Distributed Systems

10 Name Service

June 10 2009

Gerd Liefländer

System Architecture Group

Schedule of Today

- Implementing a Name Service
  - User friendly structured name
  - Distributed Name Space
  - Name Resolution
- Example Name Services
  - Domain Name Service
  - DEC's Global Name Space
- Attribute Naming
  - Directory Service
    - X 500
    - LDAP

Implementing a Name Service

Name Spaces

- Partitioned
- Replicated

Name Resolution

- Iterative
- Recursive

Example Structured Name

DNS example:

label$_i$.label $_{i-1}$. … .label$_1$

The above pathname consists of labels, each of which is not longer than 63 characters, whereas the complete pathname is limited by 255 characters

Resolutions starts at . (whereby this symbolized root name is often skipped). Further queries are sent to the corresponding name servers for label$_1$, …. label$_i$

label$_1$ can be a country, e.g. uk for United Kingdom

Distributed Name Space

Implementing a Name Server[1]

- In a LAN, the name service can be implemented as a single name server at some node of the DS
  - in order to avoid the single point of failure there might be a backup name server
- In WANs, name service is often distributed over multiple name servers
- *How to organize distributed name servers?*
  - Vertical layering
  - Horizontal partitioning (zoning)

[1]for structured names

Hierarchical Name Space

Name spaces of worldwide system are structured hierarchically, e.g. according to Cheriton

- Global layer
  - Only very few changes
  - e.g. representing a company or university
- Administrational layer
  - Occasionally changes
  - e.g. representing a division or a department
- Managerial layer
  - Regular changes
  - e.g. representing a working group or an institute

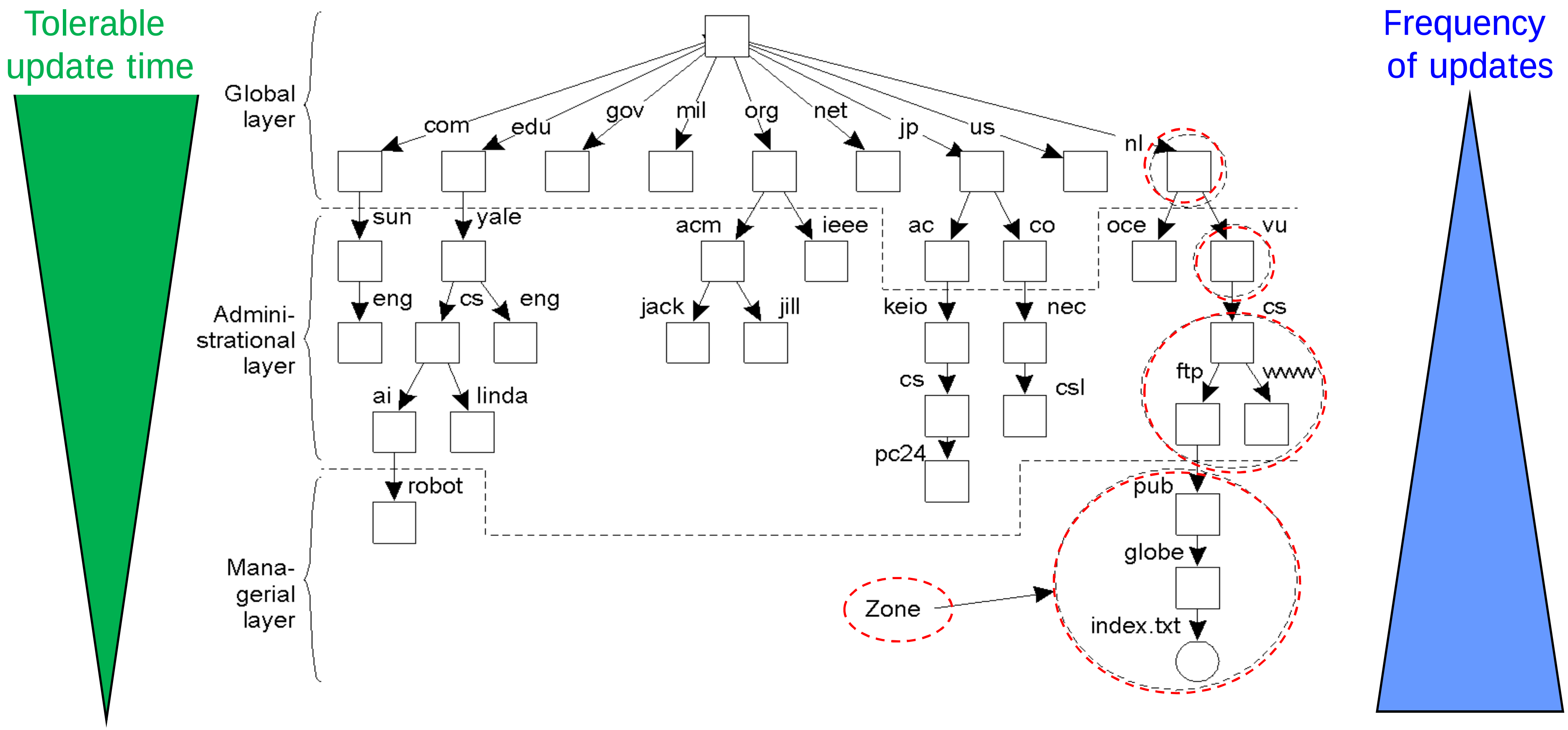
Name Space Distribution (1)

Namespace partitioned into zones:

- Non-overlapping parts of the name space
- Delegated to an authorized organization
- Organization provides servers (potentially replicated) for a zone holding records consisting of at least

  <name, access point>
- Organization has authority over name space portion

Name Space Distribution (2)



- Partitioning of DNS into zones & layers
- Tables in root and its children are relatively stable

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 of name servers from a large-scale name space partitioned into the three layers
- Responsiveness for global layers can be low, because in many cases the needed information is cached at the client

Domains

- Domain is the notion for the administrative authority responsible for a partition of the name space
  - Determines the responsibility for a domain
  - Manages and updates its name data base
- Management may be extended to sub domains
  - Sub domain **`i30www.ira.uka.de`** is relatively autonomous
  - ... but has to be fitted to the department's domain **`www.ira.uka.de`**
- Naming data of different domains should never be put into one data base

Name Resolution

Name Resolution

Name resolution:
Given a structured name,. i.e. path name, it should be possible to look up any information stored in the node (unless access is forbidden)

Possible result:

1. Entity found → **identifier or address**
2. Entity not found → **invalid name**
3. Access not allowed → **access forbidden**

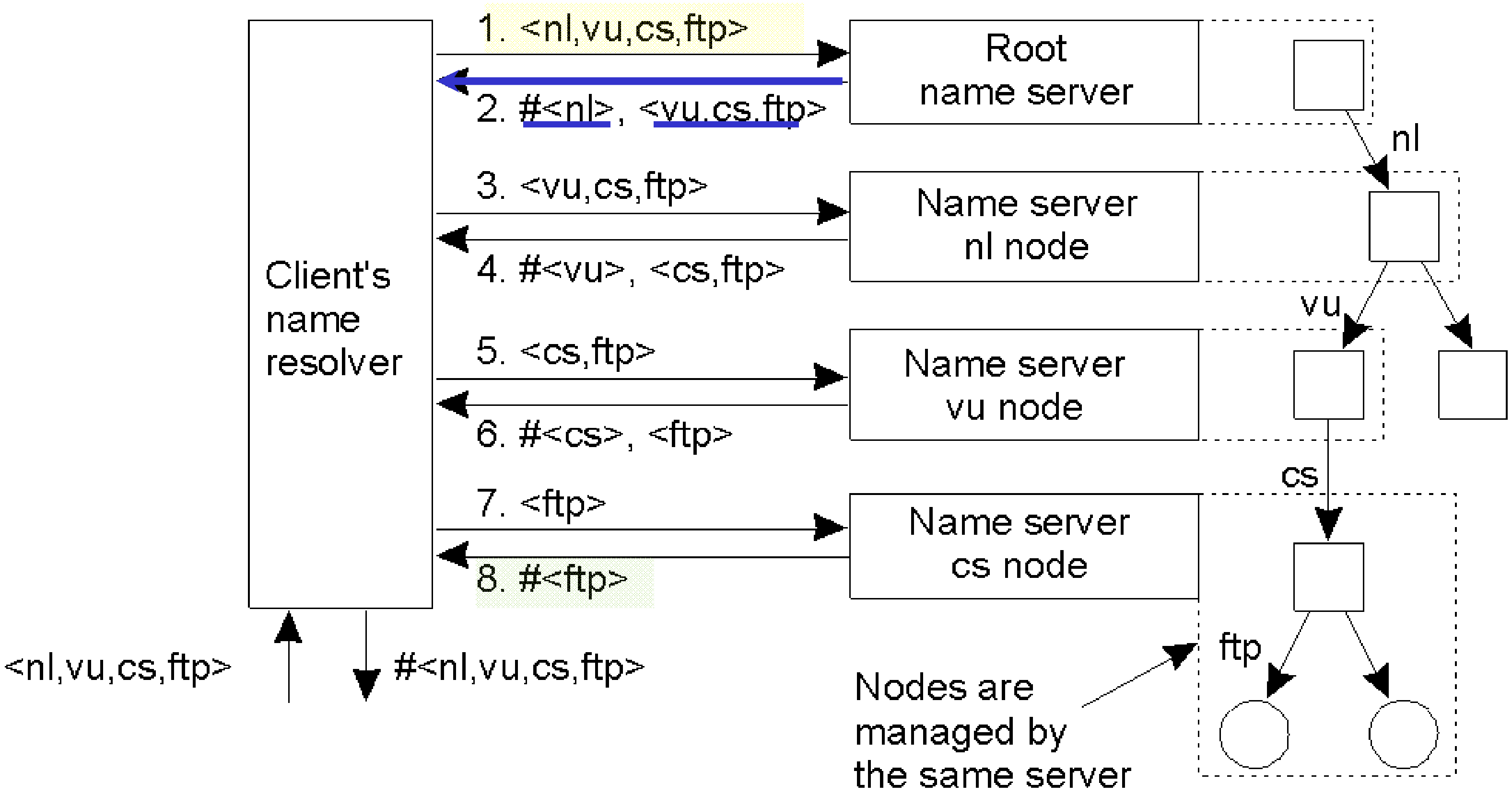
Name Resolution

- Resolving structured names via different name servers is also called navigation

<u>Three orthogonal design parameters</u>

- Resolving method
  - Iterative
  - Recursive
- Site of resolving instance
  - Client site
  - Server site
- With or without caching

Iterative Client Based Navigation



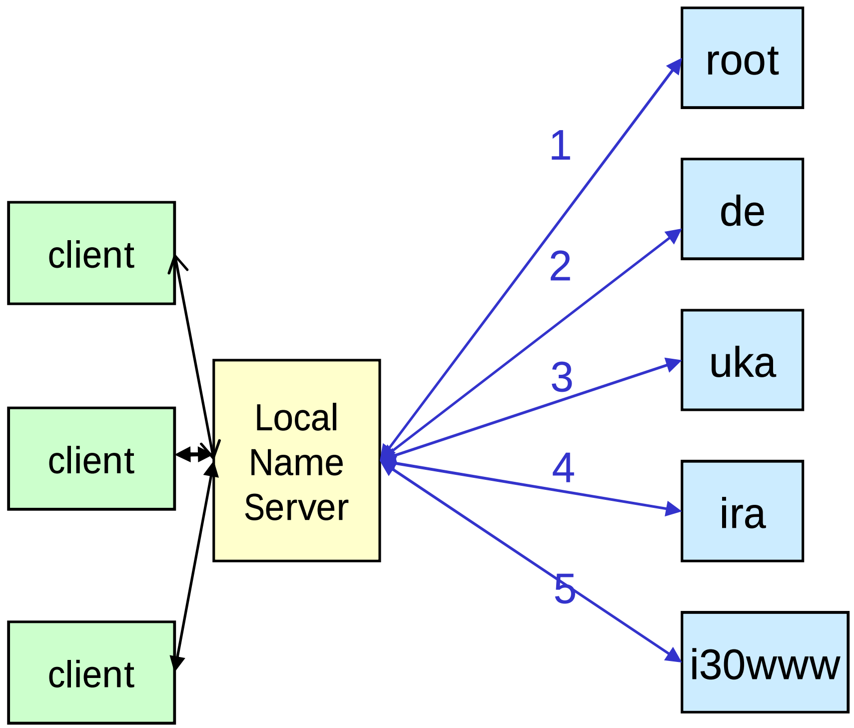
- Principle of client based iterative name resolution resolving
  `root:<nl, vu, cs, ftp, pub, globe, index.txt>`

Iterative Client Based Navigation with Local Name Resolver

Example: i30www.ira.uka.de

- Client initiates resolution

1. de is resolved by a root server, link to “de server” is given back
2. uka is resolved by “de server” + link to “university KA server”
3. ira is resolved ...
4. ...

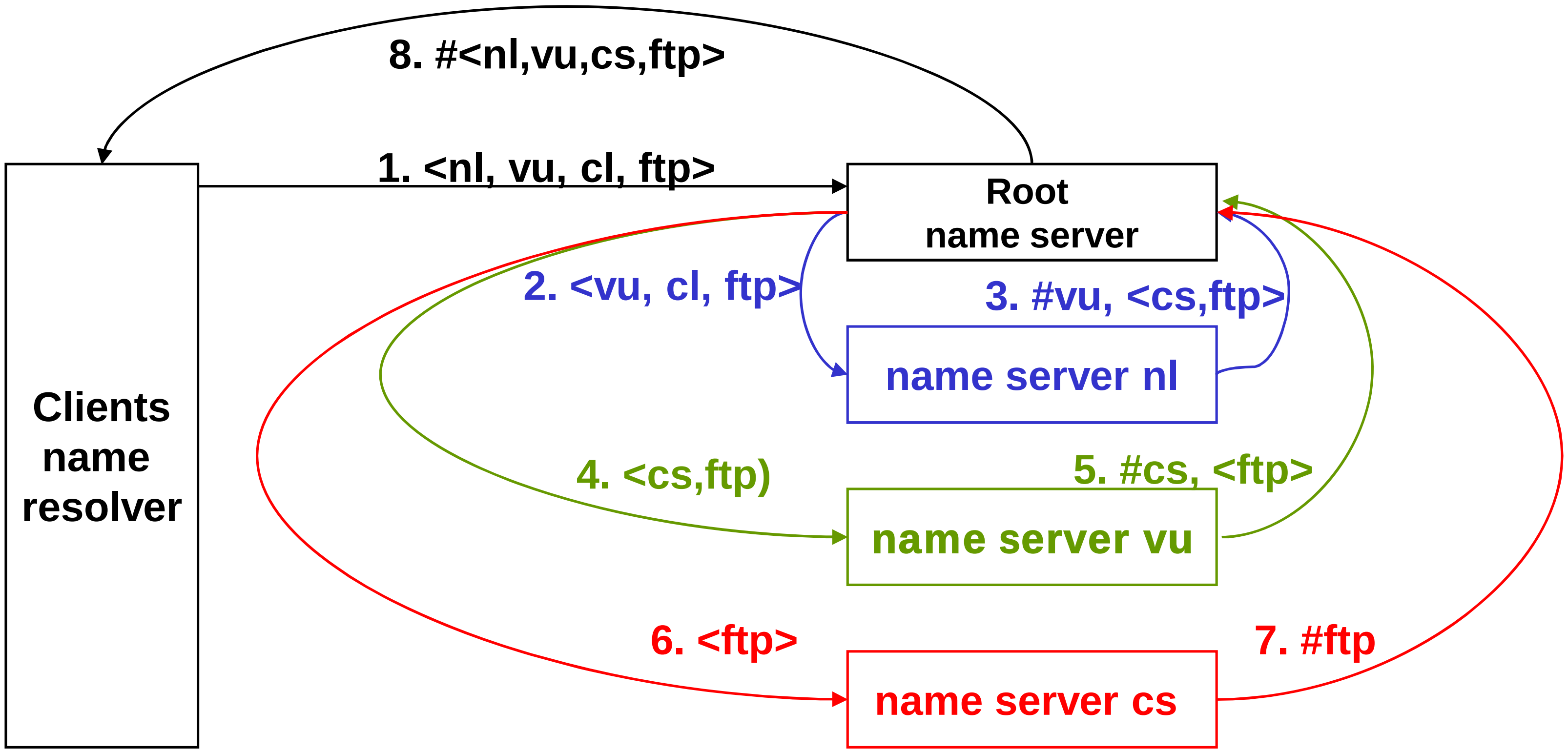


Advantage: Only local server needs to know the root server and it can cache multiple name server addresses

