CSCI-1680
DNS

Chen Avin

Based partly on lecture notes by Scott Shenker and John Jannotti and Rodrigo Fonseca

And “Computer Networking: A Top Down Approach” - 6th edition

Host names and IP Addresses

- **IP Addresses**
  - Numerical address appreciated by routers
  - Fixed length, binary numbers
  - Hierarchical, related to host location (in the network)
  - Examples: 128.148.32.110, 212.58.224.138
- **Host names**
  - Mnemonics appreciated by humans
  - Variable length, ASCII characters
  - Provide little (if any) information about location
  - Examples: www.cs.brown.edu, bbc.co.uk

Separating Naming and Addressing

- **Names are easier to remember**
  - www.cnn.com vs 157.166.224.26
- **Addresses can change underneath**
  - e.g, renumbering when changing providers
- **Name could map to multiple addresses**
  - www.cnn.com maps to at least 6 ip addresses
  - Enables
    - Load balancing
    - Latency reduction
    - Tailoring request based on requester's location/device/identity
- **Multiple names for the same address**
  - Aliases: www.cs.brown.edu and cs.brown.edu
  - Multiple servers in the same node (e.g., apache virtual servers)

Scalable Address <-> Name Mappings

- **Originally kept in a local file, hosts.txt**
  - Flat namespace
  - Central administrator kept master copy (for the Internet)
  - To add a host, emailed admin
  - Downloaded file regularly
- **Completely impractical today**
  - File would be huge (gigabytes)
  - Traffic implosion (lookups and updates)
    - Some names change mappings every few days (dynamic IP)
  - Single point of failure
  - Impractical politics (security, ownership, etc…)

Goals for an Internet-scale name system

- **Scalability**
  - Must handle a huge number of records
    - With some software synthesizing names on the fly
  - Must sustain update and lookup load
- **Distributed Control**
  - Let people control their own names
- **Fault Tolerance**
  - Minimize lookup failures in face of other network problems

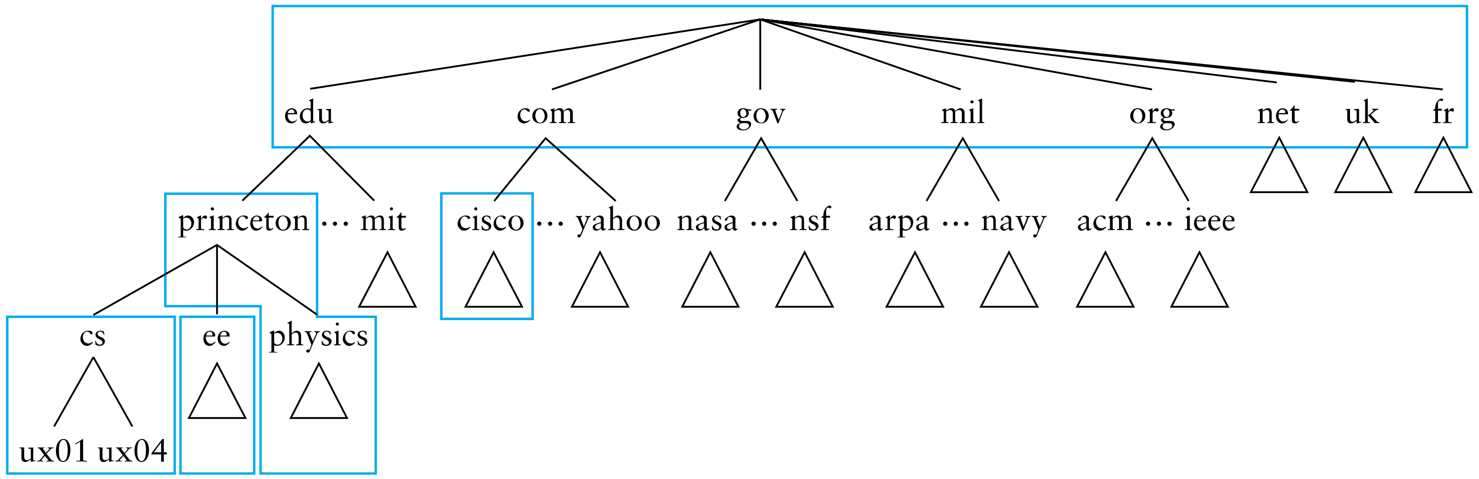
The good news

- **Properties that make these goals easier to achieve**
  1. Read-mostly database

