CSE 534
DNS

Vyas Sekar
Stony Brook University

Based on content from: Srini Seshan
Assigned reading

• DNS performance and effectiveness of caching
  – Jaeyeon Jung et al

• Development of DNS
  – Mockapetris and Dunlap
Naming

• How do we efficiently locate resources?
  – DNS: name $\rightarrow$ IP address
  – Service location: description $\rightarrow$ host

• Other issues
  – How do we scale these to the wide area?
Obvious Solutions (1)

Why not centralize DNS?

• Single point of failure
• Traffic volume
• Distant centralized database

• Doesn’t scale!
Obvious Solutions (2)

Why not use /etc/hosts?

• Original Name to Address Mapping
  – Flat namespace
  – /etc/hosts
  – SRI kept main copy
  – Downloaded regularly

• Count of hosts was increasing: machine per domain
  → machine per user
  – Many more downloads
  – Many more updates
Domain Name System (DNS) Goals

• Basically building a wide area distributed database
  – Scalability
  – Decentralized maintenance
  – Robustness
  – Global scope
    • Names mean the same thing everywhere

• Don’t need
  – Atomicity
  – Strong consistency
DNS Design

• DB contains tuples called resource records (RRs)
  – RR contains type, class and application data
  – Classes = Internet (IN), Chaosnet (CH), etc.

• Administrative hierarchy
  – “.” as separator
  – Zone = contiguous section of name space
    • Complete tree, single node or subtree
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 Records

RR format: (class, name, value, type, ttl)

FOR IN class:

- **Type=A**
  - **name** is hostname
  - **value** is IP address
- **Type=NS**
  - **name** is domain (e.g. foo.com)
  - **value** is name 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 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
• 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.StonyBrook.EDU created by StonyBrook.EDU administrators
Key roles: Servers + Resolvers

• Each host has a resolver
  – Typically a library that applications can link
  – Local name servers configured (e.g. /etc/resolv.conf)

• Name servers
  – Configured with well-known root servers
    • Currently {a-m}.root-servers.net
  – Local servers
    • Do recursive lookup of distant host names for local hosts
    • Typically answer queries about local zone
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
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
DNS Lookup Example

Client

Local DNS server

root & edu DNS server

cmu.edu DNS server

cs.cmu.edu DNS server

www.cs.cmu.edu

NS cmu.edu

NS cs.cmu.edu

www=IPaddr
Caching

- 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

- Cached data periodically times out
  - Lifetime (TTL) of data controlled by owner of data
  - TTL passed with every record
  - No explicit caps on cache size (records are small)
Subsequent Lookup Example
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 → Why not TCP?
  – Try alternate servers on timeout
  – Exponential backoff when retrying same server
  – Same identifier for all queries
    • Don’t care which server responds
Prefetching

• Name servers can add addition data on 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”
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
DNS Experience

• 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 on caching local server
    • Why is this a problem?

• Solution
  – Monitor for misbehaving name servers?
Assigned reading

• DNS performance and effectiveness of caching
  – Jaeyeon Jung et al

• Development of DNS
  – Mockapetris and Dunlap
Motivation for Jung paper

• DNS is critical infrastructure

• Little understood about performance

• People are using it in new, unexpected ways!
  – Web
  – Server selection
  – Fine-grained timescales?
Key driving questions

• How reliable and efficient?
  – User perspective

• How well are the scaling mechanisms working?
  – Infrastructure perspective

• How to pick TTL?
  – How low is too low?
Methodology

• Collect
  – DNS
  – TCP request/responses

• Why TCP also?

• Multiple vantage points – why?
DNS Experience - Client performance

• 23% of lookups with no answer
  – Inverse lookups and bogus NS records

• 13% error response $\rightarrow$ most = no name exists

• Retransmit aggressively $\rightarrow$ most packets in trace for unanswered lookups

• Increasing share of low TTL records

• Worst 10% lookup latency got much worse
One cause for errors:
Reverse Name Lookup

- 128.2.206.138?
  - Lookup 138.206.2.128.in-addr.arpa
  - Why is the address reversed?
  - Why is it that forward lookup can have multiple answers but not reverse?
Workload: Zipf-like

Name distribution = Zipf-like = \(1/x^\alpha\)
# Hit Rates

<table>
<thead>
<tr>
<th></th>
<th>mit-jan00</th>
<th>mit-dec00</th>
<th>kaist-may01</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>74.62%</td>
<td>81.17%</td>
<td>86.09%</td>
</tr>
<tr>
<td>1</td>
<td>24.07%</td>
<td>17.86%</td>
<td>10.43%</td>
</tr>
<tr>
<td>2</td>
<td>1.16%</td>
<td>0.87%</td>
<td>2.10%</td>
</tr>
<tr>
<td>3</td>
<td>0.11%</td>
<td>0.07%</td>
<td>0.38%</td>
</tr>
<tr>
<td>≥ 4</td>
<td>0.04%</td>
<td>0.03%</td>
<td>1.00%</td>
</tr>
</tbody>
</table>

Hit rates = 80 – 86%
Importance of NS caching

NS Caching seems more critical
Impact of TTL

TTL need not be very high to improve hit rate!
Interesting anecdote

• Most A-lookups followed by TCP connection

• Exceptions?

• Use of DNS for blacklists!
DNS Hacks: Blackhole Lists

• First: Mail Abuse Prevention System (MAPS)
  – Paul Vixie, 1997

• Spamhaus, spamcop, dnsrbl.org, etc.

```bash
% dig 91.53.195.211.bl.spamcop.net

;; ANSWER SECTION:
91.53.195.211.bl.spamcop.net. 2100 IN   A       127.0.0.2

;; ANSWER SECTION:
91.53.195.211.bl.spamcop.net. 1799 IN   TXT     "Blocked - see http://www.spamcop.net/bl.shtml?211.195.53.91"
```

Different addresses refer to different reasons for blocking.
Open questions

- Security?
- Availability?
- Misconfigurations?
- Mobility? Portability?
- Names become valuable?
- Typosquatting?
Key Takeaways

• Hierarchy
  – Delegate ownership

• Caching
  – Performance, scalability
  – Also for negative responses

• Datagram
  – Avoid state on top name servers

• New uses
  – CDN, Server load balancing, Spam blacklist