Reminders

• Homework 2 due Oct 1 @ 3:50 pm
• Oct 1 is this Wednesday…
• Project 1 checkpoint due Oct 6 @ 11:59:59 pm
Goals of Today’s Lecture

• Concepts & principles underlying the Domain Name System (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
  – Use of the `dig` utility

• Security analysis

• FYI: Project 2 is a dynamic DNS server!

Host Names vs. IP addresses

• Host names
  – Mnemonic name appreciated by humans
  – Variable length, full alphabet of characters
  – Provide little (if any) information about location
  – Examples: www.cnn.com and bbc.co.uk

• IP addresses
  – Numerical address appreciated by routers
  – Fixed length, binary number
  – Hierarchical, related to host location
  – Examples: 64.236.16.20 and 212.58.224.131
Separating Naming and Addressing

• Names are easier to remember
  – www.cnn.com vs. 64.236.16.20

• Addresses can change underneath
  – Move www.cnn.com to 4.125.91.21
  – E.g., renumbering when changing providers

• Name could map to multiple IP addresses
  – www.cnn.com to multiple (8) replicas of the Web site
  – Enables
    o Load-balancing
    o Reducing latency by picking nearby servers
    o Tailoring content based on requester’s location/identity

• Multiple names for the same address
  – E.g., aliases like www.cnn.com and cnn.com

Scalable (Name ↔ Address) Mappings

• Originally: per-host file
  – Flat namespace
  – /etc/hosts (what is this on your computer today?)
  – SRI (Menlo Park) kept master copy
  – Downloaded regularly

• Single server doesn’t scale
  – Traffic implosion (lookups & updates)
  – Single point of failure
  – Amazing politics

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

- Properties of DNS
  - Hierarchical name space divided into zones
  - Zones distributed over collection of DNS servers

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

- Performing the translations
  - Local DNS servers
  - Resolver software

Distributed Hierarchical Database

```
com  edu  • • •  org
    bar
west  east
      foo  my
    my.east.bar.edu

ac  • • •  uk  zw  arpa
    ac
      cam
      usr
    usr.cam.ac.uk

Top-Level Domains (TLDs)
generic domains  country domains

in-addr
```
DNS Root

- Located in Virginia, USA
- How do we make the root scale?

Verisign, Dulles, VA

DNS Root Servers

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

A Verisign, Dulles, VA
C Cogent, Herndon, VA
D U Maryland College Park, MD
G US DoD Vienna, VA
H ARL Aberdeen, MD
J Verisign

K RIPE London
I Autonomica, Stockholm
M WIDE Tokyo

E NASA Mt View, CA
F Internet Software Consortium
  Palo Alto, CA
B USC-ISI Marina del Rey, CA
L ICANN Los Angeles, CA
DNS Root Servers

- 13 root servers (see http://www.root-servers.org/)
  - Labeled A through M
- Replication via **any-casting** (localized routing for addresses)

TLD and Authoritative DNS Servers

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

- Authoritative DNS servers
  - Provide public records for hosts at an organization
    - Private records may differ, though **not** part of original design’s intent
  - For the organization’s servers (e.g., Web and mail)
  - Can be maintained locally or by a service provider
Using DNS

• Local DNS server (“default name server”)
  – Usually near the endhosts that use it
  – Local hosts configured with local server (e.g., `/etc/resolv.conf`) or learn 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 `cis.poly.edu` wants IP address for `gaia.cs.umass.edu`

1. Requesting host
2. Local DNS server
3. TLD DNS server
4. Authoritative DNS server
5. Root DNS server
6. TLD DNS server
7. Local DNS server
8. Requesting host
How did it know the root server IP?

- Hard-coded
- What if it changes?

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

ICANN Blog
Internet Corporation for Assigned Names and Numbers

Advisory — “L Root” changing IP address on 1st November
Creative Commons 3.0 License by ICANN

This is an advance notice that there is a scheduled change to the IP address for one of the authorities listed for the DNS root zone. The change is to L.ROOT-SERVERS.NET, which is administered by ICANN.

The new IPv4 address for this authority is 199.7.83.42.

This change is anticipated to be implemented in the root zone on 1 November 2007, however the new address is operational now. It will replace the previous IP address of 198.32.64.12.

