CSCI-1680
DNS

Rodrigo Fonseca

Based partly on lecture notes by Scott Shenker and John Jannotti

- **We know how to open TCP connections to a server/port:**
  - E.g., 128.148.32.110, port 80

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)

Another Change in Layers…

- **Another mapping**
  - ARP: maps IP addresses to MAC addresses

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

```
320 -- ************************
10-Jun-82 17:48:41-PDT,114828;000000000000
Mail-from: ARPANET host SRI-NIC rcvd at 10-Jun-82 1747-PDT
Date: 10 Jun 1982 1742-PDT
From: Dyer
Subject: Hostname table, 10-June-82
To:    dcacode252 at USC-ISI
cc:    nic

       ARPANET HOST NAMES AND LIAISON                  10-Jun-82

HOST NAME     HOST ADDRESS      SPONSOR         LIAISON

ACC             10.2.0.54   VDH  ARPA   Lockwood, Gregory (LOCKWOOD@BBNC)
                                         Associated Computer Consultants
                                         414 East Cota Street
                                         Santa Barbara, California 93101
                                         (805) 965-1023
   CPUtype: PDP-11/70(UNIX)
ACCAT-TIP       10.2.0.35        ARPA   McBride, William T.
                                         (MCBRIDE@USC-ISIC)
                                         Naval Ocean Systems Center
                                         Code 8321
                                         271 Catalina Boulevard
                                         San Diego, California 92152
                                         (714) 225-2083 (AV) 933-2083
   CPUtype: H-316
AEROSPACE       10.2.0.65        AFSC   Nelson, Louis C. (LOU@AEROSPACE)
                                         Aerospace Corporation
                                         A2/1013
                                         P.O. Box 92957
                                         Los Angeles, California 90009
                                         (213) 615-4424
   CPUtype: VAX-11/780(UNIX)
AFGL            10.1.0.66        AFSC   Cosentino, Antonio
                                         (COSENTINO@AFSC-HQ)
                                         Air Force Geophysics Laboratory
                                         SUNA
                                         Mail Stop 30
                                         Hanscom Air Force Base,
                                          Massachusetts 01731
                                         (617) 861-4161 (AV) 478-4161
   CPUtype: PDP-11/50(RSX11M) -> CDC-6600(NOS/BE)
AFGL-TAC        10.2.0.66        AFSC   Cosentino, Antonio
                                         (COSENTINO@AFSC-HQ)
                                         Air Force Geophysics Laboratory
                                         SUNA
                                         Mail Stop 30
                                         Hanscom Air Force Base,
                                          Massachusetts 01731
                                         (617) 861-4161 (AV) 478-4161
   CPUtype: C/30
```

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

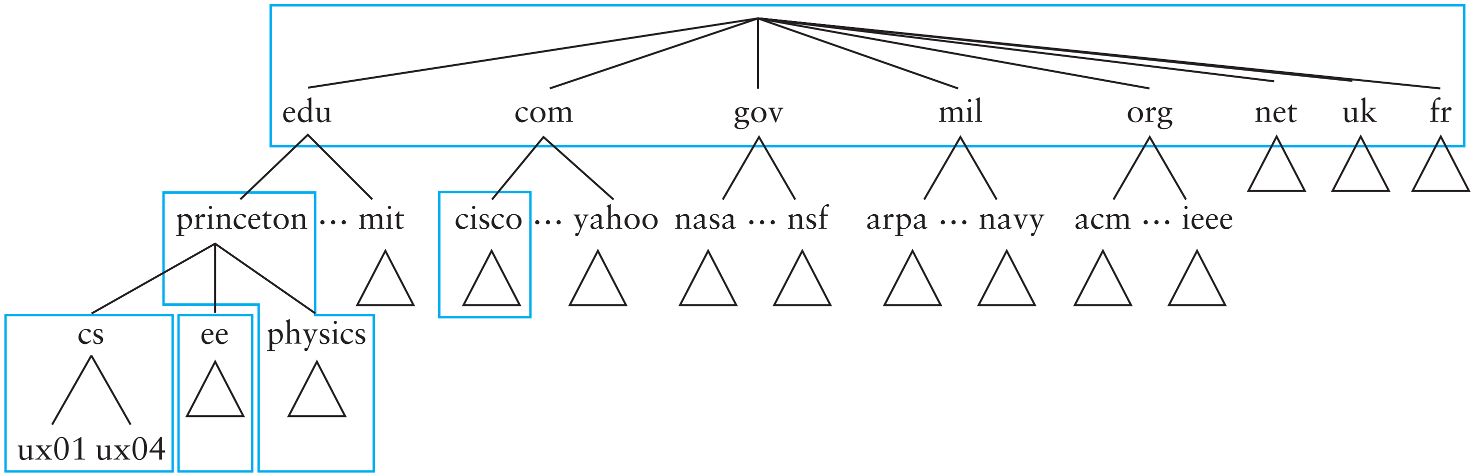
     When adding a machine, not end of the world if it takes minutes or hours to propagate
- **These suggest aggressive *caching***
  - Once you've lookup up a hostname, remember
  - Don't have to look again in the near future

