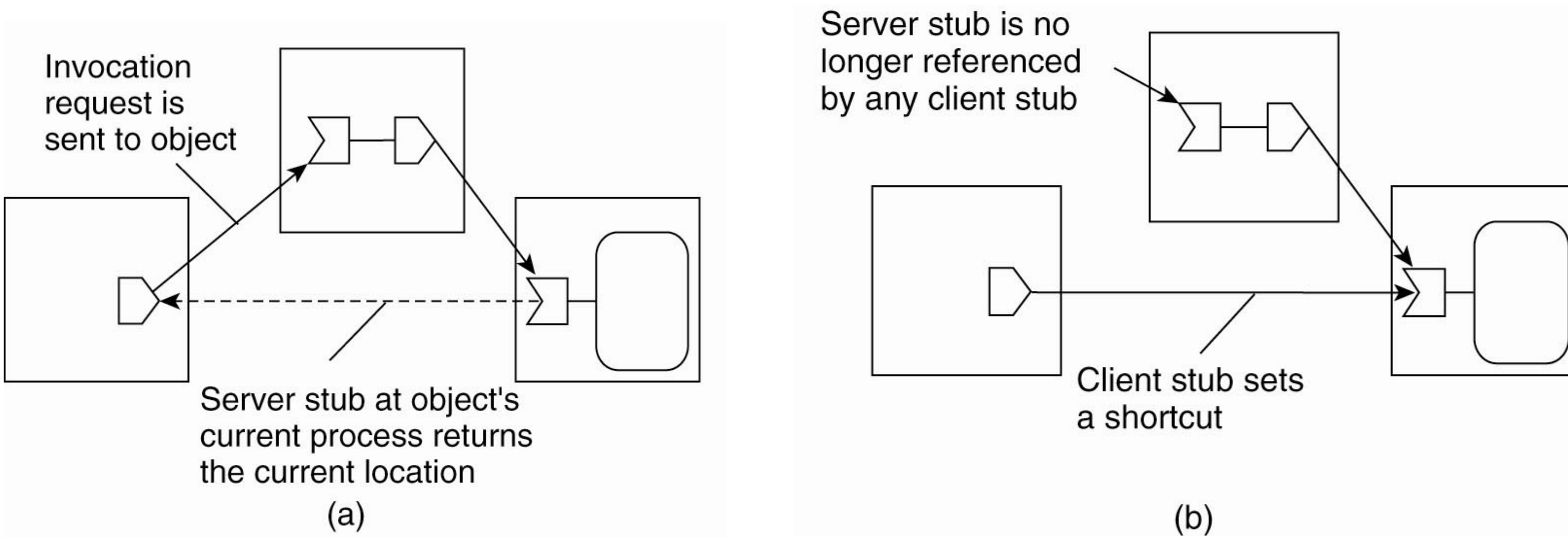
- Advantage:
  - ✓ Dereferencing can be made transparent to client – follow the pointer chain
- Geographical scalability problems:
  - ✓ Chain can be very long for highly mobile entities
  - ✓ Long chains not fault tolerant
  - ✓ High latency when dereferencing
- Need chain reduction mechanisms
  - ✓ Update client's reference when the most recent location is found

Forwarding via Client-Server Stubs

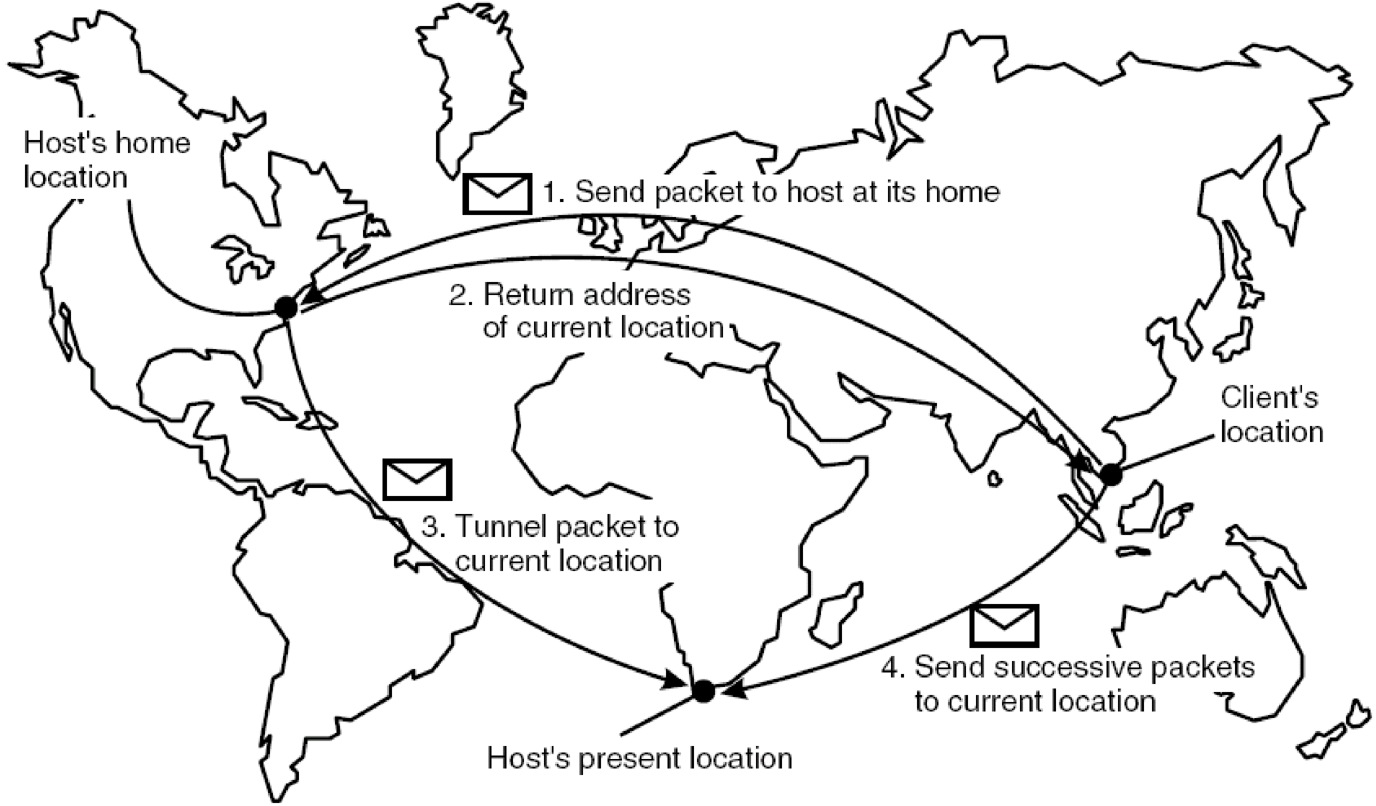


The principle of forwarding pointers
using (client stub, server stub) pairs.

