SC250
Computer Networking I

Peer-to-Peer Networks and the DNS

Prof. Matthias Grossglauser

School of Computer and Communication Sciences
EPFL

http://lcawww.epfl.ch

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

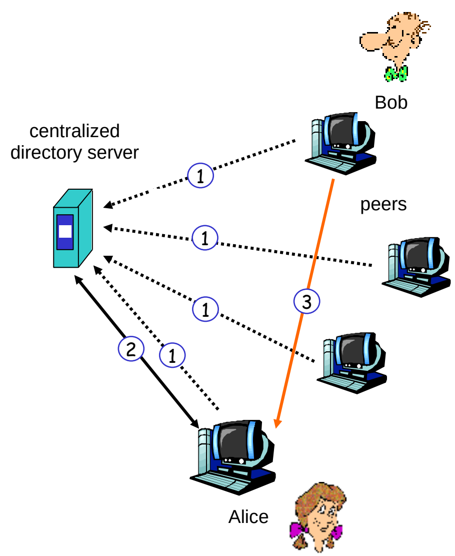
Peer-to-Peer File Sharing

- Example
  - Alice runs P2P client application on her notebook computer
  - Intermittently connects to Internet; gets new IP address for each connection
  - Asks for “Hey Jude”
  - Application displays other peers that have copy of Hey Jude.
  - Alice chooses one of the peers, Bob.
  - File is copied from Bob’s PC to Alice’s notebook: HTTP
  - While Alice downloads, other users uploading from Alice.
- Alice’s peer is both a Web client and a transient Web server.
  - All peers are servers = highly scalable!

P2P: Centralized Directory

original “Napster” design

1) when peer connects, it informs central server:
   - IP address
   - content
2) Alice queries for “Hey Jude”
3) Alice requests file from Bob



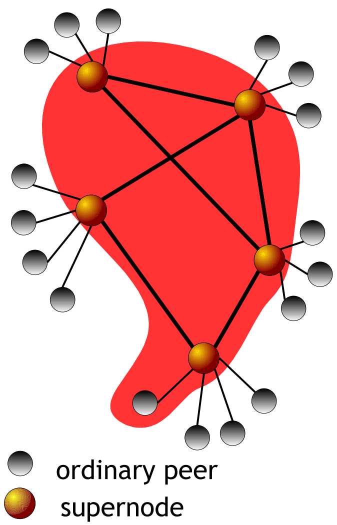
P2P: Problems with Centralized Directory

- Single point of failure
- Performance bottleneck
- Copyright infringement
  - Napster has been shut down by lawsuit

> file transfer is decentralized, but locating content is highly centralized

P2P: Decentralized Directory

- Each peer is either a group leader or assigned to a group leader.
- Group leader tracks the content in all its children.
- Peer queries group leader; group leader may query other group leaders
- Example:
  - FastTrack protocol (Kazaa, reverse-engineered by others): supernodes chosen dynamically
  - eDonkey2000/ed2k (eMule, etc.): supernodes are fixed servers

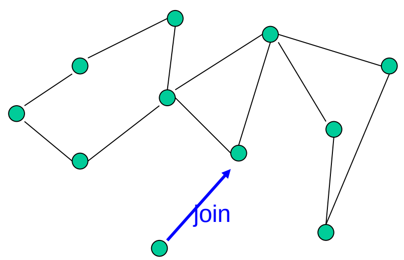


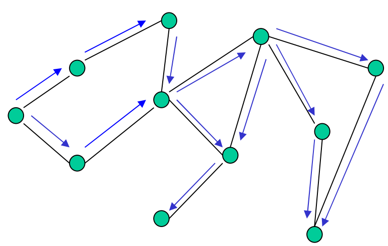
More about Decentralized Directory

- Overlay network
  - peers are nodes
  - edges between peers and their group leaders
  - edges between some pairs of group leaders
  - virtual neighbors
- Bootstrap node
  - connecting peer is either assigned to a group leader or designated as leader
- Advantages of approach
  - no centralized directory server
    - location service distributed over peers
    - more difficult to shut down
- Disadvantages of approach
  - bootstrap node needed
  - group leaders can get overloaded

P2P: Query Flooding

- Example: Gnutella
- no hierarchy
- use bootstrap node to learn about others
- join message
- Send query to neighbors
- Neighbors forward query
- If queried peer has object, it sends message back to querying peer





P2P: More on Query Flooding

- Pros
  - peers have similar responsibilities: no group leaders
  - highly decentralized
  - no peer maintains directory info
- Cons
  - excessive query traffic
  - query radius: may not have content when present
  - bootstrap node
  - maintenance of overlay network

The Domain Name System (DNS)

- Background and motivation
- Name space
- DNS architecture
- DNS protocol
- `nslookup` command

DNS: Domain Name System

- People: many identifiers:
  - SSN (CH: AVS), name, passport #
- Internet hosts, routers:
  - IP address (32 bit) - used for addressing datagrams
  - “name”, e.g., gaia.cs.umass.edu - used by humans
- How to map between IP addresses and names?

