Introduction to Computer Networks

COSC 4377

Lecture 4

Spring 2012

January 30, 2012

Announcements

- HW2 due this week
- Start working on HW3

Today’s Topics

- HTTP Performance
- Domain Name System (DNS)

HTTP Performance

- What matters for performance?
- Depends on type of request
  - Lots of small requests (objects in a page)
  - Some big requests (large download or video)

Small Requests

- Latency matters
- RTT dominates
- Two major causes:
  - Opening a TCP connection
  - Actually sending the request and receiving response
  - And a third one: DNS lookup!
- Mitigate the first one with persistent connections (HTTP/1.1)
  - Which also means you don't have to "open" the connection each time

Browser Request

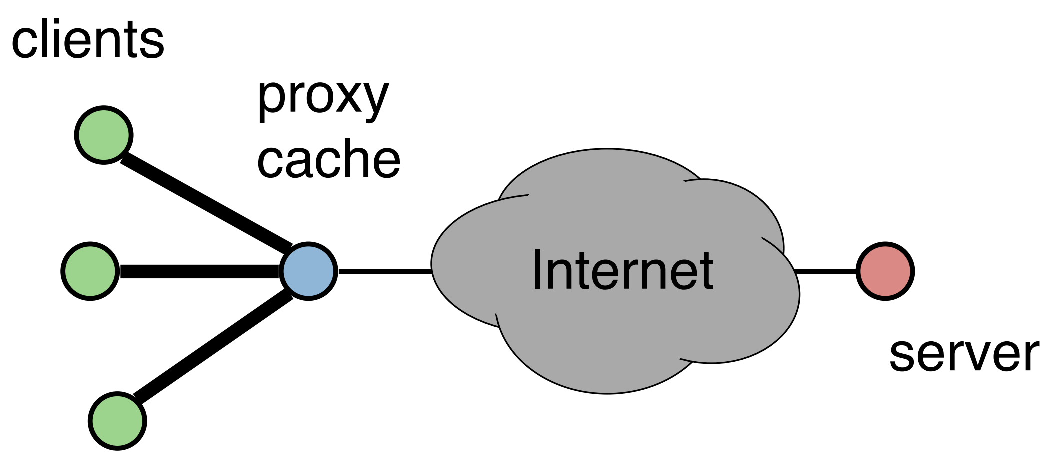
```
GET / HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Macinto ...
Accept: text/xml,application/xm ...
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

Small Requests (cont)

- Second problem is that requests are serialized
  - Similar to stop-and-wait protocols!
- Two solutions
  - Pipelined requests (similar to sliding windows)
  - Parallel Connections