     Lookups MUCH more frequent than updates
  2. Loose consistency

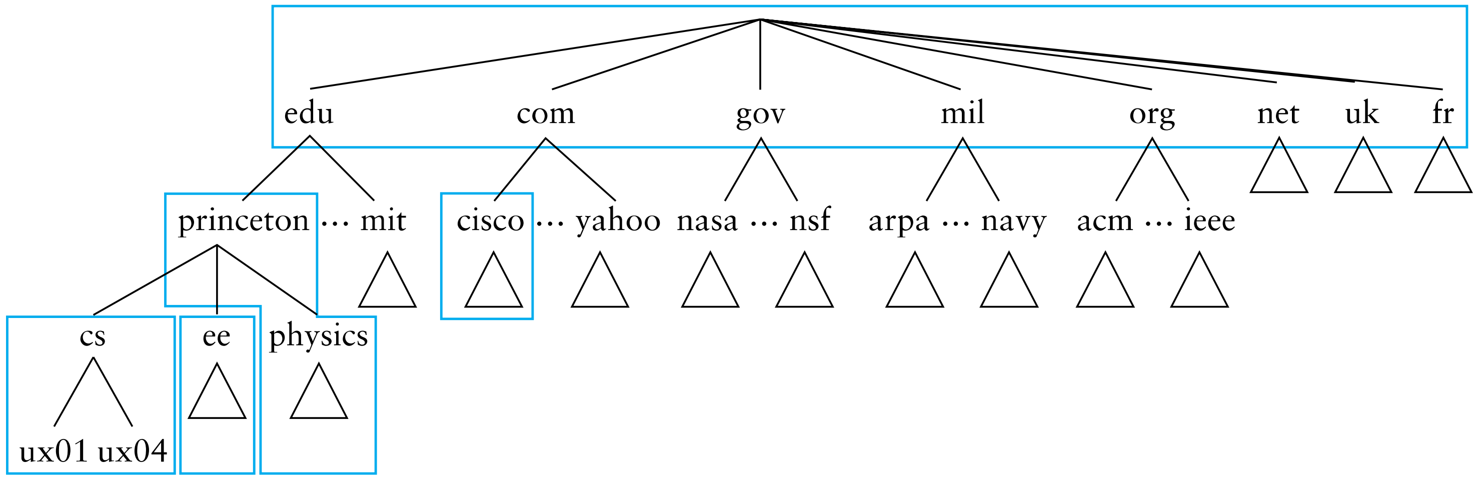
     When adding a machine, not end of the world if it takes minutes or hours to propagate
- **These suggest aggressive *caching***
  - Once you’ve lookup up a hostname, remember
  - Don’t have to look again in the near future

Domain Name System (DNS)



- **Hierarchical namespace broken into *zones***
  - root (.), edu., brown.edu., cs.brown.edu.,
  - Zones separately administered :: delegation
  - Parent zone tells you how to find servers for subdomains
- **Each zone served from multiple replicated servers**

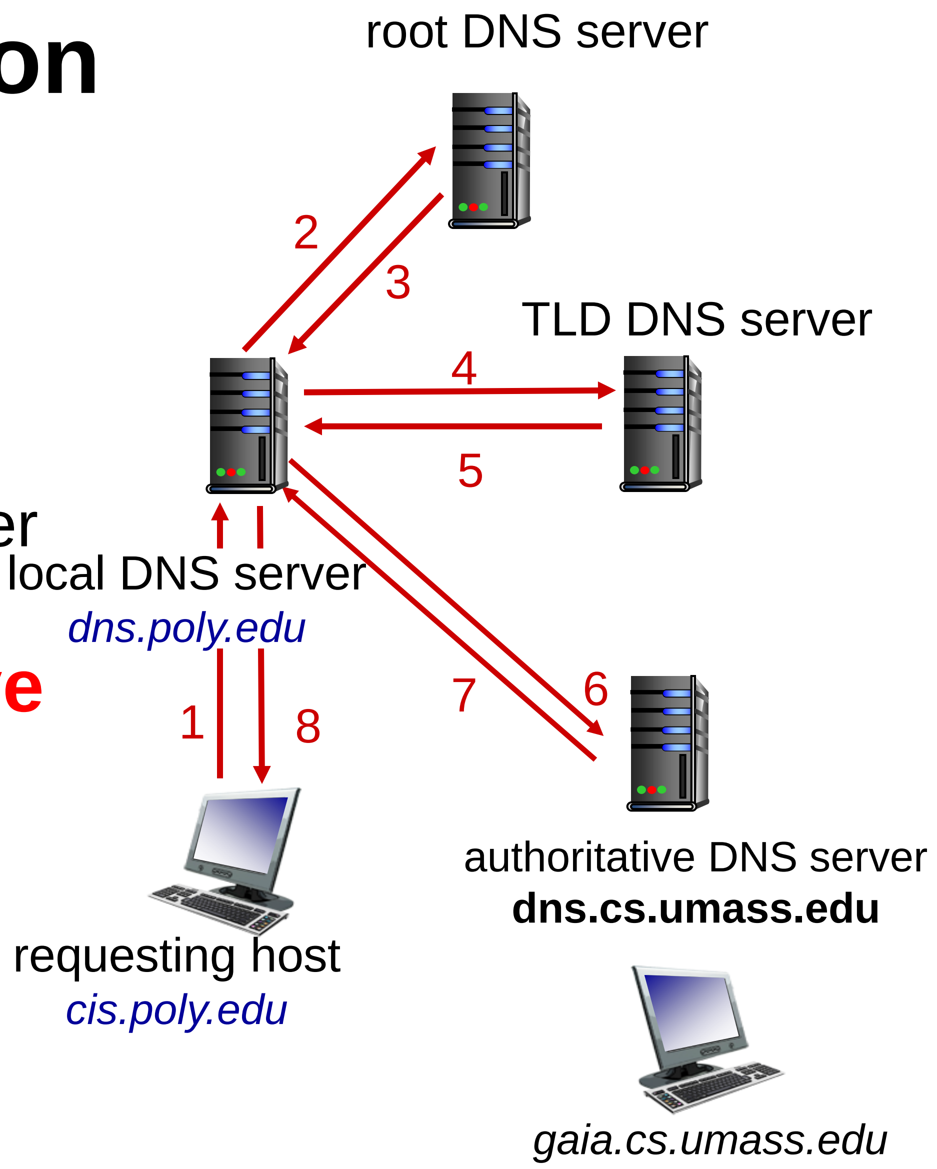
DNS Architecture



- **Hierarchy of DNS servers**
  - Root servers
  - Top-level domain (TLD) servers
  - Authoritative DNS servers
- **Performing the translation**
  - Local DNS servers (aka “default name server”)
  - Resolver software

