Domain Name System
DNS

Behnam Momeni
Computer Engineering Department
Sharif University of Technology

Acknowledgments: Lecture slides are from Computer networks course thought by Jennifer Rexford at Princeton University. When slides are obtained from other sources, a reference will be noted on the bottom of that slide. A full list of references is provided on the last slide.
Separating Names and IP Addresses

• Names are easier (for us!) to remember
  – www.cnn.com vs. 64.236.16.20

• IP addresses can change underneath
  – Move www.cnn.com to 173.15.201.39
  – E.g., renumbering when changing providers

• Name could map to multiple IP addresses
  – www.cnn.com to multiple replicas of the Web site

• Map to different addresses in different places
  – Address of a nearby copy of the Web site
  – E.g., to reduce latency, or return different content

• Multiple names for the same address
  – E.g., aliases like ee.mit.edu and cs.mit.edu
Domain Name System (DNS)

Proposed in 1983 by Paul Mockapetris
Outline: Domain Name System

• Computer science concepts underlying DNS
  – Indirection: names in place of addresses
  – Hierarchy: in names, addresses, and servers
  – Caching: of mappings from names to/from addresses

• DNS software components
  – DNS resolvers
  – DNS servers

• DNS queries
  – Iterative queries
  – Recursive queries

• DNS caching based on time-to-live (TTL)
Strawman Solution #1: Local File

• Original name to address mapping
  – Flat namespace
  – /etc/hosts
  – Someone keeps main copy
  – Downloaded regularly

• Count of hosts was increasing: moving from a machine per domain to machine per user
  – Many more downloads
  – Many more updates
Strawman Solution #2: Central Server

• Central server
  – One place where all mappings are stored
  – All queries go to the central server

• Many practical problems
  – Single point of failure
  – High traffic volume
  – Distant centralized database
  – Single point of update
  – Does not scale

Need a distributed, hierarchical collection of servers
Domain Name System (DNS)

- Properties of DNS
  - Hierarchical name space divided into zones
  - Distributed over a collection of DNS servers

- Hierarchy of DNS servers
  - Root servers
  - Top-level domain (TLD) servers
  - Authoritative DNS servers

- Performing the translations
  - Local DNS servers
  - Resolver software
DNS Root Servers

- Labeled A through M

A Verisign, Dulles, VA
C Cogent, Herndon, VA (also Los Angeles)
D U Maryland College Park, MD
G US DoD Vienna, VA
H ARL Aberdeen, MD
J Verisign, (11 locations)
K RIPE London (also Amsterdam, Frankfurt)
I Autonomica, Stockholm (plus 3 other locations)
M WIDE Tokyo

B USC-ISI Marina del Rey, CA
F Internet Software C. Palo Alto, CA (and 17 other locations)
E NASA Mt View, CA
L ICANN Los Angeles, CA
TLD and Authoritative DNS Servers

- **Top-level domain (TLD) servers**
  - Generic domains (e.g., com, org, edu)
  - Country domains (e.g., uk, fr, ca, jp, ir)
  - Typically managed professionally
    - Network Solutions maintains servers for “com”
    - Educause maintains servers for “edu”
    - IPM maintains servers for “ir”

- **Authoritative DNS servers**
  - Provide public records for hosts at an organization
  - For the organization’s servers (e.g., Web and mail)
  - Can be maintained locally or by a service provider
Distributed Hierarchical Database

```
com  edu  •••  org
      generic domains

foo

west  east

my.east.foo.edu

ac  •••  uk  ir
      country domains

ac

cam

usr

usr.cam.ac.uk
```
Using DNS

- Local DNS server ("default name server")
  - Usually near the end hosts who use it
  - Local hosts configured with local server (e.g., /etc/resolv.conf) or learn the server via DHCP

- Client application
  - Extract server name (e.g., from the URL)
  - Do `gethostbyname()` to trigger resolver code

- Server application
  - Extract client IP address from socket
  - Optional `gethostbyaddr()` to translate into name
Example

Host at ce.sharif.edu wants IP address for mail.umass.edu
Recursive vs. Iterative Queries

- **Recursive query**
  - Ask server to get answer for you
  - E.g., request 1 and response 8

- **Iterative query**
  - Ask server who to ask next
  - E.g., all other request-response pairs
DNS Caching

- Performing all these queries take time
  - And all this before the actual communication takes place
  - E.g., 1-second latency before starting Web download

- Caching can substantially reduce overhead
  - The top-level servers very rarely change
  - Popular sites (e.g., www.cnn.com) visited often
  - Local DNS server often has the information cached

- How DNS caching works
  - DNS servers cache responses to queries
  - Responses include a “time to live” (TTL) field
  - Server deletes the cached entry after TTL expires
Negative Caching

- Remember things that don’t work
  - Misspellings like www.cnn.comm and www.cnnnn.com
  - These can take a long time to fail the first time
  - Good to remember that they don’t work
  - … so the failure takes less time the next time around
DNS Resource 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 hostname of authoritative name server for this domain

• Type=CNAME
  – name is alias name for some “canonical” (the real) name
    www.ibm.com is really servereast.backup2.ibm.com
  – value is canonical name

• Type=MX
  – value is name of mailserver associated with name
**DNS Protocol**

**DNS protocol**: *query* and *reply* messages, both with same *message format*

**Message header**

- **Identification**: 16 bit # for query, reply to query uses same #
- **Flags**:
  - Query or reply
  - Recursion desired
  - Recursion available
  - Reply is authoritative

<table>
<thead>
<tr>
<th>identification</th>
<th>flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of questions</td>
<td>number of answer RRs</td>
</tr>
<tr>
<td>number of authority RRs</td>
<td>number of additional RRs</td>
</tr>
<tr>
<td>questions (variable number of questions)</td>
<td></td>
</tr>
<tr>
<td>answers (variable number of resource records)</td>
<td></td>
</tr>
<tr>
<td>authority (variable number of resource records)</td>
<td></td>
</tr>
<tr>
<td>additional information (variable number of resource records)</td>
<td></td>
</tr>
</tbody>
</table>

12 bytes
Reliability

- DNS servers are replicated
  - Name service available if at least one replica is up
  - Queries can be load balanced between replicas
- UDP used for queries
  - Need reliability: must implement this on top of UDP
- Try alternate servers on timeout
  - Exponential backoff when retrying same server
- Same identifier for all queries
  - Don’t care which server responds
Inserting Resource Records into DNS

- **Example:** just created startup “FooBar”
- **Register foobar.com at Network Solutions**
  - Provide registrar with names and IP addresses of your authoritative name server (primary and secondary)
  - Registrar inserts two RRs into the com TLD server:
    - (foobar.com, dns1.foobar.com, NS)
    - (dns1.foobar.com, 212.212.212.1, A)
- **Put in authoritative server dns1.foobar.com**
  - Type A record for www.foobar.com
  - Type MX record for foobar.com
- **Play with “dig” on *NIX**