- Domain Name System:
  - distributed database implemented in hierarchy of many name servers
  - application-layer protocol host, routers, name servers to communicate to resolve names (address/name translation)
    - note: core Internet function, implemented as application-layer protocol
    - complexity at network’s “edge”

Early ARPANET: hosts.txt

- Centralized file containing entire name-address mapping
  - updated and disseminated every few days

```
NET : 127.0.0.0 : LOOPBACK :
NET : 128.2.0.0 : CMU-NET :
NET : 128.3.0.0 : LBL-IP-NET1 :
NET : 128.4.0.0 : DCNET :
NET : 128.6.0.0 : RUTGERS :
NET : 128.7.0.0 : EKONET :
NET : 128.8.0.0 : UMDNET :
NET : 128.9.0.0 : ISI-NET :
NET : 128.11.0.0 : BBN-CRONUS :
NET : 128.12.0.0 : SU-NET :
....
HOST : 26.1.0.205 : STEWART-EMH1.ARMY.MIL : VAX-11/750 : VMS :TCP/TELNET,TCP/FTP
:
HOST : 26.0.0.206 : GORDON.MT.DDN.MIL : C/30 : TAC : TCP,ICMP :
HOST : 26.14.0.206 : NETPMSA-CHARL4.NAVY.MIL : WANG-VS100 : VSOS ::
HOST : 26.29.0.207, 137.209.8.2, 137.209.51.1, 137.209.18.2 : OAKLAND-IP.DDN.MIL ::::
HOST : 26.0.0.210 : BARKSDALE.MT.DDN.MIL : C/30 : TAC : TCP,ICMP :
HOST : 26.0.0.211 : LORING.MT.DDN.MIL : C/30 : TAC : TCP,ICMP :
```


Problems with Centralized Approach

- Scalability
  - with n names -> n^2 work
- Single point of failure
  - e.g., if file is corrupted, entire Arpanet would collapse
- Cumbersome
  - updating, regular downloads, installing file
- Collisions
  - no mechanism to avoid allocating same name multiple times
- Consistency
  - “view” of network not always the same

DNS: Decentralized Approach

- Distributed database
  - relation **name - IP address**
  - clear delegation of authority – who owns parts of namespace, who updates the database?
  - scales well
  - name servers
    - primary, secondary - authoritative data
    - cache - non-authoritative data
  - resolver:
    - **gethostbyname()**
    - **gethostbyaddr()**
- Hierarchical name space
  - similar to Unix pathnames, but reversed
    - unix: /usr/local/bin/emacs
    - DNS: tudor.eecs.berkeley.edu

DNS Overview

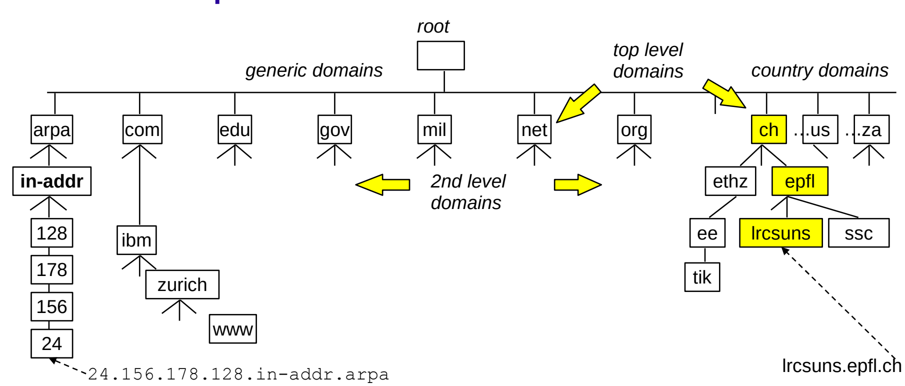
- DNS offers one distributed world-wide database
  - distributed according to the zone concept: every zone has a master file describing all records under the zone's authority
  - name servers hold their part of the database
    - for one zone, at least two name servers have the zone information, copied from master file
      - example: **stisun1.epfl.ch, stisun2.epfl.ch; dns1.ethz.ch, dns2.ethz.ch**
    - zone information held by the name server is called *authoritative* data
    - one name server may hold zone data for one or more zones
  - zone data contains pointers to name servers holding authoritative data for subzones
- Root servers
  - 13 servers distributed all over the world
  - any primary server needs to know their addresses

Name and Address Spaces

- Sample name
  - `tudor.eecs.berkeley.edu`
- Hierarchical
  - least specific to the right ("edu")
- Mainly useful to humans
  - human-readable reference to hosts, networks, email domains, etc.
- Size (virtually) unlimited
  - variable-size names, human readable

- Sample IP address
  - `128.32.43.249`
- Hierarchical
  - least specific to the left ("128")
- Mainly useful to machines
  - machine-readable reference to hosts and networks
- Size limited
  - short, fixed-sized addresses to maximize efficiency

Name Space: Domain Name Tree



- every node on the tree represents one or a set of resources
- every node on the tree has a label (**lrcsuns**) and a domain name (**lrcsuns.epfl.ch**)

DNS Names

- Node
  - label <= 63 characters (letters, digits, and -)
  - case-insensitive
- Name
  - list of labels separated by .
    - `www.epfl.ch.` (fully qualified domain name)
    - `lcawww` (local name - evaluated with respect to the local domain)
- Analogous to unix file names
  - /usr/local/bin/emacs (root of tree to the left)
  - www.trustmymail.com (root of tree to the right)

DNS Names

- Hierarchical naming authority
  - top level: ICANN (Internet Corporation For Assigned Names and Numbers)
  - any organization can apply to become authority for a subdomain, e.g.:
    - SWITCH for `ch.` and `li.`
    - EPFL for `epfl.ch.`
  - any authority can create subdomains and delegate recursively unilaterally