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%**

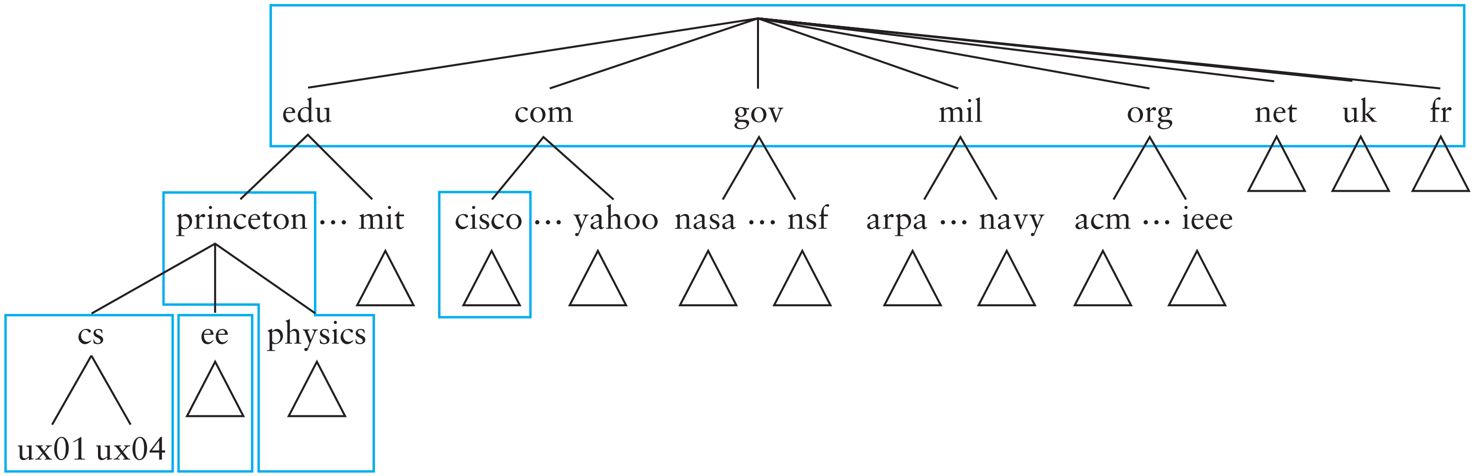
http://nms.lcs.mit.edu/papers/dns-ton2002.pdf

