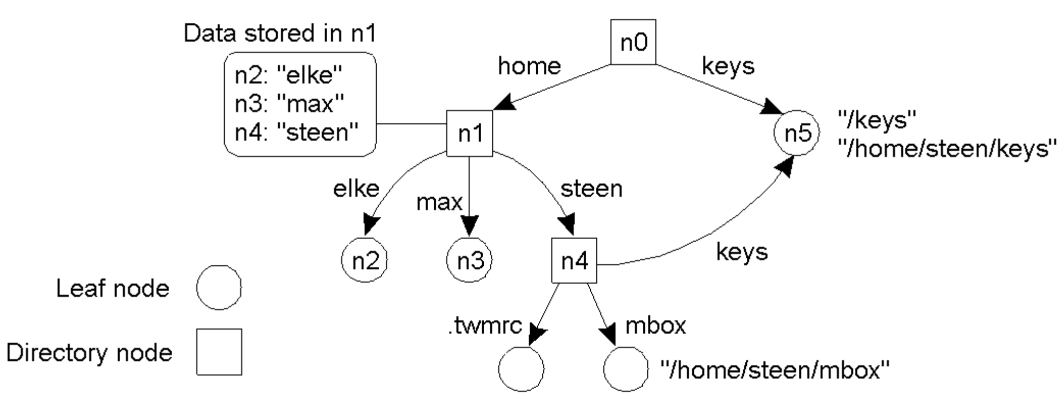
Today: Naming

- Names are used to share resources, uniquely identify entities and refer to locations
- Need to map from name to the entity it refers to
  - E.g., Browser access to www.cnn.com
  - Use name resolution
- Differences in naming in distributed and non-distributed systems
  - Distributed systems: naming systems is itself distributed
- How to name mobile entities?

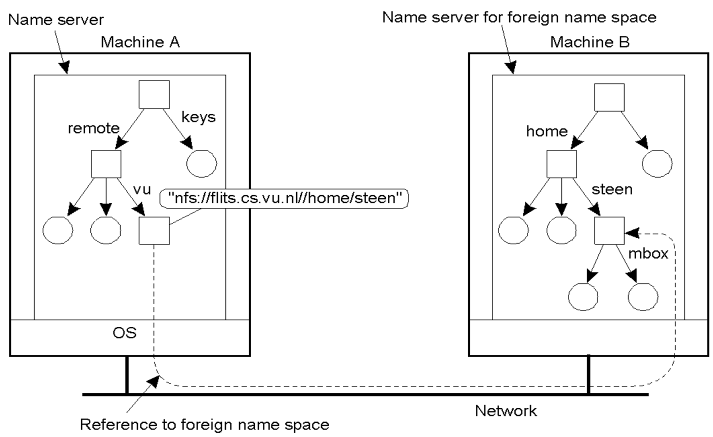
Example: File Names

- Hierarchical directory structure (DAG)
  - Each file name is a unique path in the DAG
  - Resolution of */home/steen/mbox* a traversal of the DAG
- File names are *human-friendly*



Resolving File Names across Machines

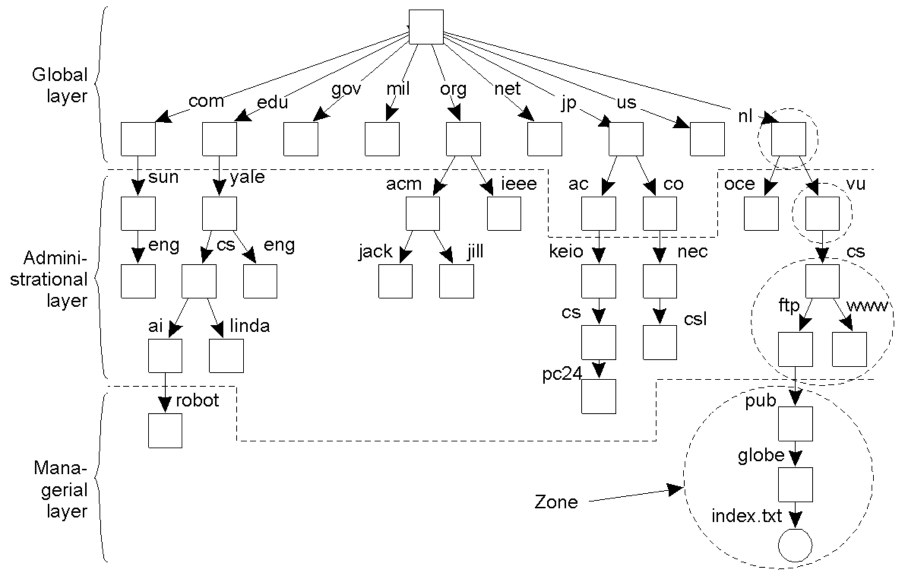
- Remote files are accessed using a node name, path name
- NFS mount protocol: map a remote node onto local DAG
  – Remote files are accessed using local names! *(location independence)*
  – OS maintains a mount table with the mappings