Resolver operation

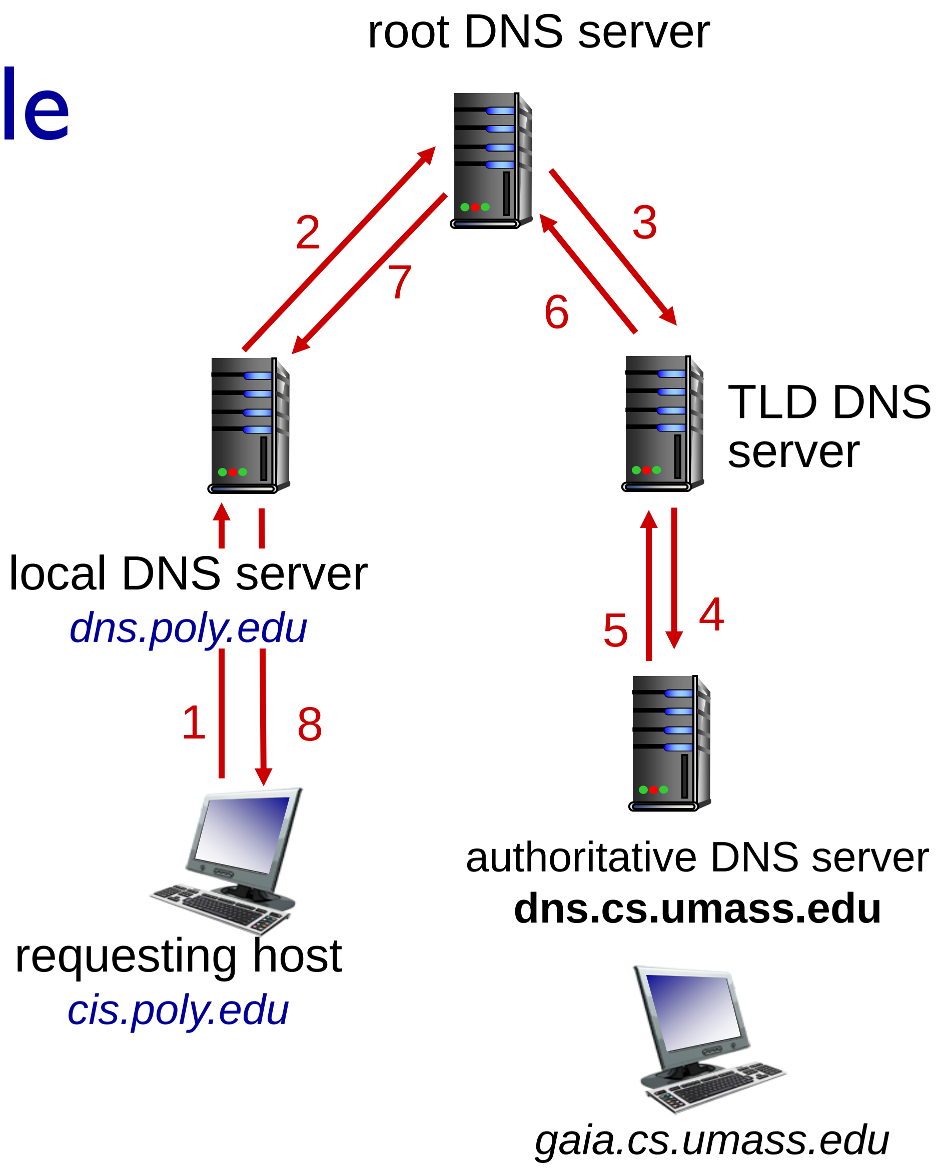
- **Apps make recursive queries to local DNS server (1)**
  - Ask server to get answer for you
- **Server makes iterative queries to remote servers (2,4,6)**
  - Ask servers who to ask next
  - Cache results aggressively



