IT441: Network Servers & Infrastructure

CLASS 6 : 11 Oct 2005
13:30 - 16:15

Except where otherwise noted, these slides are © copyright Charles Snow, 2005. Permission to copy these slides is granted so long as each slide contains the copyright notice © C. M. Snow 2005 appears on each page and the copies are made freely available, i.e., not sold. Students are advised that they should not use University consumables (ink, toner, paper) to produce printed versions of these slides. Rights for all other use of this material are reserved by the author. Permission for any other use must be obtained in writing from the author.

This Time

- DNS, DNS records, messages
- email service: SMTP, POP, IMAP
  - transfer encodings

Resolving Names

- some nameserver is the authoritative nameserver for the domain in question
- provides authoritative answer: either address or indication that the sought name doesn’t exist
- UNIX provides command-line resolvers
  - e.g., `dig osf1.gmu.edu`

```
192.168.1.2

struct hostent

struct hostent *gethostbyname();
struct hostent *gethostbyaddr();
```

Last Time

- TCP
- boot-time issues
- IP address: how allocated to host
- DNS
Sample DNS Record

- db in nameserver is a text file containing resource records
  - syntax:
    `<domain_name><ttl><class><type><value>`
  - example entry: from Tanenbaum, fig. 7-8
    ```
    fllts.cs.vu.nl 86400 IN A 192.31.231.165
    fllts.cs.vu.nl 86400 IN MX 1 fllts.cs.vu.nl
    www.cs.vu.nl 86400 IN A 192.31.231.165
    ```

Sample Resource Record

Host Info: CPU OS

```
+-------------------------+-----------------------------+
| Flights.cs.vu.nl 86400  |
| Flights.cs.vu.nl 86400  |
| Flights.cs.vu.nl 86400  |
| Flights.cs.vu.nl 86400  |
+-------------------------+-----------------------------+
```

DNS Messages

Identification (16) value assigned by client, returned by server, to enable match requests with answers

DNS Messages

```
| Operation Code (1) 0 = normal query |
| TC[1]: 0 if not, 1 if truncated answer (max > 512 bytes) |
| QR[1]: 0 if query, 1 if response |
| AA[1]: 0 if not, 1 if authoritative answer |
| RZ[1]: 0 if not, 1 if recursion available |
| RC[1]: 0 if not, 1 if recursion desired |
```

DNS Messages

- sent via UDP
A Familiar Content Service

- electronic mail service

MTA to MTA

- mail transfer agent (MTA) moves messages to their destination
  - from a queue to some (recipient) MTA
  - to a user agent from some (sender) MTA
- MTAs speak simple mail transfer protocol (SMTP) to each other
  - originally in RFC821
  - currently RFC2821
- SMTP is a TCP service
  - runs on port 25

MTA’s view: sending mail

```
220 something.org Sendmail 8.12 ready
HELO mynode.subdomain.com
250 OK
MAIL From:<scarter@mynode.subdomain.com>
250 <scarter@mynode.subdomain.com>: OK
Rcpt To: <jone1@modex.spc.mil>
250 <jone1@modex.spc.mil> OK
DATA
354 Enter mail, end with \r\n\n mail message inserted here ...
250 OK mail accepted
QUIT
221 something.org closing connection
```

Server Response Codes

- first defined for SMTP in RFC821
- now widely used in other services
- based on 3-digit xyz values:

<table>
<thead>
<tr>
<th>xyz</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1yz</td>
<td>positive preliminary reply</td>
</tr>
<tr>
<td>2yz</td>
<td>positive completion reply</td>
</tr>
<tr>
<td>3yz</td>
<td>positive intermediate reply</td>
</tr>
<tr>
<td>4yz</td>
<td>transient negative completion reply</td>
</tr>
<tr>
<td>5yz</td>
<td>permanent completion reply</td>
</tr>
</tbody>
</table>

Server Commands

1. HELO, EHLO:
   - identify the SMTP client to the SMTP server
   - client sends its fully-qualified domain name
   - HELO used by older clients
   - all servers support HELO
   - newer ones support extended services, hence EHLO

- and z is context-specific code providing more detail
**Server Commands**

1. HELO, EHLO:
2. MAIL
   - start mail transaction where mail is sent to server
3. RCPT
   - identifies a single recipient of the mail message
   - use multiple instances of RCPT for multiple recipients
4. DATA
   - server replies with 354 line then copies everything sent to it by client up to the '.' line
   - data should consist only of 7-bit ASCII characters
   - and avoid ctrl chars other than SP, HT, CR, and LF
5. RSET
   - aborts current mail operation in progress
   - any data received as part of this operation is deleted from server
   - has no effect if appears immediately after EHLO
6. VRFY
   - asks receiver to confirm that argument identifies a user or mailbox
7. EXPN
   - asks receiver to confirm that argument identifies a mailing list and, if so, return list membership
Server Commands

