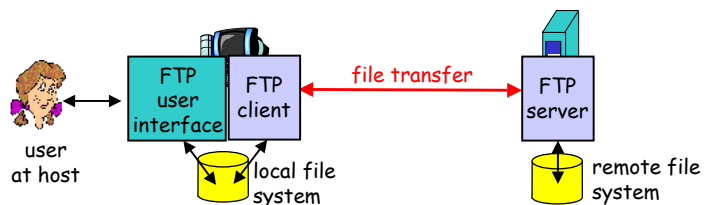


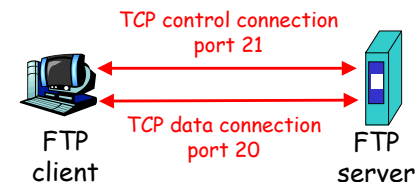
ftp: the file transfer protocol



- ❑ transfer file to/from remote host
- ❑ client/server model
 - *client*: side that initiates transfer (either to/from remote)
 - *server*: remote host
- ❑ ftp: RFC 959
- ❑ ftp server: port 21

ftp: separate control, data connections

- ❑ ftp client contacts ftp server at port 21, specifying TCP as transport protocol
- ❑ two parallel TCP connections opened:
 - **control**: exchange commands, responses between client, server. "out of band control"
 - **data**: file data to/from server
- ❑ ftp server maintains "state": current directory, earlier authentication



ftp commands, responses

Sample commands:

- ❑ sent as ASCII text over control channel
- ❑ `USER username`
- ❑ `PASS password`
- ❑ `LIST` return list of file in current directory
- ❑ `RETR filename` retrieves (gets) file
- ❑ `STOR filename` stores (puts) file onto remote host

Sample return codes

- ❑ status code and phrase (as in http)
- ❑ `331 Username OK, password required`
- ❑ `125 data connection already open; transfer starting`
- ❑ `425 Can't open data connection`
- ❑ `452 Error writing file`

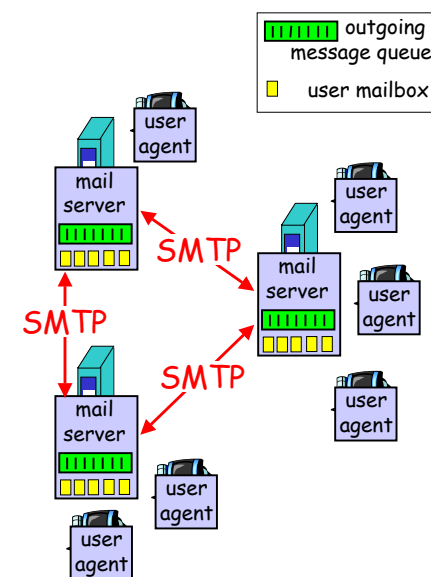
Electronic Mail

Three major components:

- ❑ user agents
- ❑ mail servers
- ❑ simple mail transfer protocol: smtp

User Agent

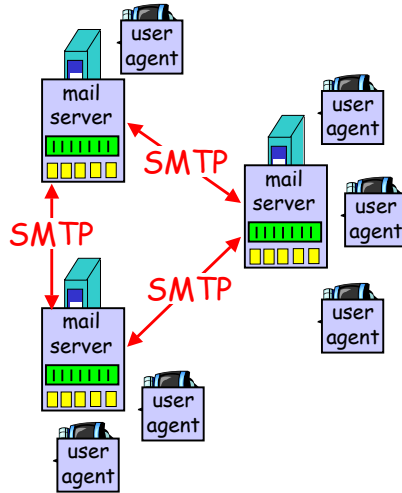
- ❑ a.k.a. "mail reader"
- ❑ composing, editing, reading mail messages
- ❑ e.g., Eudora, Outlook, elm, Netscape Messenger
- ❑ outgoing, incoming messages stored on server



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



2: Application Layer 31

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

2: Application Layer 32

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
```

2: Application Layer 33

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)

2: Application Layer 34

smtp: final words

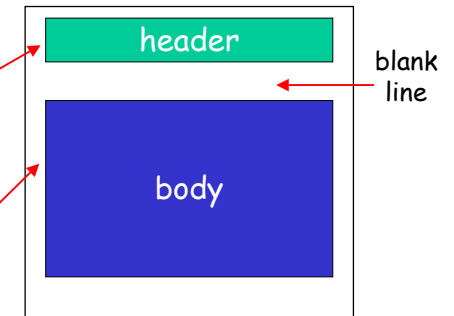
- smtp uses persistent connections
 - smtp requires that message (header & body) be in 7-bit ascii
 - certain character strings are not permitted in message (e.g., CRLF.CRLF). Thus message has to be encoded (usually into either base-64 or quoted printable)
 - smtp server uses CRLF.CRLF to determine end of message
- Comparison with http**
- http: pull
 - email: push
 - both have ASCII command/response interaction, status codes
 - http: each object is encapsulated in its own response message
 - smtp: multiple objects of message sent in a multipart message

Mail message format

smtp: protocol for exchanging email msgs

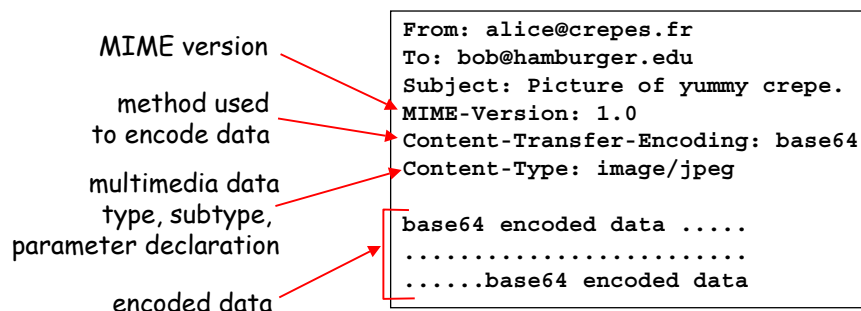
RFC 822: standard for text message format:

- header lines, e.g.,
 - To:
 - From:
 - Subject:*different from smtp commands!*
- 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 types

Content-Type: type/subtype; parameters

Text

- example subtypes: plain, html

Video

- example subtypes: mpeg, quicktime

Image

- example subtypes: jpeg, gif

Application

- other data that must be processed by reader before "viewable"
- example subtypes: msword, octet-stream

Audio

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

Multipart Type

From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=98766789

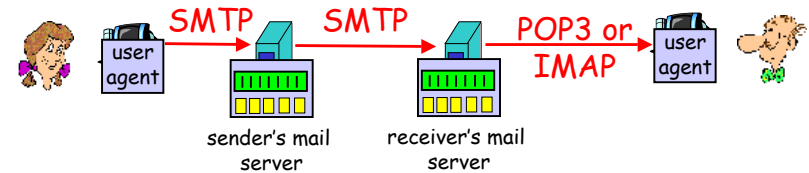
--98766789
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain

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
```