DNS name resolution example

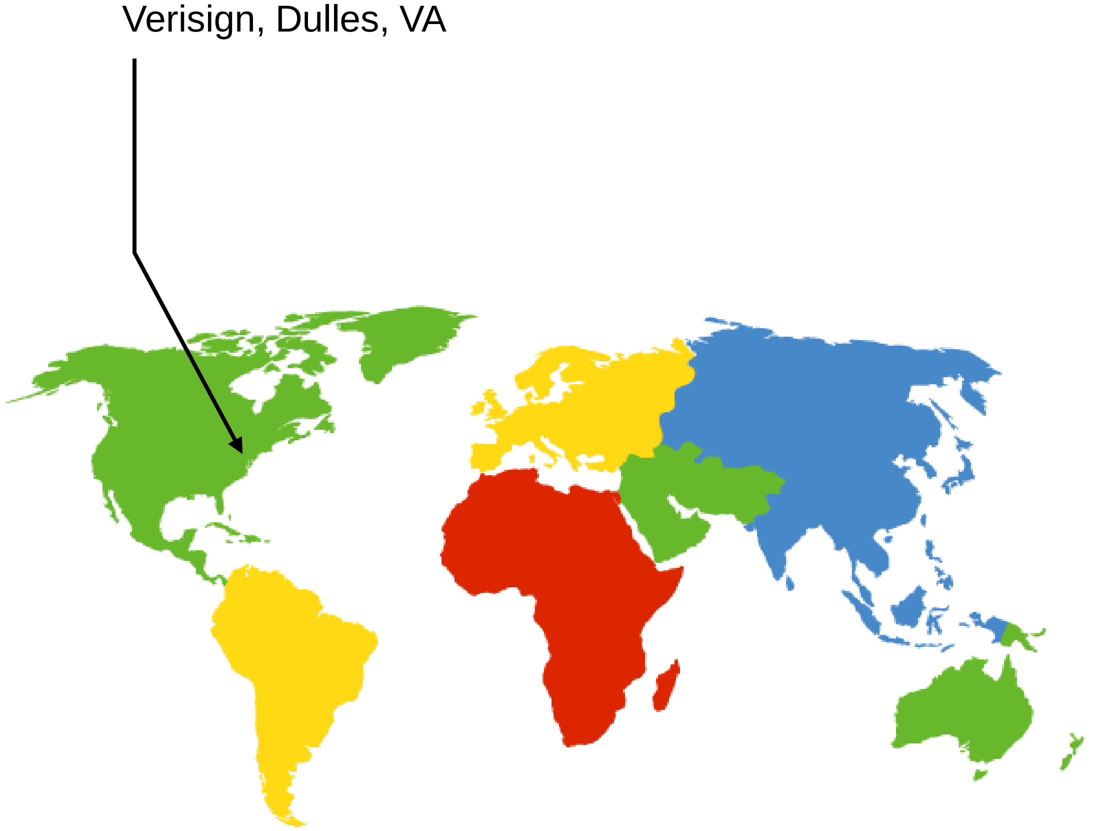
recursive query:

- puts burden of name resolution on contacted name server
- heavy load at upper levels of hierarchy?



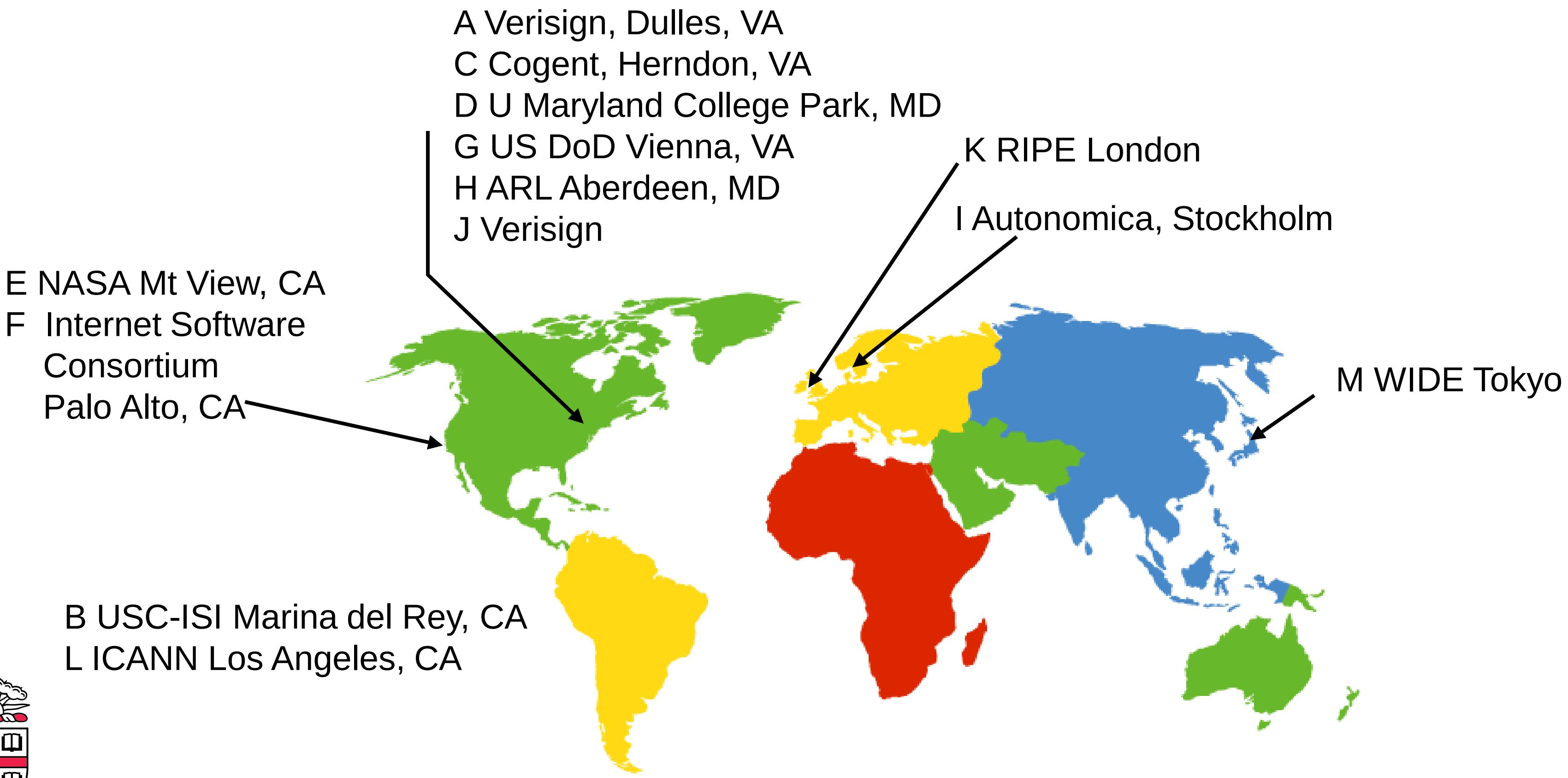
DNS Root Server

- Located in Virginia, USA
- How do we make the root scale?