Chain Reduction via Shortcuts



Home-based Approaches



The principle of Mobile IP.

Issues with Home-based approaches

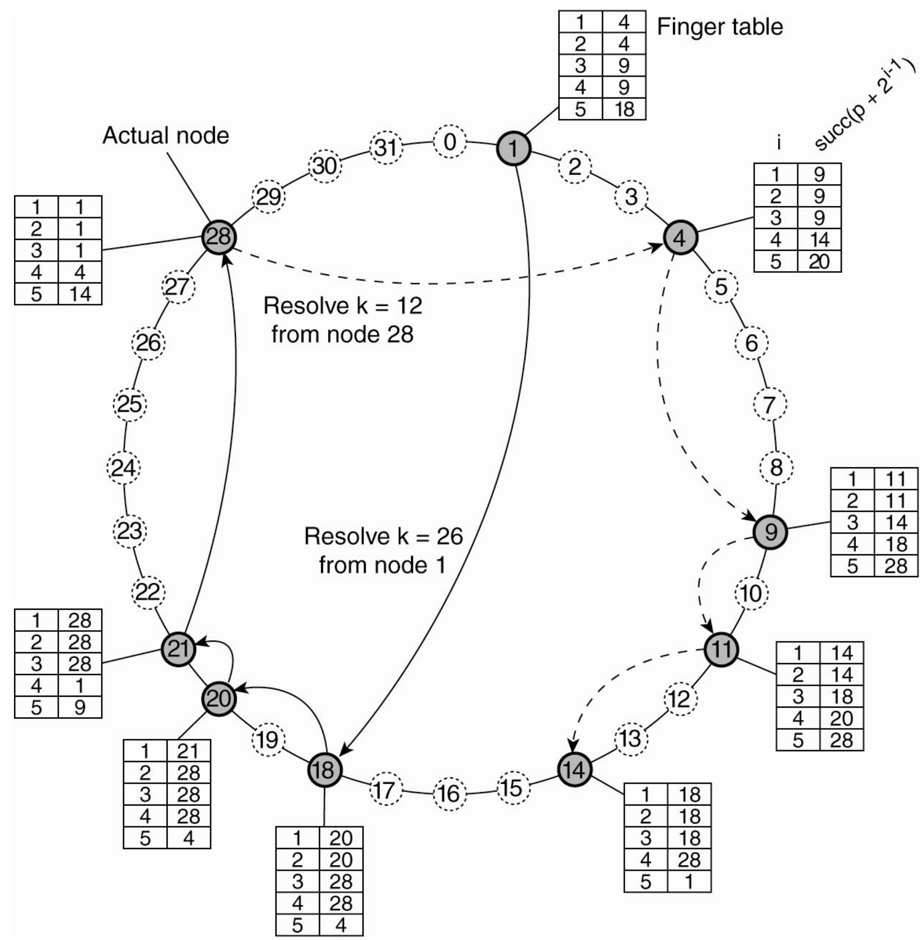
- Home address has to be supported as long as entity lives
- Home address is fixed – unnecessary burden if entity permanently moves
- Poor geographical scalability

Distributed Hash Table

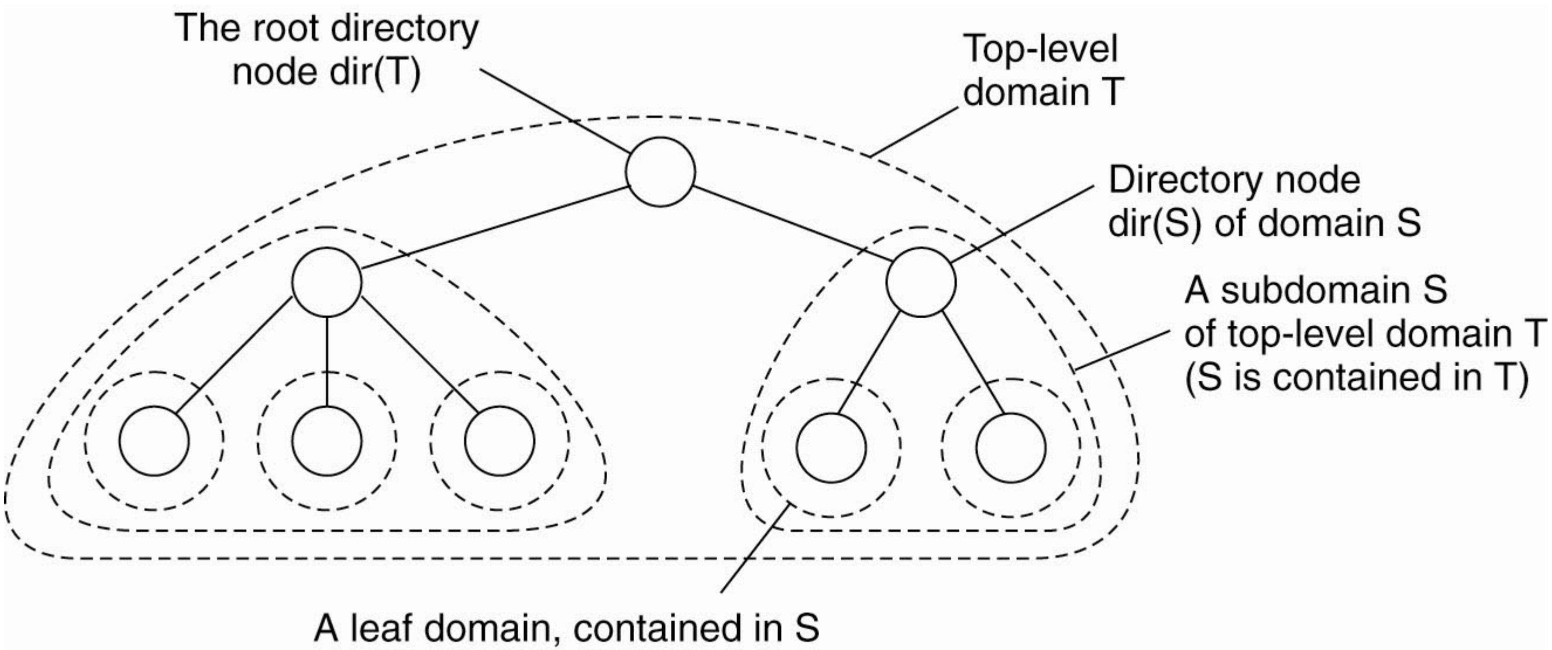
- Review of DHT-based Chord system
  - ✓ Each node has an m-bit random identifier
  - ✓ Each entity has an m-bit random key
  - ✓ An entity with key k is located on a node with the smallest identifier
    - That satisfies id >=k, denoted as succ(k)
- The major task is key lookup
  - ✓ i.e., to resolve an m-bit key to the address of succ(k)
  - ✓ Two approaches: linear approach and finger table
- The simplest form of chord does not consider network proximity

Key Lookup in Chord

Resolving key 26 from node 1 and key 12 from node 28 in a Chord system.

