CS 352
Simple Mail Transfer Protocol

CS 352, Lecture 5.1
http://www.cs.rutgers.edu/~sn624/352

Srinivas Narayana
We’re all familiar with email. How does it work?
Electronic Mail

Three major components:

1. **User agents**
   - a.k.a. “mail reader”
   - e.g., Applemail, Outlook
   - Web-based user agents (ex: gmail)
Electronic Mail: Mail servers

2. Mail Servers
   - Mailbox contains incoming messages for user
   - Message queue of outgoing (to be sent) mail messages
   - Sender mail server makes connection to Receiver mail server
     - IP address, port 25

3. SMTP protocol
   - Used to send messages
   - Client: sending user agent or sending mail server
   - server: receiving mail server
Scenario: Alice sends message to Bob

1) Alice (alice@rutgers.edu) uses UA to compose message to bob@nyu.edu
2) Alice’s UA sends message to her mail server; message placed in outgoing 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 incoming mailbox
6) Sometime later, Bob invokes his user agent to read message
Observations on these exchanges

• Mail servers are useful “always on” endpoints
  • Receiving the email on behalf of Bob, should Bob’s machine be turned off
  • Retrying the delivery of the email to Bob on behalf of Alice, should Bob’s mail server be unavailable in the first attempt

• The same machine can act as client and server based on context
  • Rutgers’s mail server is the server when Alice sends the mail
  • It is the client when it sends mail to Bob’s mail server

• SMTP is push-heavy: info is pushed from client to server
  • Contrast to HTTP or DNS where info is pulled from the server
Sample SMTP interaction

• A small demo
Sample SMTP interaction

220 hill.com SMTP service ready

HELO town.com

250 hill.com Hello town.com, pleased to meet you

MAIL FROM: <jack@town.com>

250 <jack@town.com>… Sender ok

RCPT TO: <jill@hill.com>

250 <jill@hill.com>… Recipient ok

DATA

354 Enter mail, end with “.” on a line by itself

Jill, I’m not feeling up to hiking today. Will you please fetch me a pail of water?
.

250 message accepted

QUIT

221 hill.com closing connection
# MAIL command response codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>211</td>
<td>System status or help reply</td>
</tr>
<tr>
<td>214</td>
<td>Help message</td>
</tr>
<tr>
<td>220</td>
<td>Service ready</td>
</tr>
<tr>
<td>221</td>
<td>Service closing transmission channel</td>
</tr>
<tr>
<td>250</td>
<td>Request command completed</td>
</tr>
<tr>
<td>251</td>
<td>User not local; the message will be forwarded</td>
</tr>
<tr>
<td>354</td>
<td>Start mail input</td>
</tr>
<tr>
<td>421</td>
<td>Service not available</td>
</tr>
<tr>
<td>450</td>
<td>Mailbox not available</td>
</tr>
<tr>
<td>451</td>
<td>Command aborted: local error</td>
</tr>
<tr>
<td>452</td>
<td>Command aborted: insufficient storage</td>
</tr>
<tr>
<td>500</td>
<td>Syntax error; unrecognized command</td>
</tr>
<tr>
<td>501</td>
<td>Syntax error in parameters or arguments</td>
</tr>
<tr>
<td>502</td>
<td>Command not implemented</td>
</tr>
<tr>
<td>503</td>
<td>Bad sequence of commands</td>
</tr>
<tr>
<td>504</td>
<td>Command temporarily not implemented</td>
</tr>
<tr>
<td>550</td>
<td>Command is not executed; mailbox unavailable</td>
</tr>
<tr>
<td>551</td>
<td>User not local</td>
</tr>
<tr>
<td>552</td>
<td>Requested action aborted; exceeded storage location</td>
</tr>
<tr>
<td>553</td>
<td>Requested action not taken; mailbox name not allowed</td>
</tr>
<tr>
<td>554</td>
<td>Transaction failed</td>
</tr>
</tbody>
</table>

- **220**: Service ready
- **250**: Request command complete
- **354**: Start mail input
- **421**: Service not available
Mail message (stored on server) format

SMTP: protocol for exchanging email msgs
RFC 822: standard for text message format:

- header lines, e.g.,
  - To:
  - From:
  - Subject: different from SMTP commands!
    (these would still be under “DATA”)

- 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
```

- MIME version
- method used to encode data
- multimedia data type, subtype, parameter declaration
- encoded data
CS 352

Mail: Access Protocols

CS 352, Lecture 5.2

http://www.cs.rutgers.edu/~sn624/352

Srinivas Narayana
Mail access protocols

- **SMTP**: delivery/storage to receiver’s server
- Mail access protocol: retrieval from server
  - **POP**: Post Office Protocol [RFC 1939]
    - Client connects to POP3 server on TCP port 110
  - **IMAP**: Internet Mail Access Protocol [RFC 1730]
    - Client connects to TCP port 143
  - **HTTP**: gmail, outlook, etc.
POP vs IMAP

• POP3
• Stateless server
• UA-heavy processing
• UA retrieves email from server, then typically deleted from server
• Latest changes are at the UA
• Simple protocol (list, retr, del within a POP session)

• IMAP4
• Stateful server
• UA and server processing
• Server sees folders, etc. which are visible to UAs
• Changes visible at the server
• Complex protocol
What about web-based email?

• Connect to mail servers via web browser
  • Ex: gmail, outlook, etc.

• Browsers speak HTTP
• Email servers speak SMTP
• Need a bridge to retrieve email using HTTP
Web based email

HTTP server → HTTP

SMTP Client

HTTP

Internet

HTTP server → HTTP

SMTP server
Comparing SMTP 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

• HTTP: can put non-ASCII data directly in response
• SMTP: need ASCII-based encoding
More themes from app-layer protocols

- **Separation of concerns.** Examples:
  - Content rendering for users (browser, UA) separate from protocol operations (mail server)
  - Reliable mail sending and receiving: mail UA doesn’t need to be “always on” to send or receive email reliably

- **In-band vs. out-of-band control:**
  - In-band: headers determine the actions of all the parties of the protocol
  - There are protocols with out-of-band control, e.g., FTP

- **Keep it simple until you really need complexity**
  - ASCII-based design; stateless servers. Then introduce:
    - Cookies for HTTP state
    - IMAP for email organization
    - Security extensions (e.g., TLS)
    - Different methods to set up and use underlying connections (e.g., persistence)