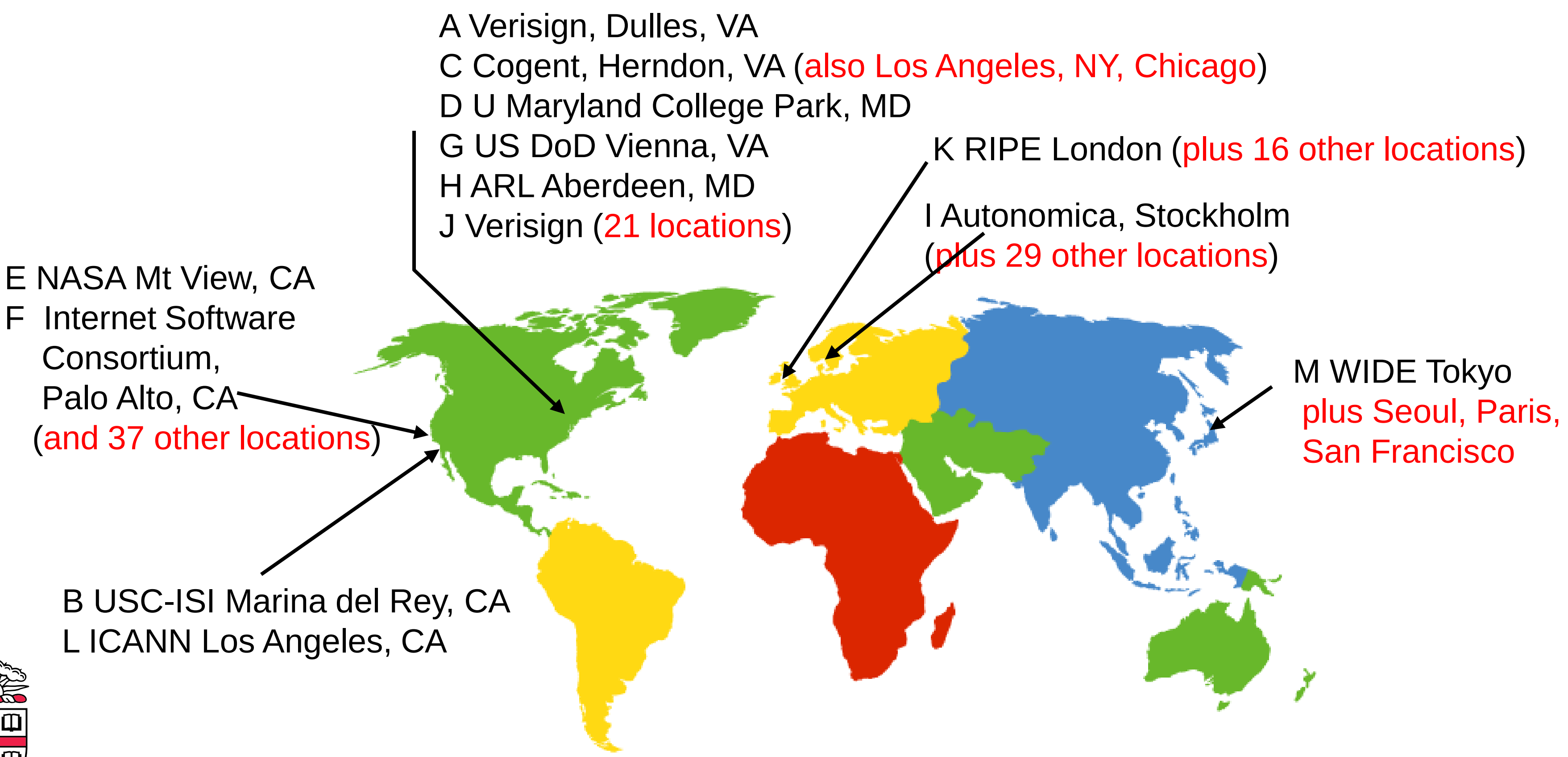
DNS Root Servers

- **13 Root Servers (www.root-servers.org)**
  - Labeled A through M (e.g, A.ROOT-SERVERS.NET)
- **Does this scale?**



DNS Root Servers

- **13 Root Servers (www.root-servers.org)**
  - Labeled A through M (e.g, A.ROOT-SERVERS.NET)
- **Replication via anycasting**


A Verisign, Dulles, VA
C Cogent, Herndon, VA (also Los Angeles, NY, Chicago)
D U Maryland College Park, MD
G US DoD Vienna, VA
H ARL Aberdeen, MD
J Verisign (21 locations)
K RIPE London (plus 16 other locations)
I Autonomica, Stockholm (plus 29 other locations)
E NASA Mt View, CA
F Internet Software Consortium, Palo Alto, CA (and 37 other locations)
M WIDE Tokyo plus Seoul, Paris, San Francisco
B USC-ISI Marina del Rey, CA
L ICANN Los Angeles, CA

TLD and Authoritative DNS Servers

- **Top Level Domain (TLD) servers**
  - Generic domains (e.g., com, org, edu)
  - Country domains (e.g., uk, br, tv, in, ly)
  - Special domains (e.g., arpa)
  - Typically managed professionally
- **Authoritative DNS servers**
  - Provides public records for hosts at an organization
    - e.g, for the organization’s own servers (www, mail, etc)
  - Can be maintained locally or by a service provider