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**

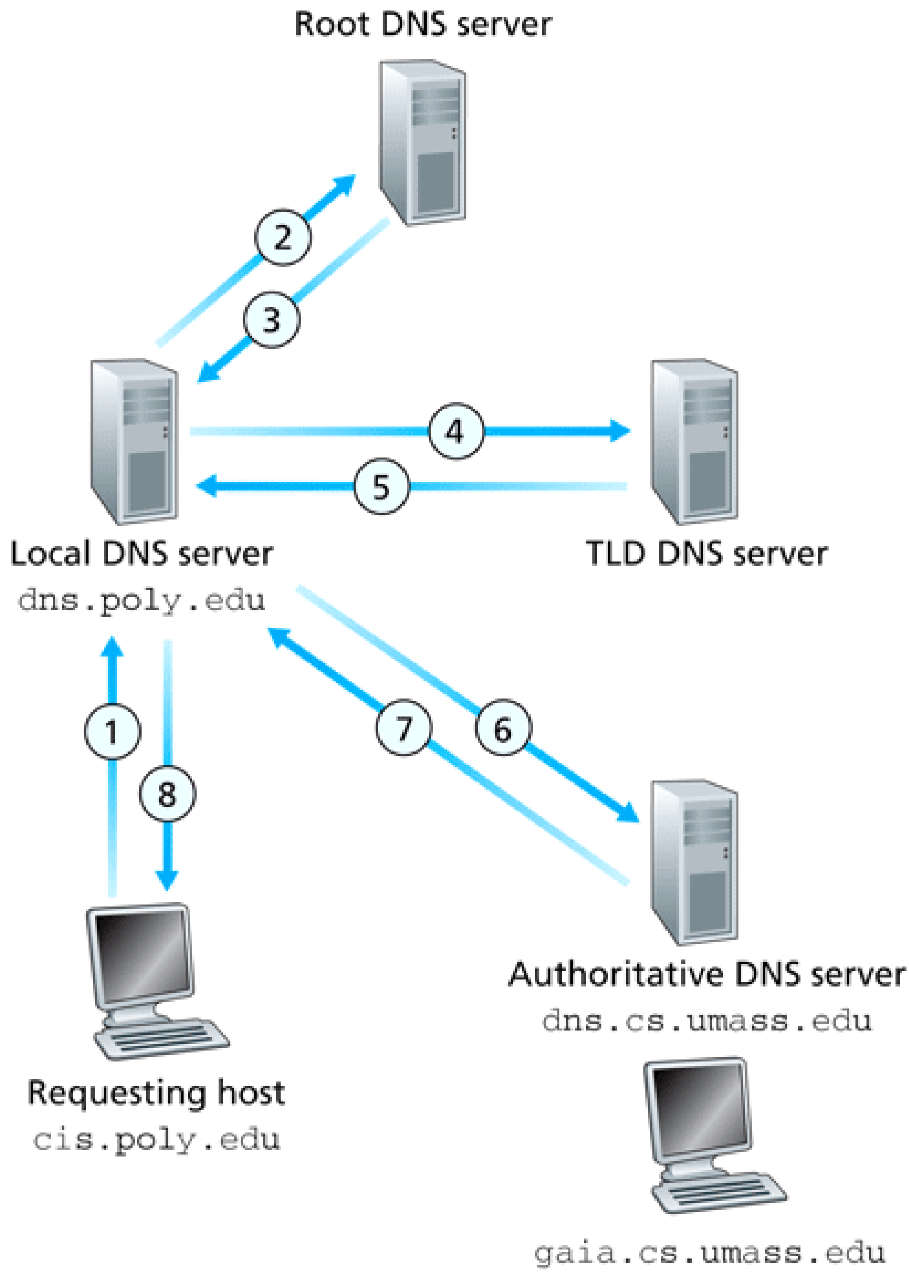
DNS Architecture



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

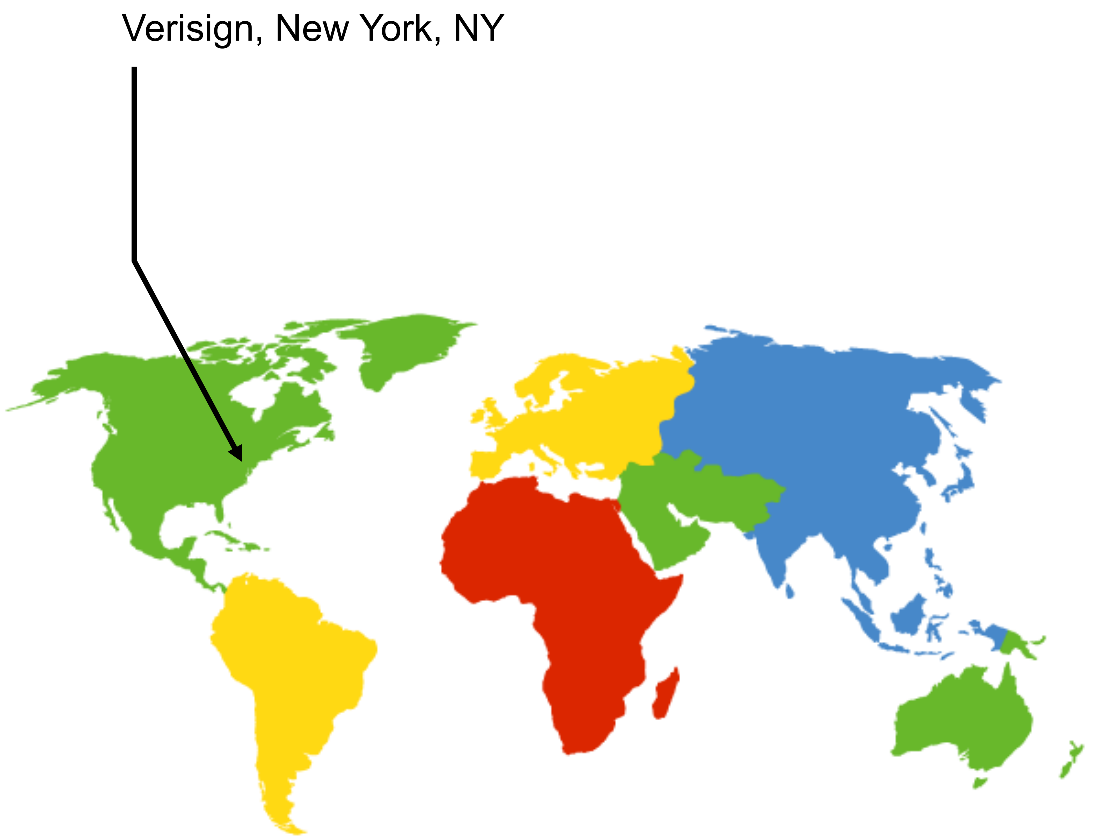
