CSE 123b
Communications Software

Spring 2004
Lecture 10: Domain Name System (DNS)
Stefan Savage

Some slides courtesy David Wetherall & Sriniv Seshan

Topic #1: Homework
Bring your homework up front

Topic #2: Midterm
Quick Survey

Topic #3: Project
You have started right?
- Unix network programming review
  - Cover somewhat in chapter #1 of the book
  - http://www.ecst.csuchico.edu/~beej/guide/net/html/ also not bad
  - Quick review – simple client-side APIs:
    - Socket data structure: just like a file descriptor
    - Create with `int socket (int domain, int type, int protocol)`
      - `domain = PF_INET` (defined in `sys/socket.h`)
      - `type = SOCK_STREAM` for TCP or `SOCK_DGRAM` for UDP
      - `protocol = IPPROTO_TCP` for TCP or `IPPROTO_UDP` for UDP
        (can also just use 0 and let it pick)

Overview for today
- What is naming about?
- How Domain Naming System (DNS) works
  - Namespace
  - Data distribution
  - Request/response protocol
  - Caching
  - Bootstraping
- Experience with DNS and new DNS uses

More socket programming
- If using TCP then
  - Connect to other host using `int connect(int socket, struct sockaddr *address, int addr_len)` – only tricky part is address structure
  - Send and receive using `send(int socket, char *buffer, int buf_len, int flags)`
    `recv(int socket, char *buffer, int buf_len, int flags)`
- If using UDP then send and receive using:
  - `int sendto(int socket, char *buffer, int len, unsigned int flags, struct sockaddr *toaddr, int tolen)`
  - `int recvfrom(int socket, char *buffer, int len, unsigned int flags, struct sockaddr *fromadd, int fromlen)`
Names and Addresses

- Names are identifiers for objects/services (high level)
- Addresses are locators for objects/services (low level)
- Resolution is the process of mapping name to address

Goals for a naming system

- How do we locate resources?
  - Machine name -> IP address
  - IP address -> Machine name

- How do we scale to the wide area?
  - Global scope
  - Robustness
  - Consistency: additions, deletions, modifications
  - Performance/overhead
  - Different administrative authorities

Internet Hostnames

- Hostnames are human-readable identifiers for end-systems based on an administrative hierarchy
  - risk64.ucsd.edu is my desktop machine

- IP addresses are a fixed-length binary encoding for end-systems based on their position in the network
  - 132.239.9.64 is risk64's IP address

Original Hostname System

- When the Internet was really young …

  - Flat namespace
    - Simple (host, address) pairs
  
  - Centralized management
    - Updates via a single master file called HOSTS.TXT
    - Manually coordinated by the SRI's Network Information Center (NIC)
    - You ftp'd the file over each day

  - Resolution process
    - Look up hostname in the HOSTS.TXT file

Scaling Problems

- Coordination
  - Between all users to avoid conflicts

- Inconsistencies
  - Between update and distribution of new version

- Reliability
  - Single point of failure

- Performance
  - Competition for centralized resources
  - Size of HOSTS.TXT

Domain Name System (DNS)

- Designed by Mockapetris and Dunlap in the mid 80s
  - Distributed database

- Namespace is hierarchical
  - Allows much better scaling of data structures
  - e.g., www.cs.ucsd.edu

- Namespace is distributed
  - Decentralized administration and access
  - e.g. xxx.ucsd.edu is managed only by UCSD

- Resolution is by query/response
  - With replicated servers for redundancy
  - With heavy use of caching for performance
**DNS Design**

- Administrative hierarchy
  - `.` as separator
  - Zone = contiguous section of name space with its own database and administrative control
    - E.g., Complete tree, single node or subtree
- Zones are created by convincing owner node to create/delegate a subzone
  - E.g. cs.ucsd.edu could be a zone under ucsd.edu

**DNS Hierarchy**

- `.` is the root of the hierarchy
- Top levels now controlled by ICANN (in theory)
- Lower level control is delegated
- Usage governed by conventions
- FQDN = Fully Qualified Domain Name

**DNS Records**

- DNS: distributed db storing resource records (RR)
- RR format: (name, value, type, ttl)
- Type=A
  - name is hostname
  - value is IP address
- Type=NS
  - name is domain (e.g. foo.com)
  - value is IP address 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

**DNS Distribution**

- One or more nameservers manage each zone
  - Primary nameserver updated manually
  - Secondary nameservers updated using zone transfers performed between nameservers (uses TCP)
  - Multiple nameservers provide redundancy
- Client resolvers query nameservers for specified records
  - Multiple messages may be exchanged per DNS lookup to navigate the name hierarchy

