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

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
    - Load-balancing
    - Reducing latency by picking nearby servers
    - 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
Distributed Hierarchical Database

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

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

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)

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
**Example**

Host at cis.poly.edu wants IP address for gaia.cs.umass.edu

1. Requesting host: cis.poly.edu
2. Local DNS server: dns.poly.edu
3. Authority DNS server: dns.cs.umass.edu
4. TLD DNS server
5. Root DNS server
6. gaia.cs.umass.edu

**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

**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**

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

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
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:
  - (foobar.com, dns1.foobar.com, NS)
    - (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 212.44.9.0/25 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:
  - Identification
  - Flags
  - Questions (variable # of resource records)
  - # Authority RRs
  - # Additional RRs
  - Answers (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
; 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
```

Evidence of the attack disappears 5 seconds later!

```
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 `RELO 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)
  - 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:
- 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 Starbucks customer is served the bogus answer out of the local server’s cache
  - 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 September @ 3:59 pm (this Wed)
- Project 1 checkpoint due Oct 7 @ 11:59:59 pm