Resolver operation

- **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



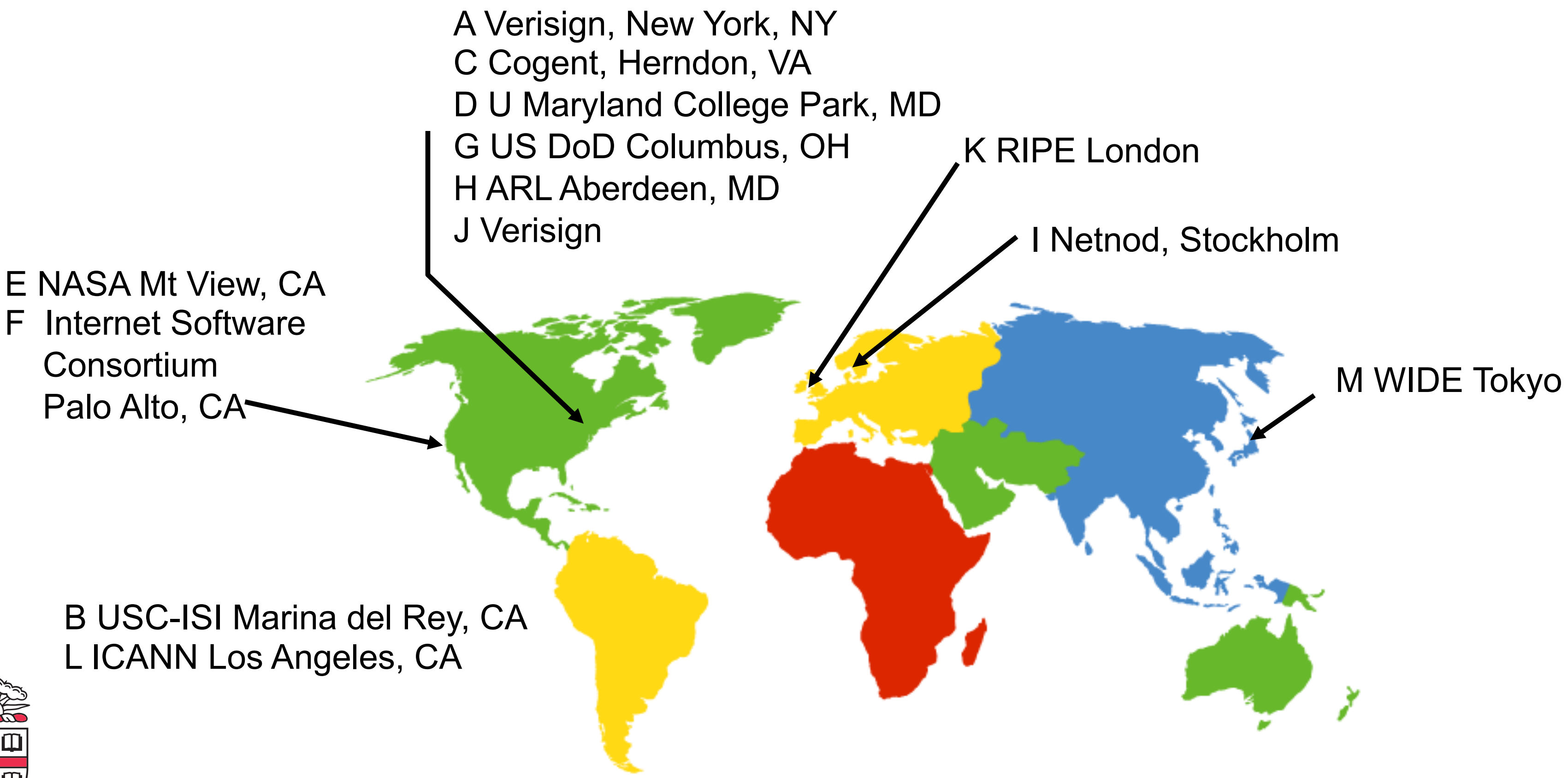
DNS Root Server

- Located in New York
- How do we make the root scale?



