Application Layer: DNS & Mail

Srinidhi Varadarajan
DNS: Domain Name System

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

Internet hosts, routers:
- IP address (32 bit) - used for addressing datagrams
- Need human interface to numbers
- “name”, e.g., gaia.cs.umass.edu - 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 communicate to resolve names (address/name translation)
  - note: core Internet function implemented as application-layer protocol
  - complexity at network’s “edge”
DNS name servers

Why not centralize DNS?

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

doesn’t scale!

- no server has all name-to-IP address mappings

local name servers:
- each ISP, company has local (default) name server
- host DNS query first goes to local name server

authoritative name server:
- for a host: stores that host’s IP address, name
- can perform name/address translation for that host’s name

doesn’t scale!
DNS: Root name servers

- contacted by local name server that cannot resolve name
- root name server:
  - contacts authoritative name server if name mapping not known
  - gets mapping
  - returns mapping to local name server
- ~ dozen root name servers worldwide
Simple DNS example

**host** surf.eurecom.fr wants IP address of gaia.cs.umass.edu

1. **Contacts its local DNS server**, dns.eurecom.fr

2. **contacts** root name server, if necessary

3. **root name server contacts authoritative name server**, dns.umass.edu, if necessary

```
1 2 5
surf.eurecom.fr

3 4
root name server
dns.eurecom.fr

1 6
dns.umass.edu
```

```
1
local name server
dns.eurecom.fr

2

3

4

5

6
requesting host
surf.eurecom.fr

gaia.cs.umass.edu
```
DNS example

Root name server:
- may not know authoritative name server
- may know intermediate name server: who to contact to find authoritative name server

```
dns.eurecom.fr
surf.eurecom.fr
```

```
dns.umass.edu
gaiac.s.umass.edu
```

```
dns.cs.umass.edu
dns.eurecom.fr
```

```
root name server
telem.eurecom.fr
telem.cs.umass.edu
```

```
local name server
telem.eurecom.fr
telem.cs.umass.edu
```

```
intermediate name server
telem.eurecom.fr
telem.cs.umass.edu
```

```
requesting host
telem.eurecom.fr
telem.cs.umass.edu
```

```
authoritative name server
telem.eurecom.fr
telem.cs.umass.edu
```
DNS: iterated 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* mapping
  - cache entries timeout (disappear) after some time

- Dynamic DNS updates:
  - update/notify mechanisms under design by IETF
  - RFC 2136
DNS 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 IP address 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 protocol, messages

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

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

<table>
<thead>
<tr>
<th>identification</th>
<th>flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of questions</td>
<td>number of answer RR</td>
</tr>
<tr>
<td>number of authority RR</td>
<td>number of additional RR</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 RR's</td>
</tr>
<tr>
<td>Number of authority RR's</td>
<td>Number of additional RR's</td>
</tr>
</tbody>
</table>

**Name, type fields for a query**

**RRs in response to query**

**Records for authoritative servers**

**Additional "helpful" info that may be used**
Load Balancing

- DNS can be used to associate multiple IP addresses with a single canonical name
  - Why?

- During address resolution, the server returns all IP addresses associated with the name.

- Every time it returns this list of IP addresses, it rotates the list. The client usually uses the first IP address in the list.
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, Netscape Messenger
- outgoing, incoming messages stored on server
Electronic Mail: mail servers

Mail Servers

- **mailbox** contains incoming messages (yet to be read) 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 821]

- uses tcp to reliably transfer email msg 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
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 vtopus.cs.vt.edu 25
- see 220 reply from server
- enter HELO, MAIL FROM, RCPT TO, DATA, QUIT commands
- lets you send email without using email client (reader)
smtp: final words

- smtp uses persistent connections
- smtp requires that message (header & body) be in 7-bit ascii
- certain character strings are not permitted in message (e.g., CRLF . CRLF). Thus message has to be encoded (usually into either base-64 or quoted printable)
- 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 is encapsulated in its own response message
- smtp: multiple objects message sent in a multipart message
smtp: protocol for exchanging email msgs
RFC 822: 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
  - MIME version
  - method used to encode data
  - multimedia data type, subtype, parameter declaration
  - encoded data
  
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
MIME types
Content-Type: type/subtype; parameters

**Text**
- **example subtypes:** plain, html

**Image**
- **example subtypes:** jpeg, gif

**Audio**
- **example subtypes:** basic (8-bit mu-law encoded), 32kadpcm (32 kbps coding)

**Video**
- **example subtypes:** mpeg, quicktime

**Application**
- other data that must be processed by reader before “viewable”
- **example subtypes:** msword, octet-stream
Dear Bob,
Please find a picture of a crepe.

--98766789
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data ..... 
..........................
......base64 encoded data
--98766789--
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 1730]
    - more features (more complex)
    - manipulation of stored msgs on server
  - HTTP: 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 alice
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 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
Reading

Recommended

- DNSNet: DNS Resources
  - [http://www.dns.net](http://www.dns.net)
- DNS RFCs:
  - DNS Resource Records: RFC 1034, 1035
  - Dynamic DNS: RFC 2136
- SMTP:
  - Protocol: RFC 821, 822 (Text Message, Header format)
  - MIME Extensions: RFC 2045, 2046
- POP3: RFC 1939
- IMAP: RFC 1999