    - HTTP standard says no more than 2 concurrent connections per host name
    - Most browsers use more (up to 8 per host, ~35 total)
  - How are these two approaches different?
  - http://en.wikipedia.org/wiki/HTTP_pipelining

Larger Objects

- Problem is throughput in bottleneck link
- Solution: HTTP Proxy Caching
  - Also improves latency, and reduces server load



HTTP Proxy Protocol

Find out how the protocol we just designed is different from protocols used by http client/proxy/server

Domain Name System

Host names and IP Addresses

- Host names
  - Mnemonics appreciated by humans
  - Variable length, ASCII characters
  - Provide little (if any) information about location
  - Examples: www.facebook.com, bbc.co.uk
- IP Addresses
  - Numerical address appreciated by routers
  - Fixed length, binary numbers
  - Hierarchical, related to host location (in the network)
  - Examples: 69.171.228.14, 212.58.241.131

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 (repeated names, 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

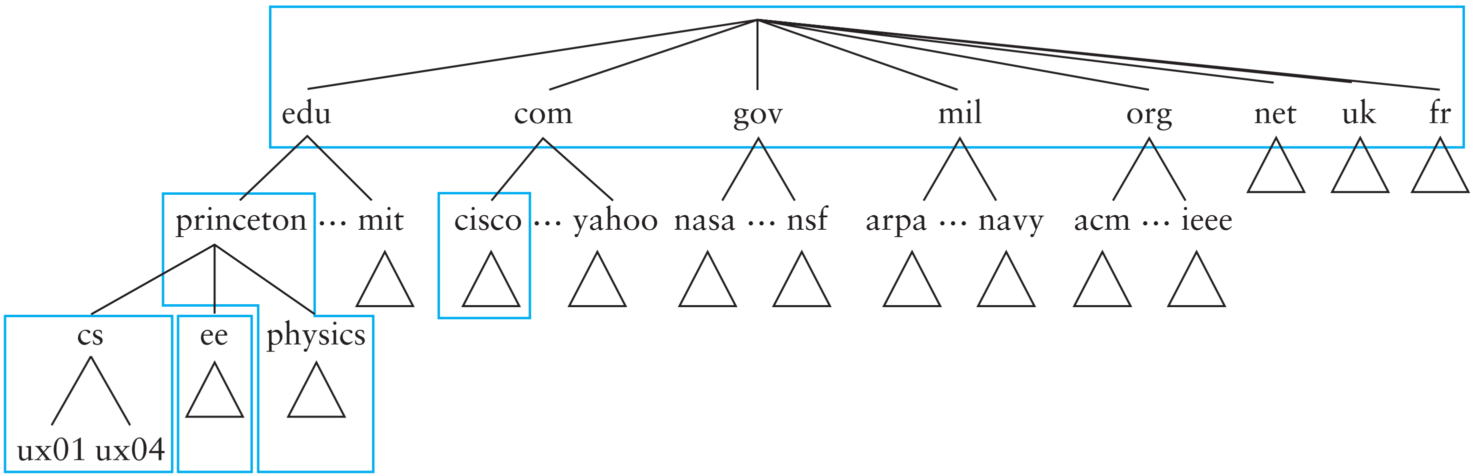
The good news

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

     Lookups MUCH more frequent than updates