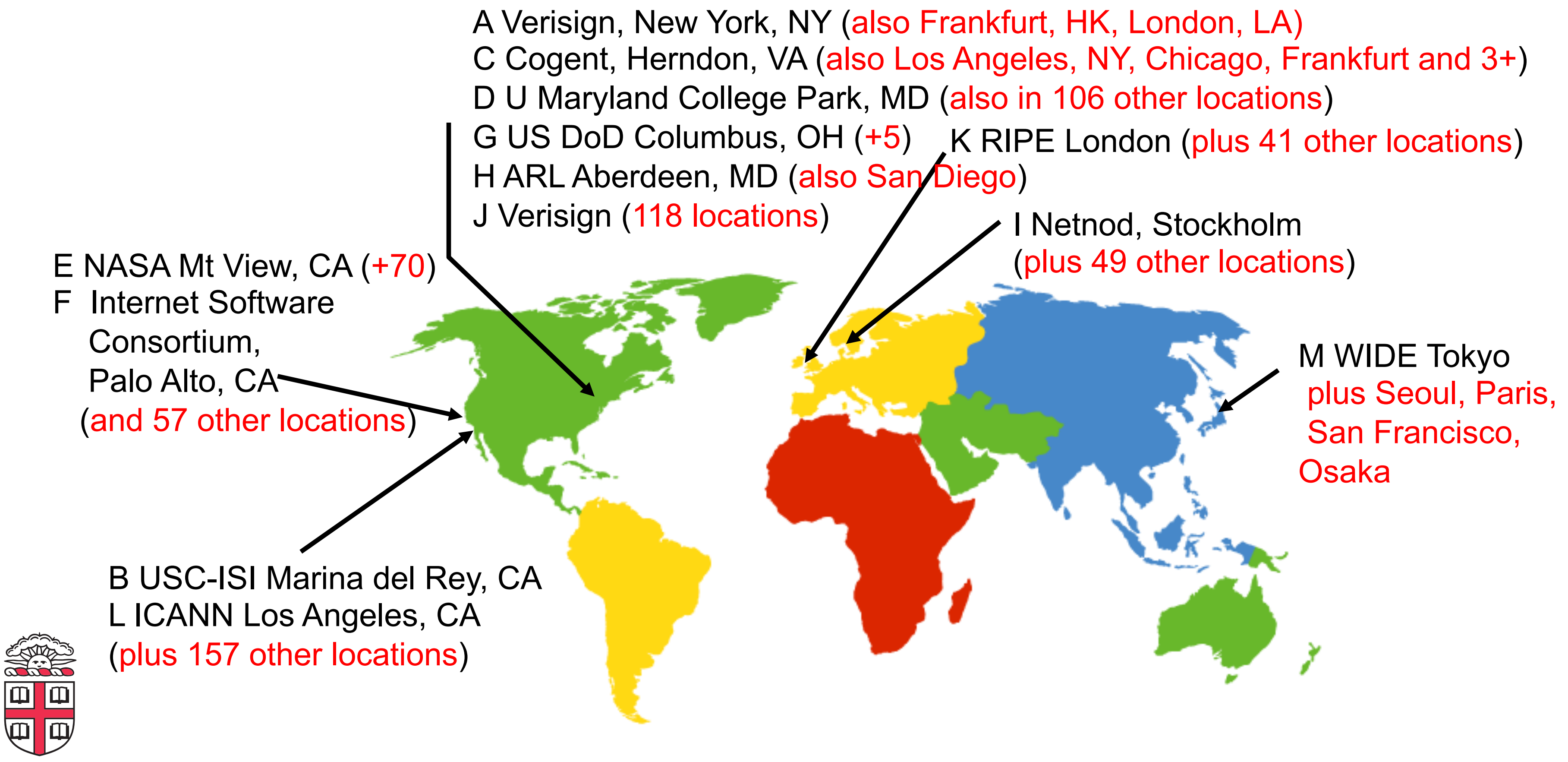
DNS Root Servers

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

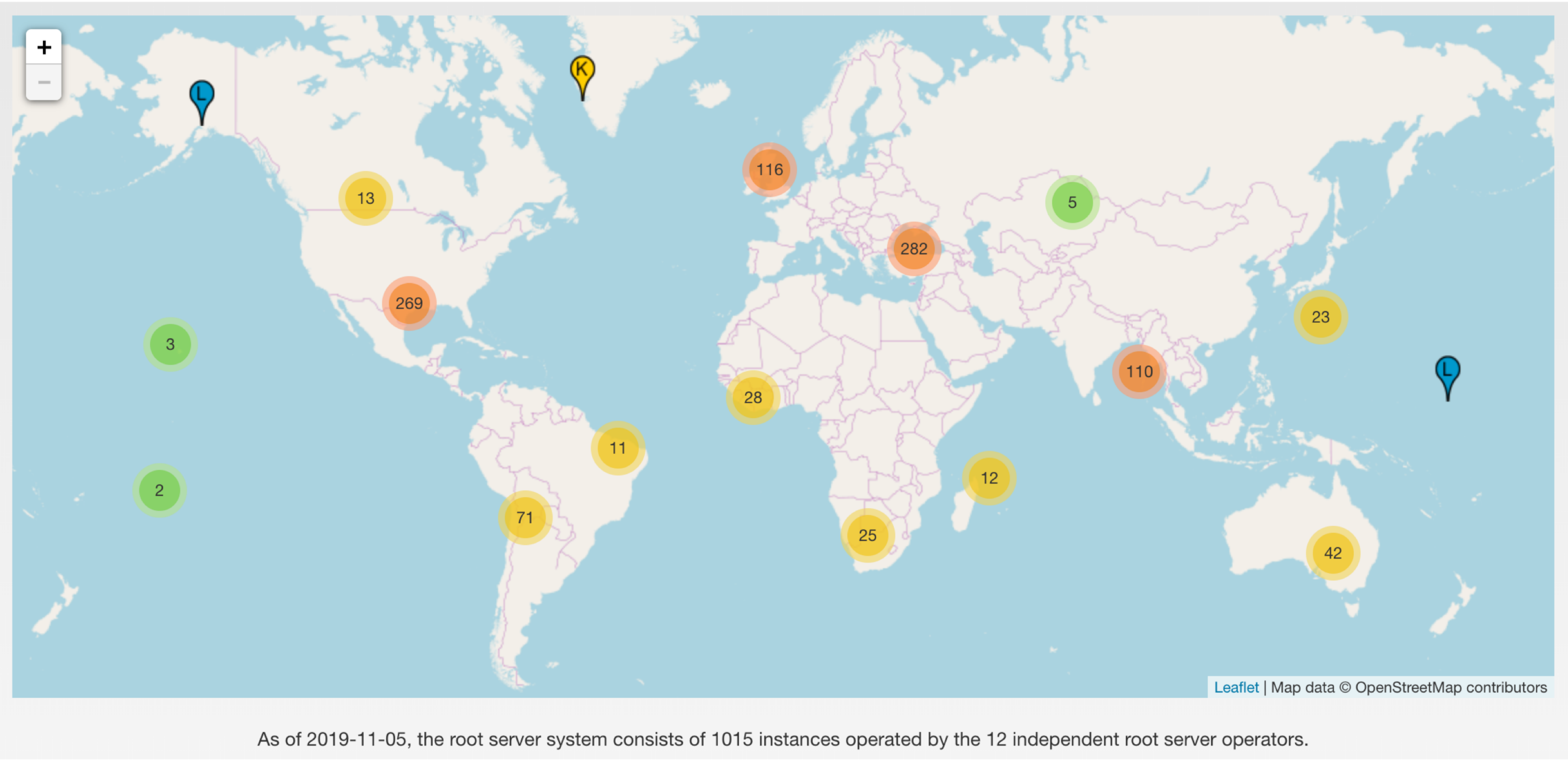


DNS Root Servers

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


A Verisign, New York, NY (also Frankfurt, HK, London, LA)
C Cogent, Herndon, VA (also Los Angeles, NY, Chicago, Frankfurt and 3+)
D U Maryland College Park, MD (also in 106 other locations)
G US DoD Columbus, OH (+5)
H ARL Aberdeen, MD (also San Diego)
J Verisign (118 locations)
K RIPE London (plus 41 other locations)
I Netnod, Stockholm (plus 49 other locations)
E NASA Mt View, CA (+70)
F Internet Software Consortium, Palo Alto, CA (and 57 other locations)
M WIDE Tokyo plus Seoul, Paris, San Francisco, Osaka
B USC-ISI Marina del Rey, CA
L ICANN Los Angeles, CA (plus 157 other locations)

DNS Root Servers



As of 2019-11-05, the root server system consists of 1015 instances operated by the 12 independent root server operators.

From: www.root-servers.org, as of Nov 2019

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

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
  - Is the cost of these two queries the same?
- **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**

Reverse Mapping

- **How do we get the other direction, IP address to name?**
- **Addresses have a natural hierarchy:**
  - 128.148.32.12
- **But, most significant element comes first**
- **Idea: reverse the numbers: 12.32.148.128 …**
  - and look that up in DNS
- **Under what TLD?**
  - Convention: in-addr.arpa
  - Lookup 12.32.148.128.in-addr.arpa
  - in6.arpa for IPv6