DNS Name Servers

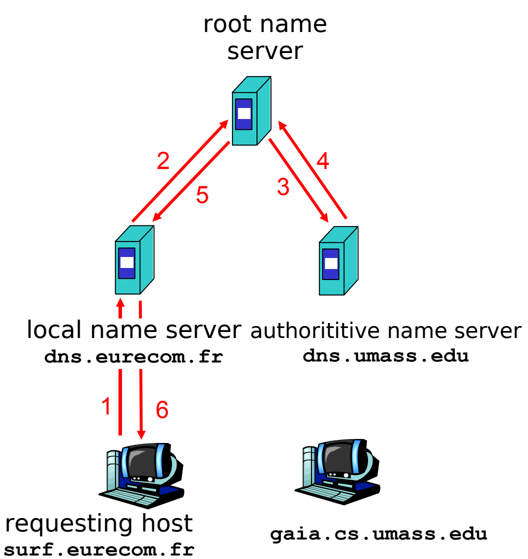
- No server has all name-to-IP address mappings
- Local name servers:
  - each ISP, company has local (default) name server
  - host DNS query first goes to local name server
- Authoritative name server:
  - for a host: stores that host's IP address, name
  - can perform name/address translation for that host's name

- Why not centralize DNS?
  - single point of failure
  - traffic volume
  - distant centralized database
  - maintenance
  - doesn't scale!

DNS Example 1

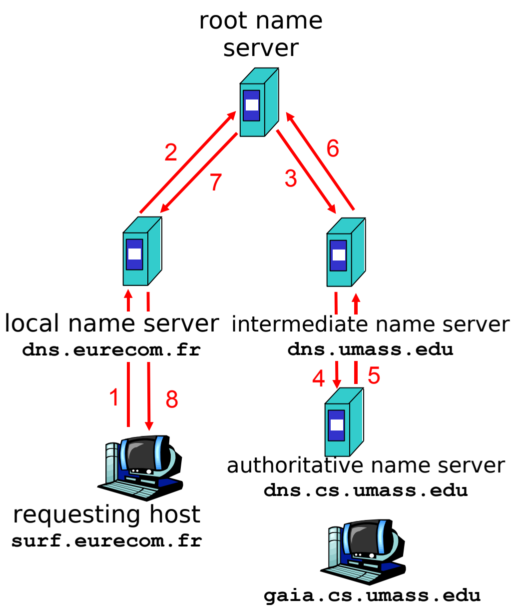
host **surf.eurecom.fr** wants IP address of **gaia.cs.umass.edu**

1. contacts its local DNS server, **dns.eurecom.fr**
2. **dns.eurecom.fr** contacts root name server, if necessary
3. root name server contacts authoritative name server, **dns.umass.edu**, if necessary



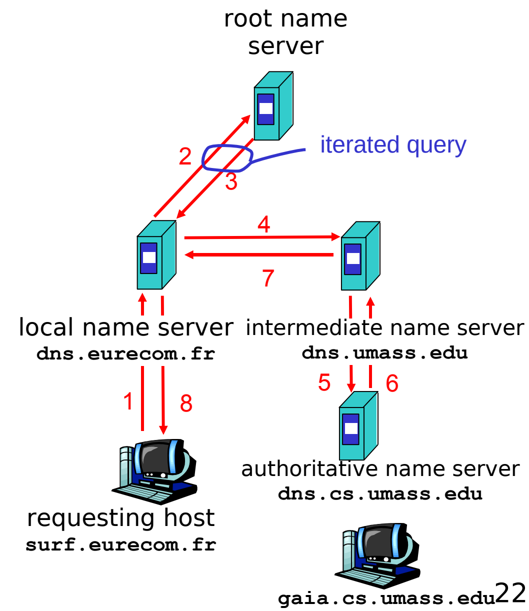
DNS Example 2

- Root name server:
  - may not know authoritative name server
  - may know intermediate name server: who to contact to find authoritative name server



DNS Example 3

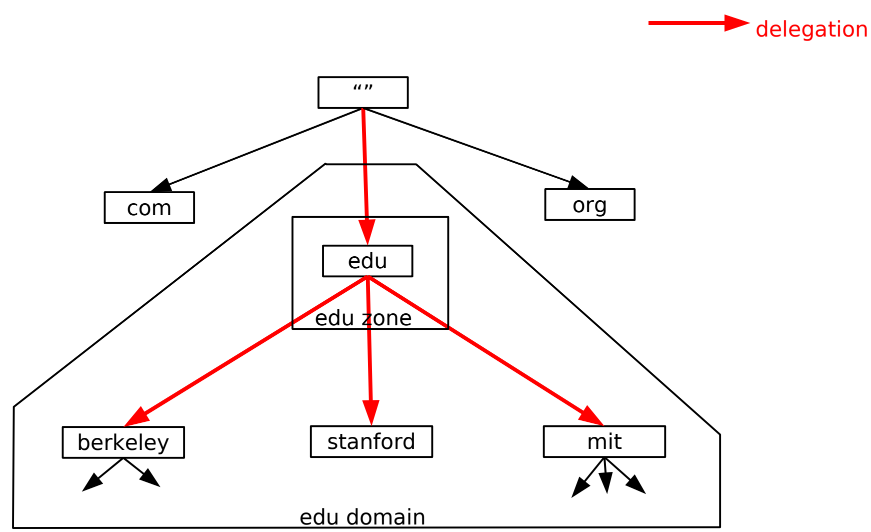
- recursive query:
  - puts burden of name resolution on contacted name server
  - heavy load?
- iterated query:
  - contacted server replies with name of server to contact
  - "I don't know this name, but ask this server"