  2. Loose consistency

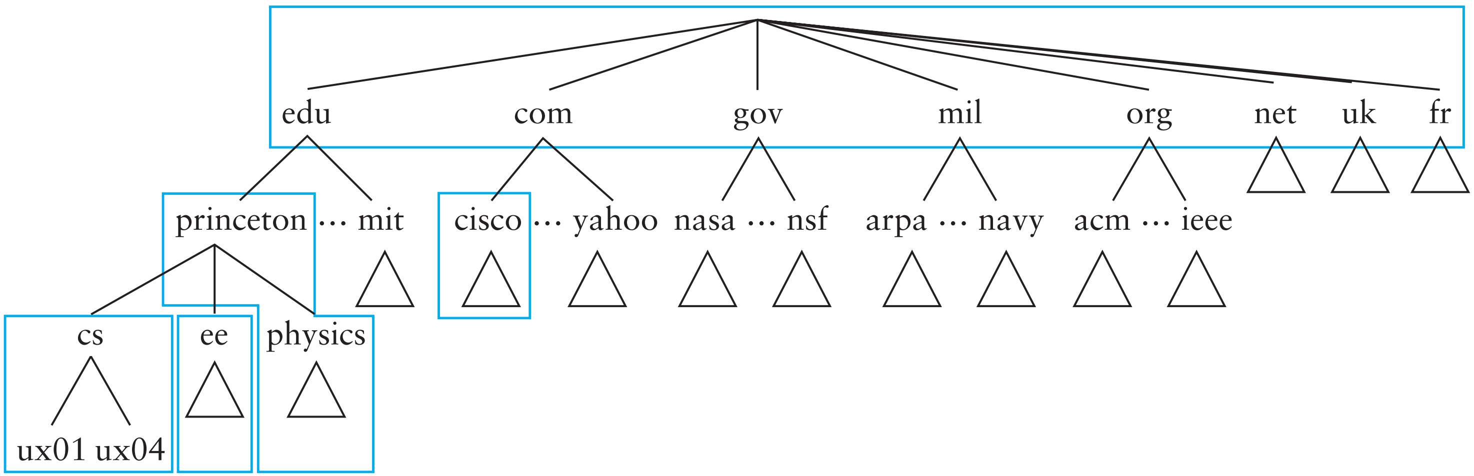
     When adding a machine, not end of the world if it takes minutes or hours to propagate
- These suggest aggressive caching
  - Once you've lookup up a hostname, remember
  - Don't have to look again in the near future

Domain Name System (DNS)



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

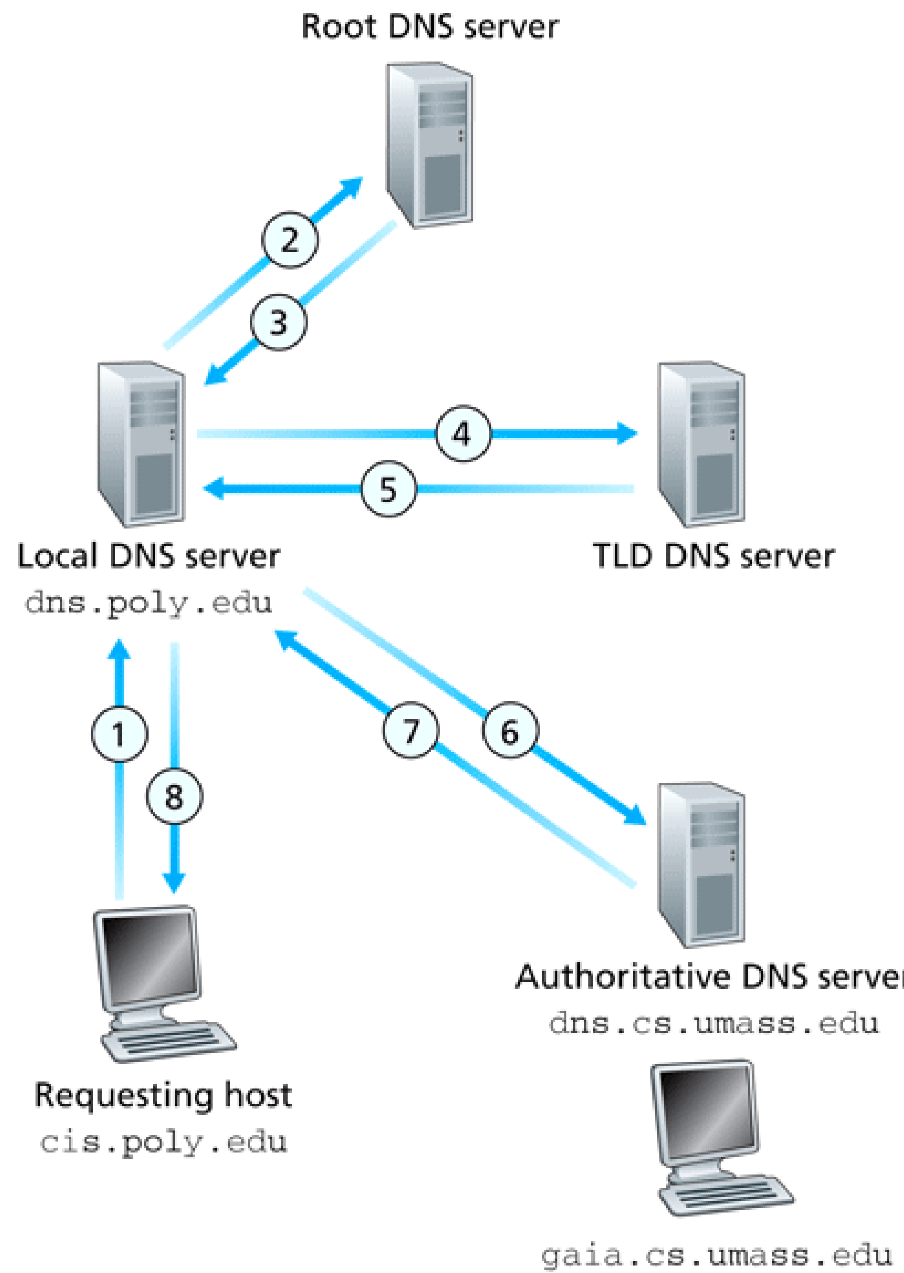
DNS Architecture



- Hierarchy of DNS servers
  - Root servers
  - Top-level domain (TLD) servers
  - Authoritative DNS servers
- Performing the translation
  - Local DNS servers