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
```

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
- **Important RR types**
  – A – Internet Address (IPv4);   AAAA – IPv6
  – 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

Other uses of DNS

- **Local multicast DNS**
  - Used for service discovery
  - Made popular by Apple
  - This is how you learn of different Apple TVs in the building
- **Load balancing**
- **CDNs**

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 212.44.9.0/24
- **Register helpme.com at namecheap.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 212.44.9.120
- **Configure your authoritative server (dns1.helpme.com)**
  - Type A record for www.helpme.com
  - Type MX record for helpme.com

Inserting a Record in DNS, cont

- **Need to provide reverse PTR bindings**
  - E.g., 212.44.9.120 -> dns1.helpme.com
- **Configure your dns server to serve the 9.44.212.in-addr.arpa zone**
  - Need to add a record of this NS into the parent zone (44.212.in-addr.arpa)
- **Insert the bindings into the 9.44.212.in-addr.arpa zone**

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

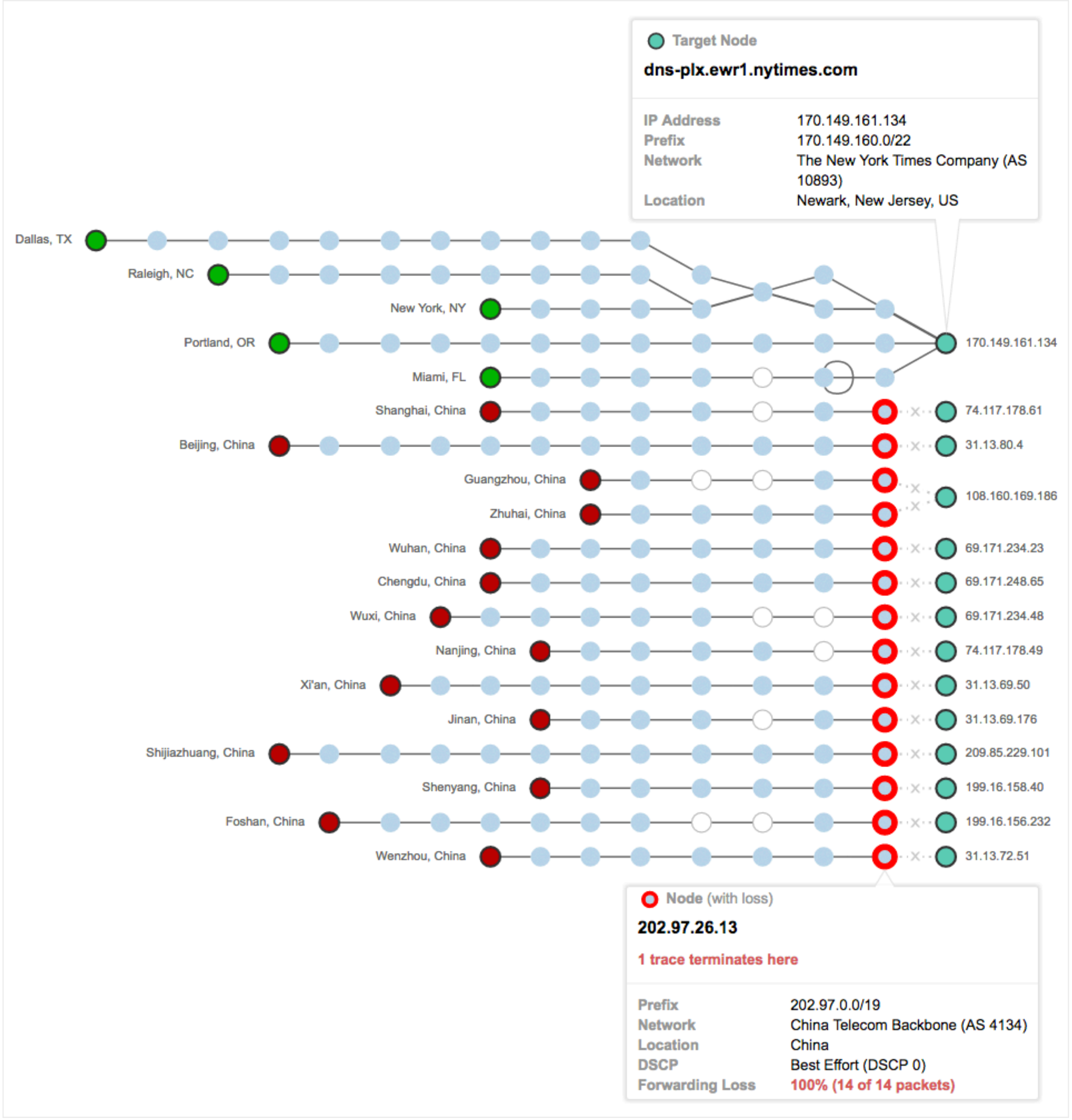
- **Randomize id**
  - Used to be sequential
- **Randomize source port number**
  - Used to be the same for all requests from the server
- **Offers some protection, but attack still possible**
- **DNS over HTTPS**
  - Connects you to a trusted resolver
  - Encrypts your traffic (no ISP spying / intercepting)
  - Does not guarantee correct results

Helpful ISPs

- **Many ISPs hijack NXDOMAIN responses to “help” by offering search and advertisement related to the domain**
- **E.g., www.bicycleisntadomain.com doesn’t (currently) exist**
  - Could return a page with search and ads on bicycles (or domain registrations?)

Great Firewall of CIT

- **If attacker is on the path (say, it is the ISP, or a malicious version of TStaff):**
  - Can sniff all DNS queries
  - Send fake responses back first
