Outline

- How DNS resolves names
- How well does DNS work today
- HTTP intro and details

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?

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

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

DNS Message Format

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification</td>
<td>Flags</td>
</tr>
<tr>
<td>No. of Questions</td>
<td>No. of Answer RRs</td>
</tr>
<tr>
<td>No. of Authority RRs</td>
<td>No. of Additional RRs</td>
</tr>
<tr>
<td>Questions (variable number of answers)</td>
<td>Answers (variable number of resource records)</td>
</tr>
<tr>
<td>RRs in response to query</td>
<td>Authority (variable number of resource records)</td>
</tr>
<tr>
<td>Records for authoritative servers</td>
<td>Additional Info (variable number of resource records)</td>
</tr>
<tr>
<td>Additional “helpful info that may be used”</td>
<td></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

Typical Resolution

Steps for resolving www.cmu.edu

- Application calls gethostbyname() (RESOLVER)
- Resolver contacts local name server (S1)
- S1 queries root server (S2) for www.cmu.edu
- S2 returns NS record for cmu.edu (S3)
- What about A record for S3?
  - This is what the additional information section is for (PREFETCHING)
  - S1 queries S3 for www.cmu.edu
  - S3 returns A record for www.cmu.edu
- Can return multiple A records → what does this mean?

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

• What workload do you expect for different servers?
  • 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

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?

Reliability

• Try alternate servers on timeout → what’s a timeout?
• What’s a good value for a timeout?
  • Hard to tell → what are the tradeoffs?
  • Better be conservative!
• Exponential backoff when retrying same server
  • Why do we need this?
• 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?
  - Happens to be www.intel-iris.net and mammoth.cmcl.cs.cmu.edu → what will reverse lookup return? Both?
    - Should only return name that reflects address allocation mechanism

Prefetching

- Name servers can add additional data to any 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

- How DNS resolves names
- How well does DNS work today
- HTTP intro and details
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

- Looped NS/CNAME record handling
- Poor static configuration (root server list)
- Lack of exponential backoff
- No centralized caching per site
  - Each machine runs own caching local server
  - Why is this a problem?
  - How many hosts do we need to share cache? → recent studies suggest 10-20 hosts

Recent Measurements

- Hit rate for DNS = 80%
  - Is this good or bad?
- Most Internet traffic is Web
  - What does a typical page look like? → average of 4-5 imbedded objects → needs 4-5 transfers
    - This alone accounts for 80% hit rate!
  - Lower TTLs for A records does not affect performance
  - DNS performance really relies more on NS-record caching

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, .name

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 handle root servers

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?

BREAK!!!
- Come see Mor & I get killed at Halo
  - Sunday 7pm, 7500 WeH
Outline

• How DNS resolves names
• How well does DNS work today
• HTTP intro and details

HTTP Basics

• HTTP layered over bidirectional byte stream
  • Almost always TCP
• Interaction
  • Client sends request to server, followed by response from server to client
  • Requests/responses are encoded in text
• Stateless
  • Server maintains no information about past client requests

How to Mark End of Message?

• Size of message → Content-Length
  • Must know size of transfer in advance
• Delimiter → MIME style Content-Type
  • Server must “escape” delimiter in content
• Close connection
  • Only server can do this

HTTP Request

• Request line
  • Method
    • GET – return URI
    • HEAD – return headers only of GET response
    • POST – send data to the server (forms, etc.)
  • URI
    • E.g. http://www.intel-iris.net/index.html with a proxy
    • E.g. /index.html if no proxy
  • HTTP version
HTTP Request

- Request headers
  - Authorization – authentication info
  - Acceptable document types/encodings
  - From – user email
  - If-Modified-Since
  - Referrer – what caused this page to be requested
  - User-Agent – client software
- Blank-line
- Body

HTTP Request Example

GET / HTTP/1.1
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)
Host: www.intel-iris.net
Connection: Keep-Alive

HTTP Response

- Status-line
  - HTTP version
    - 3 digit response code
      - 1XX – informational
      - 2XX – success
        - 200 OK
      - 3XX – redirection
        - 301 Moved Permanently
        - 303 Moved Temporarily
        - 304 Not Modified
      - 4XX – client error
        - 404 Not Found
      - 5XX – server error
        - 505 HTTP Version Not Supported
  - Reason phrase
HTTP Response

- Headers
  - Location – for redirection
  - Server – server software
  - WWW-Authenticate – request for authentication
  - Allow – list of methods supported (GET, HEAD, etc)
  - Content-Encoding – E.g. x-gzip
  - Content-Length
  - Content-Type
  - Expires
  - Last-Modified
- Blank-line
- Body

HTTP Response Example

HTTP/1.1 200 OK
Date: Tue, 27 Mar 2001 03:49:38 GMT
Server: Apache/1.3.14 (Unix) (RedHat/Linux) mod_ssl/2.7.1
OpenSSL/0.9.5a DAV/1.0.2 PHP/4.0.1pl2 mod_perl/1.24
Last-Modified: Mon, 29 Jan 2001 17:54:18 GMT
ETag: "7a11f-10ed-3a75ae4a"
Accept-Ranges: bytes
Content-Length: 4333
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
......

Cookies: Keeping “state”

Many major Web sites use cookies

Four components:
1) Cookie header line in the HTTP response message
2) Cookie header line in HTTP request message
3) Cookie file kept on user’s host and managed by user’s browser
4) Back-end database at Web site

Example:
- Susan access Internet always from same PC
- She visits a specific e-commerce site for first time
- When initial HTTP requests arrives at site, site creates a unique ID and creates an entry in backend database for ID

Cookies: Keeping “State” (Cont.)

client
Amazon server

Cookie file
ebay: 8734

usual http request msg

usual http response +
Set-cookie: 1678

server
creates ID
1678 for user

one week later:

Cookie file
amazon: 1678
ebay: 8734

usual http request msg

usual http response msg
cookie: 1678

cookie-specific
action

cookie-
specific
access

access

cookie-
specific
action

cookie-
specific
access

cookie-
specific
action

cookie-
specific
access

cookie-
specific
access
Typical Workload (Web Pages)

- Multiple (typically small) objects per page
- File sizes
  - Why different than request sizes?
  - Also heavy-tailed
    - Pareto distribution for tail
    - Lognormal for body of distribution
- Embedded references
  - Number of embedded objects = \( p(x) = a^x \cdot (a+1) \)

HTTP 0.9/1.0

- One request/response per TCP connection
  - Simple to implement
- Disadvantages
  - Multiple connection setups → three-way handshake each time
  - Several extra round trips added to transfer
  - Multiple slow starts

Single Transfer Example

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 RTT</td>
<td>SYN</td>
</tr>
<tr>
<td>1 RTT</td>
<td>ACK</td>
</tr>
<tr>
<td>2 RTT</td>
<td>ACK</td>
</tr>
<tr>
<td>3 RTT</td>
<td>ACK</td>
</tr>
<tr>
<td>4 RTT</td>
<td>ACK</td>
</tr>
</tbody>
</table>

More Problems

- Short transfers are hard on TCP
  - Stuck in slow start
  - Loss recovery is poor when windows are small
- Lots of extra connections
  - Increases server state/processing
- Server also forced to keep TIME_WAIT connection state
  - Why must server keep these?
  - Tends to be an order of magnitude greater than # of active connections, why?
Next Lecture

- HTTP Improvements
- Server Selection
- Content Distribution Networks