

# Computer Networks

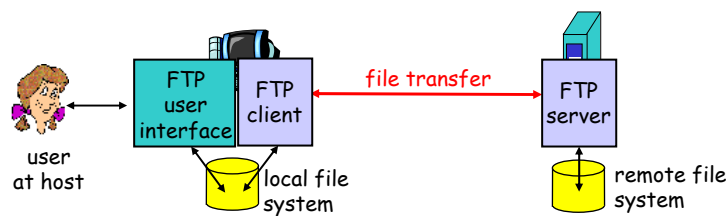
## FTP and eMail

Based on Computer Networking, 4<sup>th</sup> Edition by Kurose and Ross

Stan Kurkovsky

### FTP: the file transfer protocol

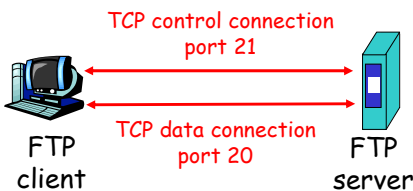
- transfer file to/from remote host
- client/server model
  - *client*: side that initiates transfer (either to/from remote)
  - *server*: remote host
- ftp: RFC 959
- ftp server: port 21



Stan Kurkovsky

## FTP: separate control, data connections

- FTP client contacts FTP server at port 21, specifying TCP as transport protocol
- Client obtains authorization over control connection
- Client browses remote directory by sending commands over control connection
- When server receives a command for a file transfer, the server opens a TCP data connection to client
- After transferring one file, server closes connection
- Server opens a second TCP data connection to transfer another file
- Control connection: **"out-of-band"**
  - As opposed to in-band control in HTTP
- FTP server maintains "state": current directory, earlier authentication



Stan Kurkovsky

## 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**

Stan Kurkovsky

## Electronic Mail

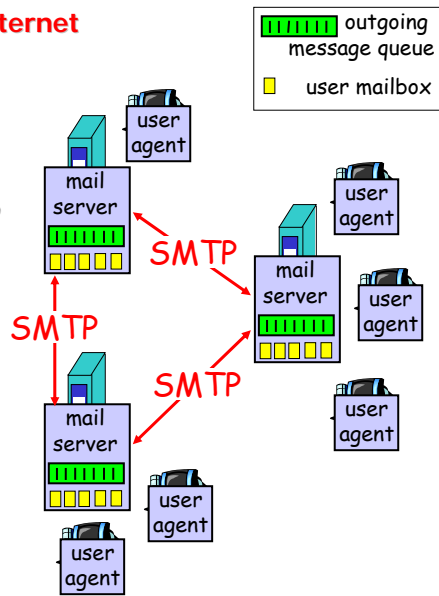
- **The original killer app of the Internet**

### Three major components:

- user agents
- mail servers
- simple mail transfer protocol: SMTP (another application-level protocol)

### User Agent

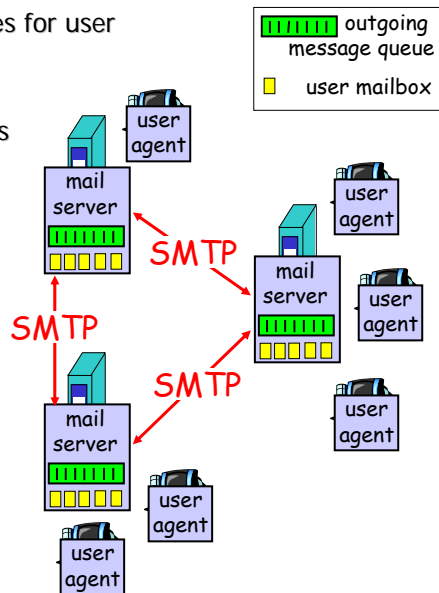
- a.k.a. "mail reader"
- composing, editing, reading mail messages
- e.g., Eudora, Outlook, elm, Thunderbird
- outgoing, incoming messages stored on server



