Distributed Systems Within the Internet

Nov. 6, 2012

Topics

- Domain Name System
  - Finding IP address
- Content Delivery Networks
  - Caching content within the network
Domain Name System (DNS)

- Mapping from Host Names to IP Addresses

Distributed database
- Each site (university, large company, ISP, ...) maintains database with its own entries
- Provide server for others to query

Implemented at Application Layer
- Runs over UDP (normally) or TCP
Both generic (e.g., “.com”) and country (e.g., “.jp” domains)

June 2009: 20 generic, 248 country

Top-level names managed by NIC

Other name zones delegated to different entities
**DNS Name Terminology**

- **Node**: Any point in hierarchy
- **Zone**: A complete subtree
- **Name Servers**: Servers that can determine IP addresses within given zone
  - With help from other servers

---

**Diagram Illustration**

- **Top Node**: unnamed root
- **Subnodes**:
  - `arpa`, `mil`, `edu`, `gov`, `com`, `ae`, `us`, `zw`, `cs`, `ece`, `ics`, `greatwhite`, `amazon` with IP `128.2.217.13`, `72.21.194.1`, `128.2.220.10`, and `128.2.220.10`.

---

15-440
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 hostent {
    char   *h_name;       /* official domain name of host */
    char   **h_aliases;   /* null-terminated array of domain names */
    int    h_addrtype;    /* host address type (AF_INET) */
    int    h_length;      /* length of an address, in bytes */
    char   **h_addr_list; /* null-terminated array of in_addr structs */
};
```

- `in_addr` is a struct consisting of 4-byte IP address

**Functions for retrieving host entries from DNS:**

- `gethostbyname`: query key is a DNS domain name.
- `gethostbyaddr`: query key is an IP address.
Properties of DNS Host Entries

- Each host entry is an equivalence class of domain names and IP addresses.

Different kinds of mappings are possible:
- Simple case: 1-1 mapping between domain name and IP addr:
  - greatwhite.ics.cs.cmu.edu maps to 128.2.220.10
- Multiple domain names mapped to the same IP address:
  - eecs.mit.edu, cs.mit.edu, and ee.mit.edu map to 18.62.1.6
- Multiple domain names mapped 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: ics.cs.cmu.edu
DNS Name Server Hierarchy

- At each level of hierarchy, have group of servers that are authorized to handle that region of hierarchy
- At bottom of hierarchy, have authority server for specific name
Nominal Root Name Servers

- 13 total

a Verisign, Dulles, VA
b USC-ISI Marina del Rey, CA
c Cogent, Herndon, VA (also Los Angeles)
d U Maryland College Park, MD
e NASA Mt View, CA
f Internet Software C, Palo Alto, CA (and 17 other locations)
g US DoD Vienna, VA
h ARL Aberdeen, MD
i Verisign, (11 locations)
j k RIPE London (also Amsterdam, Frankfurt)
l Autonomica, Stockholm (plus 3 other locations)
m WMDE Tokyo
Physical Root Name Servers

- Several root servers have multiple physical servers
- Packets routed to “nearest” server by “Anycast” protocol
- 346 servers total
DNS Records

Format: (class, name, value, type, TTL)

Database of Resource Records (RRs)

- Classes: IN = Internet
- Each class defines value associated with type

IN Class Types

- A  Address
  - Name = hostname, Value = IP address
- NS  Name Server
  - Name = domain (e.g., cs.cmu.edu)
  - Value = authoritative name server for this domain
- CNAME  Canonical Name (alias)
  - Name = alias name
  - Value = canonical name
- MX  Mail server
  - Value = mail server hostname
Getting DNS Information with `dig`

