Administrivia

- **Midterm exam Wednesday**
  - Open book, Open notes, no electronic devices allowed
  - Feel free to print out and bring lecture slides

- **SCPD students:**
  - Email cs144-aut1112-staff@lists.stanford.edu with your exam monitor information
  - Please ensure the email subject is “exam monitor”

- **Any other students with special exam needs**
  - Please email cs144-staff to make arrangements

Outline

- DNS architecture
- DNS protocol and resource records (RRs)
- Record types: A, NS, glue, MX, SOA, CNAME
- Reverse lookup
- Load balancing
- DNS security

Motivation

- Users can’t remember IP addresses
  - Need to map symbolic names (www.stanford.edu) → IP addr

- Implemented by library functions & servers
  - getaddrinfo () talks to server over UDP (sometimes TCP)

- Actually, more generally, need to map symbolic names to values

Goals of DNS

- **Scalability**
  - Must handle huge number of records
  - Potentially **exponential** in name size—because custom software may synthesize names on-the-fly

- **Distributed control**
  - Let people control their own names

- **Fault-tolerance**
  - Old software assumed hosts.txt always there
  - Bad potential failure modes when name lookups fail
  - Minimize lookup failures in the face of other network problems

Parsing a URL

```
http://cs144.scs.stanford.edu/labs/sc.html
```

```
<table>
<thead>
<tr>
<th>File</th>
<th>Host</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>192.12.69.5</td>
<td>TCP</td>
</tr>
<tr>
<td>2</td>
<td><a href="mailto:user@cs.princeton.edu">user@cs.princeton.edu</a></td>
<td>IP</td>
</tr>
<tr>
<td>3</td>
<td>192.12.69.5</td>
<td>TCP</td>
</tr>
<tr>
<td>4</td>
<td>TTP</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>IP</td>
<td></td>
</tr>
</tbody>
</table>
```

- **Parsing a URL**
  - **File**
  - **Host**
  - **Protocol**

- **Motivation**
  - Users can’t remember IP addresses
  - Implemented by library functions & servers
  - Actually, more generally, need to map symbolic names to values

- **Goals of DNS**
  - **Scalability**
    - Must handle huge number of records
    - Potentially **exponential** in name size—because custom software may synthesize names on-the-fly

- **Distributed control**
  - Let people control their own names

- **Fault-tolerance**
  - Old software assumed hosts.txt always there
  - Bad potential failure modes when name lookups fail
  - Minimize lookup failures in the face of other network problems

**hosts.txt system**

- **Originally, hosts were listed in a file, hosts.txt**
  - Email global network administrator when you add a host
  - Administrator mails out new hosts.txt file every few days

- **Would be completely impractical today**
  - hosts.txt today would be huge (Gigabytes)
  - What if two people wanted to add same name?
  - Who is authorized to change address of a name?
  - People need to change name mappings more often than every few days (e.g., Dynamic IP addresses)
The good news

- Properties that make DNS goals easier to achieve:
  1. **Read-only or read-mostly database**
     - People typically look up hostnames much more often than they are updated
  2. **Loose consistency**
     - When adding a machine, may be okay if info takes minutes or hours to propagate
  3. **These suggest approach w. aggressive caching**
     - Once you have looked up hostname, remember result
     - Don’t need to look it up again in near future

**Domain Name System (DNS)**

- **Break namespace into a bunch of zones**
  - . ("root"), edu., stanford.edu., cs.stanford.edu., ...
  - Zones separately administered → delegation
  - Parent zones tell you how to find servers for subdomains.
- **Each zone served from several replicated servers**

**Root servers**

- Root (and TLD) servers must be widely replicated
  - For some, use various tricks like IP anycast

**DNS software architecture**

- **Two types of query**
  - Recursive
  - Non-Recursive
- **Apps make recursive queries to local DNS server (1)**
- **Local server queries remote servers non-recursively (2, 4, 6)**
  - Aggressively caches result
  - E.g., only contact root on first query ending .umass.edu