Name Management

- Zone = a connected subset of nodes
  - property: a zone has one single node closest to the root = top node, used to name the zone
  - name authority matches zone boundaries:
    - names and subzones, can be created and deleted by the authority responsible for a zone; examples:
      - **zurich.ibm.com** is a subzone of **ibm.com**
      - zone **zurich.ibm.com.** has authority delegation from **ibm.com.**
  - at least 1 name server per zone (port 53)
    - primary, secondary - copy of the primary
    - **/etc/resolv.conf**: **nameserver 128.178.15.7**
      domain epfl.fr
    - replication - secondary servers
    - cache - data kept for 1 day

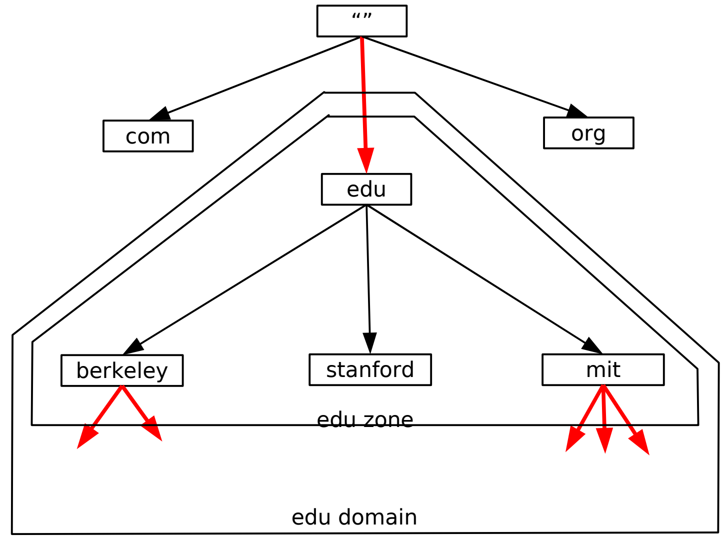
Zones and Domains



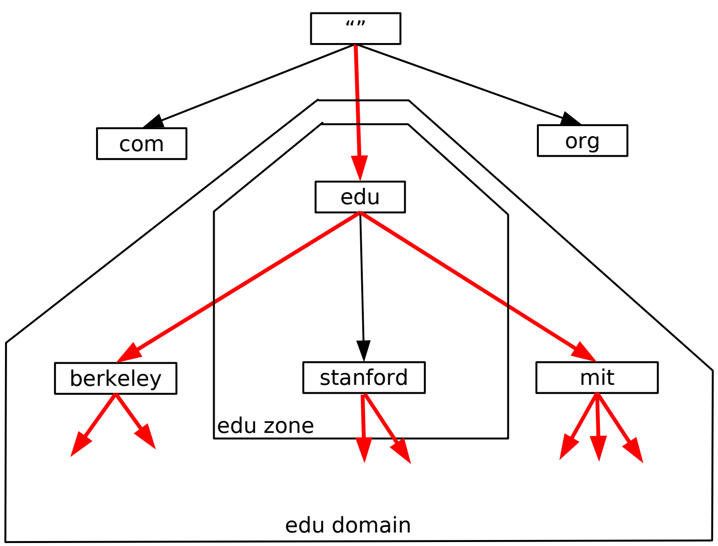
Zones and Domains

- Domains:
  - subtrees of the name space
  - domain x.y.z contains all nodes below x.y.z
  - independent of delegation relationships
- Zones:
  - nodes in name tree under single administrative control
  - zone x.y.z does not contain those nodes below x.y.z for which the zone delegates to another zone
  - delegation relationships define its boundaries

Zones and Domains

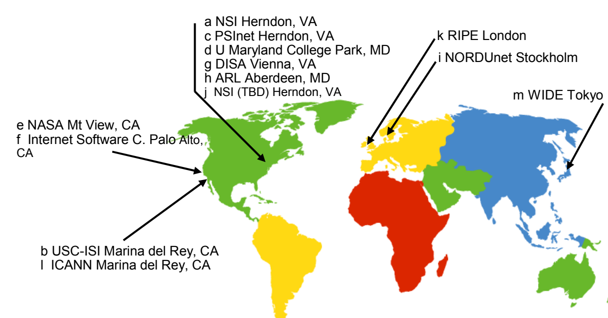


Zones and Domains



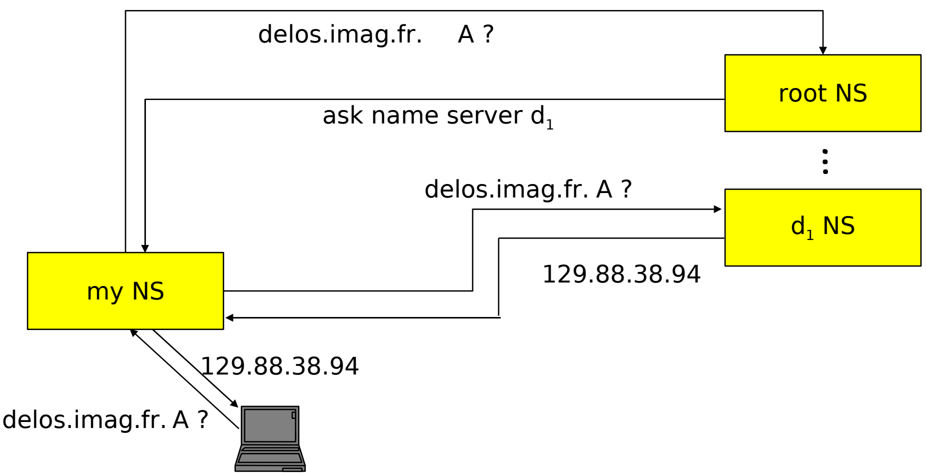
DNS Root Name Servers

- Contacted by local name server that can not resolve name
- Root name server:
  - contacts authoritative name server if name mapping not known
  - gets mapping
  - returns mapping to local name server