  - Resolver software

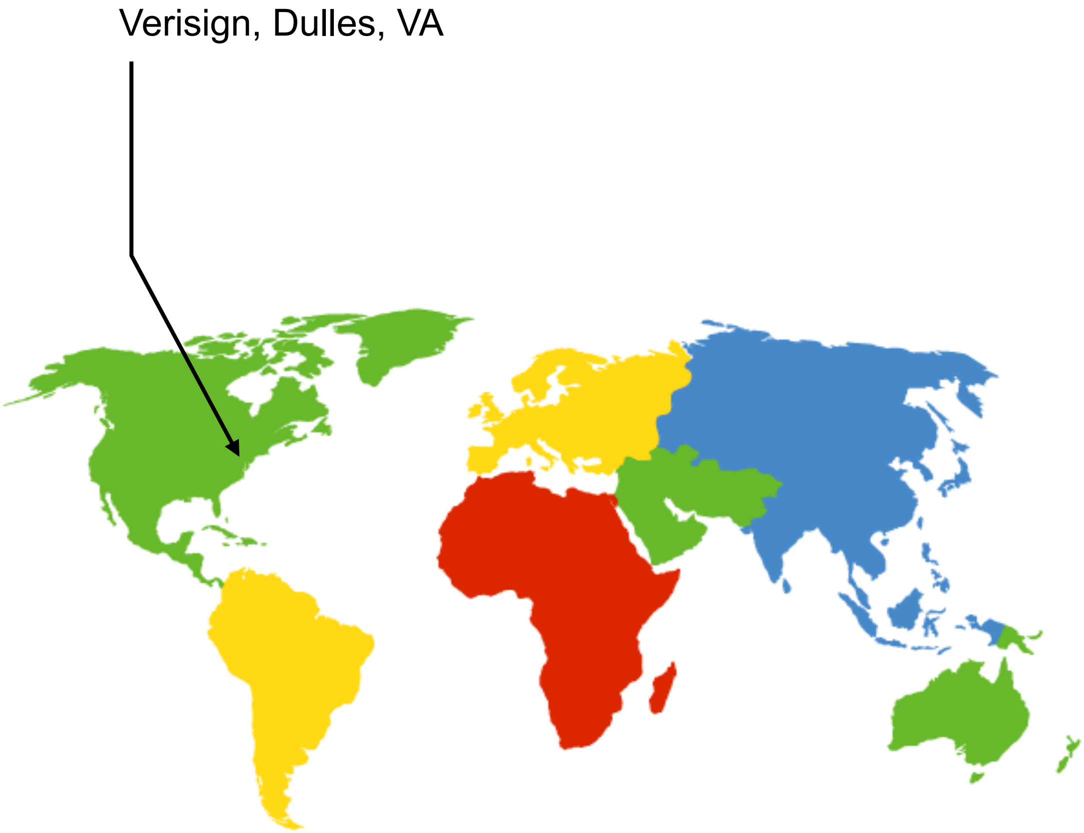
Resolver operation

- 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



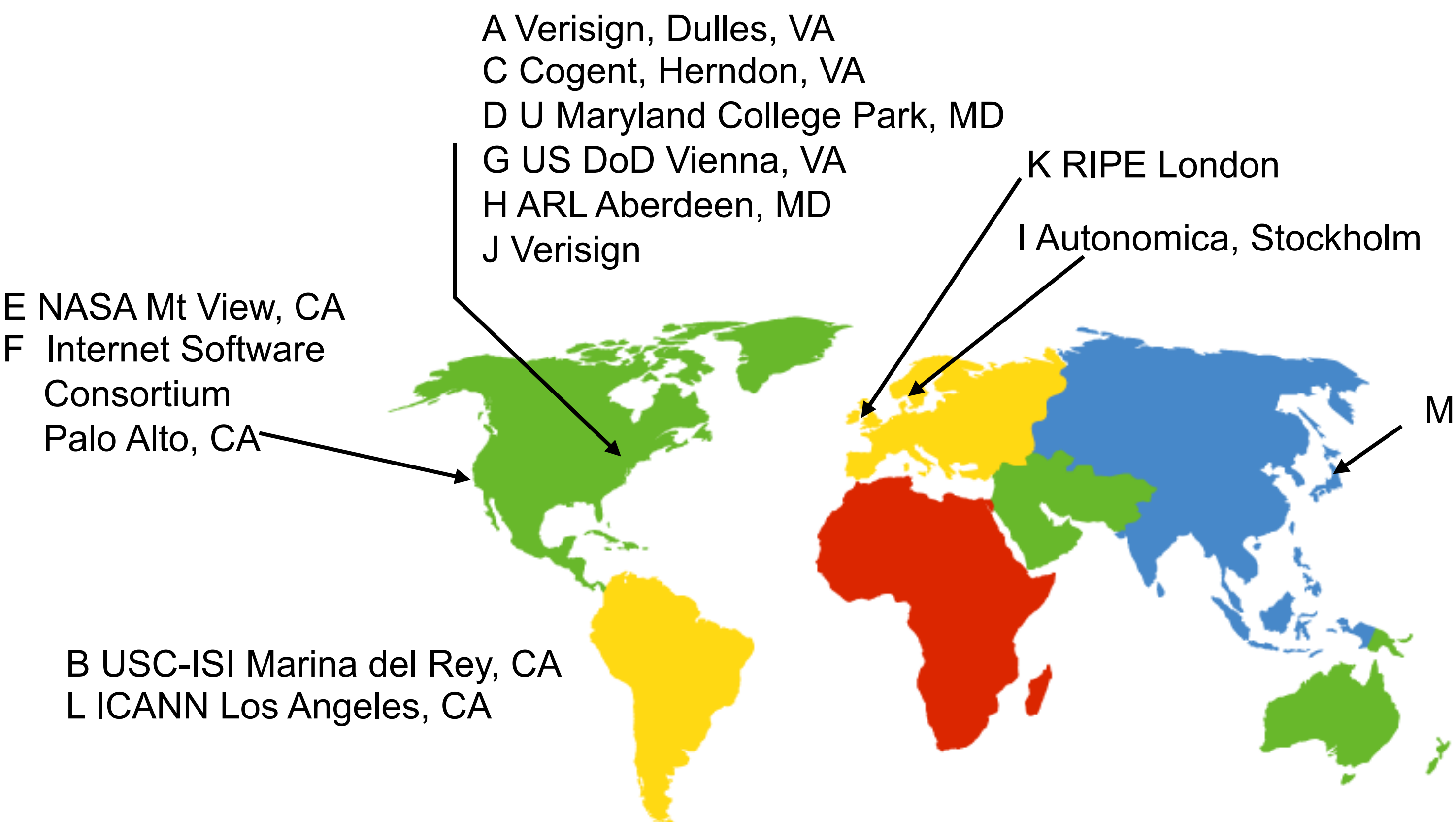
DNS Root Server

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



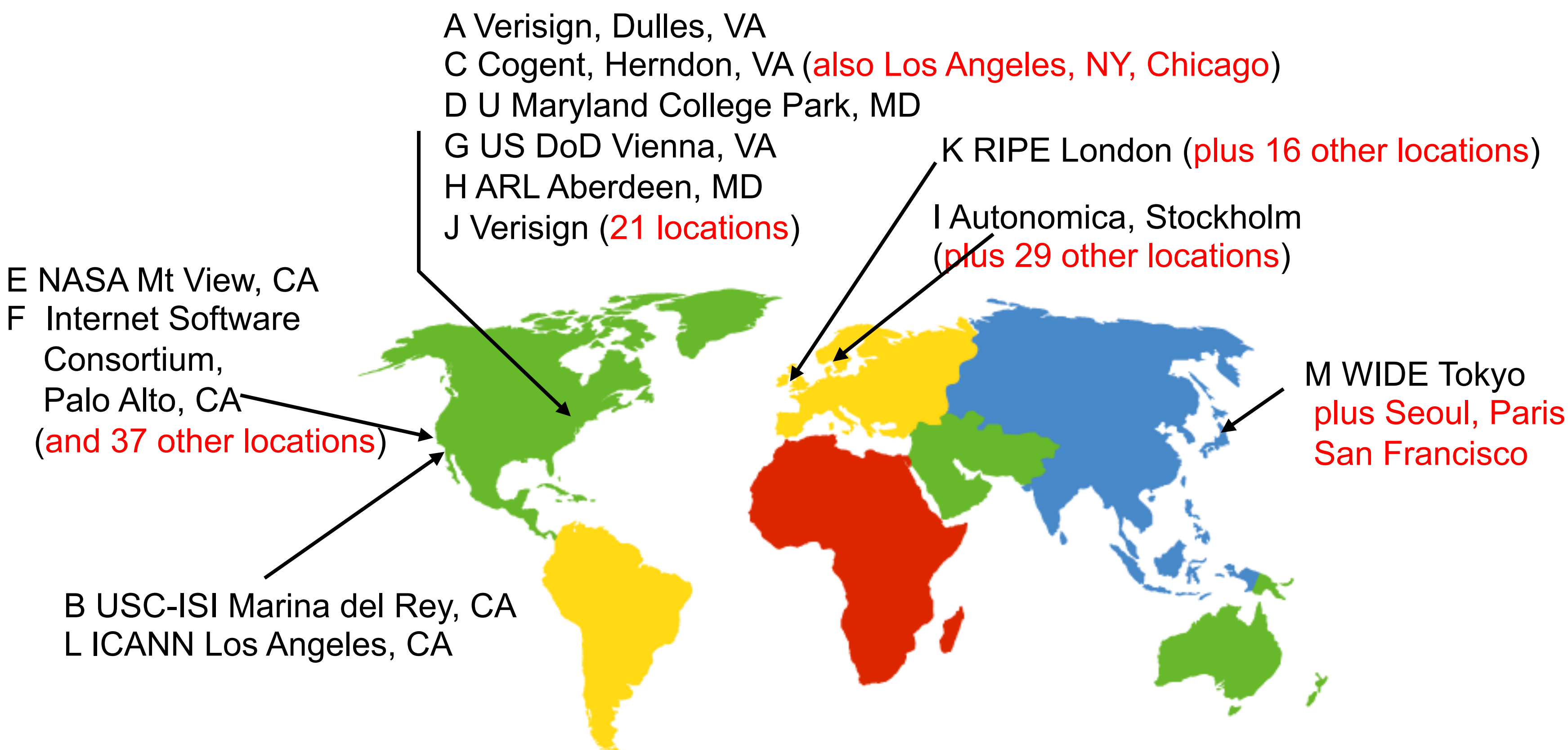
DNS Root Servers

- 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



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 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

http://en.wikipedia.org/wiki/Reverse_DNS_lookup

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

