15-440 Distributed Systems

DNS
Outline

• DNS Design

• DNS Today
Naming

• How do we efficiently locate resources?
  • DNS: name $\rightarrow$ IP address

• Challenge
  • How do we scale this to the wide area?
Obvious Solutions (1)

Why not use /etc/hosts?

• Original Name to Address Mapping
  • Flat namespace
  • /etc/hosts
  • SRI kept main copy
  • Downloaded regularly

• Count of hosts was increasing: machine per domain \(\rightarrow\) machine per user
  • Many more downloads
  • Many more updates
Obvious Solutions (2)

Why not centralize DNS?

- Single point of failure
- Traffic volume
- Distant centralized database
- Single point of update
- Doesn’t scale!
Domain Name System Goals

- Basically a wide-area distributed database
- Scalability
- Decentralized maintenance
- Robustness
- Global scope
  - Names mean the same thing everywhere
- Don’t need
  - Atomicity
  - Strong consistency
• Conceptually, programmers can view the DNS database as a collection of millions of host entry structures:

```c
/* DNS host entry structure */
struct addrinfo {
    int     ai_family;       /* host address type (AF_INET) */
    size_t  ai_addrlen;      /* length of an address, in bytes */
    struct sockaddr *ai_addr; /* address! */
    char    *ai_canonname;    /* official domain name of host */
    struct addrinfo *ai_next; /* other entries for host */
};
```

• Functions for retrieving host entries from DNS:
  • `getaddrinfo`: query key is a DNS host name.
  • `getnameinfo`: query key is an IP address.
### DNS Message Format

<table>
<thead>
<tr>
<th>Identification</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Questions</td>
<td>No. of Answer RRs</td>
</tr>
<tr>
<td>No. of Authority RRs</td>
<td>No. of Additional RRs</td>
</tr>
</tbody>
</table>

- **Name, type fields** for a query
- **RRs in response to query**
- **Records for authoritative servers**
- **Additional “helpful info that may be used**
- **Questions (variable number of answers)**
- **Answers (variable number of resource records)**
- **Authority (variable number of resource records)**
- **Additional Info (variable number of resource records)**
DNS Header Fields

- **Identification**
  - Used to match up request/response

- **Flags**
  - 1-bit to mark query or response
  - 1-bit to mark authoritative or not
  - 1-bit to request recursive resolution
  - 1-bit to indicate support for recursive resolution
DNS Records

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

- DB contains tuples called resource records (RRs)
  - Classes = Internet (IN), Chaosnet (CH), etc.
  - Each class defines value associated with type

FOR IN class:

- Type=A
  - name is hostname
  - value is IP address
- Type=NS
  - name is domain (e.g. foo.com)
  - value is canonical name

- Type=CNAME
  - name is an alias name for some “canonical” (the real) name
- Type=MX
  - value is hostname of mailserver associated with name
Properties of DNS Host Entries

- Different kinds of mappings are possible:
  - Simple case: 1-1 mapping between domain name and IP addr:
    - kittyhawk.cmcl.cs.cmu.edu maps to 128.2.194.242
  - Multiple domain names maps to the same IP address:
    - eecs.mit.edu and cs.mit.edu both map to 18.62.1.6
  - Single domain name maps to multiple IP addresses:
    - aol.com and www.aol.com map to multiple IP addrs.
  - Some valid domain names don’t map to any IP address:
    - for example: cmcl.cs.cmu.edu
DNS Design: Hierarchy Definitions

• Each node in hierarchy stores a list of names that end with same suffix
  • Suffix = path up tree
  • E.g., given this tree, where would following be stored:
    • Fred.com
    • Fred.edu
    • Fred.cmu.edu
    • Fred.cmcl.cs.cmu.edu
    • Fred.cs.mit.edu
DNS Design: Zone Definitions

- Zone = contiguous section of name space
  - E.g., Complete tree, single node or subtree
- A zone has an associated set of name servers
  - Must store list of names and tree links

• Zone = contiguous section of name space
  - E.g., Complete tree, single node or subtree
• A zone has an associated set of name servers
  - Must store list of names and tree links
DNS Design: Cont.