Name Space Distribution

- Naming in large distributed systems
  – System may be global in scope (e.g., Internet, WWW)
- Name space is organized hierarchically
  – Single root node (like naming files)
- Name space is distributed and has three logical layers
  – Global layer: highest level nodes (root and a few children)
    - Represent groups of organizations, rare changes
  – Administrational layer: nodes managed by a single organization
    - Typically one node per department, infrequent changes
  – Managerial layer: actual nodes
    - Frequent changes
  – Zone: part of the name space managed by a separate name server

Name Space Distribution Example



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

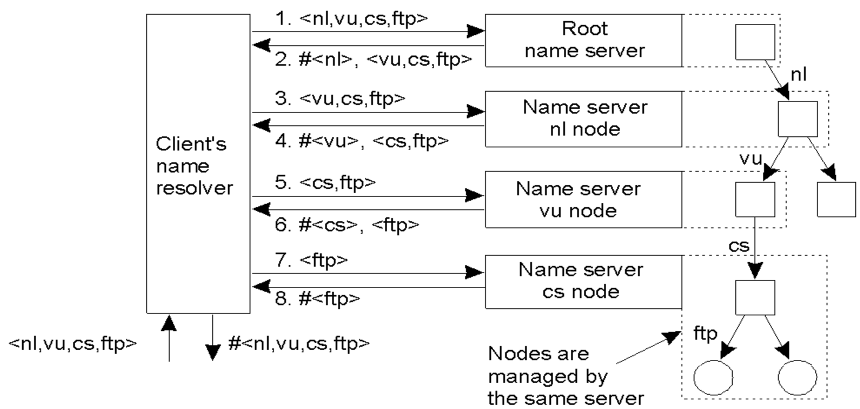
Name Space Distribution

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.
- The more stable a layer, the longer are the lookups valid (and can be cached longer)

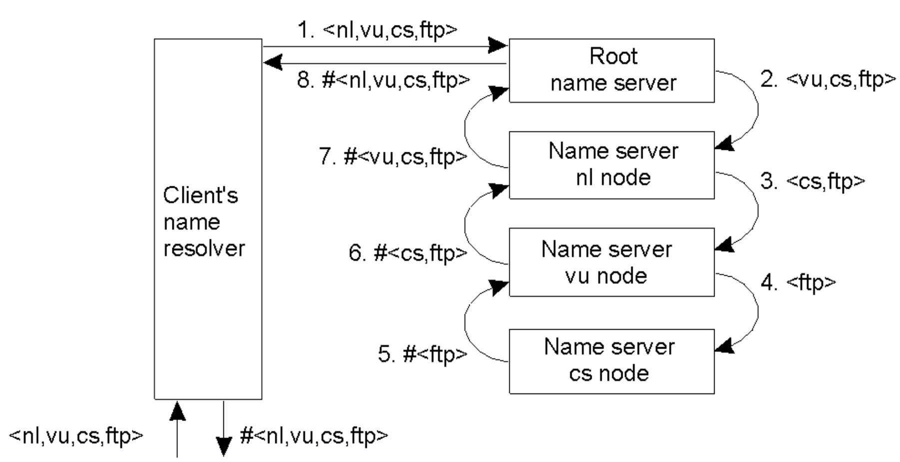
Implementing Name Resolution

- Iterative name resolution
  - Start with the root
  - Each layer resolves as much as it can and returns address of next name server



Recursive Name Resolution

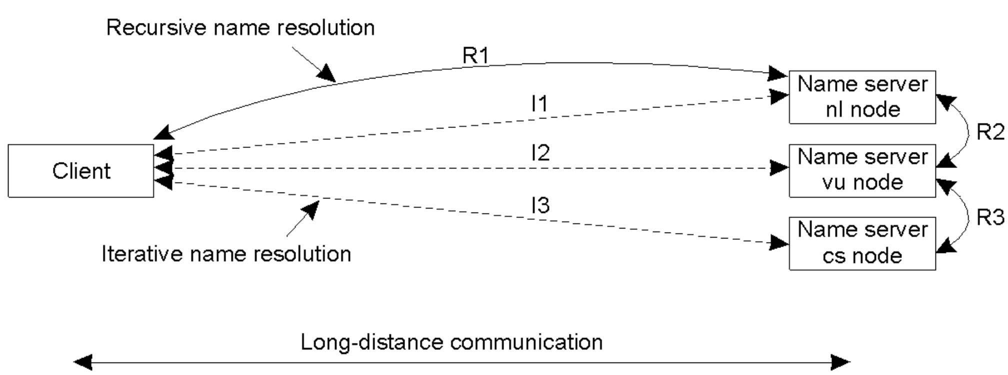
- Recursive name resolution
  - Start at the root
  - Each layer resolves as much as it can and hands the rest to the next layer



Which is better?