**Resource records**

- All DNS info represented as resource records (RR):
  
  ```
  name [TTL] [class] [type] rdata
  ```
  
  - name – domain name (e.g., www.stanford.edu.)
  - TTL – time to live in seconds
  - class – for extensibility, usually IN (1) “Internet”
  - type – type of the record
  - rdata – resource data dependent on the type
- **Two important DNS RR types:**
  - A – Internet address (IPv4)
  - NS – name server
- **Example resource records (dig stanford.edu):**
  stanford.edu. 1800 IN A 171.67.216.14
  stanford.edu. 1800 IN A 171.67.216.16
  stanford.edu. 172800 IN NS Argus.stanford.edu.
  ...

**DNS protocol**

- TCP/UDP port 53
- Most traffic uses UDP
  - Lightweight protocol has 512 byte UDP message limit
  - retry w. TCP if UDP fails (e.g., reply truncated)
- TCP requires message boundaries
  - Prefix all messages w. 16-bit length
- Bit in query determines if query is recursive
Some implementation details

- How does local name server know root servers?
  - Need to configure name server with root cache file
  - Contains root name servers and their addresses
    3600000 NS  A.ROOT-SERVERS.NET.
    3600000 A 198.41.0.4
    3600000 NS B.ROOT-SERVERS.NET.
    3600000 A 128.9.0.107

- How do you get addresses of other name servers
  - To lookup names ending .stanford.edu, ask Argus.stanford.edu.
  - Chicken and egg problem:
    How to get Argus.stanford.edu.'s address?
  - Solution: glue records – A records in parent zone
  - Name servers for edu. have A record of Argus.stanford.edu.

Glue Record Example

- Look up www.scs.stanford.edu assuming no cache
  dig +norec www.scs.stanford.edu @a.root-servers.net
  dig +norec www.scs.stanford.edu @a.edu-servers.net
  dig +norec www.scs.stanford.edu @argus.stanford.edu
  dig +norec www.scs.stanford.edu @ns1.fs.net

- Get intermediary results for .edu, stanford.edu, scs.stanford.edu, and www.scs.stanford.edu

- Where are the glue records?

Structure of a DNS message [RFC 1035]

- Same message format for queries and replies
  - Query has zero RRs in Answer/Authority/Additional sections
  - Reply includes question, plus has RRs

- Authority allows for delegation

- Additional for glue + other RRs client might need

Encoding of RRs

- A DNS name consists of a series of labels
  - www.stanford.edu. has 3 labels: www, stanford, and edu
  - Labels can contain letters, digits, and "-", but should not start or end with "-"
  - Maximum length 63 characters
  - Encoded as length byte followed by label
  - Last label always empty (zero-length) label

- Names are case insensitive
  - But server must preserve case of question in replies
  - Example: request www.sTANford.EDu, look at authority
Secondary servers

- Availability requires geographically disparate replicas
  - E.g., I ask MIT to serve scs.stanford.edu
- Typical setup: One master many slave servers
- How often to sync up servers? Trade-off
  - All the time ⇒ high overhead
  - Rarely ⇒ stale data
- Put trade-off under domain owner’s control
  - Fields in SOA record control secondary’s behavior
  - Primary can unilaterally change SOA
  - To speed propagation, primary can also notify secondary of change, providing a hint to refresh sooner [RFC 1996]

CNAME records

- CNAME record specifies an alias:
  
  \[ \text{name [TTL] [IN] CNAME canonical-name} \]
  
  - As if any RR’s associated w. canonical-name also for name
  - Can look up with AI_CANONNAME flag to getaddrinfo

- Examples, to save typing:

  \[ \begin{align*}
  \text{wb.scs.stanford.edu. CNAME williamsburg-bridge.scs.stanford.edu.} \\
  \text{mb.scs.stanford.edu. CNAME manhattan-bridge.scs.stanford.edu.}
  \end{align*} \]

- CNAME precludes any other RRs for name
  - E.g., might want: david.com CNAME david.stanford.edu
  - Illegal, because david.com would need NS records
- Note answer section can have CNAME for query name + other RR(s) for canonical-name
  - But don’t point MXes to CNAMEs, as no A recs in additional section (try bad-mx.scs.stanford.edu.)