- Zones are created by convincing owner node to create/delegate a subzone
  - Records within zone stored multiple redundant name servers
  - Primary/master name server updated manually
  - Secondary/redundant servers updated by zone transfer of name space
    - Zone transfer is a bulk transfer of the “configuration” of a DNS server – uses TCP to ensure reliability

- Example:
  - CS.CMU.EDU created by CMU.EDU administrators
  - Who creates CMU.EDU or .EDU?
DNS: Root Name Servers

- Responsible for “root” zone
- Approx. 13 root name servers worldwide
  - Currently {a-m}.root-servers.net
- Local name servers contact root servers when they cannot resolve a name
  - Configured with well-known root servers
  - Newer picture → www.root-servers.org
Physical Root Name Servers

- Several root servers have multiple physical servers
- Packets routed to “nearest” server by “Anycast” protocol
- 346 servers total
Servers/Resolvers

- Each host has a resolver
  - Typically a library that applications can link to
  - Local name servers hand-configured (e.g. /etc/resolv.conf)

- Name servers
  - Either responsible for some zone or…
  - Local servers
    - Do lookup of distant host names for local hosts
    - Typically answer queries about local zone
Typical Resolution

- Client
- Local DNS server
- www.cs.cmu.edu
- root & edu DNS server
- ns1.cmu.edu
- ns1.cs.cmu.edu

Diagram:
- Client queries www.cs.cmu.edu
- Local DNS server queries ns1.cmu.edu
- ns1.cmu.edu queries ns1.cs.cmu.edu
- A www=IP addr
- ns1.cs.cmu.edu queries root & edu DNS server
Typical Resolution

- Steps for resolving www.cmu.edu
  - Application calls gethostbyname() (RESOLVER)
  - Resolver contacts local name server ($S_1$)
  - $S_1$ queries root server ($S_2$) for (www.cmu.edu)
  - $S_2$ returns NS record for cmu.edu ($S_3$)
  - What about A record for $S_3$?
    - This is what the additional information section is for (PREFETCHING)
  - $S_1$ queries $S_3$ for www.cmu.edu
  - $S_3$ returns A record for www.cmu.edu
Lookup Methods

Recursive query:
- Server goes out and searches for more info (recursive)
- Only returns final answer or “not found”

Iterative query:
- Server responds with as much as it knows (iterative)
- “I don’t know this name, but ask this server”

Workload impact on choice?
- Local server typically does recursive
- Root/distant server does iterative
Workload and Caching

- Are all servers/names likely to be equally popular?
  - Why might this be a problem? How can we solve this problem?
- DNS responses are cached
  - Quick response for repeated translations
  - Other queries may reuse some parts of lookup
    - NS records for domains
- DNS negative queries are cached
  - Don’t have to repeat past mistakes
  - E.g. misspellings, search strings in resolv.conf
- Cached data periodically times out
  - Lifetime (TTL) of data controlled by owner of data
  - TTL passed with every record
Typical Resolution

Client

Local DNS server

www.cs.cmu.edu

www.cs.cmu.edu

NS ns1.cmu.edu

NS ns1.cs.cmu.edu

A www=IPaddr

ns1.cmu.edu DNS server

ns1.cs.cmu.edu DNS server

root & edu DNS server
Subsequent Lookup Example

Client

ftp.cs.cmu.edu

Local DNS server

root & edu DNS server

cmu.edu DNS server

ftp.cs.cmu.edu

ftp=IPaddr

cs.cmu.edu DNS server
Reliability

- DNS servers are replicated
  - Name service available if ≥ one replica is up
  - Queries can be load balanced between replicas
- UDP used for queries
  - Need reliability → must implement this on top of UDP!
  - Why not just use TCP?
- Try alternate servers on timeout
  - Exponential backoff when retrying same server
- Same identifier for all queries
  - Don’t care which server responds
Reverse DNS

- **Task**
  - Given IP address, find its name

- **Method**
  - Maintain separate hierarchy based on IP names
  - Write 128.2.194.242 as 242.194.2.128.in-addr.arpa
    - Why is the address reversed?

- **Managing**
  - Authority manages IP addresses assigned to it
  - E.g., CMU manages name space 128.2.in-addr.arpa
At each level of hierarchy, have group of servers that are authorized to handle that region of hierarchy.
Tracing Hierarchy (1)

- Dig Program
  - Use flags to find name server (NS)
  - Disable recursion so that operates one step at a time

```
unix> dig +norecurse @a.root-servers.net NS four.cmcl.cs.cmu.edu

;; ADDITIONAL SECTION:
a.edu-servers.net  172800 IN A  192.5.6.30
c.edu-servers.net  172800 IN A  192.26.92.30
d.edu-servers.net  172800 IN A  192.31.80.30
f.edu-servers.net  172800 IN A  192.35.51.30
g.edu-servers.net  172800 IN A  192.42.93.30
g.edu-servers.net  172800 IN AAAA 2001:503:cc2c::2:36
l.edu-servers.net  172800 IN A  192.41.162.30
```

