CMSC 332

Computer Networking

Email and DNS

Professor Szajda
• Last lecture we talked about design principles, and the application protocols HTTP and FTP
  ‣ Text commands sent over a port (recall telnet example)
  ‣ Difference in *statefullness*
  ‣ HTTP and FTP are primarily *pull* protocols
Chapter 2: Application layer

• 2.1 Principles of network applications
• 2.2 Web and HTTP
• 2.3 FTP
• 2.4 Electronic Mail
• 2.5 DNS
• 2.6 P2P Applications
Electronic Mail

Three major components:

- user agents
- mail servers
- simple mail transfer protocol: SMTP

User Agent

- a.k.a. “mail reader”
- composing, editing, reading mail messages
- e.g., Eudora, Outlook, elm, pine, Apple Mail, GMail
- outgoing, incoming messages stored on server
Electronic Mail: mail servers

Mail Servers

- **mailbox** contains incoming messages for user

- **message queue** of outgoing (to be sent) mail messages

- **SMTP protocol** between mail servers to send email messages
  - client: sending mail server
  - “server”: receiving mail server
Electronic Mail: SMTP [RFC 5321]

- uses TCP to reliably transfer email message from client to server, port 25
- direct transfer: sending server to receiving server
- three phases of transfer
  - handshaking (greeting)
  - transfer of messages
  - closure
- command/response interaction
  - commands: ASCII text
  - response: status code and phrase
- messages must be in 7-bit ASCII
Scenario: Alice sends message to Bob

1) Alice uses UA to compose message and “to” bob@someschool.edu

2) Alice’s UA sends message to her mail server; message placed in message queue

3) Client side of SMTP opens TCP connection with Bob’s mail server

4) SMTP client sends Alice’s message over the TCP connection

5) Bob’s mail server places the message in Bob’s mailbox

6) Bob invokes his user agent to read message
Sample SMTP interaction

S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
Try SMTP interaction for yourself:

- `telnet servername 25`
- `see 220 reply from server`
- `enter HELO, MAIL FROM, RCPT TO, DATA, QUIT commands`

above lets you send email without using email client (reader)
SMTP: final words

- SMTP uses persistent connections
  - Just like...?
- SMTP requires message (header & body) to be in 7-bit ASCII
- SMTP server uses CRLF.CRLF to determine end of message

Comparison with HTTP:
- HTTP: pull
- SMTP: push
- both have ASCII command/response interaction, status codes
- HTTP: each object encapsulated in its own response msg
- SMTP: multiple objects sent in multipart msg
Mail message format

SMTP: protocol for exchanging email msgs

RFC 2822: standard for text message format:

- header lines, e.g.,
  - To:
  - From:
  - Subject:

  different from SMTP commands!

- body
  - the “message”, ASCII characters only
Message format: multimedia extensions

- MIME: multimedia mail extension, RFC 2045, 2056
- additional lines in msg header declare MIME content type

From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data ......
..........................
......base64 encoded data
Mail access protocols

- SMTP: delivery/storage to receiver’s server
- Mail access protocol: retrieval from server
  - POP: Post Office Protocol [RFC 1939]
    - authorization (agent <--> server) and download
  - IMAP: Internet Mail Access Protocol [RFC 3501]
    - more features (more complex)
    - manipulation of stored msgs on server
  - HTTP: Gmail, Hotmail, Yahoo! Mail, etc.


POP3 protocol

authorization phase

• client commands:
  ‣ **user**: declare username
  ‣ **pass**: password

• server responses
  ‣ +OK
  ‣ -ERR

transaction phase, client:

• **list**: list message numbers

• **retr**: retrieve message by number

• **dele**: delete

• **quit**

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 2 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```
More about POP3

- Previous example uses “download and delete” mode.
- Bob cannot re-read e-mail if he changes client.
- “Download-and-keep”: copies of messages on different clients.
- POP3 is stateless across sessions.

IMAP

- Keep all messages in one place: the server.
- Allows user to organize messages in folders.
- IMAP keeps user state across sessions:
  - names of folders and mappings between message IDs and folder name.
Chapter 2: Application layer

- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 Electronic Mail
- 2.5 DNS
- 2.6 P2P Applications
DNS: Domain Name System

People: many identifiers:
  - SSN, name, passport #

Internet hosts, routers:
  - IP address (32 bit) - used for addressing datagrams
  - “name”, e.g., www.yahoo.com - used by humans

Q: map between IP addresses and name?

Domain Name System:
  - distributed database implemented in hierarchy of many name servers
  - application-layer protocol host, routers, name servers to communicate to resolve names (address/name translation)
    - note: core Internet function, implemented as application-layer protocol
    - complexity at network’s “edge”
DNS

DNS services

- Hostname to IP address translation
- Host aliasing
  - Canonical and alias names
- Mail server aliasing
- Load distribution
  - Replicated Web servers: set of IP addresses for one canonical name

Why not centralize DNS?