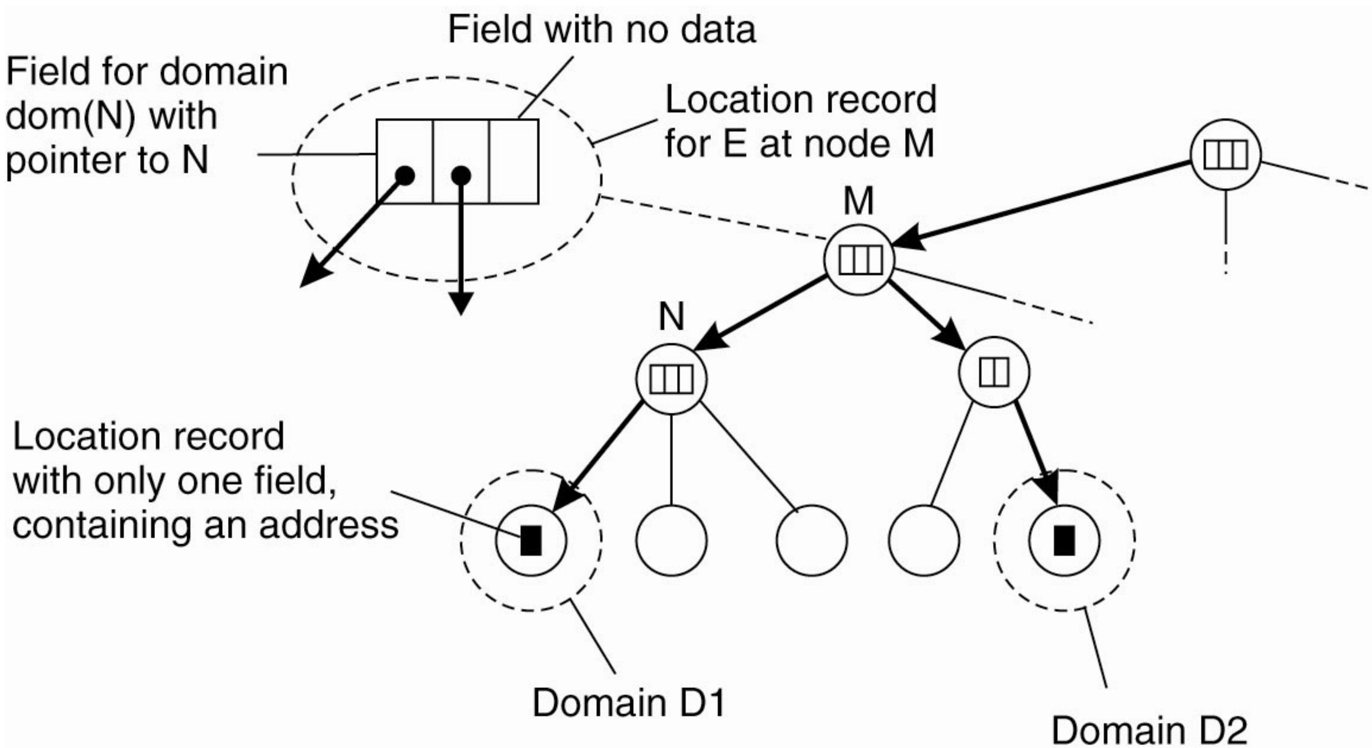


Hierarchical Approaches (1/3)



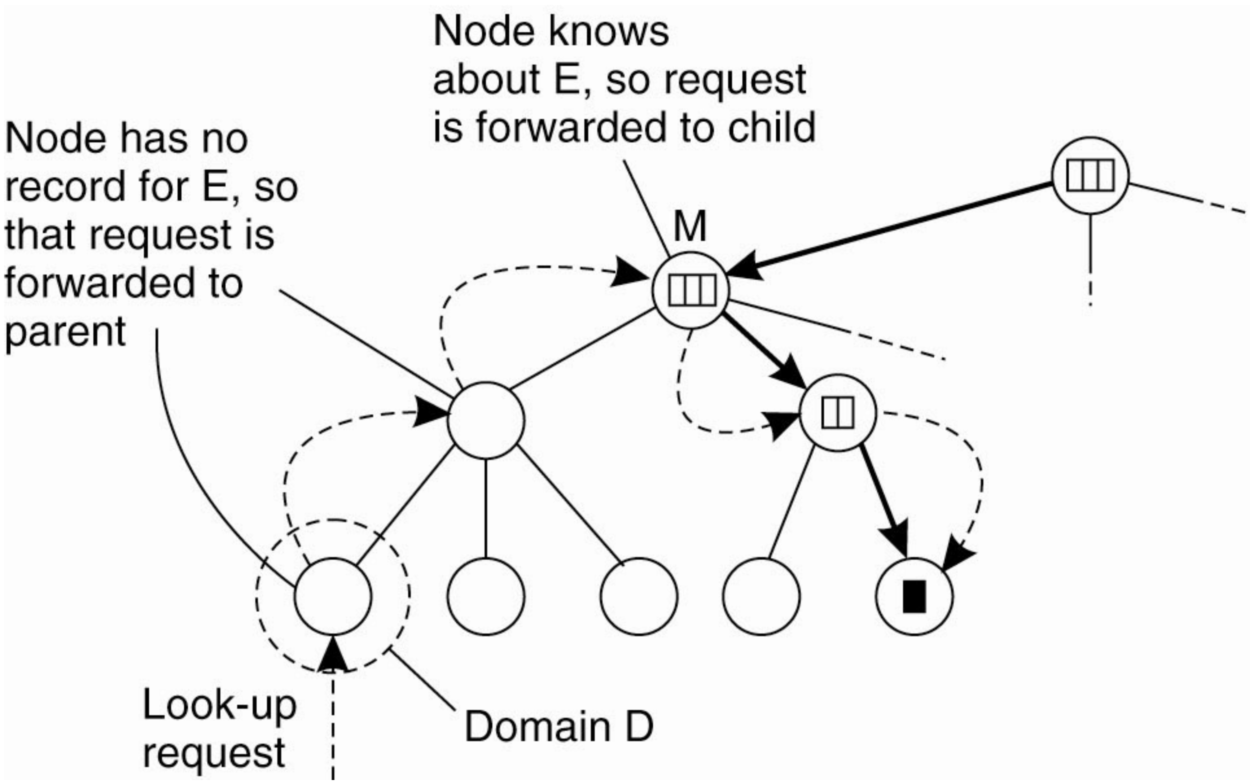
Hierarchical organization of a location service into domains, each having an associated directory node.