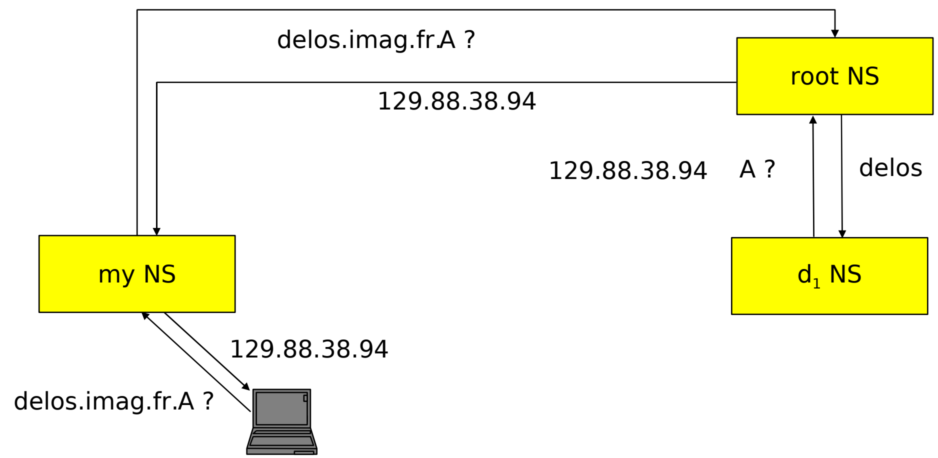
13 root name servers worldwide

Iterative Query



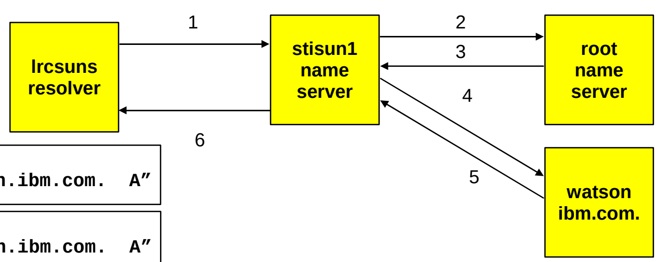
- Note: servers usually issue iterative queries to other servers

Recursive Query



- Note: resolvers always issue recursive queries to their local nameservers

Example: Query Processing



1
```
query, RD=yes
question = "www.zurich.ibm.com.   A"
```

2,4
```
query, RD=no
question = "www.zurich.ibm.com.   A"
```

3
```
answer
question = "www.zurich.ibm.com.      A"
answer = ""
authority= "ibm.com.   NS  watson.ibm.com.
                       NS  ns.austin.ibm.com.
                       NS  ns.almaden.ibm.com."
additional="watson.ibm.com.    A 192.35.232.34
            ns.austin.ibm.com. A 129.34.139.4
            ns.almaden.ibm.com A 198.4.83.134"
```

5,6
```
answer
question = "www.zurich.ibm.com.    A"
answer = "www.zurich.ibm.com.    A  193.5.61.131"
```


Reverse Mapping IP-address → name

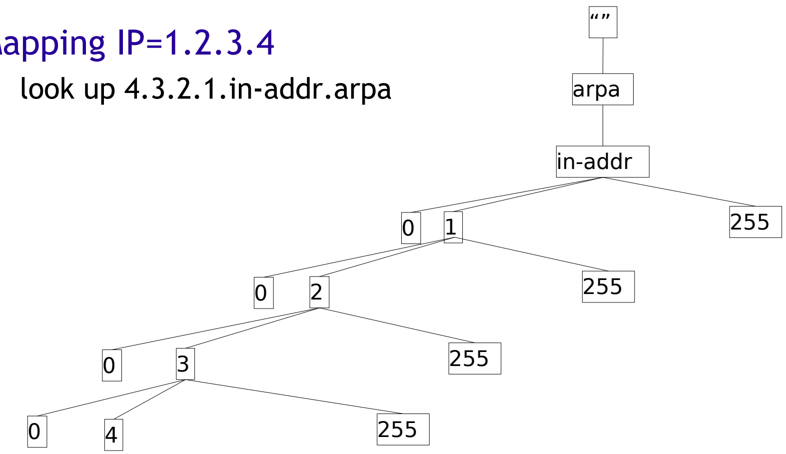
- Question:
  - How can we find the name(s) that an IP address corresponds to?
- Answer:
  - Conceptually, just search through all resource records and find the ones that match
- How to do this in a distributed way?