- single point of failure
- traffic volume
- distant centralized database
- maintenance

In summary, *it doesn’t scale*!
**Distributed, Hierarchical Database**

Client wants IP for www.amazon.com; 1st approx:

- Client queries a root server to find com DNS server
- Client queries com DNS server to get amazon.com DNS server
- Client queries amazon.com DNS server to get IP address for www.amazon.com
DNS: Root name servers

- contacted by local name server that can not resolve name
- root name server:
  - contacts authoritative name server if name mapping not known
  - gets mapping
  - returns mapping to local name server

13 root name servers worldwide:

- a Verisign, Dulles, VA
- b USC-ISI Marina del Rey, CA
- c Cogent, Herndon, VA (also Los Angeles)
- d U Maryland College Park, MD
- e NASA Mt View, CA
- f Internet Software C. Palo Alto, CA (and 17 other locations)
- g US DoD Vienna, VA
- h ARL Aberdeen, MD
- i Autonomica, Stockholm (plus 3 other locations)
- j Verisign, (11 locations)
- k RIPE London (also Amsterdam, Frankfurt)
- m WIDE Tokyo
- l ICANN Los Angeles, CA
TLD and Authoritative Servers

- **Top-level domain (TLD) servers:** responsible for com, org, net, edu, etc, and all top-level country domains uk, fr, ca, jp.
  - Network Solutions maintains servers for com TLD
  - Educause for edu TLD

- **Authoritative DNS servers:** organization’s DNS servers, providing authoritative hostname to IP mappings for organization’s servers (e.g., Web and mail).
  - Can be maintained by organization or service provider
Local Name Server

• Does not strictly belong to hierarchy

• Each ISP (residential ISP, company, university) has one.
  ‣ Also called “default name server”

• When a host makes a DNS query, query is sent to its local DNS server
  ‣ Acts as a proxy, forwards query into hierarchy.
Example

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

**recursive query:**
- puts burden of name resolution on contacted name server
- heavy load?

**iterated query:**
- contacted server replies with name of server to contact
- “I don’t know this name, but ask this server”
Recursive queries

**recursive query:**
- puts burden of name resolution on contacted name server
- heavy load?

**iterated query:**
- contacted server replies with name of server to contact
- “I don’t know this name, but ask this server”
DNS: caching and updating records

• once (any) name server learns mapping, it *caches* the mapping
  ‣ cache entries timeout (disappear) after some time
  ‣ TLD servers typically cached in local name servers
    • Thus root name servers not often visited

• update/notify mechanisms
  ‣ RFC 2136, 3007
DNS records

**DNS:** distributed db storing resource records (RR)

RR format: \((\text{name}, \text{value}, \text{type}, \text{ttl})\)

- **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=CNAME**
  - name is alias name for some “canonical” (the real) name
    - www.ibm.com is really severeast.backup2.ibm.com
  - value is canonical name

- **Type=MX**
  - value is name of mailserver associated with name
DNS protocol, messages

**DNS protocol**: query and reply messages, both with same message format

msg header
- **identification**: 16 bit # for query, reply to query uses same #
- **flags**:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>identification</td>
<td>16 bit #</td>
</tr>
<tr>
<td>flags</td>
<td>query or reply, recursion desired, recursion available, reply is authoritative</td>
</tr>
<tr>
<td>number of questions</td>
<td>number of answer RRIs</td>
</tr>
<tr>
<td>number of authority RRIs</td>
<td>number of additional RRIs</td>
</tr>
<tr>
<td>questions</td>
<td>(variable number of questions)</td>
</tr>
<tr>
<td>answers</td>
<td>(variable number of resource records)</td>
</tr>
<tr>
<td>authority</td>
<td>(variable number of resource records)</td>
</tr>
<tr>
<td>additional information</td>
<td>(variable number of resource records)</td>
</tr>
</tbody>
</table>
## DNS protocol, messages

<table>
<thead>
<tr>
<th>identification</th>
<th>flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of questions</td>
<td>number of answer RRs</td>
</tr>
<tr>
<td>number of authority RRs</td>
<td>number of additional RRs</td>
</tr>
</tbody>
</table>

### Name, type fields
- for a query

### RR in response to query
- records for authoritative servers

### Additional “helpful” info that may be used
- (variable number of resource records)
Viewing DNS Queries

- Text recommends `nslookup`
- I use `dig`
Inserting records into DNS

• Example: just created startup “Network Utopia”

• Register name networkuptopia.com at a registrar (e.g., Network Solutions)
  ‣ Need to provide registrar with names and IP addresses of your authoritative name server (primary and secondary)
  ‣ Registrar inserts two RRs into the com TLD server:
    
    (networkutopia.com, dns1.networkutopia.com, NS)
    (dns1.networkutopia.com, 212.212.212.1, A)

• Put in authoritative server Type A record for www.networkuptopia.com and Type MX record for networkutopia.com

• How do people get the IP address of your Web site?
DNS Security Issues

• Given that so many different servers can respond to your request, *how do you know that what you get back is correct?*
  
  ‣ Are you sure that you spoke to the resolver you think you spoke to?

• What happens if you manage to give a resolver false look-up information?
DNS Cache Poisoning

1. Victim nameserver
   - QID=599
   - IP for www.neighborhoodbank.com
   - referral to neighborhoodbank.com

2. QID=599
   - referral to neighborhoodbank.com
   - IP for www.neighborhoodbank.com

3. QID=600
   - IP for www.neighborhoodbank.com
   - ns.neighborhoodbank.com

4. QID=600
   - IP: 192.168.1.100
   - www.neighborhoodbank.com

5. Transaction

Client

Root Servers

ns.neighborhoodbank.com

www.neighborhoodbank.com

evil.net

fake!

www.neighborhoodbank.com

evil client
DNS Attacks - Real?

- Golden Shield Project
- Kaminsky Attack
- Others?
  - Why is it difficult to know?
Same Bat Time...

- Peer-to-Peer architectures/applications
  - Read Section 2.6

- Socket Programming
  - The book uses Java, we are going to use C
  - If you haven’t already done so, look at the Pocket Sockets Guide.