Hierarchical Approaches (2/3)



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

Hierarchical Approaches (3/3)

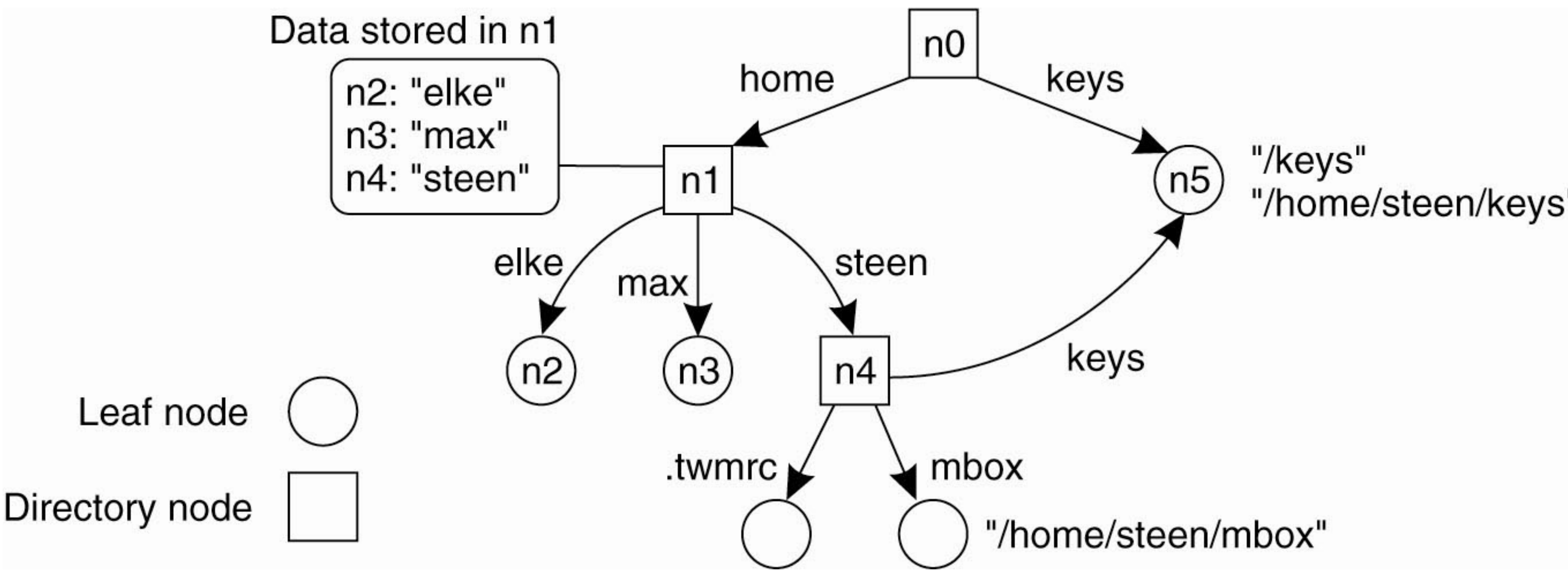


Looking up a location in a hierarchically
organized location service.

Structured Naming

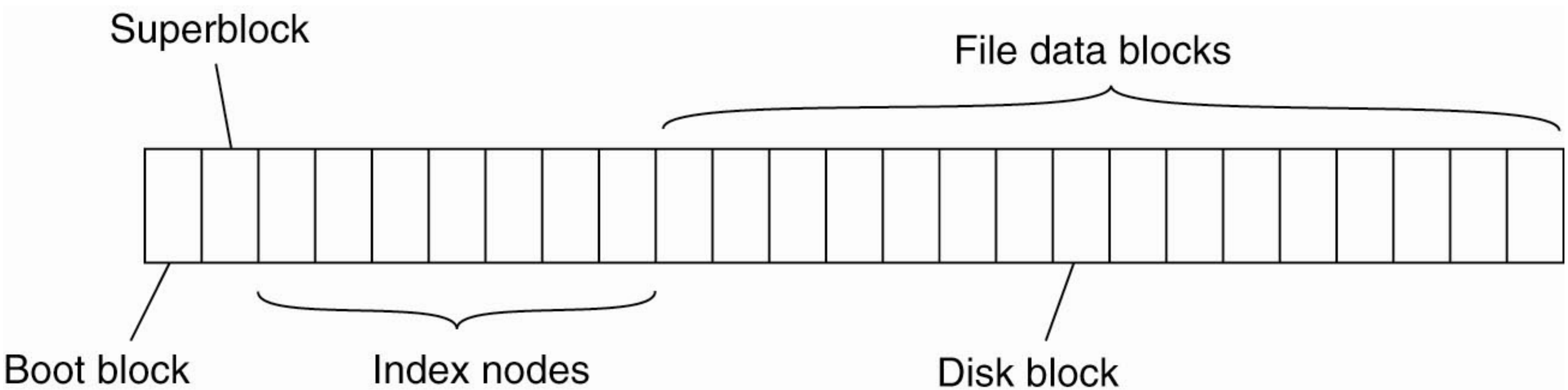
- Flat names are not convenient for humans to use
- As a result, naming systems often support structured names that
  - ✓ Are composed from simple, human-readable names, e.g., file names, Internet domain names
- Structured names are often organized into what is called a name space
  - ✓ A labeled, directed graph with two types of nodes, leaf node and directory node

Name Space



A general naming graph with a single root node.