Reverse Mapping

- **How do we get the other direction, IP address to name?**
- **Addresses have a natural hierarchy:**
  - 128.148.34.7
- **But, most significant element comes first**
- **Idea: reverse the numbers: 7.34.148.128 …**
  - and look that up in DNS
- **Under what TLD?**
  - Convention: in-addr.arpa
  - Lookup 7.34.148.128.in-addr.arpa
  - in6.arpa for IPv6

DNS Caching

- **All these queries take a long time!**
  - And could impose tremendous load on root servers
  - This latency happens before any real communication, such as downloading your web page
- **Caching greatly reduces overhead**
  - Top level servers very rarely change
  - Popular sites visited often
  - Local DNS server caches information from many users
- **How long do you store a cached response?**
  - Original server tells you: TTL entry
  - Server deletes entry after TTL expires

Negative Caching

- **Remember things that don’t work**
  - Misspellings like www.cnn.comm, ww.cnn.com
- **These can take a long time to fail the first time**
  - Good to cache negative results so it will fail faster next time
- **But negative caching is optional, and not widely implemented**

DNS Protocol

- **TCP/UDP port 53**
- **Most traffic uses UDP**
  - Lightweight protocol has 512 byte message limit
  - Retry using TCP if UDP fails (e.g., reply truncated)
- **TCP requires messages boundaries**
  - Prefix all messages with 16-bit length
- **Bit in query determines if query is recursive**

Resource Records

- **All DNS info represented as resource records (RR)**

  name [ttl] [class] type rdata

  - name: domain name
  - TTL: time to live in seconds
  - class: for extensibility, normally IN (1) “Internet”
  - type: type of the record
  - rdata: resource data dependent on the type
- **Two important RR types**
  - A – Internet Address (IPv4)
  - NS – name server
- **Example RRs**

```
www.cs.brown.edu.    86400   IN   A    128.148.32.110
cs.brown.edu.        86400   IN   NS   dns.cs.brown.edu.
cs.brown.edu.        86400   IN   NS   ns1.ucsb.edu.
```

Some important details

- **How do local servers find root servers?**
  - DNS lookup on a.root-servers.net ?
  - Servers configured with *root cache* file
  - Contains root name servers and their addresses

```
.                     3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.     3600000     A     198.41.0.4
...
```

- **How do you get addresses of other name servers?**
  - To obtain the address of www.cs.brown.edu, ask a.edu-servers.net, says a.root-servers.net
  - How do you find a.edu-servers.net?
  - Glue records: A records in parent zone

Example

```
dig . ns

dig +norec www.cs.brown.edu @a.root-servers.net

dig +norec www.cs.brown.edu @a.edu-servers.net

dig +norec www.cs.brown.edu @bru-ns1.brown.edu

    www.cs.brown.edu.  86400   IN   A    128.148.32.110
```

Structure of a DNS Message

```
+---------------------+
|        Header       |
+---------------------+
|       Question      | the question for the name server
+---------------------+
|        Answer       | RRs answering the question
+---------------------+
|      Authority      | RRs pointing toward an authority
+---------------------+
|      Additional     | RRs holding additional information
+---------------------+
```

- **Same format for queries and replies**
  - Query has 0 RRs in Answer/Authority/Additional
  - Reply includes question, plus has RRs
- **Authority allows for delegation**
- **Additional for glue, other RRs client might need**

Header format

```
                                1  1  1  1  1  1
  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      ID                       |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|QR|   Opcode  |AA|TC|RD|RA|   Z    |   RCODE   |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    QDCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ANCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    NSCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ARCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

- **Id: match response to query; QR: 0 query/1 response**
- **RCODE: error code.**
- **AA: authoritative answer, TC: truncated,**
- **RD: recursion desired, RA: recursion available**

Other RR Types

- **CNAME (canonical name): specifies an alias**

```
www.google.com.          446199   IN   CNAME  www.l.google.com.
www.l.google.com.   300  IN   A    72.14.204.147
```

- **MX record: specifies servers to handle mail for a domain (the part after the @ in email addr)**
  - Different for historical reasons
- **SOA (start of authority)**
  - Information about a DNS zone and the server responsible for the zone
- **PTR (reverse lookup)**