Other Records

- Start of Authority (SOA) record
  - States administrative information for a zone
  - dig stanford.edu soa
  - Tells you how long you can cache negative results
- Mail Exchange (MX) record
  - For historical reasons, mail does not have to use A records directly
  - Example: ping scs.stanford.edu
  - No such host, but you can still mail CS144 staff there
  - dig scs.stanford.edu mx

Mapping addresses to names

- PTR records specify names
  
  \[ \text{name [TTL] [IN] PTR “ptrdname”} \]
  
  - name – somehow encode address...how?
  - ptrdname – domain name for this address

- IPv4 addr stored under in-addr.arpa domain
  - Reverse name, append in-addr.arpa
  - To look up 171.66.3.9 → 9.3.66.171.in-addr.arpa.
  - Why reversed? Delegation!

- IPv6 under ip6.arpa
  - Historical note: ARPA funded original Internet
  - Acronym now re-purposed [RFC 3172]: Address and Routing Parameter Area

Reverse Lookups

- Remember traceroute…
  - Traceroute can learn names of hosts through reverse lookup
  - 128.30.2.121 → 121.2.30.128.in-addr.arpa
- PTR record points to canonical name
  - Example:
    - tinyos.stanford.edu → sing.stanford.edu
    - sing.stanford.edu → 171.67.76.65
    - 65.76.67.171.in-addr.arpa → sing.stanford.edu

2-minute stretch
Using DNS for load-balancing

- Can have multiple RR of most types for one name
  - Required for NS records (for availability)
  - Useful for A records
  - (Not legal for CNAME records)

- Servers rotate order in which records returned
  - getaddrinfo returns a linked list of addrinfo structures
  - Most apps just use first address returned
  - Even if your name server caches results, clients will be spread amongst servers

- Example: dig cnn.com multiple times

SRV records

- Service location records
  
  _service_.proto.name [...] SRV prio weight port target

  - _service_ – E.g., _sip for SIP (VOIP) protocol
  - _proto_ – _tcp or _udp
  - name – domain name record applies to
  - prio – as with MX records, lower # → higher priority
  - weight – within priority, affects randomization of order
  - port – TCP or UDP port number (particularly useful for SIP)
  - target – Server name, for which client needs A record

- Like a generalization of MX records for arbitrary services

TXT records

- Can place arbitrary text in DNS

  name [TTL] [IN] TXT “text” …

  - text – whatever you want it to mean

- Great for prototyping new services
  - Don’t need to change DNS infrastructure

- Example: dig gmail.com txt

  - What’s this? SPF = “sender policy framework”
    (previously known as “sender permitted from”)
  - Much spam is forged email
  - SPF specifies IP addresses allowed to send mail from @gmail.com
  - Can have incremental deployment
  - Only mail servers must change, DNS can stay the same
  - Now SPF standardized (sort of), has RR type 99 [RFC 4408]

Editorial

- SPF is based on envelope sender address
  - Nice because available earlier in SMTP protocol
  - So some users can reject forged mail while some accept

- Microsoft proposed competing standard, Sender ID [RFC 4406]
  - Instead of simple language, used XML monstrosity
  - Instead of envelope sender, extract address from message

- No agreement between camps, couldn’t standardize
  - Compromise: kill XML, but use address in message
  - But Microsoft patented extracting address from message!