UNIX File Systems

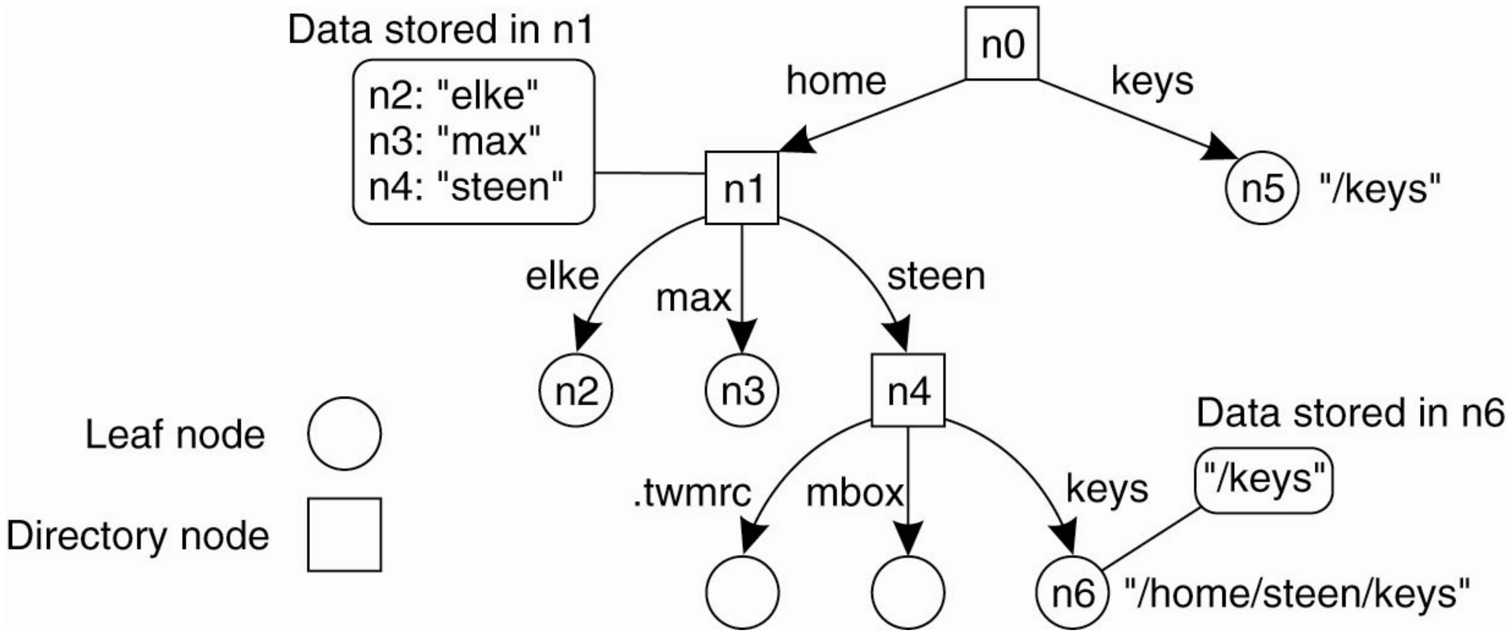


The general organization of the UNIX file system implementation on a logical disk of contiguous disk blocks.

Name Resolution

- The process of looking up a name in a name space
- Name resolution can take place only if we know where and how to start
  - ✓ A closure mechanism, e.g., starting from a well known root directory, or start from home
- Linking
  - ✓ Aliases are commonly used in a name space
  - ✓ An alias can be a hard link or a symbolic link

Symbolic Link

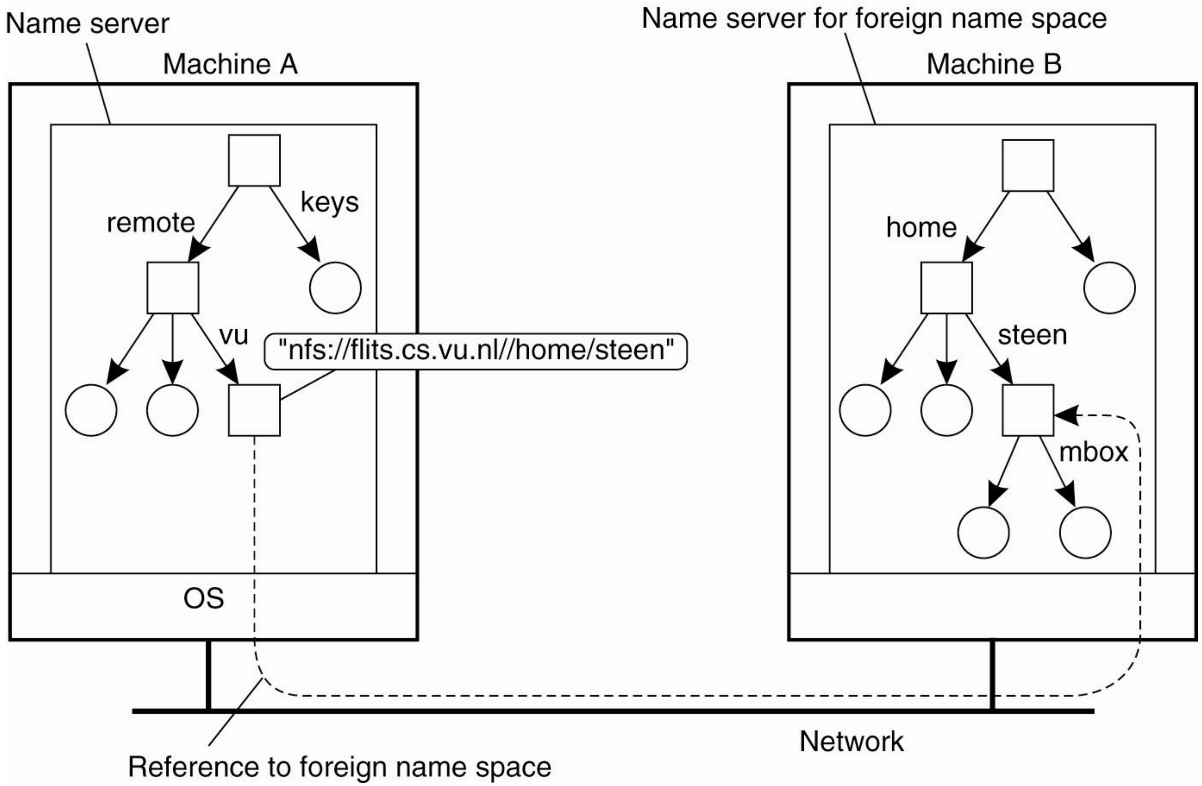


The concept of a symbolic link explained in a naming graph.