Name Resolution (3)

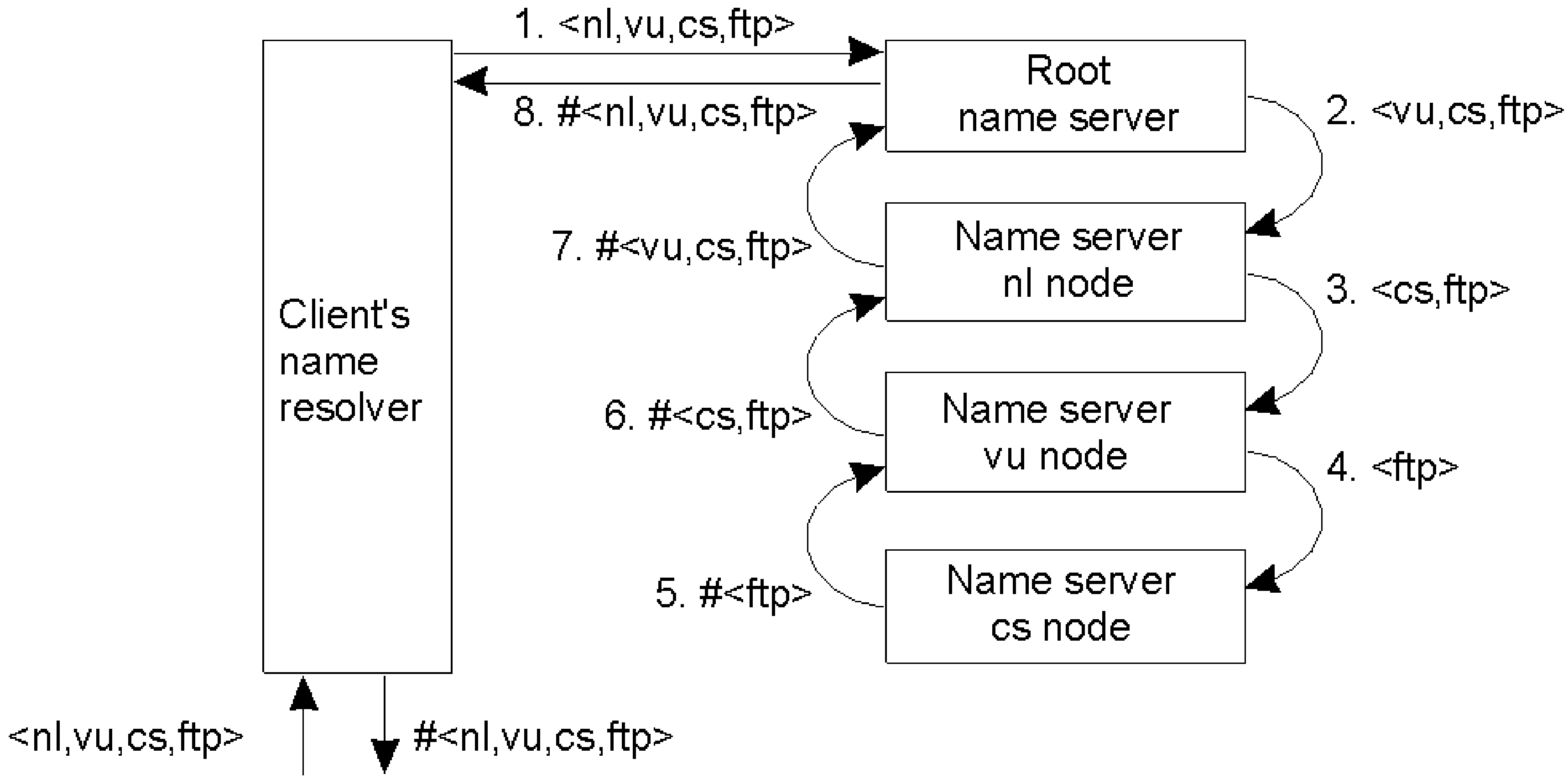
- Server based navigation: name service coordinates name resolving process contacting other name servers collecting results until resolution is completed
  - Iteratively ~ initial name server communicates with other name servers of same level
  - Recursively ~ initial name server requests another intermediate name server of the same level to continue with the name resolution process
    - Each server can cache name resolution results for a while

Sever Based Iterative Name Resolution



- Principle of non recursive server based name resolution
- Initial server must keep complete info until resolution has completed

Recursive Name Resolution

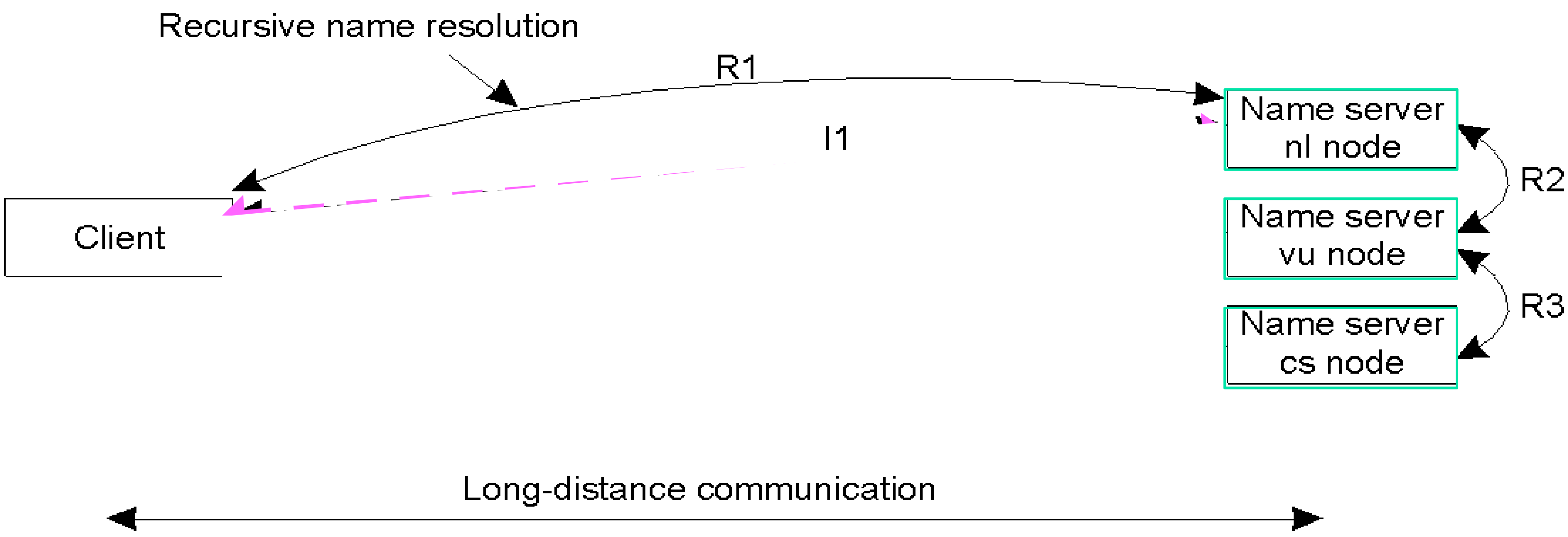


- Principle of server based recursive name resolution

Analysis: Name Resolution

- Each method requires the same number of messages if the name has to be resolved for the first time
- Client based iterative name resolution:
  - Uncomfortable, client site involved in each resolution step
  - Dedicated caching can be done per client, i.e. with a user defined refresh policy
  - At client site, the local name resolver can cache name resolution results for multiple clients
- Server based recursive name resolution:
  - root name server has to control the complete resolution $\Rightarrow$ potential performance bottleneck
  - Caching at server sites can be more efficient when different clients from different client sited request the same info

Tanenbaum’s Analysis



- Both methods need 6 messages, but in the above example
  - iterative with 6 long distance messages
  - recursive with only 2 long distance and 4 short distance messages

Example Name Services

DNS: Domain Name System

GNS: Global Name Space

Goals for DNS

- Implement a wide area distributed DB enabling:
  - Scalability & extensibility
  - Decentralized maintenance
  - Robustness
  - Fault-tolerance
  - Global scope
    - Names mean the same thing everywhere
  - No need for
    - Atomicity
    - Strong consistency

Domain Name System (DNS)*

Primary task:

- Mapping from a symbolic name to 32 bit IP address, e.g.

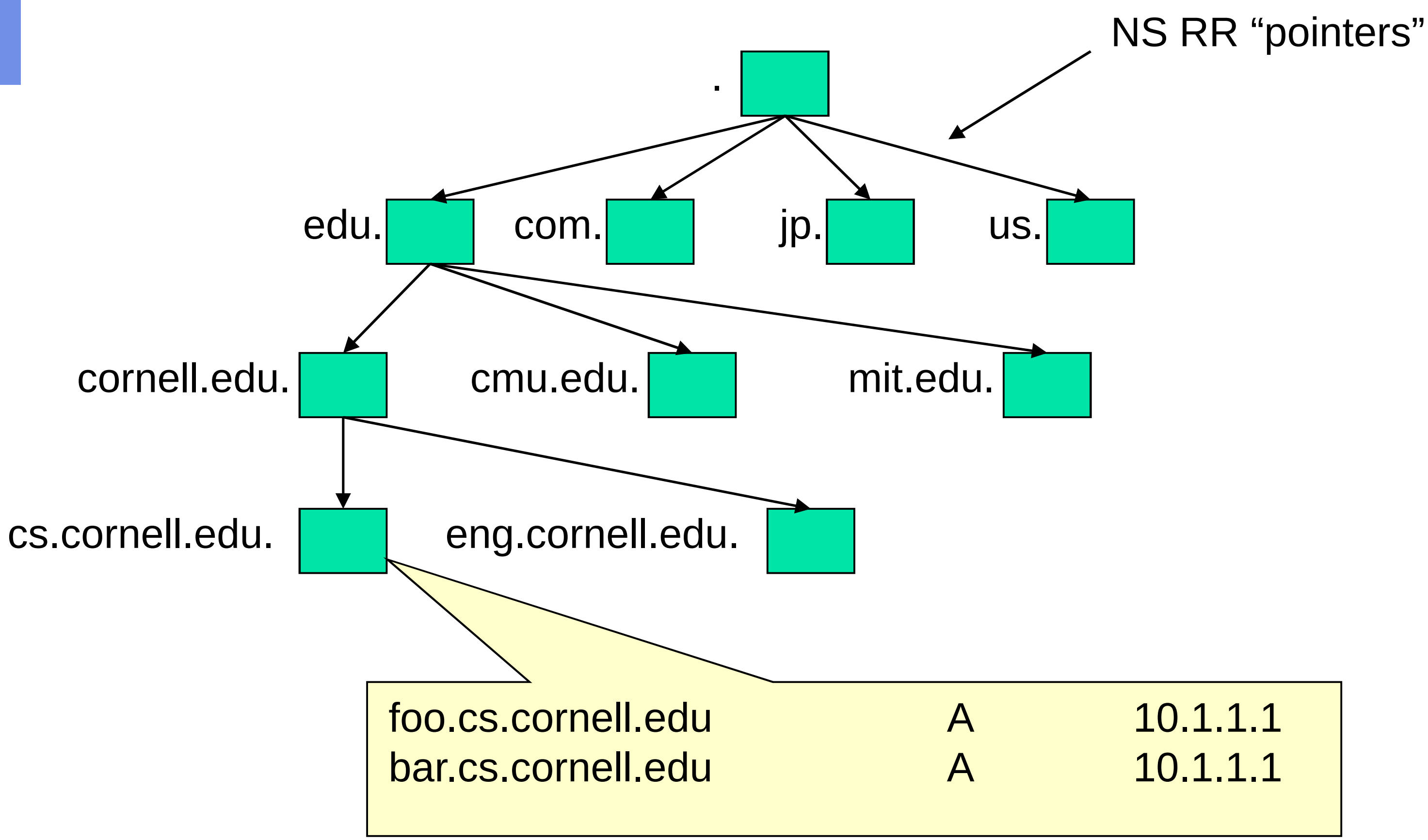
  smtp.uni-karlsruhe.de → 129.13.185.217

- Complete pathnames names are alphanumeric strings ≤ 255 characters of labels ≤ 63 characters, e.g. **ira.uka.de, i30www.ira.uka.de**

- DNS name space is implemented as a rooted tree

*Paul Mockapetris (1984, standard in the Internet since 1987)

Principle: DNS Tree Structure



DNS Names and IP Addresses are

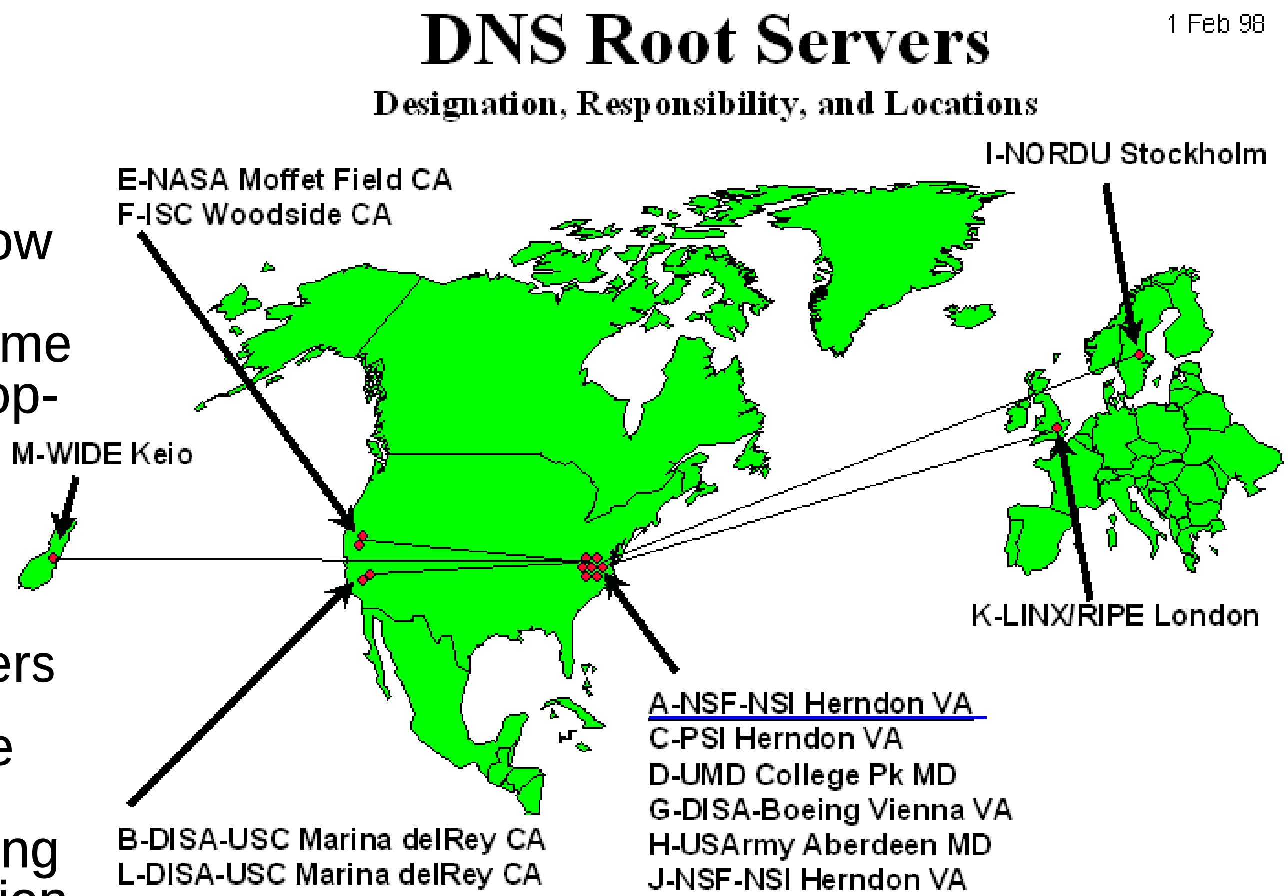
- ... Identifiers and Locators
- Both are typically non-persistent
- Private IP addresses identify only in the context of an IP realm
- Domain names are good identifiers, e.g.
  - **woodstock.cs.cornell.edu** identifies a host
  - **www.cnn.com** identifies a service

