Outline

- DNS Design
- DNS Today

Naming

- How do we efficiently locate resources?
  - DNS: name → 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 → 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

Programmer’s View of DNS
- 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;
    size_t ai_addrlen;
    struct sockaddr *ai_addr;
    char   *ai_canonname;
    struct addrinfo *ai_next;
  };
  
  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>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification</td>
<td><strong>12 bytes</strong> Identification has an ID and flags.</td>
</tr>
<tr>
<td>No. of Questions</td>
<td>No. of questions in the query.</td>
</tr>
<tr>
<td>No. of Answer RRs</td>
<td>No. of answer resource records in response.</td>
</tr>
<tr>
<td>No. of Authority RRs</td>
<td>No. of authoritative resource records in response.</td>
</tr>
<tr>
<td>Questions (variable number of answers)</td>
<td>Questions (variable number of answers)</td>
</tr>
<tr>
<td>Answers (variable number of resource records)</td>
<td>Answers (variable number of resource records)</td>
</tr>
<tr>
<td>Authority (variable number of resource records)</td>
<td>Authority (variable number of resource records)</td>
</tr>
<tr>
<td>Records for authoritative servers</td>
<td>Additional info that may be used</td>
</tr>
<tr>
<td>Additional Info (variable number of resource records)</td>
<td>Additional Info (variable number of resource records)</td>
</tr>
</tbody>
</table>
**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 name of authoritative name server for this domain
- Type=CNAME
  - name is an alias name for some "canonical" (the real) name
  - value is canonical 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

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

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

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`

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

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

Subsequent Lookup Example

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

arpa Name Server Hierarchy

- At each level of hierarchy, have group of servers that are authorized to handle that region of hierarchy

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"

Mail Addresses

- MX records point to mail exchanger for a name
  - E.g. mail.acm.org is MX for acm.org
  - Addition of MX record type proved to be a challenge
  - How to get mail programs to lookup MX record for mail delivery?
  - Needed critical mass of such mailers
Outline

• DNS Design
• DNS Today

Root Zone

• 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

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

gTLDs

• Un-sponsored
  • .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
Tracing Hierarchy (1)

- **Dig Program**
  - Allows querying of DNS system
  - 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 kittyhawk.cmcl.cs.cmu.edu
```

```
;; AUTHORITY SECTION:
edu.                    172800  IN      NS      L3.NSTLD.COM.
edu.                    172800  IN      NS      D3.NSTLD.COM.
edu.                    172800  IN      NS      A3.NSTLD.COM.
edu.                    172800  IN      NS      E3.NSTLD.COM.
edu.                    172800  IN      NS      C3.NSTLD.COM.
edu.                    172800  IN      NS      F3.NSTLD.COM.
edu.                    172800  IN      NS      G3.NSTLD.COM.
edu.                    172800  IN      NS      B3.NSTLD.COM.
edu.                    172800  IN      NS      M3.NSTLD.COM.
```

- All .edu names handled by set of servers

Tracing Hierarchy (2)

- 3 servers handle CMU names

```
unix> dig +norecurse @e3.nstld.com NS kittyhawk.cmcl.cs.cmu.edu
```

```
;; AUTHORITY SECTION:
cmu.edu.                172800  IN      NS      CUCUMBER.SRV.cs.cmu.edu.
cmu.edu.                172800  IN      NS      T-NS1.NET.cmu.edu.
cmu.edu.                172800  IN      NS      T-NS2.NET.cmu.edu.
```

Tracing Hierarchy (3 & 4)

- 4 servers handle CMU CS names

```
unix> dig +norecurse @t-ns1.net.cmu.edu NS kittyhawk.cmcl.cs.cmu.edu
```

```
;; AUTHORITY SECTION:
cs.cmu.edu.             86400   IN      NS      MANGO.SRV.cs.cmu.edu.
cs.cmu.edu.             86400   IN      NS      PEACH.SRV.cs.cmu.edu.
cs.cmu.edu.             86400   IN      NS      BANANA.SRV.cs.cmu.edu.
cs.cmu.edu.             86400   IN      NS      BLUEBERRY.SRV.cs.cmu.edu.
```

- Quasar is master NS for this zone

```
unix> dig +norecurse @blueberry.srv.cs.cmu.edu NS kittyhawk.cmcl.cs.cmu.edu
```

```
;; AUTHORITY SECTION:
cs.cmu.edu.             300     IN      SOA     QUASAR.FAC.cs.cmu.edu.
```

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?
Protecting the Root Nameservers

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 16 hours, Internet users worldwide were largely unaffected, experts said.

**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>ns.udk.edu</td>
<td>UCI</td>
<td>Marina Del Rey, California, USA</td>
</tr>
<tr>
<td>C</td>
<td>c-pa.net</td>
<td>Cogent Communications</td>
<td>distributed using anycast</td>
</tr>
<tr>
<td>D</td>
<td>tmp.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.fc.org</td>
<td>ISOC</td>
<td>distributed using anycast</td>
</tr>
<tr>
<td>G</td>
<td>ns.nic.dh.gov</td>
<td>U.S. DoD NIC</td>
<td>Columbus, Ohio, USA</td>
</tr>
<tr>
<td>H</td>
<td>i.ric.army.mil</td>
<td>U.S. Army Research Lab</td>
<td>Aberdeen Proving Grounds, Maryland, USA</td>
</tr>
<tr>
<td>I</td>
<td>nic.nons.net</td>
<td>Autorite@</td>
<td>distributed using anycast</td>
</tr>
<tr>
<td>J</td>
<td>VeriSign</td>
<td>distributed using anycast</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>RIPE NCC</td>
<td>distributed using anycast</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>ICANN</td>
<td>Los Angeles, California, USA</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>WIDE Project</td>
<td>distributed using anycast</td>
<td></td>
</tr>
</tbody>
</table>

source: wikipedia

DNS Hack #2: Blackhole Lists

- First: Mail Abuse Prevention System (MAPS)
  - Paul Vixie, 1997
- Today: Spamhaus, spamcop, dnsrbl.org, etc.

`% dig 91.53.195.211.bl.spamcop.net

; ANSWER SECTION:
91.53.195.211.bl.spamcop.net. 2100 IN A 127.0.0.2

; ANSWER SECTION:
91.53.195.211.bl.spamcop.net. 1799 IN TXT "Blocked - see http://www.spamcop.net/bl.shtml?211.195.53.91"`

Different addresses refer to different reasons for blocking

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?