Mounting (1/2)

- The process of merging different name spaces
- A common approach is to
  - ✓ Let a directory node (mount point) store the identifier of a directory node (mounting point) from the foreign name space
- Information required to mount a foreign name space in a distributed system
  - ✓ The name of an access protocol
  - ✓ The name of the server
  - ✓ The name of the mounting point in the foreign name space

Mounting (2/2)

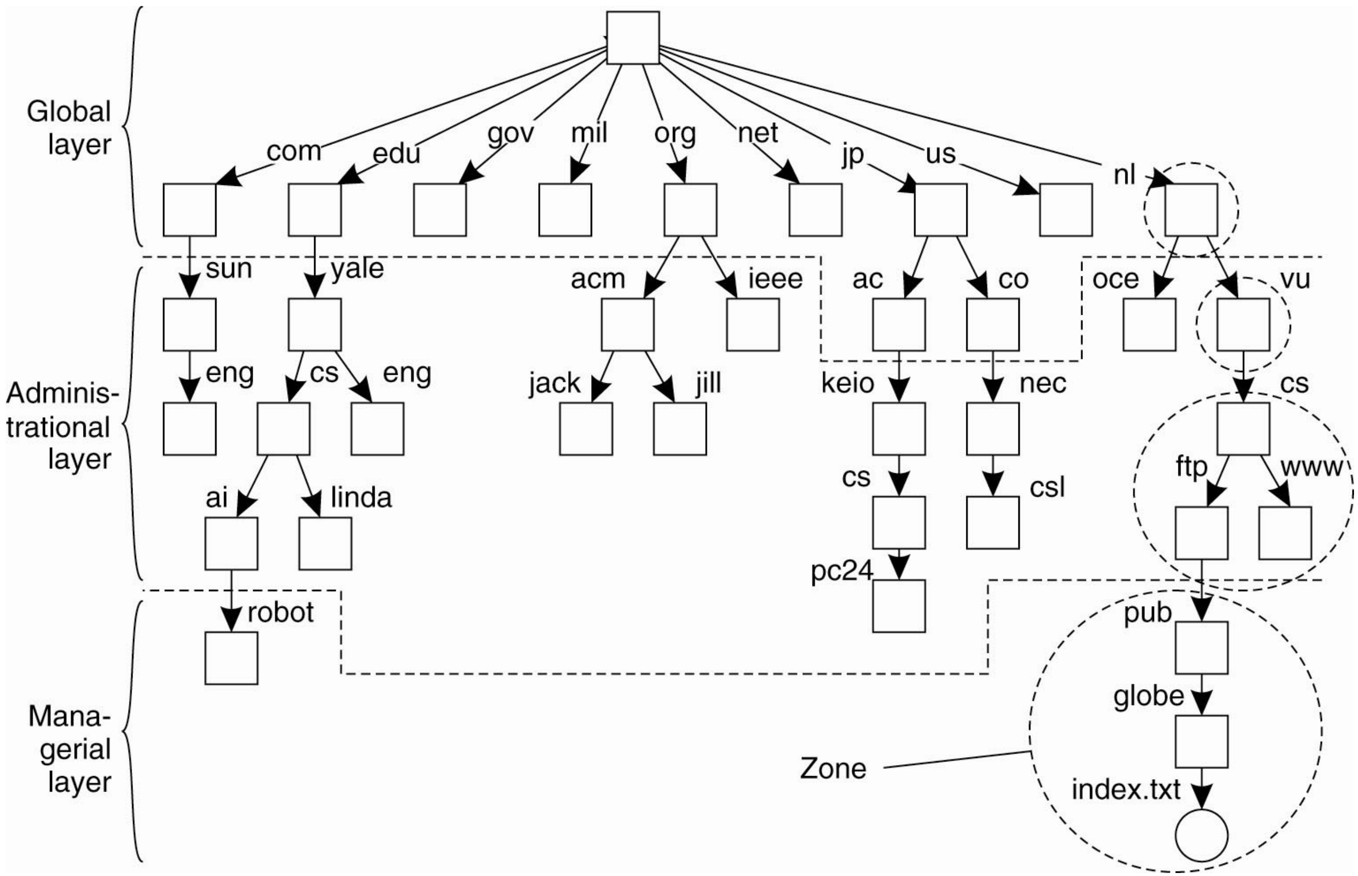


Mounting remote name spaces
through a specific access protocol.

Implementation of a Name Space

- A name space is often implemented by name servers
  - ✓ In LAN, a single name server is enough
  - ✓ In large-scale systems, the implementation of a name space is often distributed over multiple name servers
- A name space for large-scale distributed systems is often organized hierarchically
  - ✓ Global layer
    - Often stable, represents organizations of groups of organizations
  - ✓ Administrational layer
    - Represents groups of entities in a single organization
  - ✓ Managerial layer
    - Nodes often change frequently, e.g., hosts in a local network
    - May be managed by system administrators or end users

Name Space Distribution (1/2)



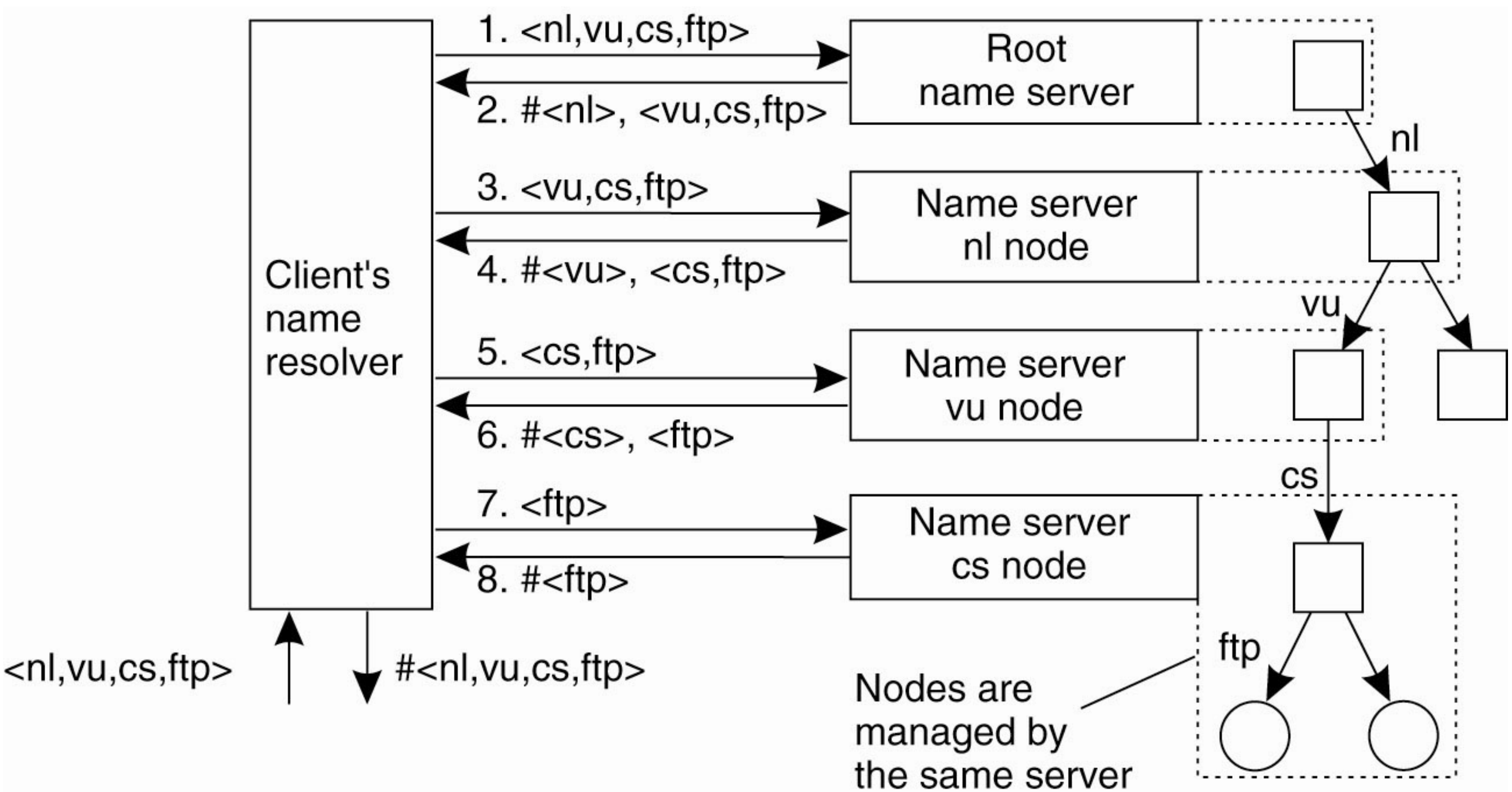
An example partitioning of the DNS name space, including Internet-accessible files, into three layers.