Domain Name System (DNS)

- Distributed directory service
- Hierarchical name space
- Each level separated by '.'
  - Analogous to '/' separator in file systems
- One global root
  - Started with 13 replicated root servers (A,B,...M), only
  - Root server A in Dulles in Virginia
  - There have been Denial of Service (DoS) attacks on these root servers, none of them really successful
  - Because of intensive caching, queries to root servers are quite rare

DNS Root Name Servers (1998)

- The root name servers know how to find the authoritative name servers for all top-level zones.
- 1998 ∃ only 13 root name servers
- Root servers are critical for the proper functioning of name resolution



Map of DNS Root Name Servers

- Map of DNS Root Name Servers (Feb. 2007 currently 123 RNS)
- Up to 13 ORSN DNS server in Europe

DNS is simple but powerful

- Only one type of query
  - Query(domain name, RR type)
    - Resource Record (RR) type is like an attribute type
  - Answer(values, additional RRs)
- Limited number of RR types
- Hard to make new RR types
  - Not for technical reasons...
  - Rather because each requires global agreement

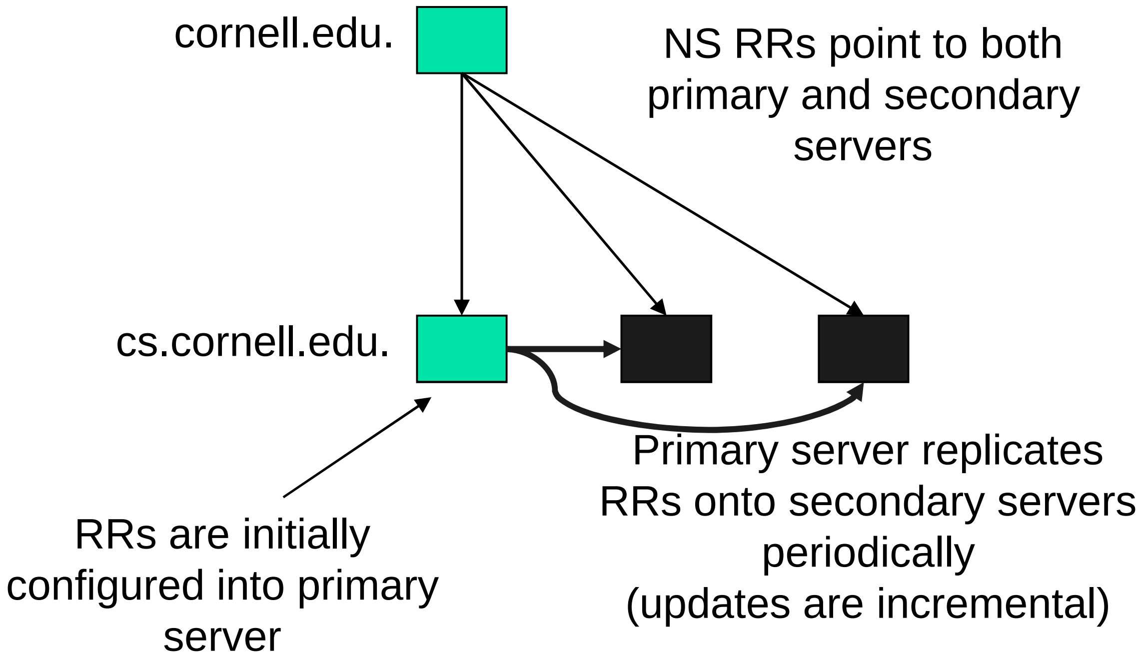
DNS is the Core of the Internet

- Global name space
  - Can be the core of a naming or identifying scheme
- Global directory service
  - Can resolve a name to nearly every computer on the planet

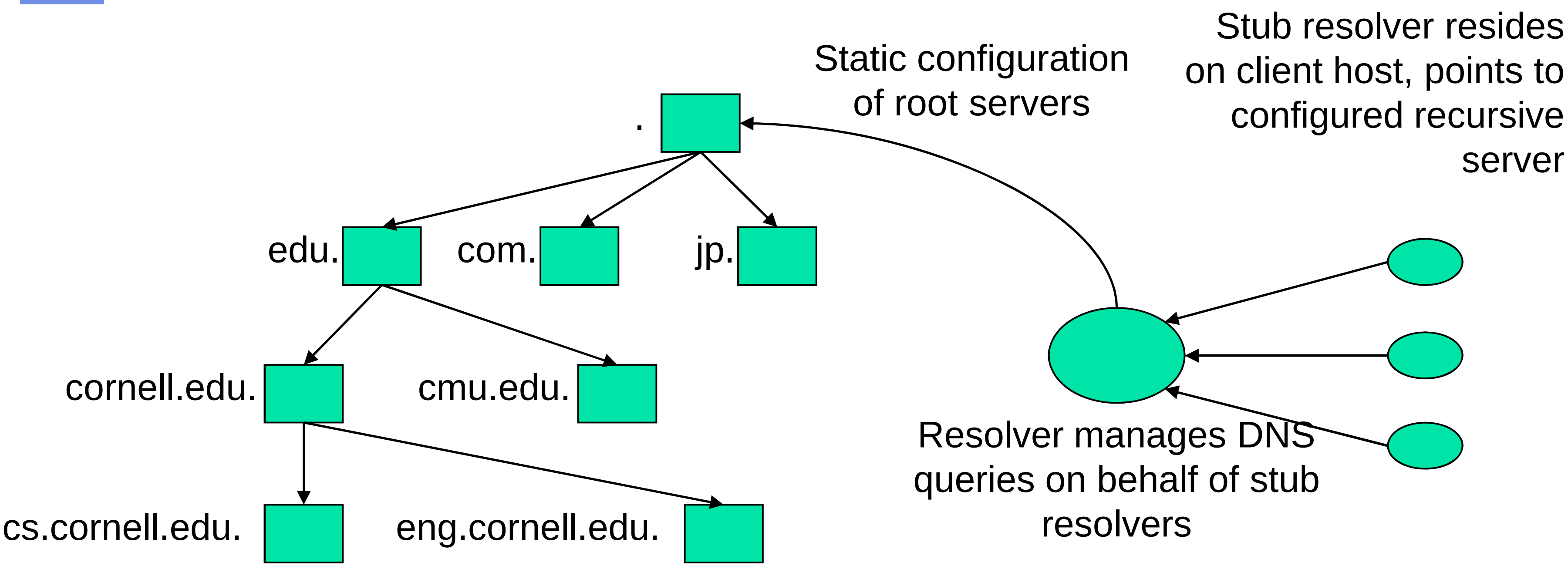
Important DNS RR Types

- **NS**: Points to next Name Server down the tree
- **A**: Contains the IP address
  - **AAAA** for IPv6
- **MX**: Contains the name of the mail server
- Service-oriented RR types
  - **SRV**: Contains addresses and ports of services on servers
    - One way to learn what port number to use
  - **NAPTR**: Essentially a generalized mapping from one name space (i.e. phone numbers) to another (i.e. SIP URL)

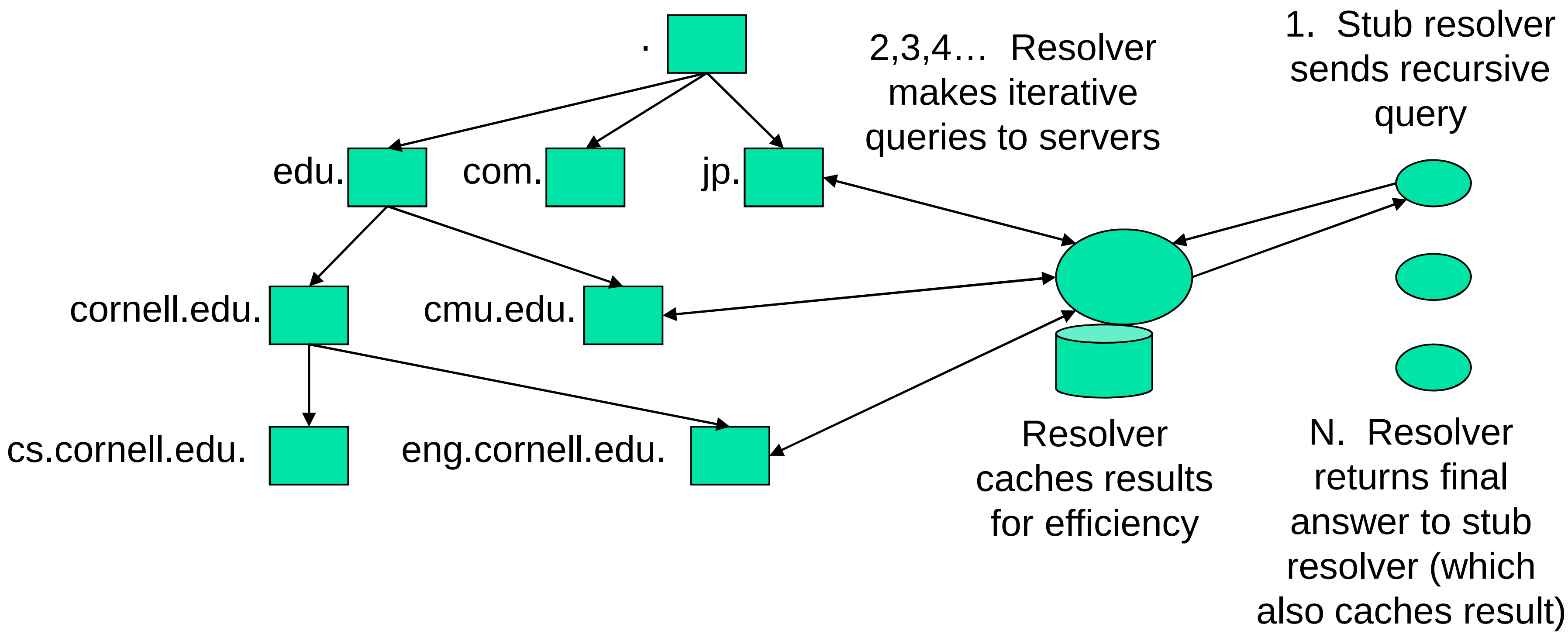
Primary and Secondary Servers



Resolver Structure & Configuration



Resolver Structure & Configuration



DNS Cache Management

- All RRs have Time-to-live (TTL) values
- When TTL expires, cache entries are removed
- NS RRs tend to have long TTLs
  - Cached for a long time
  - Reduces load on higher level servers
- A RRs may have very short TTLs
  - Order one minute for some web services
  - Order one day for typical hosts

Why DNS iterative, not recursive?

- AT/MvS* teach that recursive is more efficient
  - Better caching characteristics
    - Caches in servers, not just resolvers
  - Shorter paths
- However, high-performance recursive server are much harder to implement
  - Maintain state for thousands of concurrent queries
  - Manage cache
- Recursive server prone to DoS attacks

* AT/MvS = Andrew Tanenbaum/Martin van Steen text

URLs, URNs, and URIs

- Uniform Resource <Locator, Name, Identifier>
- URL tells a computer where and how to reach a resource
  - These came first
- URN is a true identifier
  - Unique, persistent
- URI refers to both URLs and URNs
  - Defines syntax for current and future URLs and URNs
- *For now we only really care about URLs*

URL

- Consists of:

<scheme>:<scheme-specific-part>

URL

- Consists of:

<scheme>:<scheme-specific-part>

A protocol

Information the protocol needs

URL Examples

- HTTP (web)
  - http://www.cnn.com/news/story.html
- Email
  - mailto://francis@cs.cornell.edu
- Newsgroups
  - news:cornell/class/cs514
- SIP (Session Initiation Protocol)
  - sip://service@phone.verizon.com

Note the Central Role of DNS

- HTTP (web)
  - http://***www.cnn.com***/news/story.html
- Email
  - mailto://francis@***cs.cornell.edu***
- Newsgroups
  - news:cornell/class/cs514
- SIP (Session Initiation Protocol)
  - sip://service@***phone.verizon.com***

Locating Mobile Entities

- *What is a mobile entity?*
- From naming perspective, it is an entity whose address changes often
- This doesn't require physical mobility!
  - Every time you dial up, you may get a new address
- So, “mobility” existed well before laptops became common
  - Though laptops create more mobility

Is Mobility a Problem for DNS?

- Not really
  - Even though DNS was designed with relatively stable IP addresses in mind
- Because mobility only effects leaf DNS servers
  - Recall: A RR TTL is short, but NS RR TTL is long
- Note: *non-mobile* web server's A RRs often have very short TTLs
  - To allow quick failover to another web server

Is Mobility a Problem at all?