Stan Kurkovsky

## Electronic Mail: 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



Stan Kurkovsky

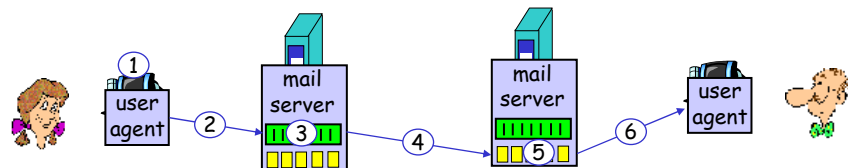
## Electronic Mail: SMTP [RFC 2821]

- 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

Stan Kurkovsky

## 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



Stan Kurkovsky

### 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

Stan Kurkovsky

### Great exercise

- `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)

Stan Kurkovsky

## SMTP and HTTP

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

### Comparison with HTTP:

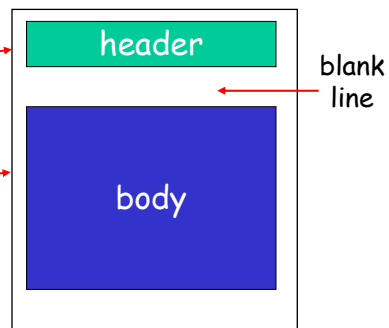
- HTTP: pull
- SMTP: push
- both have ASCII command/response interaction, status codes
- HTTP: each object encapsulated in its own response msg
- SMTP: multiple objects sent in multipart msg

Stan Kurkovsky

## Mail message format

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

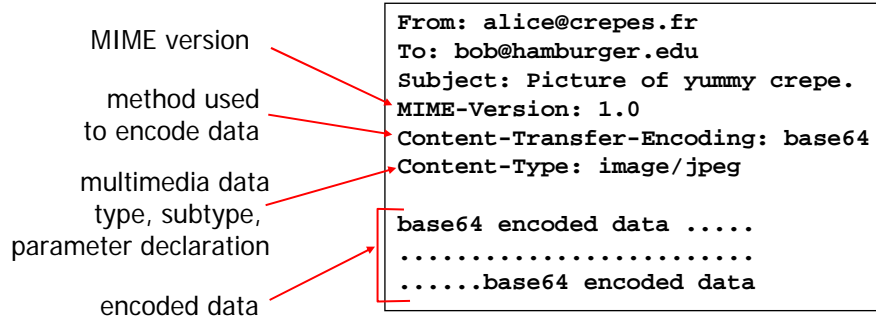
- header lines, e.g.,
  - To:
  - From:
  - Subject:*different from SMTP commands!*
- body
  - the "message", ASCII characters only



Stan Kurkovsky

## Message format: multimedia extensions

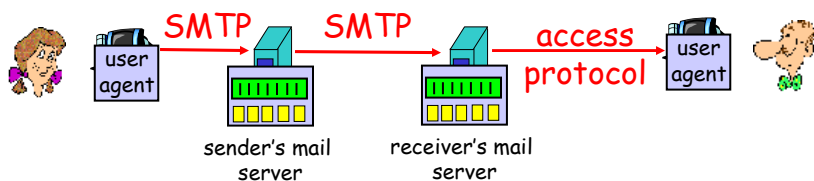
- **MIME: Multipurpose Internet Mail Extensions**
- MIME: multimedia mail extension, RFC 2045, 2056
- additional lines in message header declare MIME content type



Stan Kurkovsky

## 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: GMail, Hotmail , Yahoo! Mail, etc.



Stan Kurkovsky

## 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 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

Stan Kurkovsky

## Great exercise

**telnet** <your ISP's POP3 server> 110

e.g. telnet pop.sbcglobal.net 110

S: +OK hello from popgate(2.34.1)

C: user <your user name>

S: +OK password required.

C: pass <your password>

S: +OK maildrop ready, 3 messages (1144 octets)

C: list

S: +OK 3 messages (1144 octets)

S: 1 275

S: 2 138

S: 3 731

S: .

C: retr 1

S: <message 1>

S: .

C: quit

S: +OK server signing off.

Stan Kurkovsky

## POP3 and IMAP

- 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