Name Space Distribution (2/2)

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

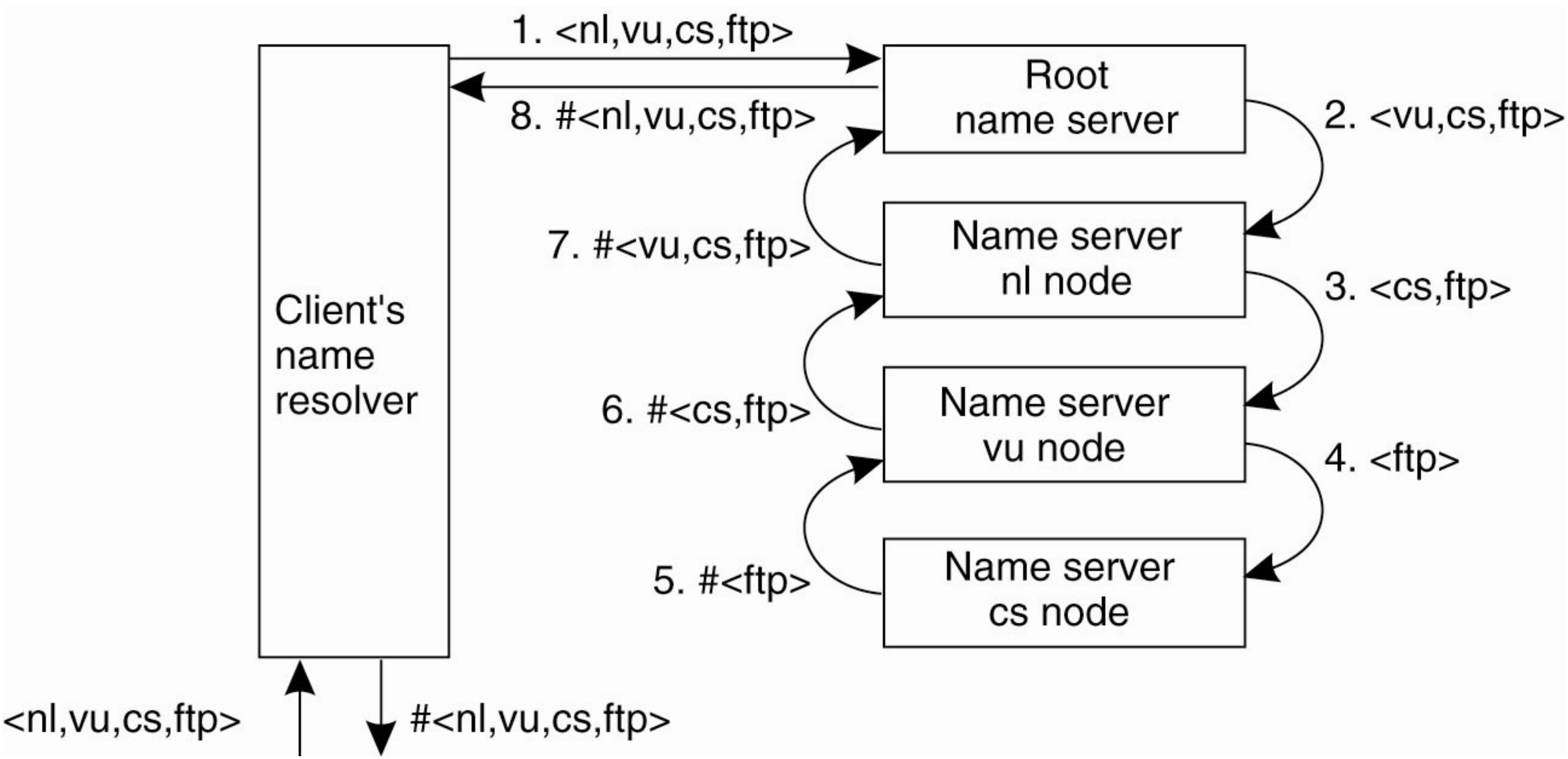
A comparison between name servers for implementing nodes from a large-scale name space partitioned into a global layer, an administrational layer, and a managerial layer.

Implementing Name Resolution (1/2)



The principle of iterative name resolution.

Implementing Name Resolution (2/2)



The principle of recursive name resolution.

Recursive v.s. Iterative

- Recursive resolution demands more on each name server
- However, it has two advantages
  - ✓ Caching is more effective than iterative name resolution
    - Intermediate nodes can cache the result
    - With iterative solution, only the client can cache
  - ✓ Overall communication cost can be reduced

Example: The Domain Name System

- The DNS name space is organized as a root tree
- Each node in this tree stores a collection of resource recodes

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

Decentralized DNS Implementation

- In standard hierarchical DNS implementation, higher-level nodes receives more requests than low-level nodes
  - ✓ Leading to a scalability problem
- Fully decentralized solution can avoid such scalability problem
  - ✓ Map DNS names to keys and look them up in a distributed hash table
  - ✓ The problem is that we lose the structure of the original names and make some operations difficult