- Less than you'd think
- Most mobile systems are clients; servers are rarely mobile
  - Clients are initiators of connections, not recipients
  - Therefore, there is not a client locating problem
- What about email, instant messaging, and VoIP (Voice over IP)?
  - Clients receive emails, instant messages, and phone calls

Application specific Registration as a Mobility Solution

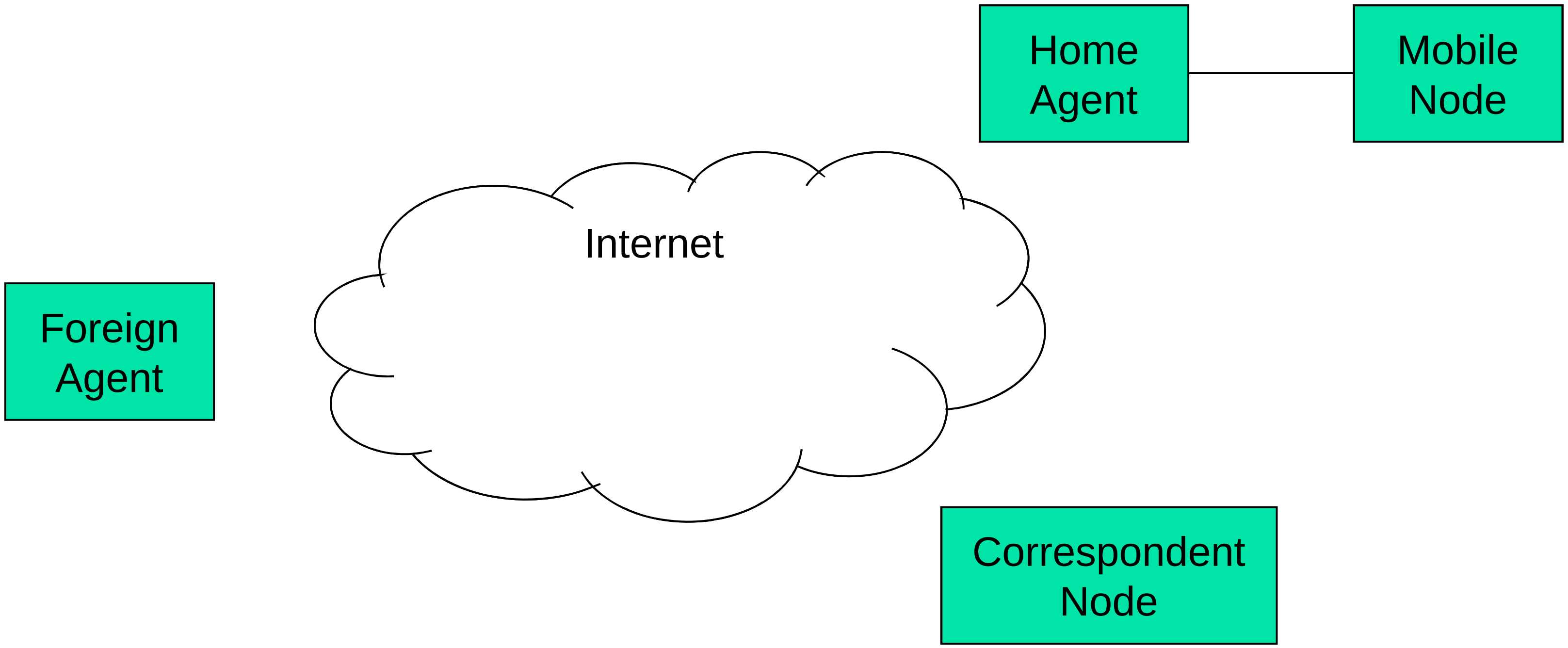
- To receive email, client connects to an email server
- To do instant messaging, client registers with an IM server
- To do VoIP, client registers with a SIP server

This is an adequate solution to 90% of mobility issues

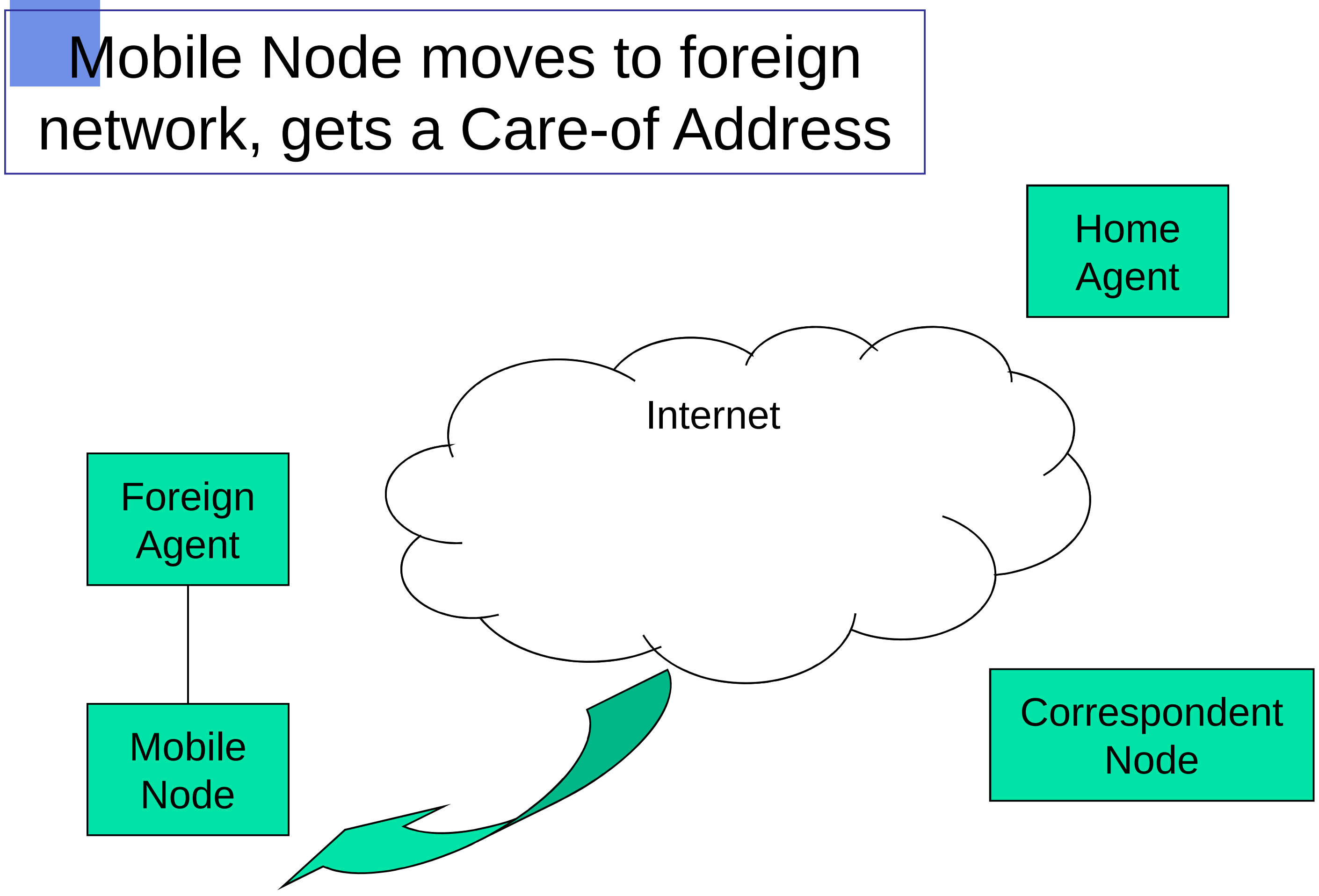
  - This is why Mobile IP hasn't gotten traction (i.e. Microsoft has not implemented it)