**Servers/Resolvers**

- Each host has a resolver
  - Typically a library that applications can link to, sometimes in kernel (e.g. Windows XP)
  - Host resolvers hand-configured (e.g. /etc/resolv.conf) or frequently key information (e.g. IP address of local nameserver) is passed via DHCP
- Name servers
  - Typically responsible for some zone (e.g. cs.ucsd.edu)
  - Local servers (also sometimes called caching servers)
    - Do lookup of distant host names for local hosts
    - Typically answer queries about local zone

**Lookup Methods**

- Iterative
  - Server responds with as much as it knows (iterative)
- Recursive
  - Server goes out and searches for more info (recursive)
  - Only returns final answer or “not found”
- Local server typically does recursive
- Root/distant server does iterative
**DNS Lookup Example**

**Recursive query:**
- Puts burden of name resolution on contacted name server.

**Iterative query:**
- Contacted server replies with name of server to contact.
- "I don't know this name, but ask this server."

---

**DNS Message Format**

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

- 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 field**
  - Used to match up request/response
  - How to pick id value?

- **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

---

**Caching**

- Servers and clients cache results of DNS lookups
  - Cache partial results too (e.g., server for princeton.edu)
  - Greatly improves system performance; lookups the rare case

- Cache using time-to-live (TTL) value from provider
  - Higher TTL means less traffic, lower TTL means less stale info

- Negative caching is used too!
  - Errors can cause repeated queries for non-existent data

---

**Impact of caching**

**DNS Lookup Example**

- Client -> Local DNS server
- Local DNS server queries root server
- Root server returns list of authoritative servers
- Iterated queries continue until authoritative server is reached

---

**Impact of caching**

**Subsequent Lookup**

- Client -> Local DNS server
- Local DNS server queries previous server
- No additional queries needed

---
Reliability
- DNS servers are replicated
  - Name service available if at least one replica is up
- UDP used for queries
  - Need reliability → Why not TCP?
  - Try alternate servers on timeout
    - How long a timeout?
  - Exponential backoff when retrying same server
  - Same identifier for all queries
    - Don’t care which server responds

Reverse Name Lookup
- 128.2.206.138?
  - Lookup 138.206.2.128.in-addr.arpa
  - Why is the address reversed?
- What if there is a many to one mapping?
  - i.e., www.cs.ucsd.edu and www-cse.ucsd.edu are the same machine
  - Reverse lookup should return primary name

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

DNS Bootstrapping
- Need to know IP addresses of root servers before we can make any queries
- Addresses for 13 root servers ([a-m].root-servers.net) handled via initial configuration (named.ca file or compiled into resolver)

DNS: Root Name Servers
- Responsible for “root” zone
- Approx. dozen 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

Building on the DNS
- Other naming designs leverage the DNS
  - Email:
    - e.g., savage@cs.ucsd.edu is savage in the domain cs.ucsd.edu
  - Uniform Resource Locators (URLs) name for Web pages
    - e.g., http://www.cs.ucsd.edu/~savage/
    - Use domain name to identify a Web server
    - Use “/” separated string to name path to page (like files)
Building on the DNS

- Load balancing of Internet services
  - If a name -> IP address mapping is one to many then can use DNS for load balancing
  - RR DNS: provide set of answers
  - Akamai/CDNs: provide different answer based on source address of local server and load on replicated content

Future Evolution of the DNS

- Design constrains us in two major ways that are increasingly less appropriate
  - Static host to IP mapping
    - What about mobility (Mobile IP) and dynamic address assignment (DHCP)
  - Location-insensitive queries
    - What if I don’t care what server a Web page comes from, as long as it’s the right page?
    - e.g., a yahoo page might be replicated

DNS Experience

- One of the greatest challenges seemed to be getting good name server implementations
  - Developers were typically happy with “good enough” implementation
  - Challenging, large scale, wide area distributed system
    - Like routing, but easier to have broken implementations that work
- Common bugs
  - Poor static configuration (root server list)
  - Lack of exponential backoff
  - No centralized caching per site
    - Each machine runs own caching local server

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 gTLDs

- .info -> general info
- .biz -> businesses
- .aero -> air-transport industry
- .coop -> business cooperatives
- .name -> individuals
- .pro -> accountants, lawyers, and physicians
- .museum -> museums
- Only new one active so far = .info, .biz

New Registrars

- Network Solutions (NSI) used to handle all registrations, root servers, etc...
  - Clearly not the democratic way
  - Large number of registrars that can create new domains
  - However NSI still handle root servers
Key Concepts

- The design of names, addresses and resolution has a significant impact on system capabilities
- Hierarchy, decentralization and caching allow the DNS to scale
  - These are general techniques!

For next time...

- HTTP: Read 9.2.2