- Recursive name resolution puts heavy burden on gobal layer nodes
  - Burden is heavy => typically support only iterative resolution
- Advantages of recursive name resolution
  - Caching possible at name servers (gradually learn about others)
    - Caching improves performance
    - Use time-to-live values to impose limits on caching duration
    - Results from higher layers can be cached for longer periods
    - Iterative: only caching at client possible

Communication costs



- The comparison between recursive and iterative name resolution with respect to communication costs
  - Recursive may be cheaper

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.

DNS Implementation

- 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

X.500 Directory Service

- OSI Standard
- Directory service: special kind of naming service where:
  - Clients can lookup entities based on attributes instead of full name
  - Real-world example: Yellow pages: look for a plumber

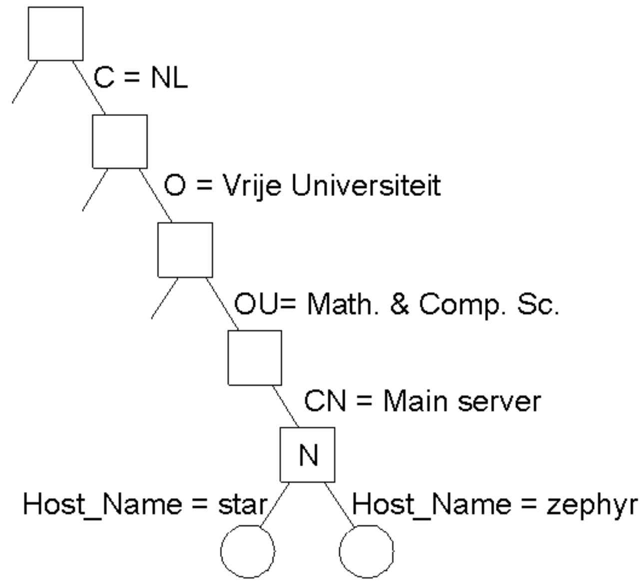
The X.500 Name Space (1)

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 (2)

- Part of the directory information tree.

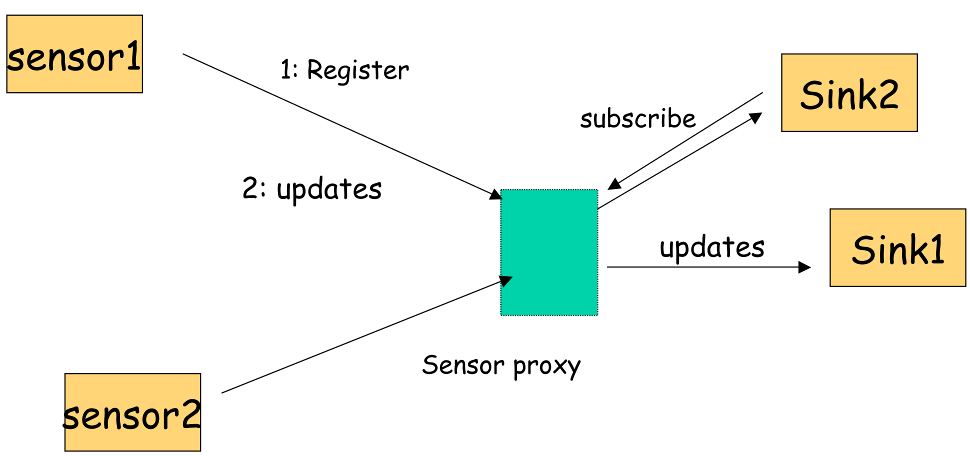


LDAP

- Lightweight Directory Access Protocol (LDAP)
  - X.500 too complex for many applications
  - LDAP: Simplified version of X.500
  - Widely used for Internet services
  - Application-level protocol, uses TCP
  - Lookups and updates can use strings instead of OSI encoding
  - Use master servers and replicas servers for performance improvements
  - Example LDAP implementations:
    - Active Directory (Windows 2000)
    - Novell Directory services
    - iPlanet directory services (Netscape)
    - Typical uses: user profiles, access privileges, network resources

Project 1

- Illustrate distributed systems principles using sensor systems/sensor networks
- Sources: a network of sensors that periodically produce new data
- Sinks: consumers of sensor data that periodically need updates
- Sensor proxies:
  - Sensors post updates to proxies
  - Sinks subscribe to one or more sources
  - Proxies disseminate data to sinks
    - Use a publish-subscribe paradigm for data dissemination



Project 1 details

- Proxy should be multi-threaded to service arbitrary number of sources and sinks
  - Sources, sinks and the proxy can reside on different machines
- Proxy should employ synchronization
  - Proxies may process data from multiple sources and disseminate it to a sink
    - Example: disseminate sum(sensor1,sensor2) -> source 1