Mobile IP uses an IP-Level Registration

Mobile Node has a stable home address at its home network

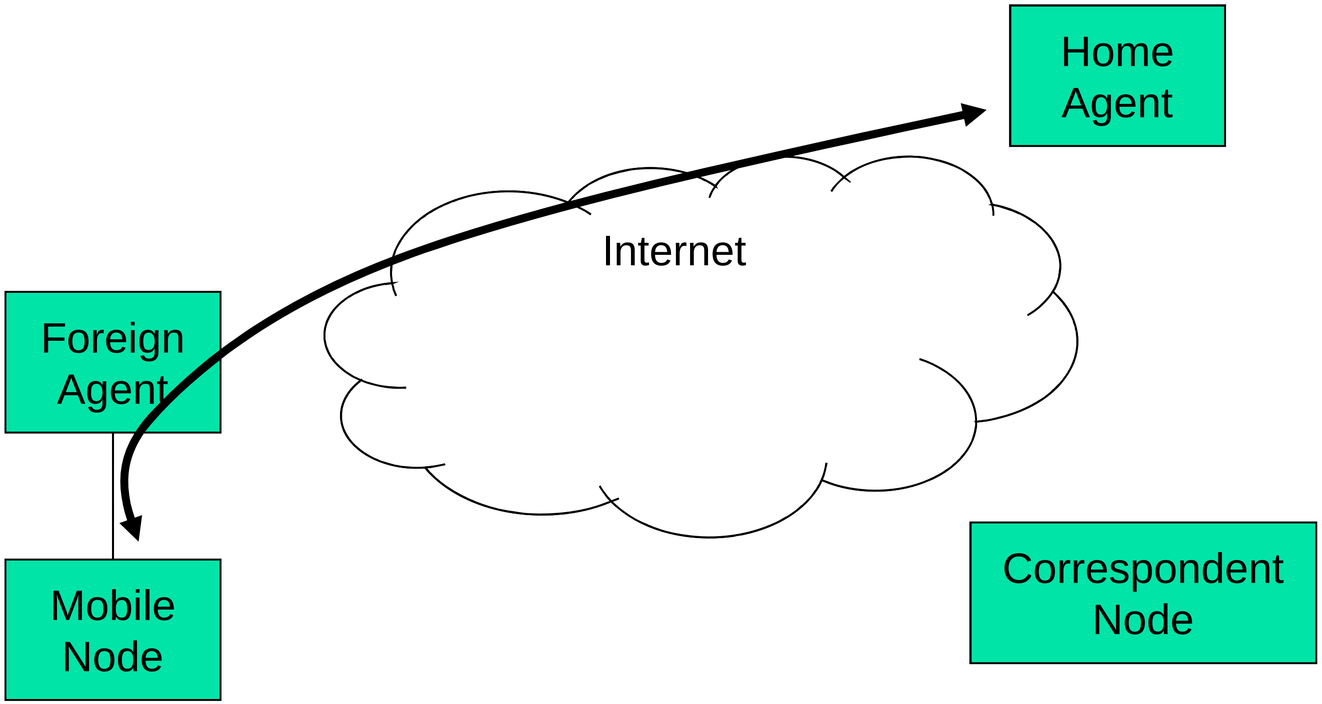


Mobile IP uses an IP-Level Registration



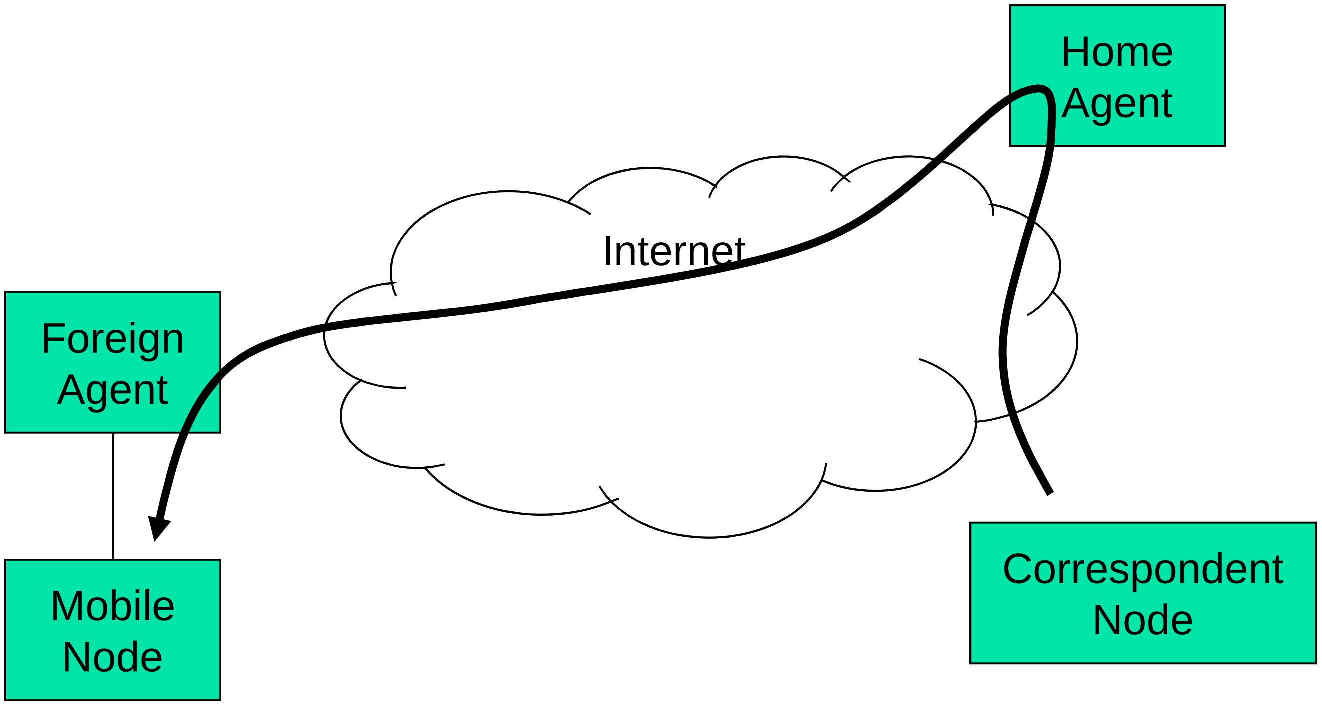
Mobile IP uses an IP-Level Registration

Mobile Node registers with Home Agent, creates IP tunnel

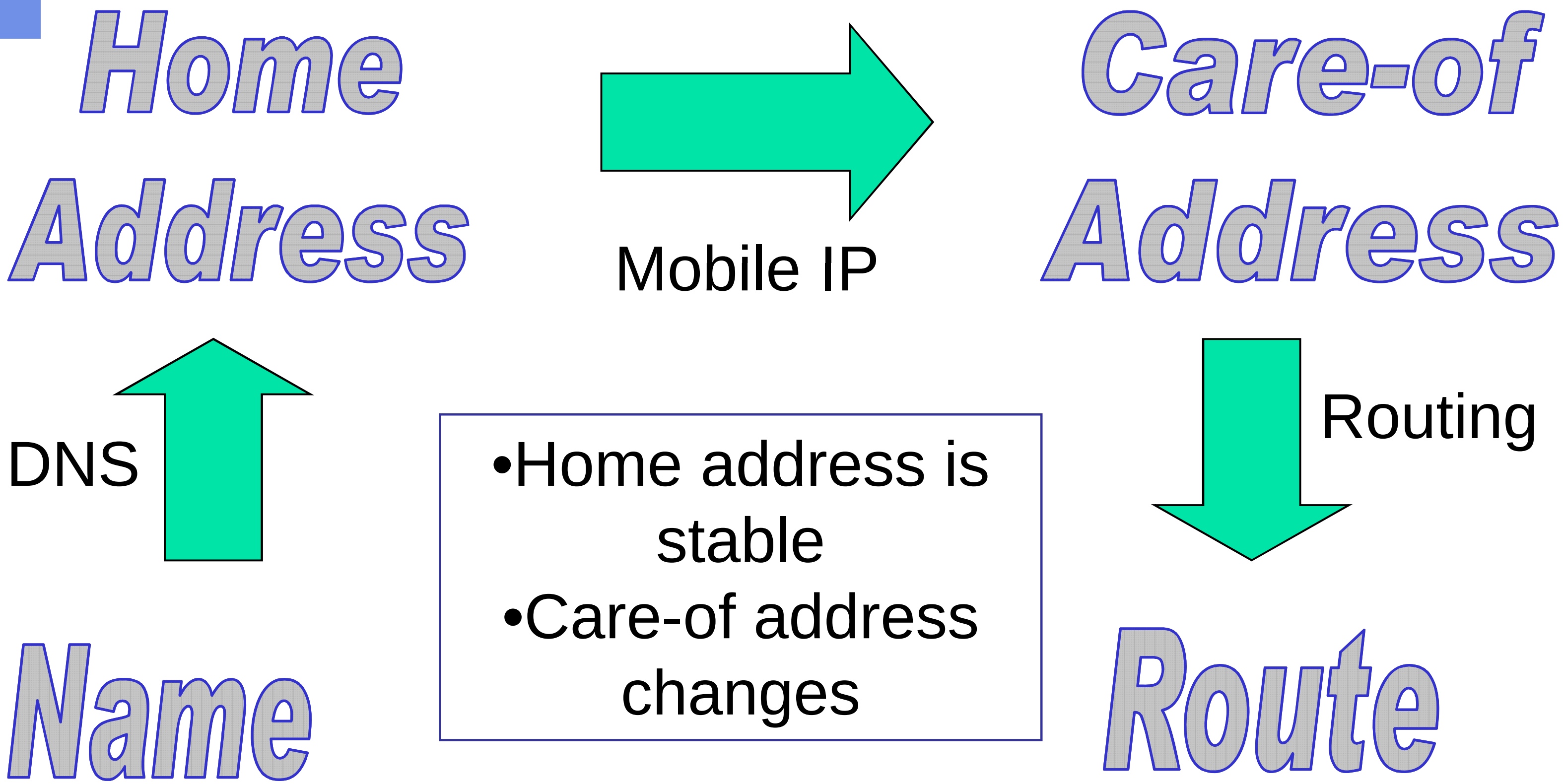


Mobile IP uses an IP-Level Registration

Connection initiated by Correspondent Node will be tunneled to Mobile Node



Mobile IP Adds a Layer of Indirection



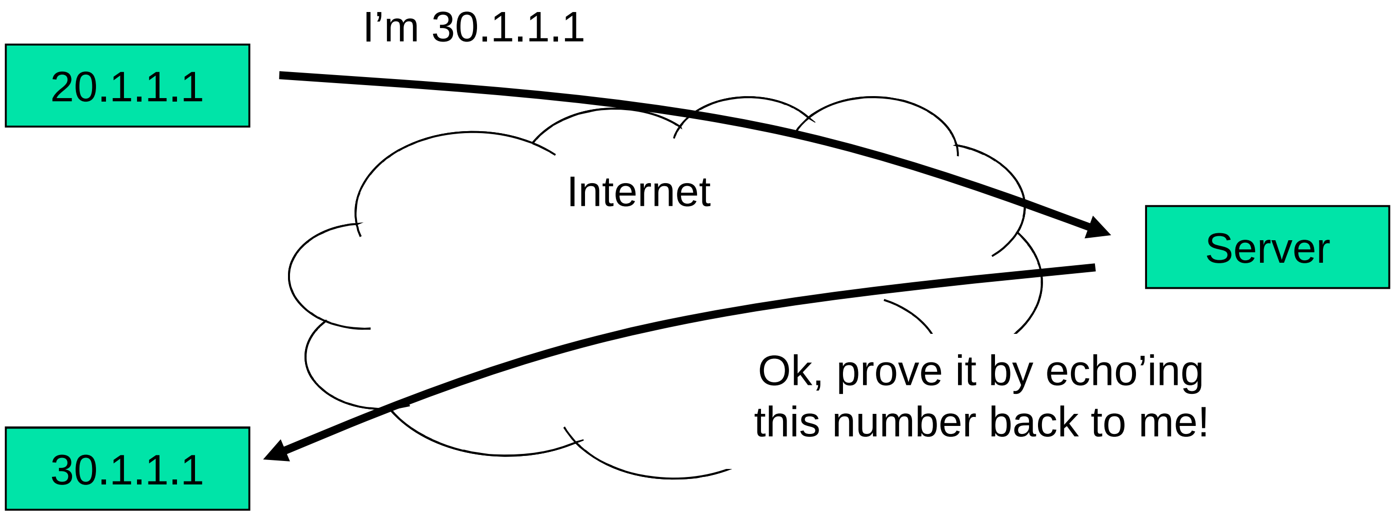
Client Identification

- Servers cannot locate clients, but often must be able to identify them
- HTTP cookies serve this role
- HTTP cookies also contain many attributes about the client or session
- They also typically contain some kind of signature
  - To prevent tampering

Identifiers: hard to spoof

- That is why driver's licenses have pictures and credit cards have signatures
- In networking, two ways:
  1. Identifier is also a locator
     - Reverse routability
  2. Some kind of secret-protected signature

Reverse Routability: DoS & Mobile IP



Since challenge doesn't go back to 20.1.1.1 (i.e. is not reverse routable), 20.1.1.1 cannot spoof 30.1.1.1

DNS Implementation

- Distributed DB implemented in hierarchy of many name servers
- Decentralized control and management of data
- Application-layer protocol used by hosts and name servers
  - Communicate to resolve names (i.e. name/address translation)
  - Core Internet function implemented as application-layer protocol

DNS Server Name DB

- DB contains entries called resource records (RR)
  - RR contains type, class, and application data
    - Before attribute type has been added, there was only record type (A, used to resolve IP address for a given domain name)
  - Classes = Internet (IN), Chaos net (CH), etc.
  - Each class defines types, e.g. for IN:
    - **A = address**
    - NS = name server
    - CNAME = canonical name (for aliasing)
    - HINFO = CPU/OS info
    - **MX = mail exchange**
    - PTR = pointer for reverse mapping of address to name

DNS Record Types

RR format: **`(name, value, type, ttl)`**

- Type=A
  - **`name`** is hostname
  - **`value`** is IP address
- Type=NS
  - **`name`** is domain (e.g. foo.com)
  - **`value`** is IP address of authoritative name server for this domain
- Type=CNAME
  - **`name`** is an alias name for some canonical name
  - **`value`** is canonical name
- Type=MX
  - **`value`** is priority and hostname of mail server associated with **`name`**

DNS Resource Records

	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 (OS + HW-type) this node represents
TXT	Any kind	Contains any entity-specific information considered useful

- Most important types of resource records forming the contents of nodes in the Internet DNS name space

DNS MX Record Type

- MX records point to mail exchanger for a name, e.g.
  - `mail.acm.org` is MX for `acm.org`
- Addition of MX record type proved to be a challenge
  - How to get mail programs to lookup MX record for mail delivery rather than an A record?
  - Needed critical mass of such mailers

Resource Records

- The database records of the distributed data base are called resource records (RR)
- Resource records are stored in configuration files (zone files) at name servers.
- Example:
  Resource records for db.mylab.com

db.mylab.com

```
                  $TTL 86400
mylab.com. IN SOA PC4.mylab.com.
                 hostmaster.mylab.com. (
                             1 ; serial
                         28800 ; refresh
                          7200 ; retry
                        604800 ; expire
                         86400 ; ttl
                             )
                  ;
mylab.com.   IN  NS  PC4.mylab.com.
                  ;
   localhost        A   127.0.0.1
   PC4.mylab.com.   A   10.0.1.41
   PC3.mylab.com.   A   10.0.1.31
   PC2.mylab.com.   A   10.0.1.21
   PC1.mylab.com.   A   10.0.1.11
```

Resource Records

db.mylab.com

```
              $TTL 86400
mylab.com. IN SOA PC4.mylab.com.
              hostmaster.mylab.com. (
                      1 ; serial
                  28800 ; refresh
                   7200 ; retry
                 604800 ; expire
                  86400 ; ttl
                      )
              ;
mylab.com.   IN  NS  PC4.mylab.com.
              ;
   localhost        A   127.0.0.1
   PC4.mylab.com.   A   10.0.1.41
   PC3.mylab.com.   A   10.0.1.31
   PC2.mylab.com.   A   10.0.1.21
   PC1.mylab.com.   A   10.0.1.11
```

- Max. age of cached data in seconds
- Start of authority (SOA) record. Means: “This name server is authoritative for the zone mylab.com”
  - PC4.mylab.com is the name server
  - hostmaster@mylab.com is the email address of the person in charge
- Name server (NS) record
- One entry for each authoritative name server
- Address (A) records
- One entry for each host address

DNS Examples

Priority of mail server

Represents domain as well as zone

Name server star.cs.vu.nl has 2 network interfaces thus increasing robustness

Backup mail server

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

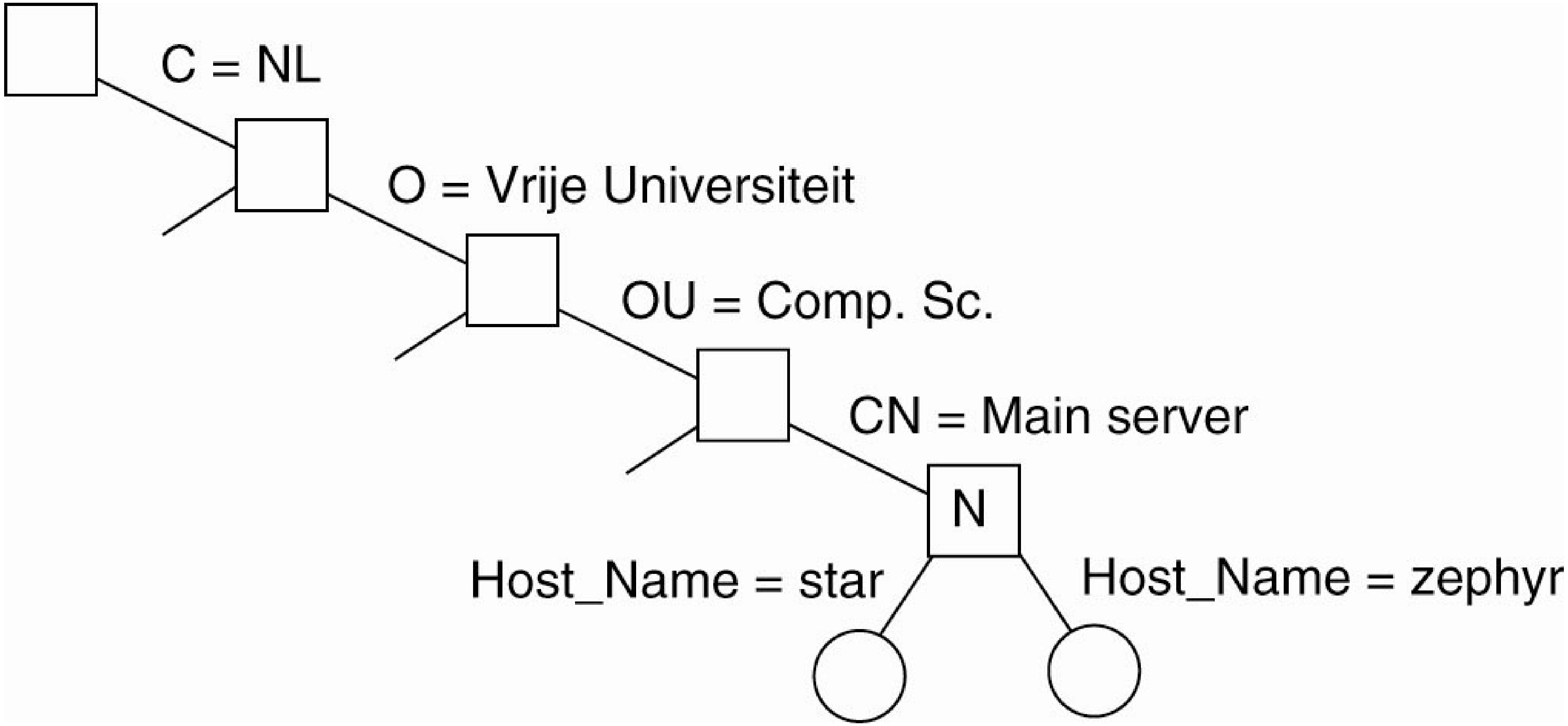
- An excerpt from the DNS database for the zone `cs.vu.nl.`

DNS Examples (2)

ftp.cs.vu.nl.	CNAME	soling.cs.vu.nl.
www.cs.vu.nl.	CNAME	soling.cs.vu.nl.
soling.cs.vu.nl.	A	130.37.20.20
soling.cs.vu.nl.	MX	1 soling.cs.vu.nl.
soling.cs.vu.nl.	MX	666 zephyr.cs.vu.nl.
soling.cs.vu.nl.	HINFO	"Sun" "Unix"
vucs-das1.cs.vu.nl.	PTR	0.198.37.130.in-addr.arpa.
vucs-das1.cs.vu.nl.	A	130.37.198.0
inkt.cs.vu.nl.	HINFO	"OCE" "Proprietary"
inkt.cs.vu.nl.	A	192.168.4.3
pen.cs.vu.nl.	HINFO	"OCE" "Proprietary"
pen.cs.vu.nl.	A	192.168.4.2
localhost.cs.vu.nl.	A	127.0.0.1

- Excerpt from the DNS database for the zone cs.vu.nl.

DNS Implementation (3)



- Part of the description for the vu.nl domain which contains the cs.vu.nl domain

DNS Implementation

Name	Record type	Record value
cs.vu.nl	NIS	solo.cs.vu.nl
solo.cs.vu.nl	A	130.37.21.1

- Part of the description for the vu.nl domain which contains the cs.vu.nl domain.

DNS Name Servers

- Authoritative name servers store parts of the DB
- Names assigned to authoritative name servers
  - For a host, authority stores host's IP address, name
  - Responds to queries for host IP addresses
  - Performs name/address translation for that host's name
  - Root name server knows authoritative servers for particular sub domains
    - Hierarchy organizes authoritative name servers
    - Reserving a domain gives you control of entry in root name server for particular names

DNS Name Lookup

- Hierarchical lookup
  - Each host has a pointer to a local name server to query for unknown names
  - Each local name server knows root of its sub tree
  - Root points to sub-levels, sub-levels point to deeper sub-levels, ... point to leaf name server representing the authority for unknown name

Top-Level Domains

- Three types of top-level domains:
  - Organizational: 3-character code indicates the function of the organization
    - Used primarily within the US
    - Examples: gov, mil, edu, org, com, net
  - Geographical: 2-character country or region code
    - Examples: us, va, jp, de
  - Reverse domains: A special domain (in-addr.arpa) used for IP address-to-name mapping

There are more than 200 top-level domains

Authority and Delegation

- Authority for the root domain is with the Internet Corporation for Assigned Numbers and Names (ICANN)
- ICANN delegates to accredited registrars (for gTLDs) and countries for country code top level domains (ccTLDs), e.g. DENIC[1]
- Authority can be delegated further
  - Chain of delegation can be obtained by reading domain name from right to left.
- Unit of delegation is a "zone"

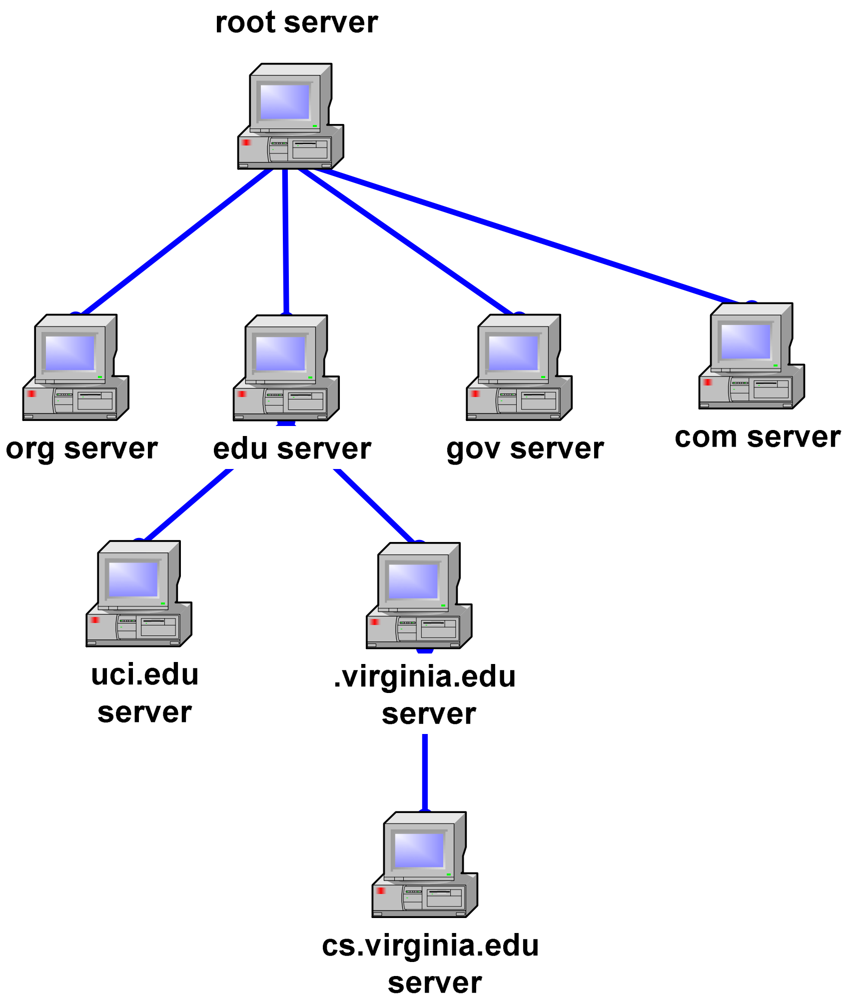
[1]DENIC takes part in the ENUM project, e.g. one address for all