Reverse Mapping

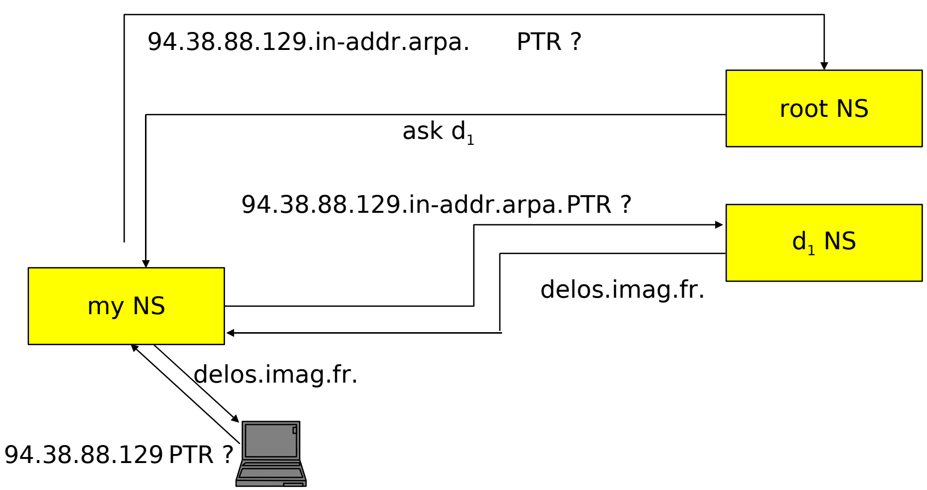
- Key observation:
  - IP address space is also hierarchical
  - ...but this hierarchy has nothing to do with naming hierarchy!
    - Example: tinycorp get IP addresses `100.101.102.0-100.101.102.255` from their ISP, and the name "`tinycorp.com`" from an ICANN-accredited registrar
- Solution:
  - build an additional domain that maintains this mapping: `in-addr.arpa`

In-addr.arpa Domain for Reverse Lookups

- Mapping IP=1.2.3.4
  - look up 4.3.2.1.in-addr.arpa



Pointer Query: IP Address back to Name



Performance and Robustness

- Replication
  - multiple servers with identical zone data
  - load balancing and failover
- Caching: once (any) name server learns mapping, it caches, i.e., remembers, this mapping
  - cache entries timeout (disappear) after some time: TTL (time to live) defined by authoritative name server
  - reduce traffic by creating "shortcuts" in walking down the tree

Replication

- Zone data is replicated
  - primary server holds master file on disk
  - secondary servers poll primary servers (ex: every 3 hours)
    - using the SERIAL field in the zone data
    - copying is called zone transfer; uses TCP (queries usually use UDP)
  - changes in zone data by system manager:
    - update master file
    - signal primary name server to reload; new value of SERIAL field automatically created
    - secondary servers will discover the change automatically
  - zone data in secondary servers is authoritative

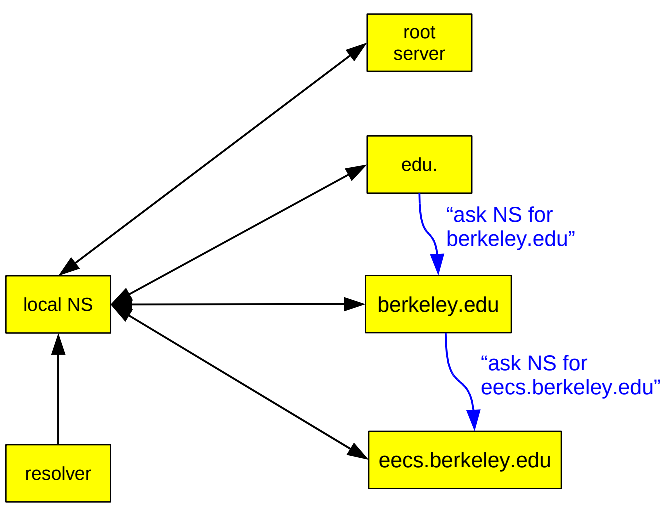
Server Selection

- How does a name server select among multiple potential servers in a lookup?
  - we'd like to use "close" servers
  - example: neste.com name servers + RTTs (from ping):

```
nestle.com        nameserver = ns2.nesusa.com.     RTT=200ms
nestle.com        nameserver = dns2.nestec.ch.     RTT= 30ms
nestle.com        nameserver = ns1.nesusa.com      RTT=200ms
```

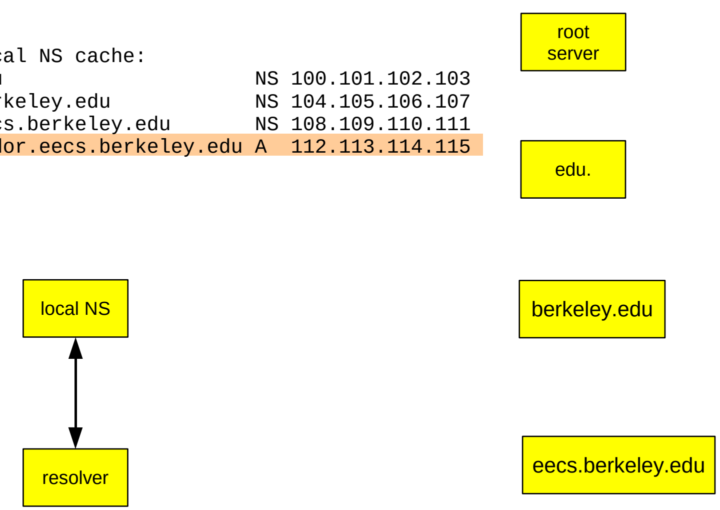
- Solution:
  - name server measures RTT of queries it sends to servers
  - over time, it will converge to using the closest and best performing of potential servers