1. HELO, EHLO:
2. MAIL
3. RCPT
4. DATA
5. RSET
6. VRFY
7. EXPN
8. HELP
   • causes server to send helpful information to client
9. NOOP
   • has no effect on server other than to cause it to respond with OK
10. QUIT
    • server must send OK and close connection

Mail Headers

• used by user agents
• simple syntax: attribute=value
  • attributes may contain ASCII chars with codes 0x21 to 0xFE except 0x3A
  • values may contain any ASCII chars except CR and LF
  • e.g., “From: user@somplace.com”
  • some start with X- or user-defined
  • e.g., X-Charset, X-Mailer

email messages

• simple structure —
  • envelope: used by MTAs for delivery
    • consists of the 2 SMTP commands MAIL and RCPT
  • header: contains non-message information like addressee, sender, date, etc.
    • has a standard syntax
  • body: contains actual content of message
    • can be ‘anything’
Mail Headers

- **To:** address-list
  - contains the address(es) of the primary recipient(s) of the message
- **CC:** address-list
  - contains the addresses of others who are to receive the message
- **BCC:** address-list
  - contains addresses of recipients of the message whose addresses are not to be revealed to other recipients of the message

Mail Headers

- **Message-ID:** <ident>
  - not intended to be human-readable
  - e.g., <200403260218.129255085620metlab.gmu.edu>
  - provides unique identifier referring to particular version of particular message: uniqueness guaranteed by host generating it
- **In-Reply-To:** <ident>
  - lists message-id of current msg (to which it is a reply)
- **References:** <ident>
  - provides id’s appearing in current msg’s references field

Mail Headers

- Informational fields, human readable:
  - **Subject:**
  - **Comments:**
  - **Keywords:**

Mail Headers

- **user defined fields:**
  - **X-Charset:** identify character set used to represent message
  - **X-Mailer:** user agent used to send message
  - **X-Sender:** duplicate of From:

email messages

- originally were only text
  - using 7-bit ASCII
  - protocols designed around text-only original version
  - still support 7-bit ASCII
  - some may support 8-bit character data (see RFC1652)
  - described in RFC822 (still often cited)
  - current version is RFC2822
  - lines must be ≤ 1000 bytes long
  - but now want more than just text...

Different Content Types

- need mechanism to support:
  - different kinds of content to appear in a msg
  - multiple different kinds of content within one msg
**Different Content Types**

- need mechanism to support:
  - different kinds of content to appear in a msg
  - multiple different kinds of content within one msg
- use Multipurpose Internet Mail Extensions (MIME) RFC1521 (now: RFCs: 2045-2049)
  - provides way to encode binary data using only printable ASCII characters
  - inflates size of data

**MIME**

- used in email, adds lines to headers:
- for multi-part messages:
  - Content-Type: multipart/alternative
    - start delimiter
      - char-set = encoding ? encoded-text end delimiter
    - start delimiter
      - Content-Type: application/x-mvar
    - Content-Disposition: inline; filename= "image.png"
  - first part of message (text)
  - Content-Type: image/png
    - name= "test.png"
    - Content-Disposition: inline; filename= "image.png"
- second part of message (png image)
  - Content-Type: multipart/alternative

**Non-ASCII in Header**

- headers may also contain non-ASCII chars
- introduced using syntax:

```
charset ? encoding ? encoded-text
```

- start delimiter
- identify character set: e.g., ISO-8859-1
- encoding scheme:
  - Q for quoted printable
  - B for base-64

**Transfer Encodings**

ways to represent non-ASCII data as ASCII:

1. quoted-printable:
   - ASCII chars with code 0x21 to 0x7E (except 0x3D) appears "as is"
   - all line breaks expressly appear as \r\n
2. quoted-printable:
   - space appears as ",", unless at end of line "=20"
   - lines longer than 76 bytes have inserted line breaks
   - everything else represented as sequence =xx
     - e.g., 'é' appears as "=E9"

**Quoted Printable Example**

```
from: <7150-8959-1@Patrick.Petlius.tst=v@netlab.kth.se> <af@netlab.kth.se>
translates into:
```

<table>
<thead>
<tr>
<th>Patrick.Petlius</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:Patrick.Petlius.tst=v@netlab.kth.se">Patrick.Petlius.tst=v@netlab.kth.se</a></td>
</tr>
</tbody>
</table>

example taken from RFC1522
Transfer Encodings

ways to represent non-ASCII data as ASCII:

2. base64 encoding
   - replace every 6 bits of binary data with a single ASCII character from special charset table
   - table contains:
     * A..Z
     * a..z
     * 0..9
     * + / =
   - send string so generated

---

Base64 Example

From: keith@cs.utk.edu
To: andre@prandwms1.utk.edu
CC: keal@lkut.edu
Subject: B1 -- a simple security technique

which a user agent renders as:

From: Keith Moore <moore@cs.utk.edu>
To: kael Simonsen <kael@lkut.edu>
CC: Andre Picard <PRANDWMS1.utk.edu>
Subject: If you can read this you understand the example.

---

Address Services

- what's in a name?
  * user@some.node.subdomain.domain
  * where does "user" get its email?
  * what about
    * user@someother.node.subdomain.domain

---

Address Services

- many organizations now use email gateway
  * node dedicated to receiving, forwarding, sending email
  * lets users have email addresses like
    * username@doma.in
  * gateway receives all inbound email
    * either forwards to particular node, or,
    * keeps local and user's user-agent retrieves the email therefrom

---

Address Services

- gateway receives all inbound email
- gateway needs list of entries associating
  * username@domain with
  * username@node.subdomain.com
- must this association be 1:1?

---

Address Services

- must this association be 1:1?
- no: can have one name: many names
  * so email gateway can match a listname with a 1 mailboxes and must generate individual copies of the message to each
  * such a gateway is hence known as a mail-explorer
  * see Comer fig. 32.5 for example
  * users must have their email address added to list in order to receive mailings
Address Services

- adding name to list can be automated procedure:
  - usually simply send email message to list
  - follow syntax rules specified by list, e.g., send subject line as "add me"
  - no body to msg (not being read by human)
- list manager on email gateway watches for these special msgs
- performs add when discovers one to do
- analogously for removal

Getting Your Mail

- today, email usually is delivered to some server
  whence you must retrieve it
  - rather than being delivered to 'your' local machine
  - i.e., your email user agent may be local, but the actual mailbox isn't
- how to retrieve?

Post Office Protocol (POP)

- currently POP-3 as described in RFC1939
- POP is client-server pair
  - server runs on system where email physically is collected
  - client runs on system where user agent runs
- so mail server runs
  - software to collect email directed to it, including exploding function
  - software to support access to collected email via POP

Post Office Protocol (POP)

- so mail client runs
  - software to send email (via SMTP)
  - software to receive email (via POP)
- POP client interacts with server:
  - copy accumulated email from mail server to user client
  - delete copied messages from email server, or leave intact
- service runs on port 110

Post Office Protocol (POP)

- example client server interaction in POP3:
  connect to port 110 on server...
  -OK POP3 server ready
  USER Fred
  -OK PASS deriv
  -OK user logged on
  LIST
  -OK 2 messages (120 octets)
  1 120
  2 200

Post Office Protocol (POP)

- example client server interaction in POP3, cont'd:
  RETR 1
  -OK 320 octets
  <the POP3 server sends the entire message here>
  .
  DELE 1
  -OK message 1 deleted
  DELE 3
  -ERR message 1 already deleted
  QUIT
  -OK POP3 server signing off
POP3

- available commands in the POP3 protocol include:
  - APOP
  - PASS
  - STAT
  - DELE
  - QUIT
  - TOP
  - LIST
  - RETR
  - UIDL
  - NOOP
  - RSET
  - USE

Another Mail Fetching Protocol

- instead of POP, a user agent may use Internet Mail Access Protocol (IMAP) RFC3501
- particular advantage: allows management of messages on a server
  - e.g., can organize messages into folders
  - also provides ability to retrieve only parts of a message
    - e.g., subject lines
    - e.g., specific part of MIME multipart message

Servers

- provide a service
- provide content
  - static
  - dynamic
    - generated on-demand
    - "streamed"
    - on-going flow of data
    - often real-time sensitive
    - may be pre-computed (e.g., mp3)
    - real-time, not pre-computable (e.g., telephony)

Internet Telephony

- real-time
- full duplex
- cannot be pre-computed, pre-compressed
- provide usual telephony services
  - call forwarding
  - call waiting
  - caller ID
  - PBX services
- interoperability with existing PSTN

Telephones

- what is a telephone?
  - any combination of hardware and software that performs these functions can be a telephone

  audio capture device (microphone)
  activation control switch
  audio reproduction device (speaker)
  activity alert device (ringer)
  address selection device (keypad)