DNS Top-Level Domain

Domain Name	Meaning
com	Commercial bussiness
edu	Universities (colleges) in USA
gov	Government departments(USA)
mil	Military institutions
net	Netprovider
org	All other business
arpa	Temporal ARPA-domain
int	International organisations
Zip code of Country(e.g. de)	Abbreviations of all countries

Hierarchy of Name Servers

- Resolution of the hierarchical name space is done by a hierarchy of name servers
- Each server is responsible (authoritative) for a contiguous portion of the DNS namespace, called a zone
- Zone is a part of the subtree
- DNS server answers queries about hosts in its zone



Primary/Secondary Name Server

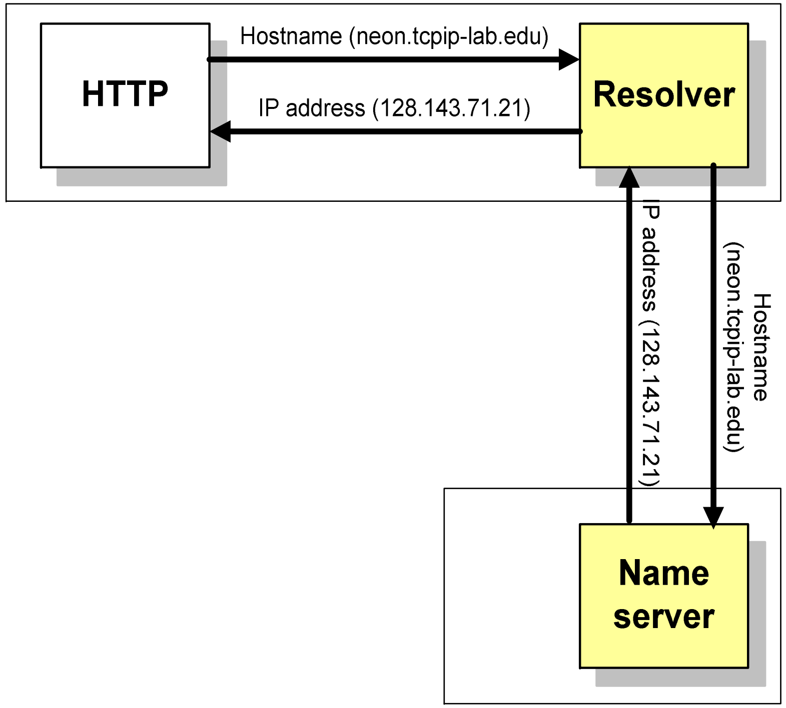
- For each zone, there must be a primary name server and a secondary name server
  - The primary server (master server) maintains a zone file which has information about the zone. Updates are made to the primary server.
  - The secondary server pulls data stored at primary server

Adding a host:

- When a new host (e.g. "**`gold.cs.virginia.edu`**") is added to a zone, the administrator of the cs-department of the Virginia State University adds the IP information of the host (IP address and name) to its primary server

Domain Name Resolution

1. User program issues a request for the IP address of a hostname
2. Local resolver formulates a DNS query to the name server of the host
3. Name server checks if it is authorized to answer the query.
   a) If yes, it responds.
   b) Otherwise, it will query other name servers, starting at the root tree
4. When the name server has the answer it sends it to the resolver.

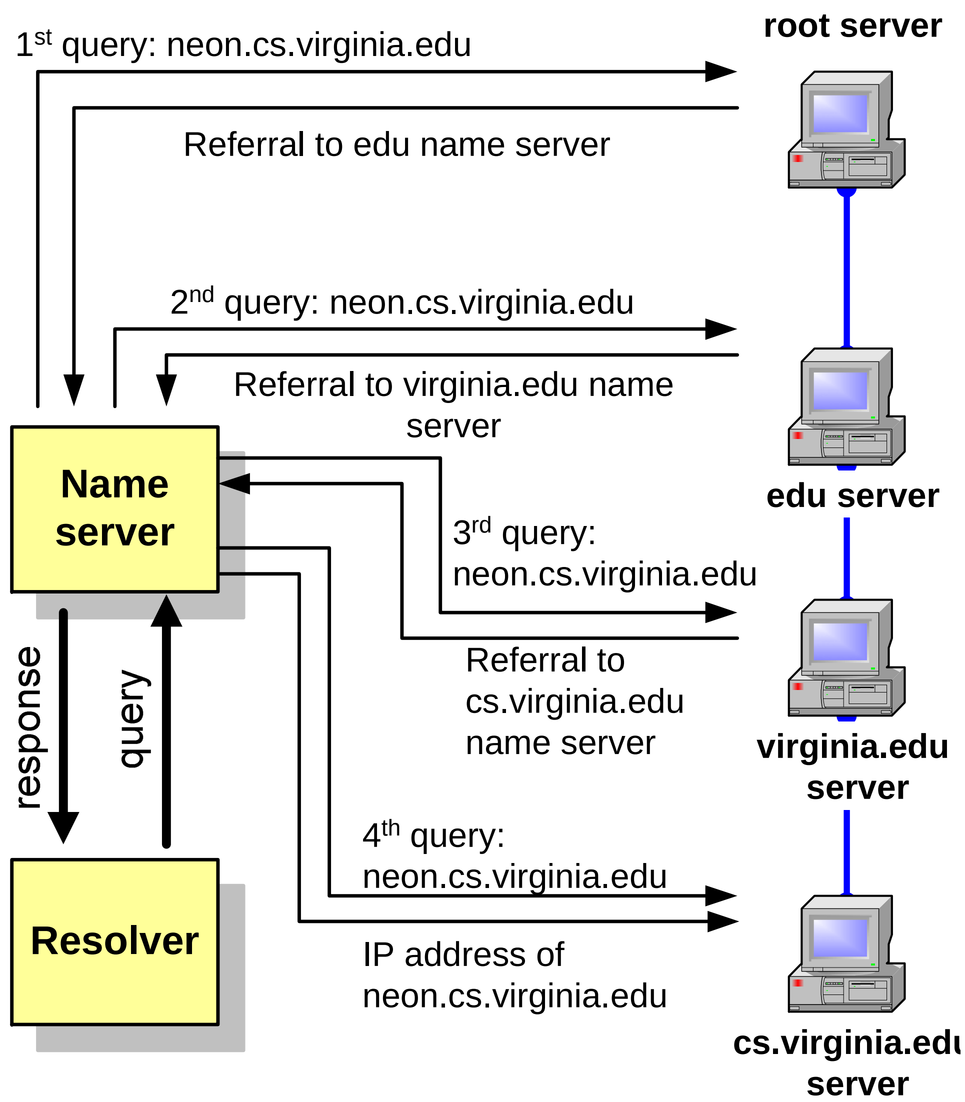


Recursive and Iterative Queries

- There are two types of queries:
  - Recursive queries
  - Iterative (non-recursive) queries
- The type of query is determined by a bit in the DNS query
- Recursive query: When the name server of a host cannot resolve a query, the server issues a query to resolve the query
- Iterative queries: When the name server of a host cannot resolve a query, it sends a referral to another server to the resolver

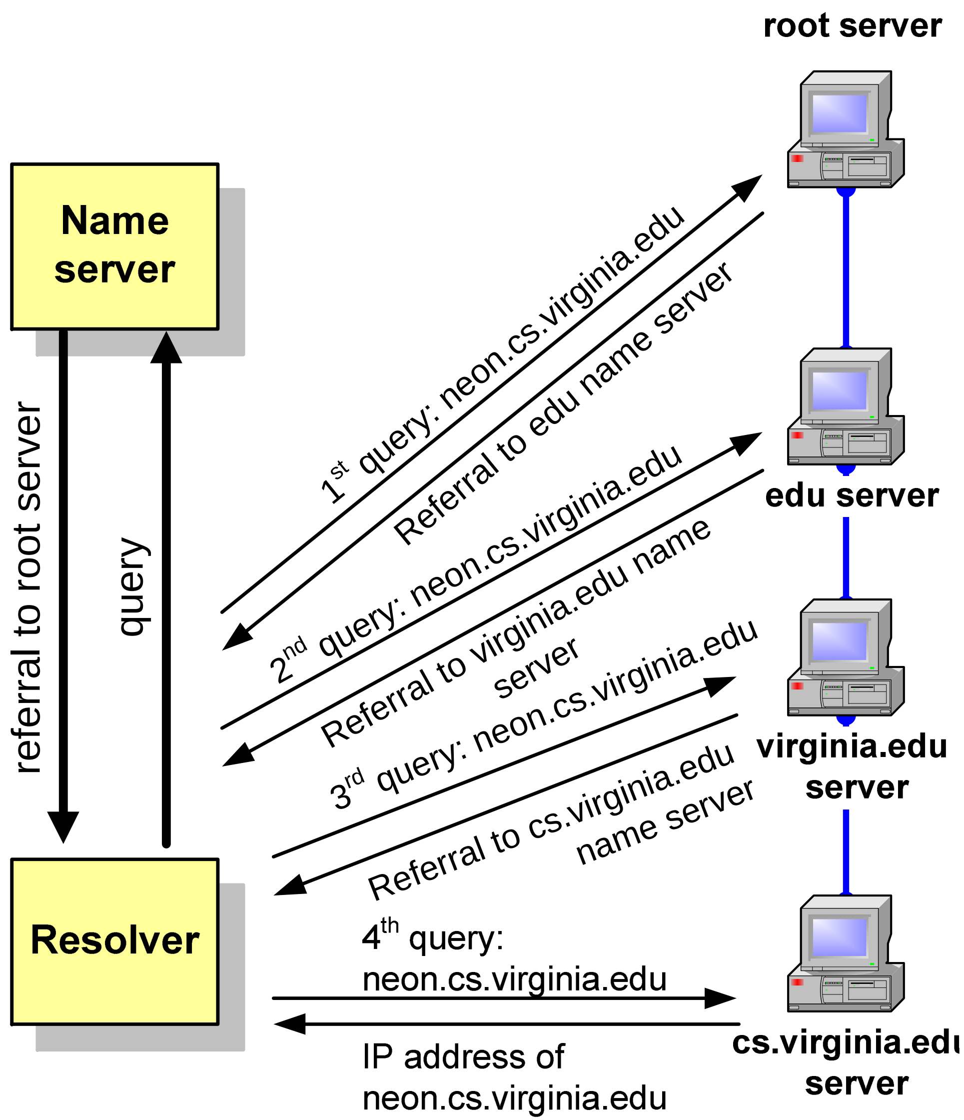
“Recursive” Query

- In a recursive query, the resolver expects the response from the name server
- If the server cannot supply the answer, it will send the query to the “closest known” authoritative name server (here: in the worst case, the closest known server is the root server)
- The root sever sends a referral to the “edu” server. Querying this server yields a referral to the server of “virginia.edu”
- ... and so on



Iterative Query

- In an iterative query, the name server sends a closest known authoritative name server a referral to the root server.
- This involves more work from the resolver



Caching

- To reduce DNS traffic, name servers cache information on previous <domain name, IP address>
- When entry for a query is cached, immediate reply
- Note: If an entry is sent from a cache, the reply from the server is marked as “unauthoritative”
- Also DNS negative queries are also cached
  - Don’t have to repeat past mistakes, e.g. misspellings

Typical DNS Name Resolution

- Client does recursive request to local name server
- Local name server does iterative request to find name
- Local name server has knowledge of root servers
- Steps for resolving www.ogi.edu
  - Application calls gethostbyname()
  - Resolver contacts local name server (S_1)
  - S_1 queries root server (RS_2) for (www.ogi.edu)
  - RS_2 returns NS record for ogi.edu (i.e. name server S_3)
  - S_1 queries S_3 for for www.ogi.edu
  - S_3 returns A record for www.ogi.de

DNS Caching

- Cached info periodically times out
  - Soft state
  - Lifetime (TTL) of date controlled by owner of data
  - TTL passed with every record
  - TTL affects DNS-based load balancing techniques
- Update/notify mechanisms under design by IETF
  - TFC 2136
  - http://www.ietf.org/html.charters/dnsind-charter.html

Replication and Caching in DNS

- Replication
  - for every root server there are at least 2 replicas
    - primary/backup principle
    - backup servers periodically request updates from their primary servers (zone transfer)
- Caching
  - Each name server implements caching

Further reading:

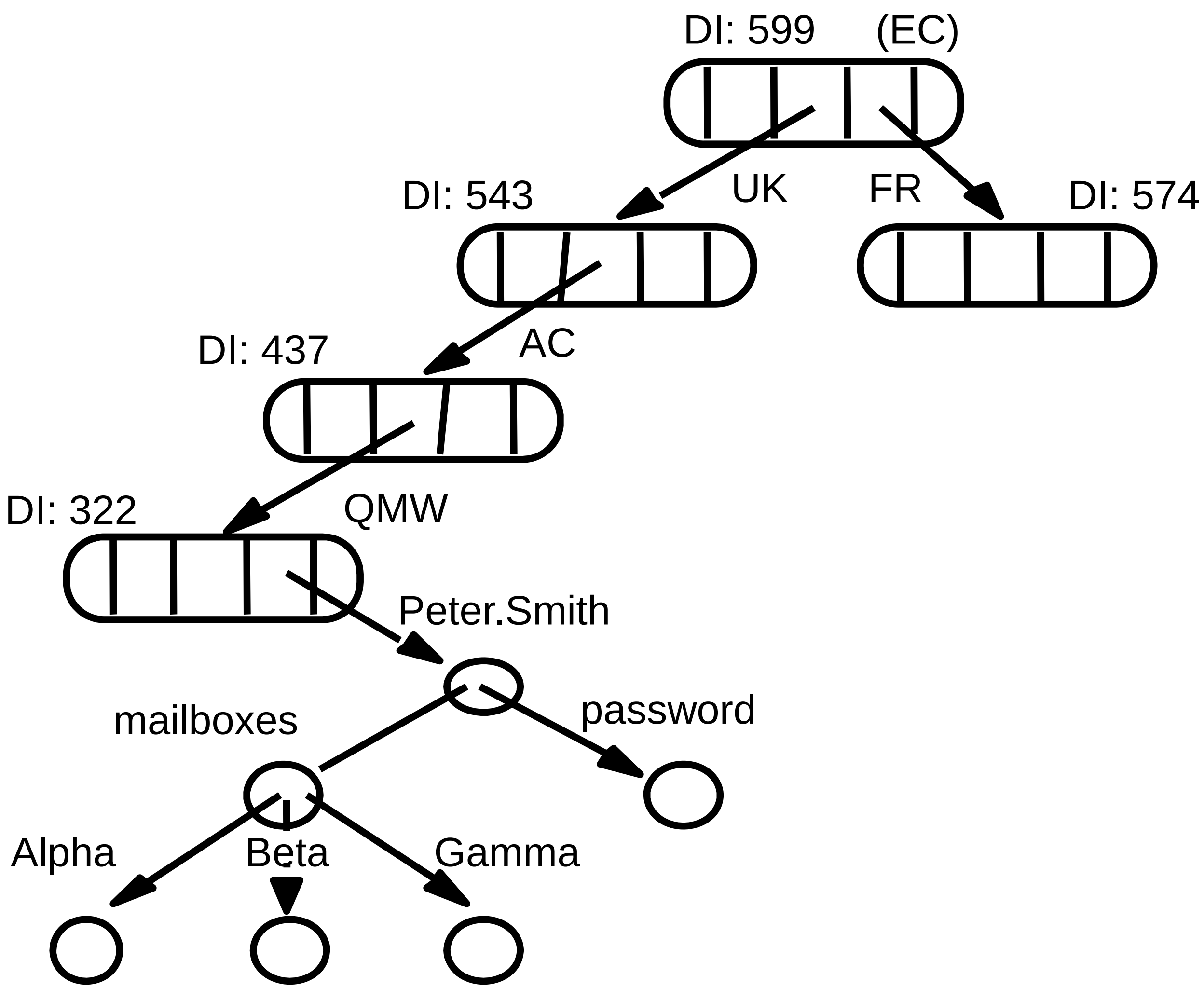
F. Halsall: “Data Communications, Computer Networks and Open System”, Addison-Wesley 1992