Caching: Request for tudor.eecs.berkeley.edu



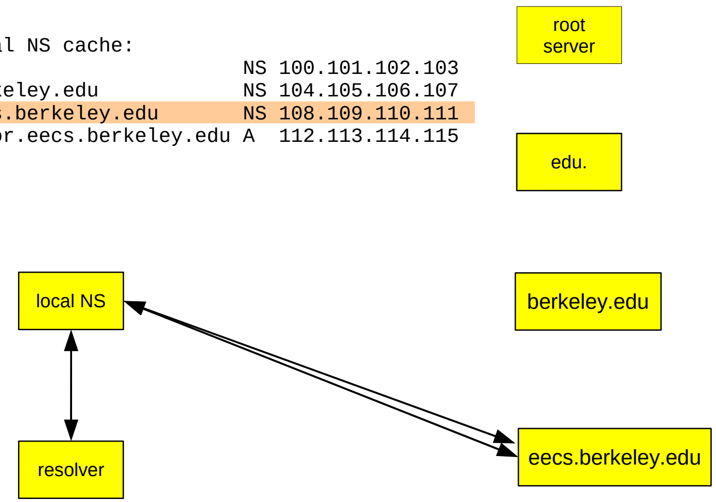
Caching: Subsequent Request for tudor.eecs.berkeley.edu

```
local NS cache:
edu                     NS 100.101.102.103
berkeley.edu            NS 104.105.106.107
eecs.berkeley.edu       NS 108.109.110.111
tudor.eecs.berkeley.edu A  112.113.114.115
```

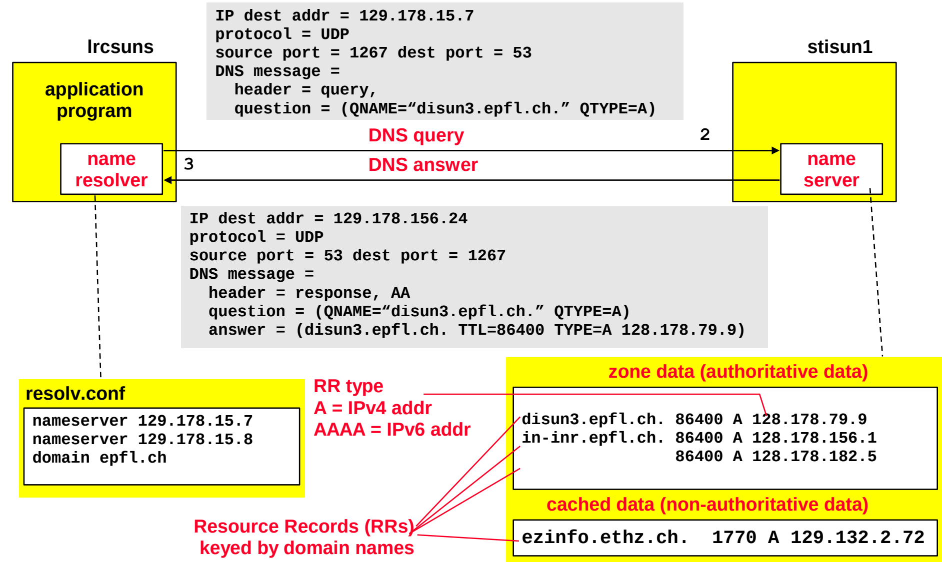


Caching: Subsequent Request for xyz.eecs.berkeley.edu

```
local NS cache:
edu                      NS 100.101.102.103
berkeley.edu             NS 104.105.106.107
eecs.berkeley.edu        NS 108.109.110.111
tudor.eecs.berkeley.edu  A  112.113.114.115
```



Details of a Query



DNS Records

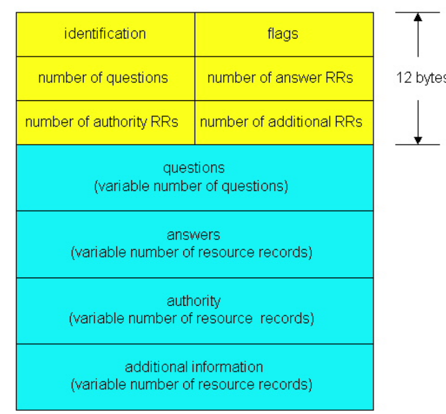
DNS: distributed db storing resource records (RR)

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

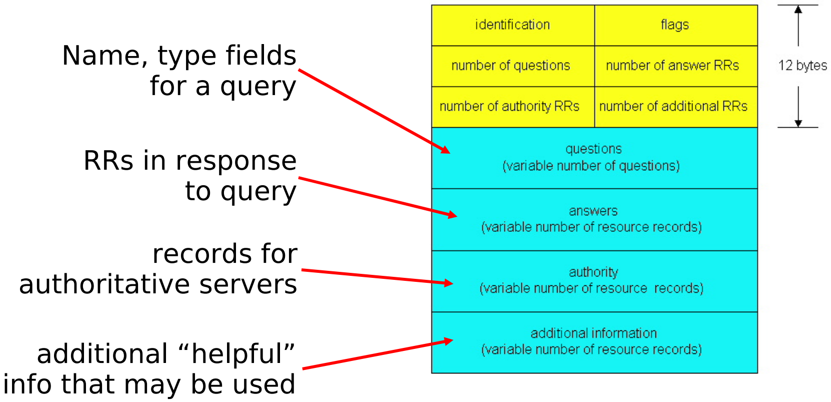
- Type=A
  - **name** is hostname
  - **value** is IP address
- Type=NS
  - **name** is domain (e.g. foo.com)
  - **value** is IP address of authoritative name server for this domain
- Type=CNAME
  - **name** is alias name for some "canonical" (the real) name
    `www.ibm.com` is really `servereast.backup2.ibm.com`
  - **value** is canonical name
- Type=MX
  - **value** is name of mailserver associated with **name**

DNS Protocol, Messages

- DNS protocol:
  - query and reply messages, both with same message format
  - usually uses UDP: query+reply fit in single packet, delay important, reliability handled by DNS itself
- Message header
  - identification: 16 bit # for query, reply to query uses same #
  - flags:
    - query or reply
    - recursion desired
    - recursion available
    - reply is authoritative