Attribute-based Naming

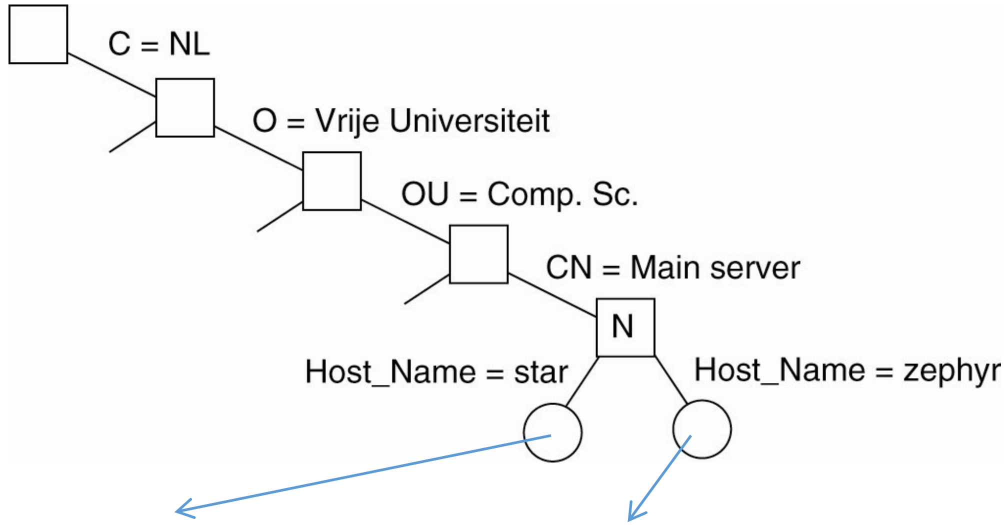
- As more information being made available, it becomes important to
  - ✓ Locate entities based on merely a description of that is needed
- Attribute-based naming
  - ✓ Each entity is associated with a collection of attributes
  - ✓ The naming system provides one of multiple entities that matches a user's description
- Attribute-based naming systems are often known as directory services

Hierarchical Implementation LDAP

Attribute	Abbr.	Value
Country	C	NL
Locality	L	Amsterdam
Organization	O	Vrije Universiteit
OrganizationalUnit	OU	Comp. Sc.
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

A simple example of an LDAP
directory entry using LDAP naming conventions.

Directory Information Tree (DIT)

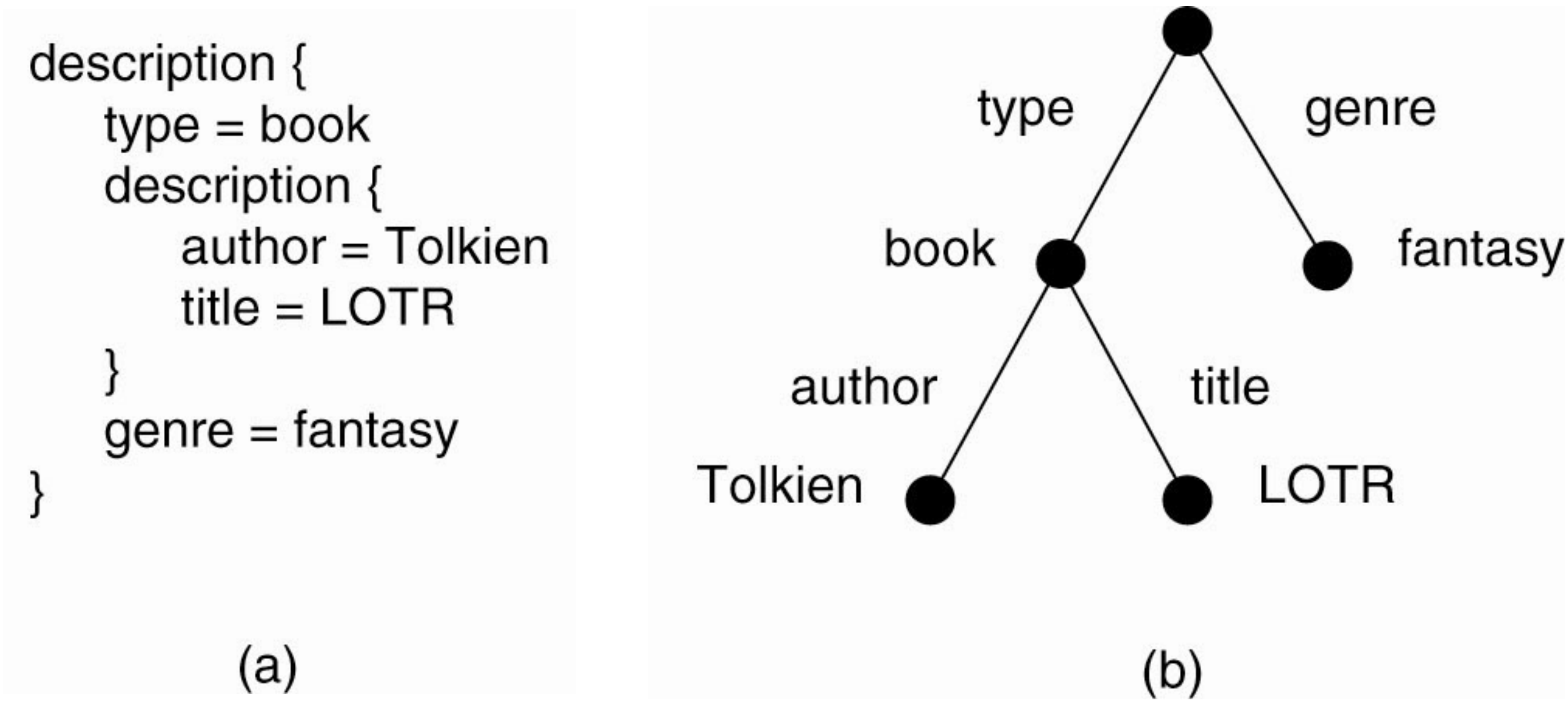


Attribute	Value
Country	NL
Locality	Amsterdam
Organization	Vrije Universiteit
OrganizationalUnit	Comp. Sc.
CommonName	Main server
Host_Name	star
Host_Address	192.31.231.42

Attribute	Value
Country	NL
Locality	Amsterdam
Organization	Vrije Universiteit
OrganizationalUnit	Comp. Sc.
CommonName	Main server
Host_Name	zephyr
Host_Address	137.37.20.10

(b)

Decentralized (DHT) Implementation



- Each path in attribute-value tree (AVT) produces a hash value and mapped to a DHT
  - ✓ h1=hash(type-book), h2=hash(type-book-author) …

Ranged Query in DHT Implementation

- Two phase approach
- Separate the name and the attribute in computing the hash value
  - ✓ Phase 1: distribute attribute names in DHT
  - ✓ Phase 2: for each name, partition the values into subranges and assign a single server for each subrange
- Drawbacks
  - ✓ Updates may need to be sent to multiple servers
  - ✓ Load balancing between different subrange servers

Semantic Overlay Networks

- Construct an overlay network where each pair of neighbors are semantically proximal neighbors
  - ✓ i.e., they have similar resources