D. Comer: “Computernetzwerke und Internets”, Prentice Hall 1997

DEC' Global Name Space

Study of your own

Global Name Service (GNS)



DEC's Global Name Space Model*

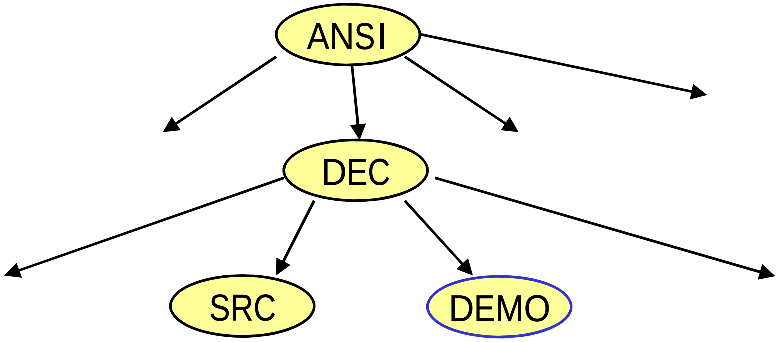
Requirements:

- Large size, i.e. handle an arbitrary number of names
- Long life, i.e. many changes may occur to the name space in the long run
- High availability, because otherwise with a broken name service the system cannot work any longer
- Fault isolation, local failures don't crash complete DS
- Tolerance of mistrust, since large scale service won't have any component which is trusted by all its clients

* Butler B. Lampson: "Designing a global name service", 1985 another major paper you should have read

Client Level

- ∃ hierarchical names and their values with operations for reading and updating them
- Client sees a structure like a Unix FS, i.e. a rooted tree with unique directory identifiers
- Arcs of tree are called directory references, i.e. a directory can be named relative to its root by a pathname (*full name*)



Nodes of the tree have 2 attributes:

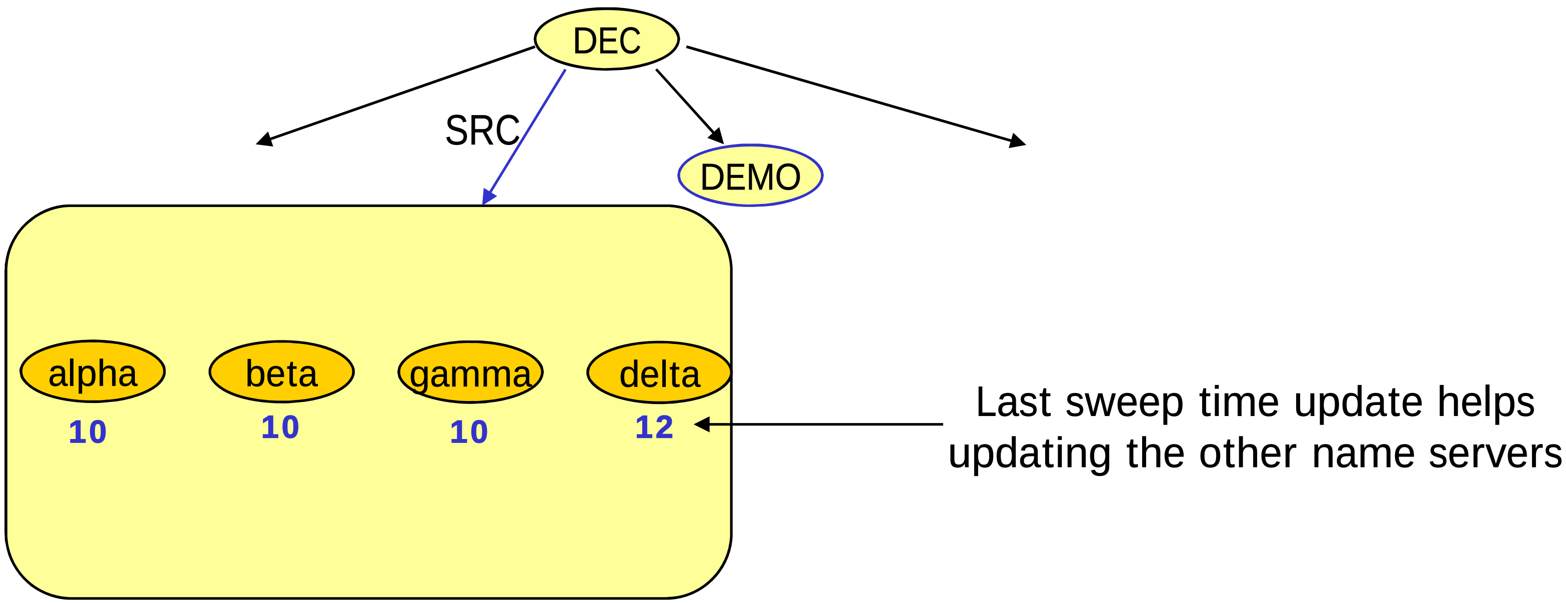
a) Time stamp

b) Present/absent mark

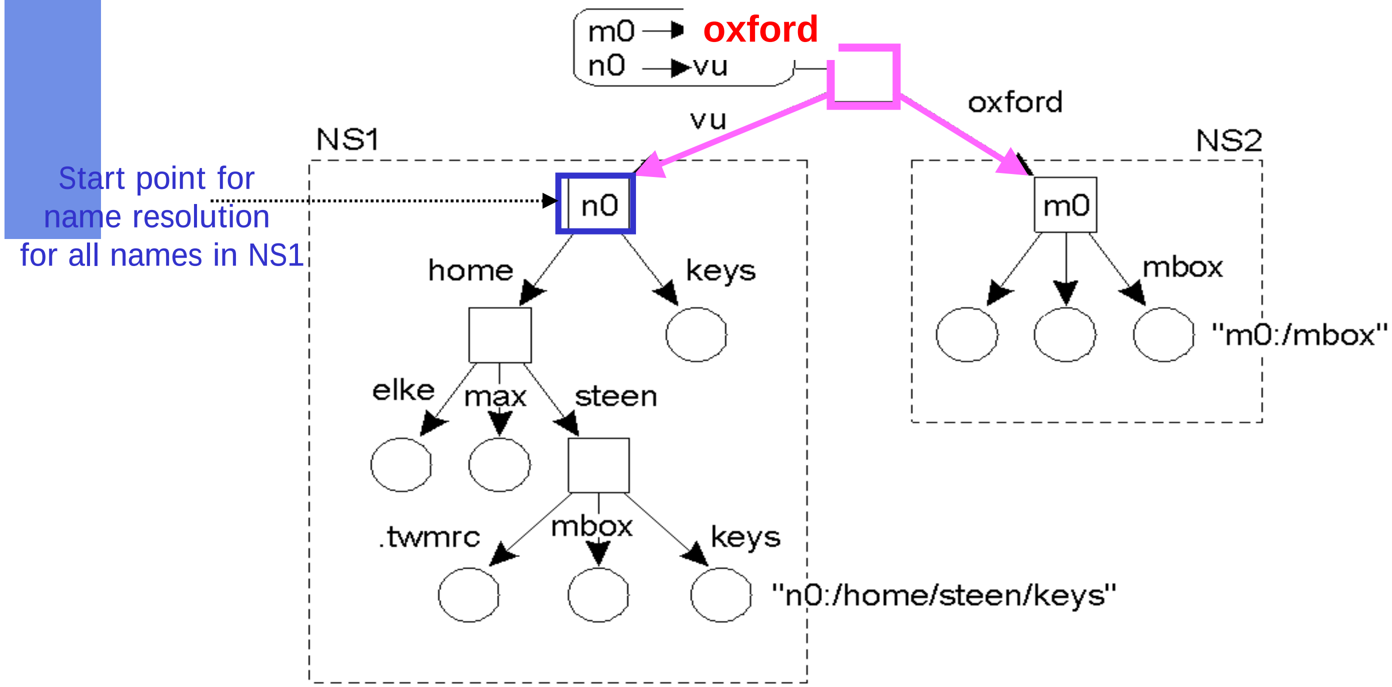
Demo has the pathname ANSI/DEC/DEMO

Administrative Level

- Administrator controls the number of replicated name server and controls the update of all replicas
- Each directory reference includes a list of all replicated servers for the referenced directory



DEC's GNS: Super Root



- Super root contains a table of all distributed roots, e.g. vu, oxford and their local names n0, m0
- Name **`/home/steen/keys`** in NS1 is expanded to:
  - **`n0://home/steen/keys`**
- This name is resolved into: **`/vu/home/steen/keys`**

Attribute-Based Naming

Directory Service

Hierarchical Implementations: Lightweight Directory Access Protocol (LDAP)

Decentralized Implementations

- Mapping to DHT
- Semantic Overlay Networks

Study of your own

LDAP is another popular DDS

- Richer and more general than DNS
  - Has generalized attribute/value scheme
  - Can search on attribute, not just name
- Simpler and more efficient than a full relational database
- Not a global directory service, though namespace is global
  - Its predecessor, X.500, was meant to be
  - But “local” LDAP services can point to each other
- Commonly used for personnel RR databases, subscriber databases

DDS = Distributed Directory Service

Resource Description Framework (RDF)

- Resources are described as triples:
  <subject, predicate, object>

- Example:
  <person, name, Alice> describes a resource "person" whose "name" is Alice

Directory Service: X 500

CCITT and ISO standard (1988):

Names

- List of tuples (attribute = value)
  - Attributes

- country	“c”
- organization	“o”
- organizational unit	“ou”
- surname	“sn”
...	
- telephone number	“telephone”

Example:

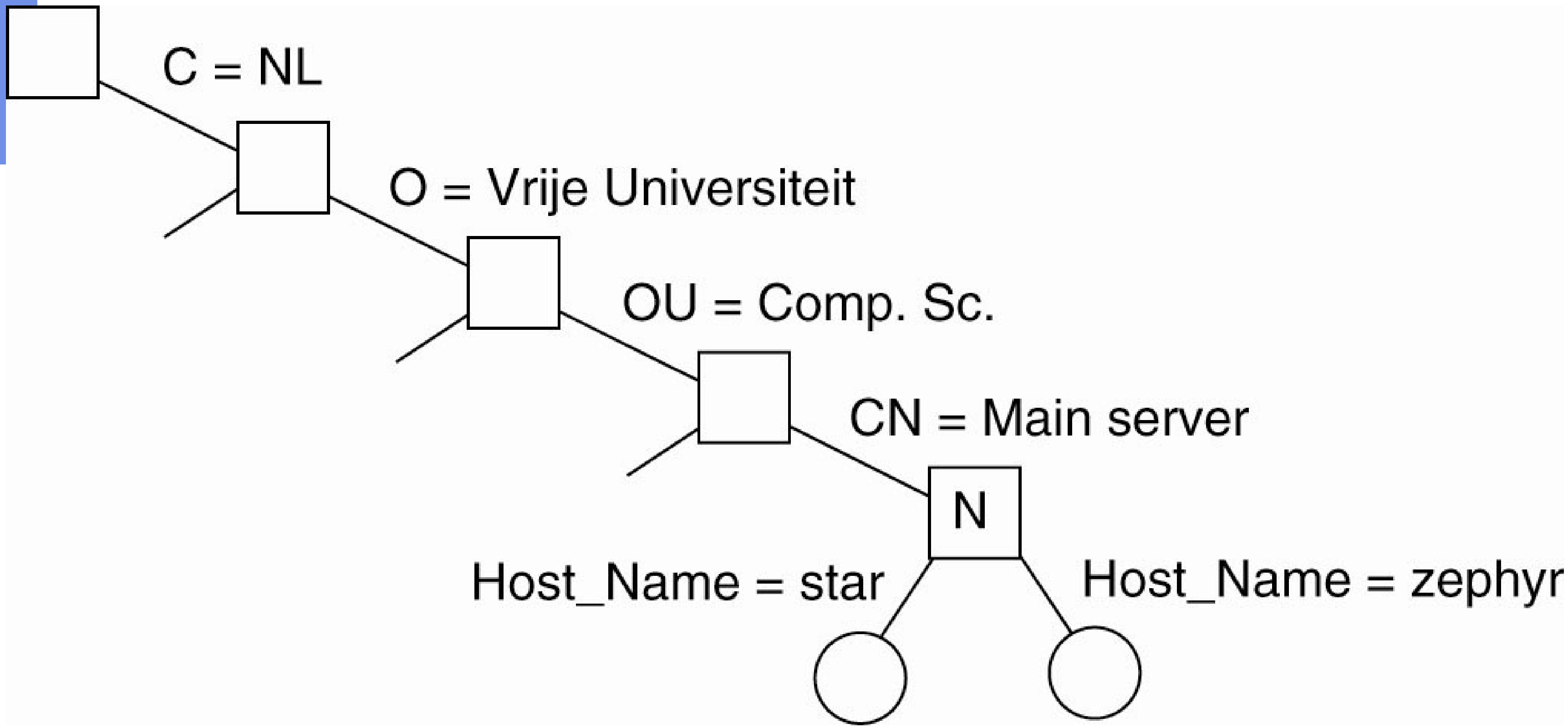
/c=de/o=uni-karlsruhe/ou=rz/sn=zoller/telephone=+49 721 608 40!