- All .edu names handled by set of servers
Prefetching

- Name servers can add additional data to response
- Typically used for prefetching
  - CNAME/MX/NS typically point to another host name
  - Responses include address of host referred to in “additional section”
Tracing Hierarchy (2)

- 3 servers handle CMU names

unix> dig +nosecure @g.edu-servers.net NS four.cmcl.cs.cmu.edu

;; AUTHORITY  SECTION:
cmu.edu.        172800 IN  NS  ny-server-03.net.cmu.edu.
cmu.edu.        172800 IN  NS  nsauth1.net.cmu.edu.
cmu.edu.        172800 IN  NS  nsauth2.net.cmu.edu.
Tracing Hierarchy (3 & 4)

- 3 servers handle CMU CS names

```
unix> dig +norecurse @nsauth1.net.cmu.edu NS four.cmcl.cs.cmu.edu

;; AUTHORITY SECTION:
cs.cmu.edu.   600 IN  NS    AC-DDNS-2.NET.cs.cmu.edu.
cs.cmu.edu.   600 IN  NS    AC-DDNS-1.NET.cs.cmu.edu.
cs.cmu.edu.   600 IN  NS    AC-DDNS-3.NET.cs.cmu.edu.
```

- Server within CS is “start of authority” (SOA) for this name

```
unix> dig +norecurse @AC-DDNS-2.NET.cs.cmu.edu NS four.cmcl.cs.cmu.edu

;; AUTHORITY SECTION:
cs.cmu.edu.   300 IN  SOA PLANISPHERE.FAC.cs.cmu.edu.
```
Tracing Hierarchy (3 & 4)

- Any will return the A record

```
unix> dig +norecurse @AC-DDNS-2.NET.cs.cmu.edu
four.cmcl.cs.cmu.edu

;; ANSWER SECTION:
FOUR.CMCL.CS.CMU.EDU. 3600 IN A 128.2.209.189
```
DNS Hack #1

- Can return multiple A records → what does this mean?

- Load Balance
  - Server sends out multiple A records
  - Order of these records changes per-client
Server Balancing Example

- **DNS Tricks**

```sh
unix1> dig www.yahoo.com

;; ANSWER SECTION:
atsv2-fp.wg1.b.yahoo.com. 49 IN A 98.139.180.149
atsv2-fp.wg1.b.yahoo.com. 49 IN A 98.138.252.30
atsv2-fp.wg1.b.yahoo.com. 49 IN A 98.139.183.24
atsv2-fp.wg1.b.yahoo.com. 49 IN A 98.138.253.109
```

```sh
unix2> dig www.yahoo.com

;; ANSWER SECTION:
atsv2-fp.wg1.b.yahoo.com. 30 IN A 98.139.180.149
atsv2-fp.wg1.b.yahoo.com. 30 IN A 98.138.253.109
atsv2-fp.wg1.b.yahoo.com. 30 IN A 98.138.252.30
atsv2-fp.wg1.b.yahoo.com. 30 IN A 98.139.183.24
```
Outline

• DNS Design

• DNS Today
Protecting the Root Nameservers

Attack On Internet Called Largest Ever

By David McGuire and Brian Krebs
washingtonpost.com Staff Writers
Tuesday, October 22, 2002; 5:40 PM

The heart of the Internet sustained its largest and most sophisticated attack ever, starting late Monday, according to officials at key online backbone organizations.

Around 5:00 p.m. EDT on Monday, a "distributed denial of service" (DDOS) attack struck the 13 "root servers" that provide the primary roadmap for almost all Internet communications. Despite the scale of the attack, which lasted about an hour, Internet users worldwide were largely unaffected, experts said.

Sophisticated? Why did nobody notice?


Defense Mechanisms

• Redundancy: 13 root nameservers
• IP Anycast for root DNS servers \{c,f,i,j,k\}.root-servers.net
  • RFC 3258
  • Most physical nameservers lie outside of the US
## Defense: Replication and Caching