  - Could do this selectively, to direct facebook.com to cs.brown.edu, for example…



Target Node
dns-plx.ewr1.nytimes.com

IP Address	170.149.161.134
Prefix	170.149.160.0/22
Network	The New York Times Company (AS 10893)
Location	Newark, New Jersey, US

Node (with loss)
202.97.26.13

1 trace terminates here

Prefix	202.97.0.0/19
Network	China Telecom Backbone (AS 4134)
Location	China
DSCP	Best Effort (DSCP 0)
Forwarding Loss	100% (14 of 14 packets)

https://blog.thousandeyes.com/monitoring-dns-in-china/

Demo

- **Find an open resolver that might be giving bad responses…**
  - https://public-dns.info/
  - Let’s try 218.107.55.108

Great Firewall of China

- **Inject false responses to DNS queries *passing through* Chinese AS'es!**

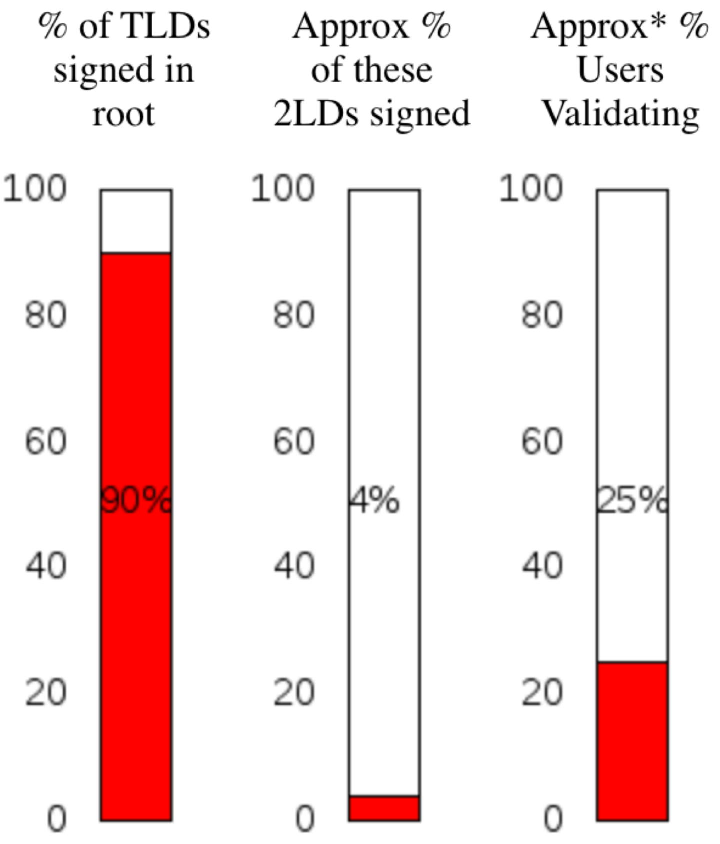
The Collateral Damage of Internet Censorship by DNS Injection. Sigcomm CCR 42, 3 (June 2012)

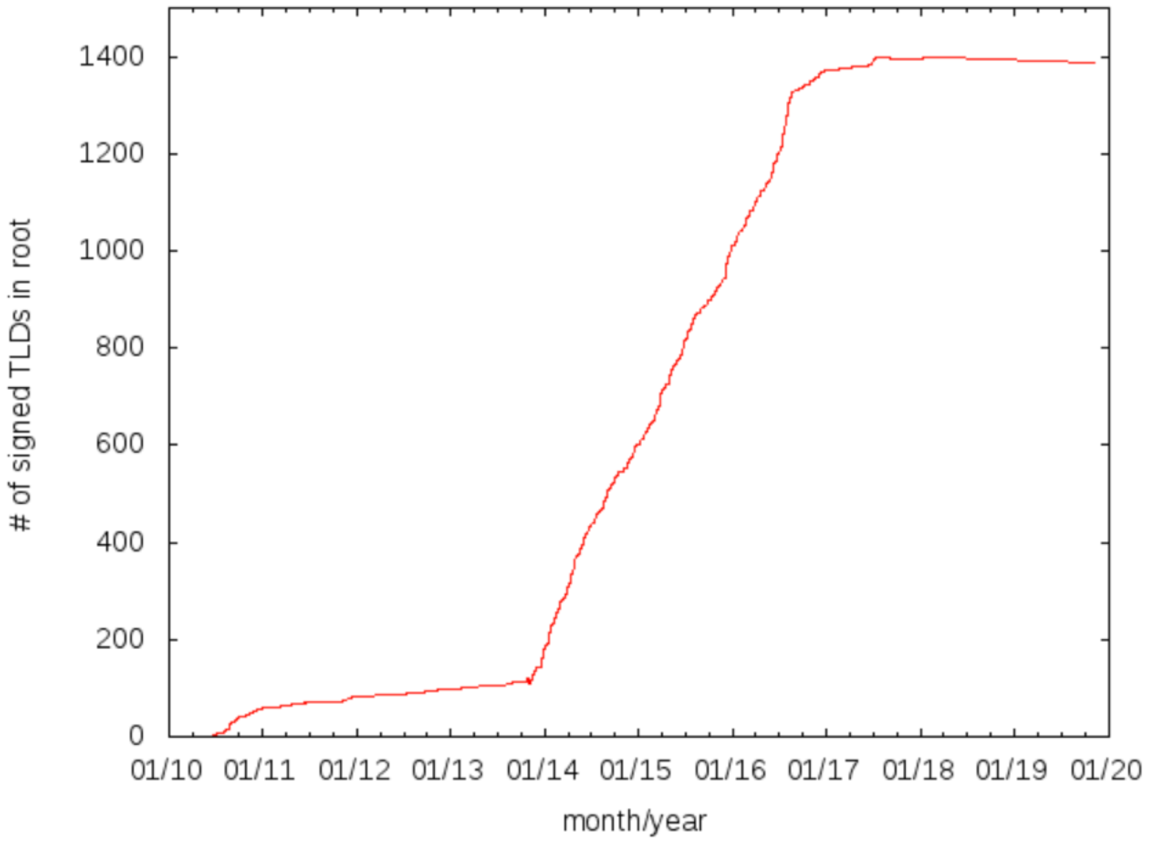
Real Solution: Signatures

- **Signature: cryptographic way to prove a party is who they say they are**
- **Requires a chain of trust**
  - Mirror the DNS hierarchy
- **DNSSEC deployment is underway**
  - Root signed in 2010
  - 1398 / 1527 TLDs are signed (2019/11/07)

DNSSEC

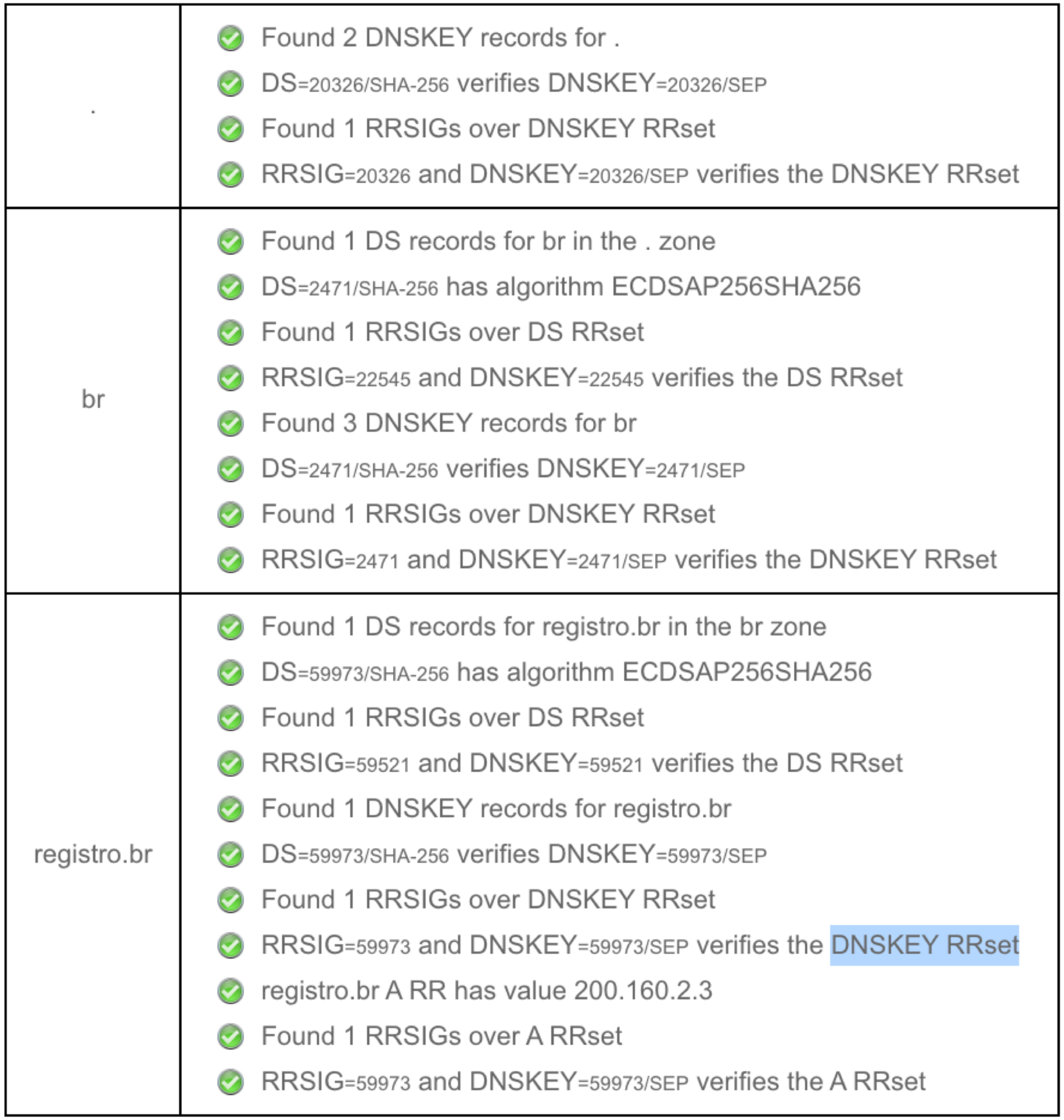
- **1. Digitally sign DNS records by authoritative servers**
- **2. Resolvers have to make sure they trust the public key**
  - Start at the root (have a copy of the root's key, or fingerprint)
  - Verify the public key for the TLD, signed by the root
  - Recurse until you get to the public key of the record you got





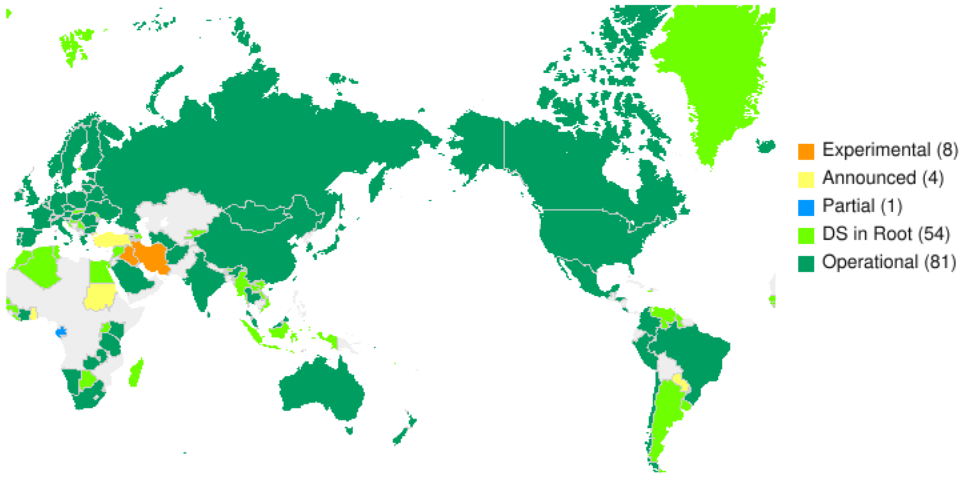
*From http://stats.labs.apnic.net/dnssec Some tools: http://www.co.tt DEWS% DEWS

http://rick.eng.br/dnssecstat/



.	✓ Found 2 DNSKEY records for .
	✓ DS=20326/SHA-256 verifies DNSKEY=20326/SEP
	✓ Found 1 RRSIGs over DNSKEY RRset
	✓ RRSIG=20326 and DNSKEY=20326/SEP verifies the DNSKEY RRset
br	✓ Found 1 DS records for br in the . zone
	✓ DS=2471/SHA-256 has algorithm ECDSAP256SHA256
	✓ Found 1 RRSIGs over DS RRset
	✓ RRSIG=22545 and DNSKEY=22545 verifies the DS RRset
