Topics

• 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
• Inner-workings of DNS
  – DNS resolvers and servers
  – Iterative and recursive queries
  – TTL-based caching
• Web and DNS
  – Influence of DNS queries on Web performance
  – Server selection and load balancing

Domain Name System (DNS)

The Basic Naming Problem:
  – Internet uses IP addresses like 129.82.100.64
  – People use names like www.colostate.edu
Solution: The Domain Name System:
  • application-layer protocol to resolve names (convert names like www.colostate.edu into IP addresses like 129.82.100.64)
  • distributed database implemented in hierarchy of many name servers
  • First step in nearly every application:
    – Use DNS to convert a name into an IP address.
DNS Query and Response

End-user

www.colostate.edu

A?

Root DNS Server

www.colostate.edu

A is 129.82.64.100

Caching DNS Server

duo DNS Server

Multiple servers for each zone in case any one server fails

- 13 root servers
- 13 edu servers
- 5 colostate.edu servers

colostate.edu DNS Server

Host Names vs. IP addresses

- Host names
  - Mnemonic name appreciated by humans
  - Variable length, alphanumeric characters
  - Provide little (if any) information about location

- IP addresses
  - Numerical address appreciated by routers
  - Fixed length, binary number
  - Hierarchical, related to host location
  - Examples: 64.236.16.20 and 193.30.227.161

Why Separate Naming and Addressing?

- Names are easier for people to remember
  - www.cnn.com vs. 64.236.16.20

- Addresses can change underneath
  - Move www.cnn.com to 64.236.16.20
  - 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
Strawman Solution: Local File

- Original name to address mapping
  - Flat namespace
  - /etc/hosts
  - SRI kept 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 (thus, never implemented)

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

- 13 root servers (see http://www.root-servers.org/)
- Labeled A through M

Types of DNS Servers

- Authoritative DNS servers:
  - provide authoritative records for a particular zone (e.g., colostate.edu, cisco.com, edu, uk, etc)
  - Can be maintained locally or by a service provider
- Top-level domain (TLD) servers:
  - Authoritative servers responsible for com, org, net, edu, etc, and all top-level country domains uk, fr, ca, jp.
  - Typically managed professionally
  - Network solutions maintains servers for com TLD
  - Educause for edu TLD

Caching Servers

- Accept queries for end hosts, lookup requested data, and cache answers for later replies.

DNS Organization

- Data organized as tree structure
  - Each zone is authoritative for its local data.
- Each zone operates a set of name servers that contain the zone data
  - Change to host cs.colostate.edu is entered at cs.colostate.edu servers.
- Tree structure directs queries to the appropriate name server
  - Root knows how to reach edu
  - Edu knows how to reach colostate.edu
  - Etc.
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 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

Types of Queries

recursive query:
• puts burden of name resolution on contacted name server
• heavy load?
• Query 1 is recursive

iterated query:
• contacted server replies with name of server to contact
• "I don't know this name, but ask this server"
• Queries 2, 4, and 6 are iterative

DNS Caching

• Performing all these queries takes 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.cnnn.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
    - Value is canonical name
  - **Type=MX**
    - Value is name of mailserver associated with name

DNS Messages (1/2)

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

  - **msg header**
    - **identification**: 16 bit # for query, reply to query uses same #
    - **flags**: query or reply, recursion desired, recursion available, reply is authoritative

  - **message format**:
    - Identification: 16 bit #
    - Flags: 16 bit #
    - Questions: variable length
    - Answers: variable length
    - Authority: variable length
    - Additional: variable length
DNS Messages (2/2)

- Name, type fields for a query
- RR's in response to query
- Records for authoritative servers
- Additional "helpful" info that may be used

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 back-off 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
DNS and the Web

DNS Query in Web Download
- User types or clicks on a URL
- Browser extracts the site name
  - E.g., www.cnn.com
- Browser calls gethostbyname() to learn IP address
  - Triggers resolver code to query the local DNS server
- Eventually, the resolver gets a reply
  - Resolver returns the IP address to the browser
- Then, the browser contacts the Web server
  - Creates and connects socket, and sends HTTP request

Multiple DNS Queries
- Often a Web page has embedded objects
  - E.g., HTML file with embedded images
- Each embedded object has its own URL
  - ... and potentially lives on a different Web server
  - E.g., http://www.myimages.com/image1.jpg
- Browser downloads embedded objects
  - Usually done automatically, unless configured otherwise
  - Requires learning the address for www.myimages.com
When are DNS Queries Unnecessary?

- Browser is configured to use a proxy
  - E.g., browser sends all HTTP requests through a proxy
  - Then, the proxy takes care of issuing the DNS request
- Requested Web resource is locally cached
  - E.g., cache has http://www.cnn.com/2006/leadstory.html
  - No need to fetch the resource, so no need to query
- Browser recently queried for this host name
  - E.g., user recently visited http://www.cnn.com/
  - So, the browser already called gethostbyname()
  - ... and may be locally caching the resulting IP address

Web Server Replicas

- Popular Web sites can be easily overloaded
  - Web site often runs on multiple server machines

<table>
<thead>
<tr>
<th>Internet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

- Simple approach: different names
  - But, this requires users to select specific replicas
- More elegant approach: different IP addresses
  - Single name (e.g., www.cnn.com), multiple addresses
  - E.g., 64.236.16.20, 64.236.16.52, 64.236.16.84, ...
- Authoritative DNS server returns many addresses
  - And the local DNS server selects one address
  - Authoritative server may vary the order of addresses
Clever Load Balancing Schemes

- Selecting the “best” IP address to return
  - Based on server performance
  - Based on geographic proximity
  - Based on network load
  - ...

- Example policies
  - Round-robin scheduling to balance server load
  - U.S. queries get one address, Europe another
  - Tracking the current load on each of the replicas

Challenge: What About DNS Caching?

- Problem: DNS caching
  - What if performance properties change?
  - Web clients still learning old “best” Web server
  - ... until the cached information expires

- Solution: Small Time-to-Live values
  - Setting artificially small TTL values
  - ... so replicas picked based on fresh information

- Disadvantages: abuse of DNS?
  - Many more DNS request/response messages
  - Longer latency in initiating the Web requests