SPF vs. Sender ID (continued)

- Compromise 2: Have two competing standards
  - After a few years, see which standard more widely used

- Use different formats for SPF vs. Sender ID
  - Start SPF records with string “v=spf1 ”
  - Start Sender ID records with string “spf2.0/pra ”

- SPF had a head start—lots of sites had adopted it

- Dirty trick appeared in final draft of Sender ID
  - If no spf2.0/pra record present, but see v=spf1, treat v=spf1 as if it were a sender ID record
  - Causes sender ID machines to reject mail from SPF sites (E.g., if you use SPF and post to mailing list, some recipients will reject)
  - Thwarts idea of independent experiment

DNS redirection for content distribution

- Play with akamai and www.microsoft.com
Classless in-addr delegation

- How to delegate on non-byte boundary?
- Solution: Use CNAME records
  - So-called classless in-addr delegation
- Example:
  1.3.66.171.in-addr.arpa. CNAME 1.ptr.your-domain.com.
  2.3.66.171.in-addr.arpa. CNAME 2.ptr.your-domain.com.
  3.3.66.171.in-addr.arpa. CNAME 3.ptr.your-domain.com.

DNS exploits

- July 29, 2008, Bruce Schneier:
  Despite the best efforts of the security community, the details of a critical internet vulnerability discovered by Dan Kaminsky about six months ago have leaked.
- One of the basic problems: DNS caching
  - If you can poison the cache, the damage stays
  - Who knows how far it spreads…

DNS exploit example

- Alice wants to look up www.google.com
- Bob the attacker knows
- Bob knows source address/port, destination address/port
- Bob generates a spoof response: www.google.com is www.evil.com
- Challenge: Bob has to guess Query ID
- If Bob guesses, RR can stay in Alice's cache a long time

Countermeasures

- Choose good QIDs (used to be incremented, now randomly generated), 16 bits
- Randomize source port, 16 bits
- Some protection, but only makes it take longer, networks are faster each day
Another exploit

- DNS clients used to trust all responses
- Problem: glue records and helpful A records
  - Ask NS of evil.com for www.evil.com
  - Says www.evil.com is a CNAME for www.amazon.com
  - Provides A record for www.amazon.com

Exploit Example

www.evil.com \[\rightarrow\] CNAME

www.amazon.com \[\rightarrow\] A

66.66.66.66

It gets worse

- Glue records can overwrite standard A records
- Even if you have a good A record for www.amazon.com, it's overwritten
- E.g., Server wants name of my IP address
  - Looks up 66.66.66.66.in-addr.arpa
- I say nameserver for 66.66.66.66.in-addr.arpa is www.amazon.com
  - Include glue A record for www.amazon.com in my reply

Solution 1

- Only use glue records for duration of query
  - Cache only end-to-end traversal of pointers, not intermediate steps
- In CNAME example www.evil.com will point to evil server
  - www.amazon.com will not point to evil server
- In in-addr.arpa example, can lie about hostname
  - But I can lie anyway
  - Have to check reverse lookup result by doing forward lookup

Example

www.evil.com \[\rightarrow\] www.amazon.com

66.66.66.66

Solution 2: bailiwick checking

- Only pay attention to answers for the domain you’ve asked
- Response from evil.com can’t tell you the A record for google.com
- Ask google.com for www.google.com
- Opponent can still race, but at least it's not deterministic
Kaminsky exploit
- Make winning the race easier
- Brute force attack
- Force Alice to look up AAAA.google.com, AAAAB.google.com, etc.
- Forge CNAME responses for each lookup, inserting A record for www.google.com
- Circumvents bailiwick checking

Solution: signatures
- Signature: cryptographic way to prove a party is who they say they are (more later in quarter)
- Requires a chain of trust
- Whom do you trust to sign DNS?
- DNSSEC extensions may finally be deployed soon [RFC 4033]

DNS Overview
- Distributed system for mapping names to values (e.g., IP addresses)
- Read-dominated workload allows caching
- Name structure allows distribution, independent administration
- Caching means bad data can stay a long time
- Standard protocol does not authenticate response is from server: DNSSec does