We encourage operators of DNS infrastructure to update any references to the old IP address and replace it with the new address.
Reverse Mapping (Address → Host)

- How do we go the other direction, from an IP address to the corresponding hostname?
- Addresses already have natural “quad” hierarchy: 12.34.56.78
- But: quad notation has most-sig. hierarchy element on left, while www.cnn.com has it on the right
- Idea: reverse the quads = 78.56.34.12 ...
  - … and look that up in the DNS
- Under what TLD?
  - Convention: in-addr.arpa
  - So lookup is for 78.56.34.12.in-addr.arpa

Distributed Hierarchical Database

```
com  edu  • • •  org  
|     |     |     |     |
bar      west  east

ac  • • •  uk  zw

my.east.bar.edu

usr.cam.ac.uk

arpa

in-addr

12 34 56

12.34.56.0/24
```
DNS Caching

- Performing all these queries takes time
  - And all this before actual communication takes place
  - E.g., 1-second latency before starting Web download
- **Caching** can greatly 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 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
- But: negative caching is **optional**
  - And not widely implemented
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=PTR**
  - `name` is reversed IP quads
    - E.g. 78.56.34.12.in-addr.arpa
  - `value` is corresponding hostname

- **Type=CNAME**
  - `name` is alias name for some “canonical” name
    - E.g. `www.cs.mit.edu` is really `eecsweb.mit.edu`
  - `value` is canonical name

- **Type=MX**
  - `value` is name of mailserver associated with `name`
  - Also includes a weight/preference

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

- Plus fields indicating size (0 or more) of optional header elements
Interactive DNS lookups using **dig**

- **dig** program on Unix
  - Allows querying of DNS system
  - Dumps each field in DNS responses
  - By default, executes recursive queries
    - Disable via `+norecurse` so that operates one step at a time

```
unix> dig +norecurse @a.root-servers.net www.cnn.com
;; <<>> DiG 9.2.2 <<>> +norecurse @a.root-servers.net www.cnn.com
;; global options:  printcmd
;; Got answer:
;; ->>>HEADER<<- opcode: QUERY, status: NOERROR, id: 21041
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 14

;; QUESTION SECTION:

;; AUTHORITY SECTION:
com. 172800 IN NS A.GTLD-SERVERS.NET.
com. 172800 IN NS B.GTLD-SERVERS.NET.
com. 172800 IN NS C.GTLD-SERVERS.NET.
com. 172800 IN NS D.GTLD-SERVERS.NET.
com. 172800 IN NS E.GTLD-SERVERS.NET.
com. 172800 IN NS F.GTLD-SERVERS.NET.
com. 172800 IN NS G.GTLD-SERVERS.NET.
com. 172800 IN NS H.GTLD-SERVERS.NET.
com. 172800 IN NS I.GTLD-SERVERS.NET.
com. 172800 IN NS J.GTLD-SERVERS.NET.
com. 172800 IN NS K.GTLD-SERVERS.NET.
com. 172800 IN NS L.GTLD-SERVERS.NET.
com. 172800 IN NS M.GTLD-SERVERS.NET.

Note, no “ANSWER” section
```
dig +norecurse @g.gtld-servers.net www.cnn.com

; ; ADDITIONAL SECTION:
A.GTLD-SERVERS.NET. 172800 IN AAAA 2001:503:a83e::2:30
A.GTLD-SERVERS.NET. 172800 IN A 192.5.6.30
G.GTLD-SERVERS.NET. 172800 IN A 192.42.93.30
H.GTLD-SERVERS.NET. 172800 IN A 192.54.112.30
C.GTLD-SERVERS.NET. 172800 IN A 192.26.92.30
I.GTLD-SERVERS.NET. 172800 IN A 192.43.172.30
B.GTLD-SERVERS.NET. 172800 IN AAAA 2001:503:231d::2:30
B.GTLD-SERVERS.NET. 172800 IN A 192.33.14.30
D.GTLD-SERVERS.NET. 172800 IN A 192.31.80.30
L.GTLD-SERVERS.NET. 172800 IN A 192.41.162.30
F.GTLD-SERVERS.NET. 172800 IN A 192.35.51.30
J.GTLD-SERVERS.NET. 172800 IN A 192.48.79.30
K.GTLD-SERVERS.NET. 172800 IN A 192.52.178.30
E.GTLD-SERVERS.NET. 172800 IN A 192.12.94.30

;; Query time: 117 msec
;; SERVER: 198.41.0.4#53(a.root-servers.net)
;; WHEN: Mon Sep 25 11:13:15 2006
;; MSG SIZE  rcvd: 501

dig +norecurse @g.gtld-servers.net www.cnn.com

; ; QUESTION SECTION:
; ; AUTHORITY SECTION:
cnn.com. 172800 IN NS twdns-01.ns.aol.com.
cnn.com. 172800 IN NS twdns-03.ns.aol.com.