```
bayou.cs.uh.edu.  3600   IN  A   129.7.240.18
cs.uh.edu.     3600   IN  NS  ns2.uh.edu.
cs.uh.edu.     3600   IN  NS  dns.cs.uh.edu.
```

Some important details

- How do local servers find root servers?
  - DNS lookup on a.root-servers.net ?
  - Servers configured with *root cache* file
  - ftp://ftp.rs.internic.net/domain/db.cache
  - 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 +norec bayou.cs.uh.edu @a.root-servers.net

dig +norec bayou.cs.uh.edu @a.edu-servers.net

dig +norec bayou.cs.uh.edu @ns1.uh.edu

dig +norec bayou.cs.uh.edu @dns.cs.uh.edu

    ;; ANSWER SECTION:
    bayou.cs.uh.edu. 3600  IN A  129.7.240.18
```

```
[gnawali@bayou ~]$ dig handy.cs.uh.edu +trace

; <<>> DiG 9.3.4-P1 <<>> handy.cs.uh.edu +trace
;; global options:  printcmd
.                   94758   IN   NS   k.root-servers.net.
.                   94758   IN   NS   j.root-servers.net.
.                   94758   IN   NS   d.root-servers.net.
.                   94758   IN   NS   b.root-servers.net.
.                   94758   IN   NS   i.root-servers.net.
.                   94758   IN   NS   l.root-servers.net.
.                   94758   IN   NS   f.root-servers.net.
.                   94758   IN   NS   m.root-servers.net.
.                   94758   IN   NS   g.root-servers.net.
.                   94758   IN   NS   h.root-servers.net.
.                   94758   IN   NS   a.root-servers.net.
.                   94758   IN   NS   c.root-servers.net.
.                   94758   IN   NS   e.root-servers.net.
;; Received 288 bytes from 129.7.240.1#53(129.7.240.1) in 0 ms

edu.                172800  IN   NS   a.edu-servers.net.
edu.                172800  IN   NS   c.edu-servers.net.
edu.                172800  IN   NS   d.edu-servers.net.
edu.                172800  IN   NS   f.edu-servers.net.
edu.                172800  IN   NS   g.edu-servers.net.
edu.                172800  IN   NS   l.edu-servers.net.
;; Received 268 bytes from 193.0.14.129#53(k.root-servers.net) in 38 ms

uh.edu.             172800  IN   NS   ns2.uh.edu.
uh.edu.             172800  IN   NS   ncc.uky.edu.
uh.edu.             172800  IN   NS   ns1.uh.edu.
uh.edu.             172800  IN   NS   mesquite.cc.uh.edu.
;; Received 181 bytes from 192.5.6.30#53(a.edu-servers.net) in 36 ms

handy.cs.uh.edu. 3600   IN   A    129.7.240.36
;; Received 49 bytes from 129.7.1.6#53(ns2.uh.edu) in 0 ms
```

```
[gnawali@bayou ~]$ dig www.google.com +trace

; <<>> DiG 9.3.4-P1 <<>> www.google.com +trace
;; global options:  printcmd
.                   94874   IN   NS   h.root-servers.net.
.                   94874   IN   NS   f.root-servers.net.
…
.                   94874   IN   NS   l.root-servers.net.
.                   94874   IN   NS   i.root-servers.net.
;; Received 244 bytes from 129.7.240.1#53(129.7.240.1) in 0 ms

com.                172800  IN   NS   a.gtld-servers.net.
com.                172800  IN   NS   b.gtld-servers.net.
com.                172800  IN   NS   c.gtld-servers.net.
com.                172800  IN   NS   d.gtld-servers.net.
com.                172800  IN   NS   e.gtld-servers.net.
…
com.                172800  IN   NS   m.gtld-servers.net.
;; Received 495 bytes from 128.63.2.53#53(h.root-servers.net) in 48 ms

google.com.         172800  IN   NS   ns2.google.com.
google.com.         172800  IN   NS   ns1.google.com.
google.com.         172800  IN   NS   ns3.google.com.
google.com.         172800  IN   NS   ns4.google.com.
;; Received 168 bytes from 192.5.6.30#53(a.gtld-servers.net) in 37 ms

www.google.com.             604800  IN   CNAME
          www.l.google.com.
www.l.google.com.   300     IN   A    74.125.227.48
www.l.google.com.   300     IN   A    74.125.227.52
www.l.google.com.   300     IN   A    74.125.227.51
www.l.google.com.   300     IN   A    74.125.227.49
www.l.google.com.   300     IN   A    74.125.227.50
;; Received 132 bytes from 216.239.34.10#53(ns2.google.com) in 40 ms
```