```
unix> dig greatwhite.ics.cs.cmu.edu

;; ANSWER SECTION:
greatwhite.ics.cs.cmu.edu. 2966 IN A 128.2.220.10

;; AUTHORITY SECTION:
cs.cmu.edu.             593     IN      NS      AC-DDNS-3.NET.cs.cmu.edu.
cs.cmu.edu.             593     IN      NS      AC-DDNS-1.NET.cs.cmu.edu.
cs.cmu.edu.             593     IN      NS      AC-DDNS-2.NET.cs.cmu.edu.
```

**Perform DNS lookup as would for `gethostbyname`**

- Lots of command-line options
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 greatwhite.ics.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

IP v6 address
3 servers handle CMU names

unix> dig +norecurse @g.edu-servers.net NS greatwhite.ics.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 greatwhite.ics.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 greatwhite.ics.cs.cmu.edu |
|-------|--------|---|---------------|
| ;; AUTHORITY SECTION:                     |
| cs.cmu.edu.  300 IN SOA PLANISPHERE.FAC.cs.cmu.edu. |
Recursive DNS Name Resolution

**Nonlocal Lookup**
- Recursively from root server downward
- Results passed up

**Caching**
- Results stored in caches along each hop
- Can shortcircuit lookup when cached entry present
Iterative DNS Name Resolution

Nonlocal Lookup
- At each step, server returns name of next server down
- Local server directly queries each successive server

Caching
- Local server builds up cache of intermediate translations
- Helps in resolving names xxx.cs.cmu.edu, yy.cmu.edu, and z.edu
Reverse DNS

Task
- Given IP address, find its name

Method
- Maintain separate hierarchy based on IP names
- Write 128.2.194.242 as 242.194.128.2.in-addr.arpa

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
Performance Issues

Challenge
- There’s way too much traffic on the Internet
- Popular sites (Google, Amazon, Facebook, ...) get huge amounts of traffic
  - Could become “hot spot”
- It takes much longer to route packets around world than next door

Opportunities
- Services can be replicated
  - Multiple servers / data center
  - Multiple data centers around world
- Content can be cached

How Can this Work?
- Compare to original Internet model
  - IP address designates unique host
Server Balancing

DNS Tricks

- Customize DNS response to location
  - Allows distribution by geography
- Return multiple host names / query
  - Client (could) choose one at random
- Update DNS entries with new servers
  - Rotate loading

Within Data Center

- Keep changing binding between IP address and host
Server Balancing Example

DNS Tricks

- Different responses to different servers, short TTL’s

unix1> dig www.google.com

;; ANSWER SECTION:
www.l.google.com.       81      IN      A       72.14.204.104
www.l.google.com.       81      IN      A       72.14.204.105
www.l.google.com.       81      IN      A       72.14.204.147
www.l.google.com.       81      IN      A       72.14.204.103

unix2> dig www.google.com

;; ANSWER SECTION:
www.l.google.com.       145     IN      A       72.14.204.103
www.l.google.com.       145     IN      A       72.14.204.104
www.l.google.com.       145     IN      A       72.14.204.147
CDN Motivation

Typical Workload:
- Multiple (typically small) objects per page
- Frame, body, ads, logos, ...

File sizes
- Heavy-tailed
  - Pareto distribution for tail
  - Lognormal for body of distribution

Embedded references
- Number of embedded objects also pareto
  \[ \text{Pr}(X>x) = \left(\frac{x}{x^m}\right)^{-k} \]

This plays havoc with performance. Why?

Solutions?

• Lots of small objects & TCP yields:
  - 3-way handshake
  - Lots of slow starts
  - Extra connection state
Content Distribution Networks (CDNs)

The content providers are the CDN customers.

Content replication

CDN company installs hundreds of CDN servers throughout Internet

- Close to users

CDN replicates its customers’ content in CDN servers. When provider updates content, CDN updates servers

CDNs:

- Akamai
- Major ISPs
Serving Through CDN

Requirement
- Route HTTP request to CDN node, rather than to original server

Methods
- CDN provider manipulates DNS tables
- Rewrite HTML pages
  - `<a href="http://www.nfl.com/images/ben_roethlisberger">`
  - `<a href="http://a989.g.akamai.net/nfl/images/ben_roethlisberger">`

unix1> `dig www.nfl.com`

;; ANSWER SECTION:
www.nfl.com.edgesuite.net. 13778 IN CNAME a989.g.akamai.net.
a989.g.akamai.net. 20 IN A 96.7.40.32
a989.g.akamai.net. 20 IN A 96.7.40.33
Caching Content in CDN

**Simplistic**
- Each CDN server caches content that flows through it

**Better**
- Create DHT among cluster of servers
- Origin of Chord led to founding of Akamai

**Challenges**
- Usual ones of staleness / consistency / replication
- Handled by TTLs

**Effectiveness**
- Can’t cache dynamic content
  - Responses to individual queries
  - But, even dynamic pages contain static links
- Great for streaming content
  - If multiple clients viewing same programs ~ simultaneously
Summary

DNS one of world’s largest distributed system
- Operation and authority delegated hierarchically
- Huge number of queries / second

Many Ways to Reduce / Balance Traffic
- Contrary to simple unique address / host model
- Time & location varying DNS entries
- CDNs