```
7.34.148.128.in-addr.arpa. 86400 IN   PTR  quanto.cs.brown.edu.
```

Reliability

- **Answers may contain several alternate servers**
- **Try alternate servers on timeout**
  - Exponential backoff when retrying same server
- **Use same identifier for all queries**
  - Don’t care which server responds, take first answer

Inserting a Record in DNS

- **Your new startup helpme.com**
- **Get a block of addresses from ISP**
  - Say 212.44.9.128/25
- **Register helpme.com at GoDaddy.com (for ex.)**
  - Provide name and address of your authoritative name server (primary and secondary)
  - Registrar inserts RR pair into the com TLD server:
    - helpme.com NS dns1.helpme.com
    - dns1.helpme.com A 212.44.9.129
- **Configure your authoritative server (dns1.helpme.com)**
  - Type A record for www.helpme.com
  - Type MX record for your mail server

Inserting a Record in DNS, cont

- **Need to provide reverse PTR bindings**
  - E.g., 212.44.9.129 -> dns1.helpme.com
- **Normally, these would go into the 9.44.212.in-addr.arpa zone**
- **Problem: you can't run the name server for that domain. Why not?**
  - Your block is 212.44.9.128/25, not 212.44.9.0/24
  - Whoever has 212.44.9.0/25 wouldn't be happy with you setting their PTR records
- **Solution: [RFC2317, Classless Delegation]**
  - Install CNAME records in parent zone, e.g:

129.9.44.212.in-addr.arpa CNAME 129.ptr.helpme.com

DNS Measurements (Data from MIT, 2000)

- **What was being looked up?**
  - 60% A, 25% PTR, 5% MX, 6% ANY
- **Latency**
  - Median ~100ms (90th percentile ~500ms)
- **Query packets per lookup: ~2.4**
- **Top 10% of domains → ~70% of lookups**
  - Great for caching!
- **9% of lookups are unique**
  - Caching can’t hit more than 91%
- **Cache hit rates actually ~75%**

DNS Measurements (Data from MIT, 2000)

- **Does DNS give back answers?**
  - ~23% of queries do not elicit an answer
  - ~13% return NXDOMAIN (or similar)
    - Mostly reverse lookups
  - Only ~64% of queries are successful
- **~63% of DNS packets in unanswered queries**
  - Failing queries are frequently retransmitted
  - 99.9% successful queries have <= 2 retransmissions

DNS Security

- **You go to starbucks, how does your browser find www.google.com?**
  - Ask local name server, obtained from DHCP
  - You implicitly trust this server
  - Can return any answer for google.com, including a malicious IP that poses as a man in the middle
- **How can you know you are getting correct data?**
  - Today, you can’t for all sources
  - HTTPS can help
  - DNSSEC extension allow you to verify

DNS Security 2 – Cache Poisoning

- Suppose you control evil.com. You receive a query for www.evil.com and reply:

```
;; QUESTION SECTION:
;www.evil.com.              IN     A

;; ANSWER SECTION:
www.evil.com.       300    IN     A      212.44.9.144

;; AUTHORITY SECTION:
evil.com.           600    IN     NS     dns1.evil.com.
evil.com.           600    IN     NS     google.com.