<table>
<thead>
<tr>
<th>Letter</th>
<th>Old name</th>
<th>Operator</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>ns.internic.net</td>
<td>VeriSign</td>
<td>Dulles, Virginia, USA</td>
</tr>
<tr>
<td>B</td>
<td>ns1.isi.edu</td>
<td>ISI</td>
<td>Marina Del Rey, California, USA</td>
</tr>
<tr>
<td>C</td>
<td>c.psi.net</td>
<td>Cogent Communications</td>
<td>distributed using anycast</td>
</tr>
<tr>
<td>D</td>
<td>terp.umd.edu</td>
<td>University of Maryland</td>
<td>College Park, Maryland, USA</td>
</tr>
<tr>
<td>E</td>
<td>ns.nasa.gov</td>
<td>NASA</td>
<td>Mountain View, California, USA</td>
</tr>
<tr>
<td>F</td>
<td>ns.isc.org</td>
<td>ISC</td>
<td>distributed using anycast</td>
</tr>
<tr>
<td>G</td>
<td>ns.nic.ddn.mil</td>
<td>U.S. DoD NIC</td>
<td>Columbus, Ohio, USA</td>
</tr>
<tr>
<td>H</td>
<td>aos.arl.army.mil</td>
<td>U.S. Army Research Lab</td>
<td>Aberdeen Proving Ground, Maryland, USA</td>
</tr>
<tr>
<td>I</td>
<td>nic.nordu.net</td>
<td>Autonomica</td>
<td>distributed using anycast</td>
</tr>
<tr>
<td>J</td>
<td></td>
<td>VeriSign</td>
<td>distributed using anycast</td>
</tr>
<tr>
<td>K</td>
<td>RIPE NCC</td>
<td></td>
<td>distributed using anycast</td>
</tr>
<tr>
<td>L</td>
<td>ICANN</td>
<td></td>
<td>Los Angeles, California, USA</td>
</tr>
<tr>
<td>M</td>
<td>WIDE Project</td>
<td></td>
<td>distributed using anycast</td>
</tr>
</tbody>
</table>

*source: wikipedia*
What Happened on Oct 21st 2016?

- DDoS attack on Dyn
- Dyn provides core Internet services for Twitter, SoundCloud, Spotify, Reddit and a host of other sites
- Why didn’t DNS defense mechanisms work in this case?
- Let’s take a look at the DNS records
What was the source of attack?

- Mirai botnet
  - Used in 620Gbps attack last month

- Source: bad IoT devices, e.g.,
  - White-labeled DVR and IP camera electronics
  - username: root and password: xc3511
  - password is hardcoded into the device firmware
Attack Waves

• DNS lookups are routed to the nearest data center
• First wave
  • On three Dyn data centers – Chicago, Washington, D.C., and New York
• Second wave,
  • Hit 20 Dyn data centers around the world.
  • Required extensive planning.
  • Since DNS request go to the closest DNS server, the attacker had to plan a successful attack for each of the 20 data centers with enough bots in each region to be able to take down the local Dyn services
Solutions?

• Dyn customers
  • Going to backup DNS providers, as Amazon did
  • Signing up with an alternative today after the attacks, as PayPal did

• Lowering their time-to-life settings on their DNS servers
  • Redirect traffic faster to another DNS service that is still available
Generic Top Level Domains (gTLD) = .com, .net, .org, etc…
Country Code Top Level Domain (ccTLD) = .us, .ca, .fi, .uk, etc…
Root server ({a-m}.root-servers.net) also used to cover gTLD domains
  - Load on root servers was growing quickly!
  - Moving .com, .net, .org off root servers was clearly necessary to reduce load → done Aug 2000
gTLDs

- **Unsponsored**
  - .com, .edu, .gov, .mil, .net, .org
  - .biz → businesses
  - .info → general info
  - .name → individuals

- **Sponsored (controlled by a particular association)**
  - .aero → air-transport industry
  - .cat → catalan related
  - .coop → business cooperatives
  - .jobs → job announcements
  - .museum → museums
  - .pro → accountants, lawyers, and physicians
  - .travel → travel industry

- **Starting up**
  - .mobi → mobile phone targeted domains
  - .post → postal
  - .tel → telephone related

- **Proposed**
  - .asia, .cym, .geo, .kid, .mail, .sco, .web, .xxx
New Registrars

• Network Solutions (NSI) used to handle all registrations, root servers, etc…
  • Clearly not the democratic (Internet) way
  • Large number of registrars that can create new domains → However NSI still handles A root server
Do you trust the TLD operators?

• Wildcard DNS record for all .com and .net domain names not yet registered by others
  • September 15 – October 4, 2003
  • February 2004: Verisign sues ICANN

• Redirection for these domain names to Verisign web portal (SiteFinder)

• What services might this break?
DNS (Summary)

- Motivations → large distributed database
  - Scalability
  - Independent update
  - Robustness
- Hierarchical database structure
  - Zones
  - How is a lookup done
- Caching/prefetching and TTLs
- Reverse name lookup
- What are the steps to creating your own domain?