DNS Protocol, Messages



nslookup: Look up a Host

```
$ nslookup www.zurich.ibm.com
Server:  stisun1.epfl.ch
Address:  128.178.15.8
Non-authoritative answer:
Name:    www.zurich.ibm.com
Address:  193.5.61.131
```

- Origin of information
  - “non-authoritative”: from some NS's cache
  - “authoritative”: from (one of the) authoritative servers

nslookup: Look up Nameserver

- Asking for name-server for zone zurich.ibm.com

```
$ nslookup -querytype=NS zurich.ibm.com
Server:        128.178.15.7
Address:        128.178.15.7#53

Non-authoritative answer:
zurich.ibm.com  nameserver = ns1.emea.ibm.com.
zurich.ibm.com  nameserver = ns2.emea.ibm.com.
zurich.ibm.com  nameserver = internet-server.zurich.ibm.com.
zurich.ibm.com  nameserver = ns.watson.ibm.com.

Authoritative answers can be found from:
ns.watson.ibm.com        internet address = 129.34.20.80
ns1.emea.ibm.com         internet address = 195.212.29.46
ns2.emea.ibm.com         internet address = 195.212.29.110
internet-server.zurich.ibm.com  internet address = 195.176.20.204
```


nslookup: Look up Nameserver

- Same as before, but ask server 129.34.20.80 rather than local server

```
$ nslookup -querytype=NS zurich.ibm.com 129.34.20.80
Server:        129.34.20.80
Address:        129.34.20.80#53

zurich.ibm.com  nameserver = ns1.emea.ibm.com.
zurich.ibm.com  nameserver = ns2.emea.ibm.com.
zurich.ibm.com  nameserver = internet-server.zurich.ibm.com.
zurich.ibm.com  nameserver = ns.watson.ibm.com.
```

nslookup: Obtaining Root Servers

- Querying for name-servers for "." - the root!

```
$ nslookup -querytype=NS .
Server:        128.178.15.7
Address:       128.178.15.7#53

Non-authoritative answer:
.      nameserver = L.ROOT-SERVERS.NET.
.      nameserver = M.ROOT-SERVERS.NET.
.      nameserver = A.ROOT-SERVERS.NET.
.      nameserver = B.ROOT-SERVERS.NET.
.      nameserver = C.ROOT-SERVERS.NET.
.      nameserver = D.ROOT-SERVERS.NET.
.      nameserver = E.ROOT-SERVERS.NET.
.      nameserver = F.ROOT-SERVERS.NET.
.      nameserver = G.ROOT-SERVERS.NET.
.      nameserver = H.ROOT-SERVERS.NET.
.      nameserver = I.ROOT-SERVERS.NET.
.      nameserver = J.ROOT-SERVERS.NET.
.      nameserver = K.ROOT-SERVERS.NET.

.....
```

```
.....

Authoritative answers can be found from:
A.ROOT-SERVERS.NET     internet address = 198.41.0.4
B.ROOT-SERVERS.NET     internet address = 192.228.79.201
C.ROOT-SERVERS.NET     internet address = 192.33.4.12
D.ROOT-SERVERS.NET     internet address = 128.8.10.90
E.ROOT-SERVERS.NET     internet address = 192.203.230.10
F.ROOT-SERVERS.NET     internet address = 192.5.5.241
G.ROOT-SERVERS.NET     internet address = 192.112.36.4
H.ROOT-SERVERS.NET     internet address = 128.63.2.53
I.ROOT-SERVERS.NET     internet address = 192.36.148.17
J.ROOT-SERVERS.NET     internet address = 192.58.128.30
K.ROOT-SERVERS.NET     internet address = 193.0.14.129
L.ROOT-SERVERS.NET     internet address = 198.32.64.12
M.ROOT-SERVERS.NET     internet address = 202.12.27.33
```


nslookup: Reverse Mapping IP → name

```
$ nslookup -querytype=PTR 193.5.61.131
Server:  stisun1.epfl.ch
Address:  128.178.15.8
131.61.5.193.in-addr.arpa   name = uetliberg.zurich.ibm.ch
61.5.193.in-addr.arpa   nameserver = ns1.zurich.ibm.ch
61.5.193.in-addr.arpa   nameserver = scsnms.switch.ch
61.5.193.in-addr.arpa   nameserver = swidir.switch.ch
ns1.zurich.ibm.ch       internet address = 193.5.61.131
scsnms.switch.ch        internet address = 130.59.10.30
scsnms.switch.ch        internet address = 130.59.1.30
swidir.switch.ch        internet address = 130.59.72.10
```


nslookup: Other Points

- Interactive and noninteractive modes
  - Interactive: session with its own prompt, issue commands
  - Noninteractive: everything from command line (like preceding examples)
- Can behave like a resolver or like a name server
  - E.g., can issue both recursive (like resolver) or iterative (usually done by name servers) queries
- Option to see query and response messages
  - Debug option

DNS: Summary

- Hierarchical name space
  - Natural way to delegate portions of the space
  - Natural way to distribute mapping functionality
- Name servers all over the world
  - Well-known set of root servers
  - Note: separate name-spaces can be created by using a different set of roots!
- Scalable
  - Distribution and authority delegation
  - Caches for efficiency (reduce traffic)
  - Replication for fault tolerance (server outage)

DNS: Summary

- One of the key features of the Internet
  - ...and key source of problems (e.g., delay)!
- Most popular implementation: BIND
- Recent trends:
  - DNS increasingly used for sophisticated tasks it was not originally designed for, e.g., load-balancing among web servers
  - Security