	✓ Found 3 DNSKEY records for br
	✓ DS=2471/SHA-256 verifies DNSKEY=2471/SEP
	✓ Found 1 RRSIGs over DNSKEY RRset
	✓ RRSIG=2471 and DNSKEY=2471/SEP verifies the DNSKEY RRset
registro.br	✓ Found 1 DS records for registro.br in the br zone
	✓ DS=59973/SHA-256 has algorithm ECDSAP256SHA256
	✓ Found 1 RRSIGs over DS RRset
	✓ RRSIG=59521 and DNSKEY=59521 verifies the DS RRset
	✓ Found 1 DNSKEY records for registro.br
	✓ DS=59973/SHA-256 verifies DNSKEY=59973/SEP
	✓ Found 1 RRSIGs over DNSKEY RRset
	✓ RRSIG=59973 and DNSKEY=59973/SEP verifies the DNSKEY RRset
	✓ registro.br A RR has value 200.160.2.3
	✓ Found 1 RRSIGs over A RRset
	✓ RRSIG=59973 and DNSKEY=59973/SEP verifies the A RRset

https://dnssec-analyzer.verisignlabs.com/registro.br ccTLD DNSSEC Status on 2019-09-09



Backup Slides

- Good information, but not covered in class

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!**

SOPA - H. R. 3261
October 2011

- **“A BILL To promote prosperity, creativity, entrepreneurship, and innovation by combating the theft of U.S. property, and for other purposes”**

…

102.c.2
- **A service provider shall take technically feasible and reasonable measures designed to prevent access by its subscribers located within the United States to the foreign infringing site (or portion thereof) that is subject to the order, including measures designed to prevent the domain name of the foreign infringing site (or portion thereof) from resolving to that domain name’s Internet Protocol address.**

- **Would this work?**

SOPA

- **Provider filtering would interact badly with DNSSEC**
- **Filtering would be easy to circumvent**
  - Circumvention could expose users to malicious DNS servers
- **Could cause collateral damage**
  - E.g., blog1.blogspot.com vs blog2.blogspot.com

- **The bill was withdrawn in 2012 after major online outcry, but highlights the importance of understanding technical underpinnings.**

See http://www.circleid.com/pdf/PROTECT-IP-Technical-Whitepaper-Final.pdf for details