;; ADDITIONAL SECTION:
google.com.         5      IN     A      212.44.9.155
```

- Glue record pointing to your IP, not Google’s
- Gets cached!

Cache Poisoning # 2

- **But how do you get a victim to look up evil.com?**
- **You might connect to their mail server and send**
  - HELO www.evil.com
  - Which their mail server then looks up to see if it corresponds to your IP address (SPAM filtering)
- **Mitigation (bailiwick checking)**
  - Only accept glue records from the domain you asked for

Cache Poisoning

- **Another possibility: bad guy at Starbucks, can sniff or guess the ID field the local server will use**
  - Not hard if DNS server generates ID numbers sequentially
  - Can be done if you force the DNS server to look up something in *your* name server
  - Guessing has 1 in 65535 chance (Or does it?)
- **Now:**
  - Ask the local server to lookup google.com
  - Spoof the response from google.com using the correct ID
  - Bogus response arrives before legit one (maybe)
- **Local server caches first response it receives**
  - Attacker can set a long TTL

Countermeasures

- **Randomize id**
  - Used to be sequential
- **Randomize source port number**
  - Used to be the same for all requests from the server
- **Offers some protection, but attack still possible**

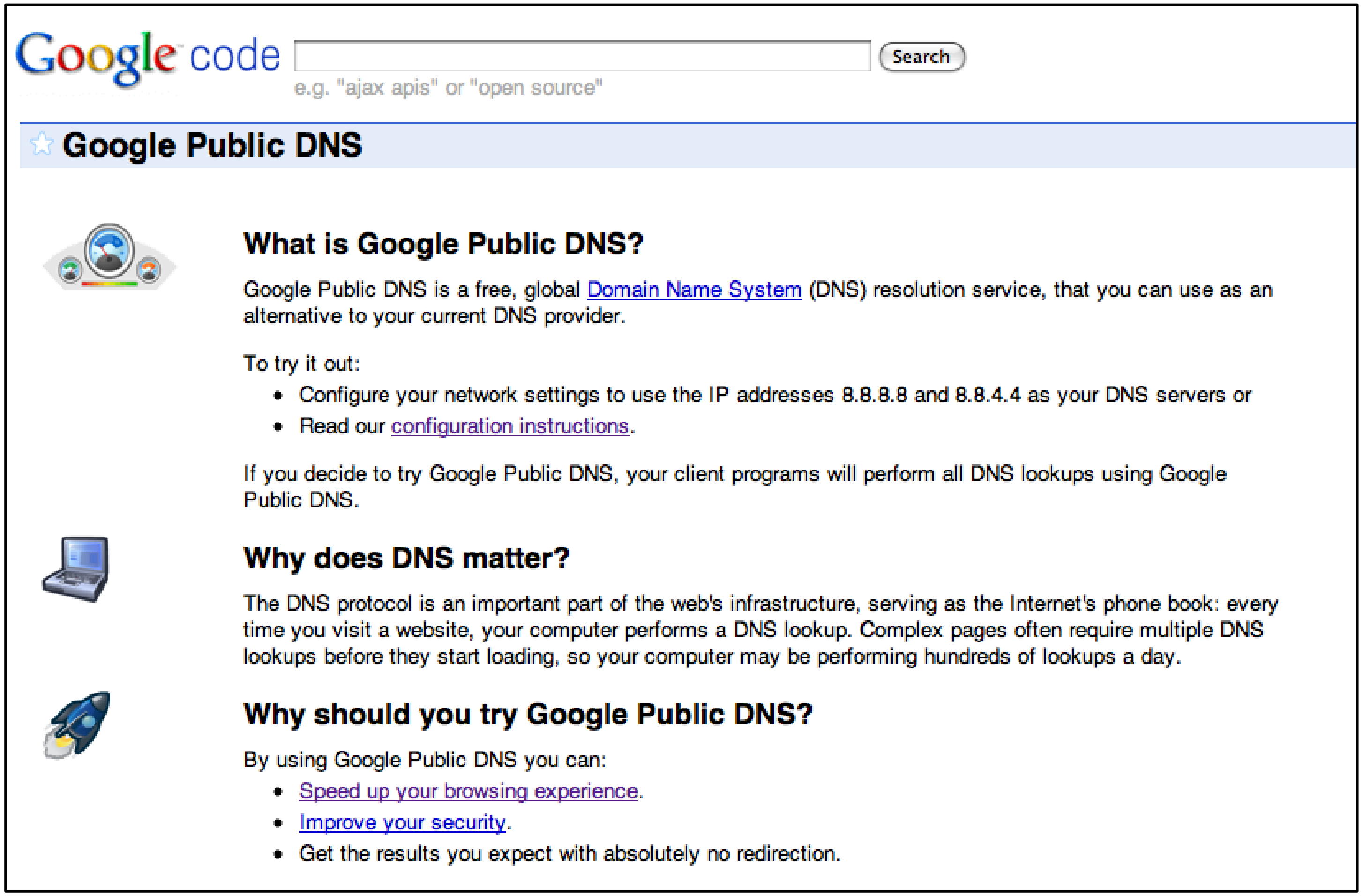
Solution: signatures

- Signature: cryptographic way to prove a party is who they say they are
- Requires a chain of trust
- DNSSEC deployment is underway

Some more DNS fun

- **You can use DNS to tunnel data!**
- **Steps:**
  - Start up a Name Server for a domain you control
  - Send info encoding data in the domain name part of a query
  - Server encodes response in a TXT record
- **Why? DNS is often *not* blocked in airports, etc**
- **This has been a final project in this class!**

Updates



Google Public DNS

What is Google Public DNS?

Google Public DNS is a free, global Domain Name System (DNS) resolution service, that you can use as an alternative to your current DNS provider.

To try it out:

- Configure your network settings to use the IP addresses 8.8.8.8 and 8.8.4.4 as your DNS servers or
- Read our configuration instructions.

If you decide to try Google Public DNS, your client programs will perform all DNS lookups using Google Public DNS.

Why does DNS matter?

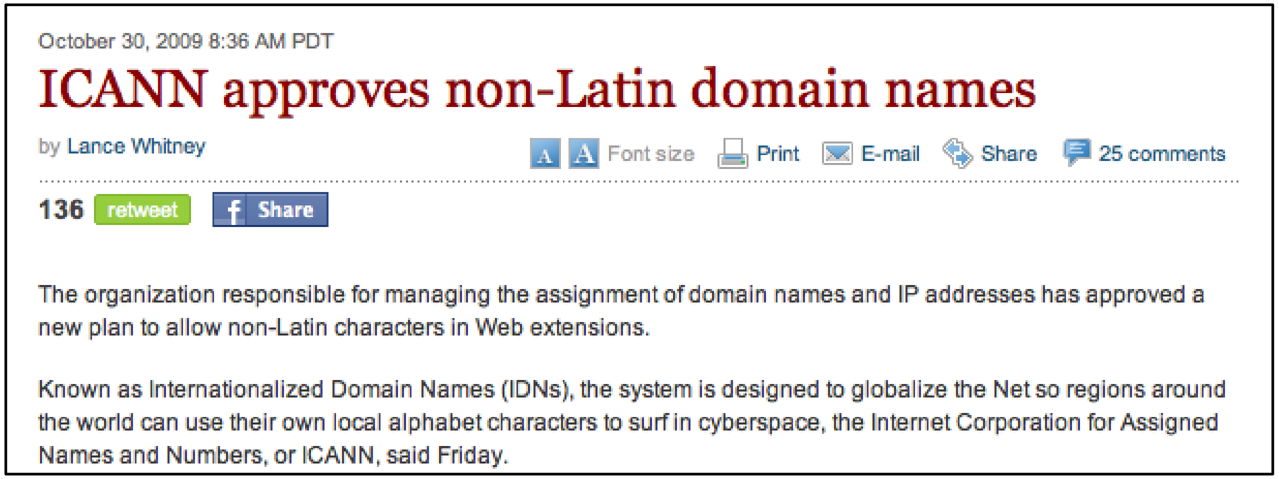
The DNS protocol is an important part of the web's infrastructure, serving as the Internet's phone book: every time you visit a website, your computer performs a DNS lookup. Complex pages often require multiple DNS lookups before they start loading, so your computer may be performing hundreds of lookups a day.

Why should you try Google Public DNS?

By using Google Public DNS you can:

- Speed up your browsing experience.
- Improve your security.
- Get the results you expect with absolutely no redirection.

Updates



October 30, 2009 8:36 AM PDT

ICANN approves non-Latin domain names

by Lance Whitney

The organization responsible for managing the assignment of domain names and IP addresses has approved a new plan to allow non-Latin characters in Web extensions.

Known as Internationalized Domain Names (IDNs), the system is designed to globalize the Net so regions around the world can use their own local alphabet characters to surf in cyberspace, the Internet Corporation for Assigned Names and Numbers, or ICANN, said Friday.