; ; ADDITIONAL SECTION:
twdns-01.ns.aol.com. 172800 IN A 149.174.213.151
twdns-02.ns.aol.com. 172800 IN A 152.163.239.216
twdns-03.ns.aol.com. 172800 IN A 207.200.73.85
twdns-04.ns.aol.com. 172800 IN A 64.12.147.120
Reliability

- DNS servers are replicated
  - Name service available if at least one replica is up
  - Queries can be load-balanced between replicas

- Usually, UDP used for queries
  - Need reliability: must implement this on top of UDP
  - Spec supports TCP too, but not always implemented

- 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”
- Get a block of address space from ISP
  - Say 212.44.9.128/25
- Register foobar.com at Network Solutions (say)
  - Provide registrar with names and IP addresses of your authoritative name server (primary and secondary)
  - Registrar inserts RR pairs into the com TLD server:
    0 (foobar.com, dns1.foobar.com, NS)
    0 (dns1.foobar.com, 212.44.9.129, A)
- Put in your (authoritative) server dns1.foobar.com:
  - Type A record for www.foobar.com
  - Type MX record for foobar.com

Setting up foobar.com, con’t

- In addition, need to provide reverse PTR bindings
  - E.g., 212.44.9.129 → dns1.foobar.com
- Normally, these would go in 9.44.212.in-addr.arpa
- Problem: you can’t run the name server for that domain. Why not?
  - Because your block is 212.44.9.128/25, not 212.44.9.0/24
  - And whoever has 212.44.9.0/24 won’t be happy with you owning their PTR records
- Solution: ISP runs it for you
  - Now it’s more of a headache to keep it up-to-date :-( 
Security Analysis of DNS

• What security issues does the design & operation of the Domain Name System raise?

• Degrees of freedom:

<table>
<thead>
<tr>
<th>16 bits</th>
<th>16 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification</td>
<td>Flags</td>
</tr>
<tr>
<td># Questions</td>
<td># Answer RRs</td>
</tr>
<tr>
<td># Authority RRs</td>
<td># Additional RRs</td>
</tr>
</tbody>
</table>

| Questions (variable # of resource records) |
| Answers (variable # of resource records) |
| Authority (variable # of resource records) |
| Additional information (variable # of resource records) |

Security Problem #1: Starbucks

• As you sip your latte and surf the Web, how does your laptop find google.com?

• Answer: it asks the local name server per Dynamic Host Configuration Protocol (DHCP) …
  – … which is run by Starbucks or their contractor
  – … and can return to you any answer they please
  – … including a “man in the middle” site that forwards your query to Google, gets the reply to forward back to you, yet can change anything they wish in either direction

• How can you know you’re getting correct data?
  – Today, you can’t. (Though if site is HTTPS, that helps)
  – One day, hopefully: DNSSEC extensions to DNS
Security Problem #2: Cache Poisoning

- Suppose you are a Bad Guy and you control the name server for foobar.com. You receive a request to resolve www.foobar.com and reply:

```
;; QUESTION SECTION:
;www.foobar.com.    IN    A    Evidence of the attack disappears 5 seconds later!

;; ANSWER SECTION:
www.foobar.com. 300 IN A 212.44.9.144

;; AUTHORITY SECTION:
foobar.com. 600 IN NS google.com.

;; ADDITIONAL SECTION:
google.com. 5 IN A 212.44.9.155
```

A foobar.com machine, not google.com

Cache Poisoning, con’t

- Okay, but how do you get the victim to look up www.foobar.com in the first place?

- Perhaps you connect to their mail server and send
  - HELO www.foobar.com
  - Which their mail server then looks up to see if it corresponds to your source address (anti-spam measure)

- Note, with compromised name server we can also lie about PTR records (address → name mapping)
  - E.g., for 212.44.9.155 = 155.44.9.212.in-addr.arpa return google.com (or whitehouse.gov, or whatever)
    o If our ISP lets us manage those records as we see fit, or we happen to directly manage them
Cache Poisoning, con’t

• Suppose Bad Guy is at Starbucks and they can sniff (or even guess) the identification field the local server will use in its next request:

<table>
<thead>
<tr>
<th>16 bits</th>
<th>16 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification</td>
<td>Flags</td>
</tr>
</tbody>
</table>

• They:
  – Ask local server for a (recursive) lookup of google.com
  – Locally spoof subsequent reply from correct name server using the identification field
  – Bogus reply arrives sooner than legit one

• Local server duly caches the bogus reply!
  – Now: every future Starbuck customer is served the bogus answer out of the local server’s cache
  o In this case, the reply uses a large TTL

Summary

• Domain Name System (DNS)
  – Distributed, hierarchical database
  – Indirection gets us human-readable names, ability to change address, etc.
  – Caching to improve performance
  – Examine using dig utility

• DNS lacks authentication
  – Can’t tell if reply comes from the correct source
  – Can’t tell if correct source tells the truth
  – Malicious source can insert extra (mis)information
  – Malicious bystander can spoof (mis)information
  – Playing with caching lifetimes adds extra power to attacks
Next Lecture

• An application protocol: The Web
• Reading: K&R 2.2
• Homework 2 due Oct 1 @ 3:50 pm (this Wed)

• Project 1 checkpoint due Oct 6 @ 11:59:59 pm