Hierarchical Implementations: 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

- Example of a lightweight directory access protocol LDAP directory entry using LDAP naming conventions
- LDAP derived from X.500

Hierarchical Implementations: LDAP



- Part of a directory information tree

Hierarchical Implementations: LDAP

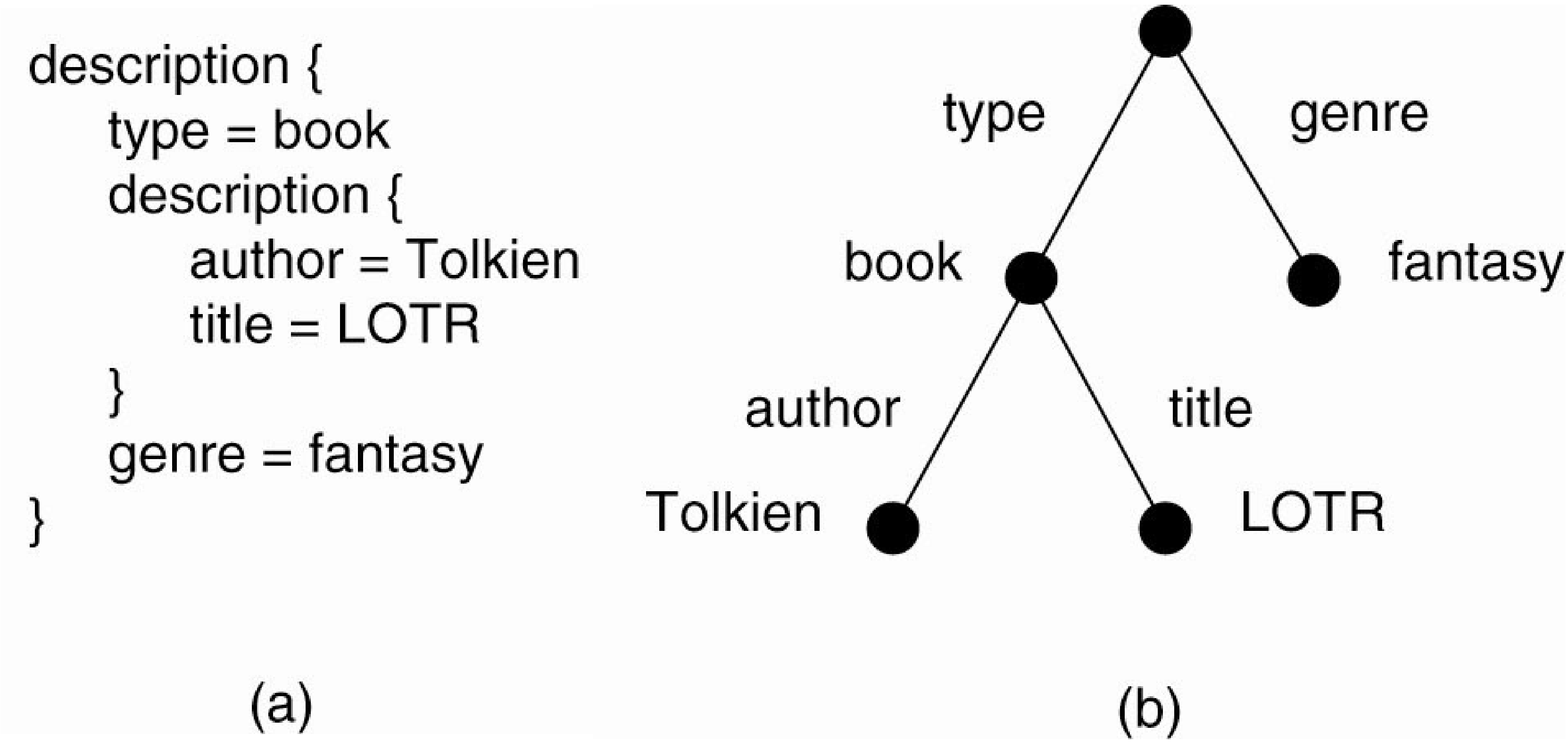
Attribute	Value
Country	NL
Locality	Amsterdam
Organization	Vrije Universiteit
OrganizationalUnit	Comp. Sc.
CommonName	Main server
Host_Name	star
Host_Address	192.31.231.42

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)

- (b) Two directory entries having *Host_Name* as RDN

Mapping to Distributed Hash Tables

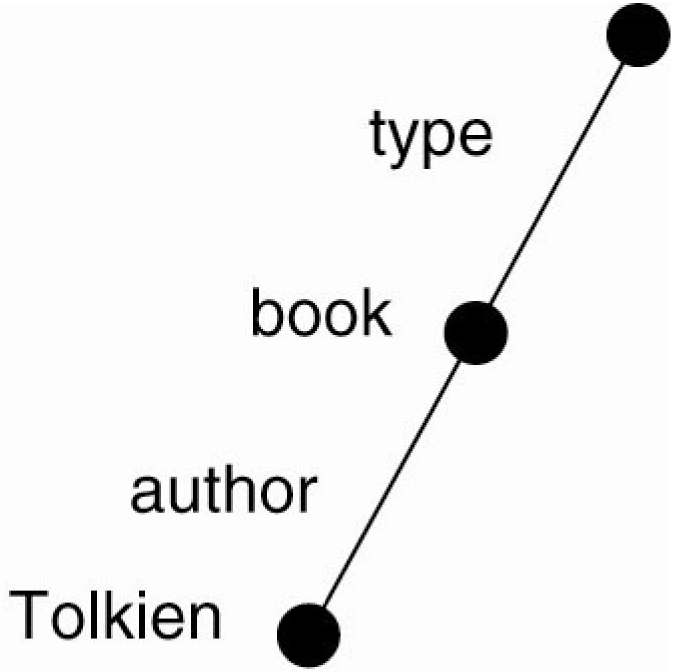


- (a) A general description of a resource
- (b) Its representation as an AVTree.

Mapping to Distributed Hash Tables

```
description {
    type = book
    description {
        author = Tolkien
        title = *
    }
    genre = *
}
```

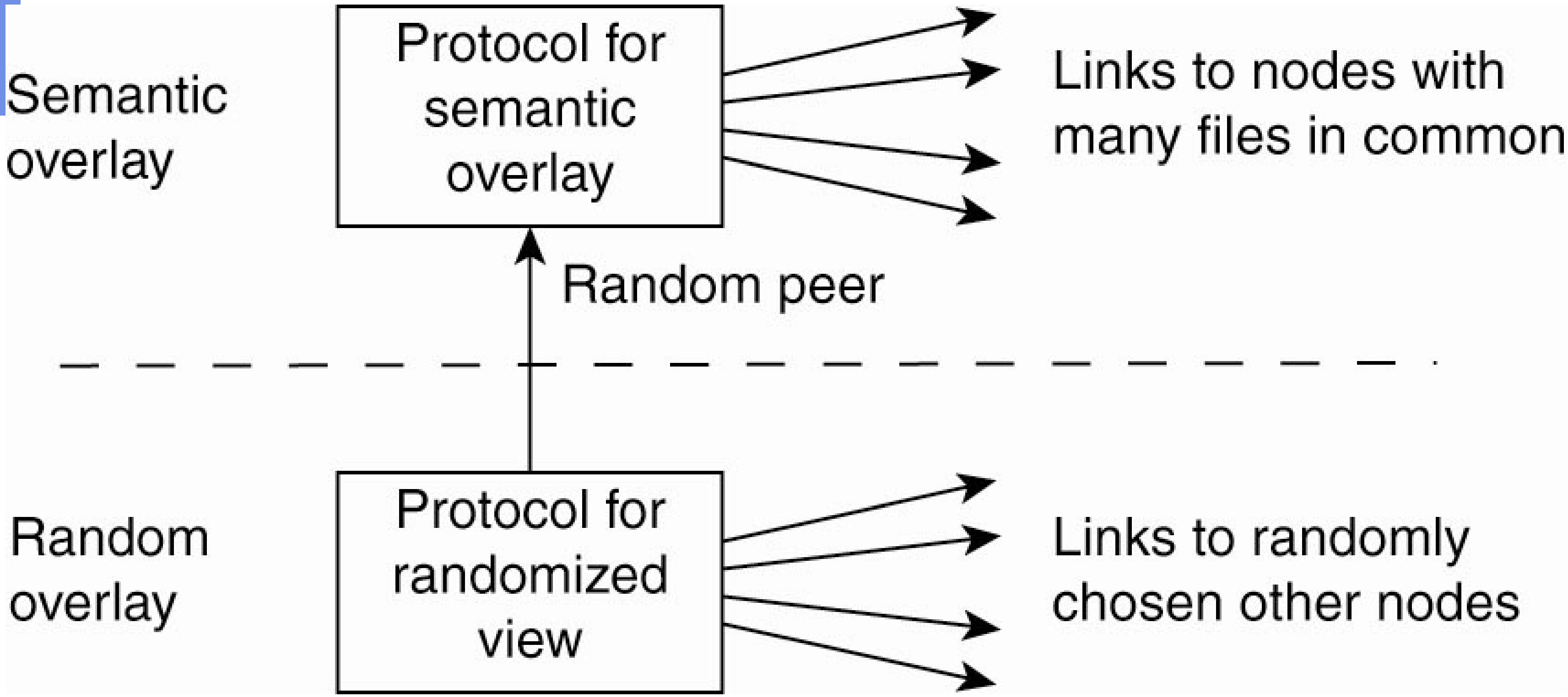
(a)



(b)

- (a) The resource description of a query.
- (b) Its representation as an AVTree.

Semantic Overlay Networks



- Maintaining a semantic overlay through gossiping

Directory Service: X.500

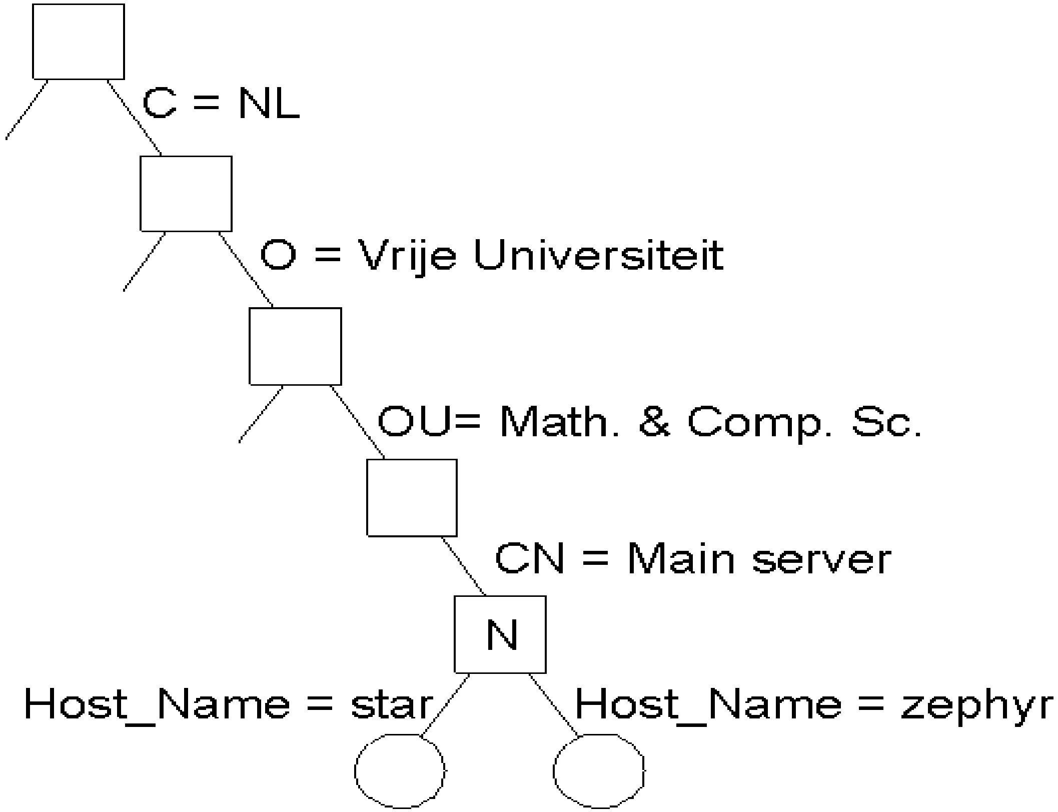
- A Directory Service supports lookup based on a set of attribute values (yellow pages)
- Directory entries contain <attrib, value> pairs
- Set of entries forms Directory Information Base (DIB)
- Naming attributes of an entry jointly identify an entry uniquely.
- Canonical sequences of naming attributes form the Directory Information Tree (DIT)
  - Edges are labeled with <attrib, value> pairs
- Each name attribute is a so called RDN (relative distinguished name)

The X.500 Directory Entries

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, 192.31.231,192.31.231.66
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 DIB



- Part of the directory information tree

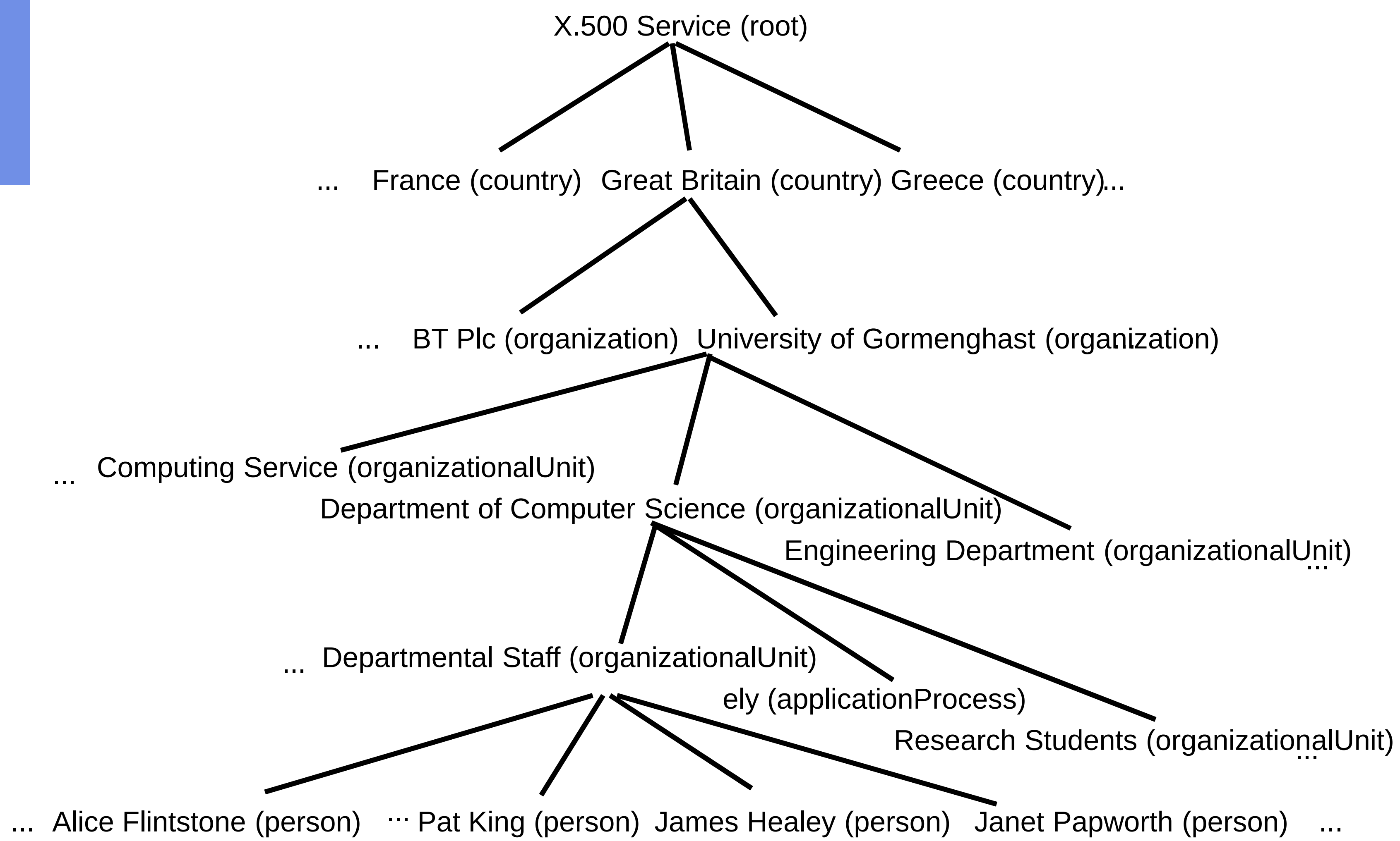
The X.500 Name Space

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**

- Two directory entries having *Host_Name* as RDN

X.500 Directory Information Tree



X.500 Lookup

Name lookup

`list(/C=NL/O=Vrije Universiteit/`

`OU=Math.&Comp.Sci./CN=Main server)`

returns corresponding names

`star zephyr`

Directory lookup

search **`&(C=NL)(O=Vrije Universiteit)`**

`(OU=*)(CN=Main server)`

returns all entries with matching attributes