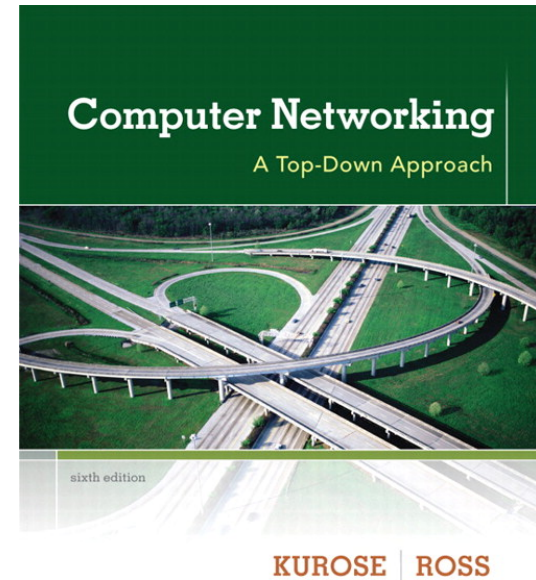


CSEE 4119 Computer Networks

Chapter 2 Application (3/6)



Chapter 2: Application layer

- ❖ 2.1 Principles of network applications
- ❖ 2.2 Web and HTTP
- ❖ 2.3 FTP
- ❖ 2.4 Electronic Mail
 - SMTP, POP3, IMAP
- ❖ 2.5 DNS

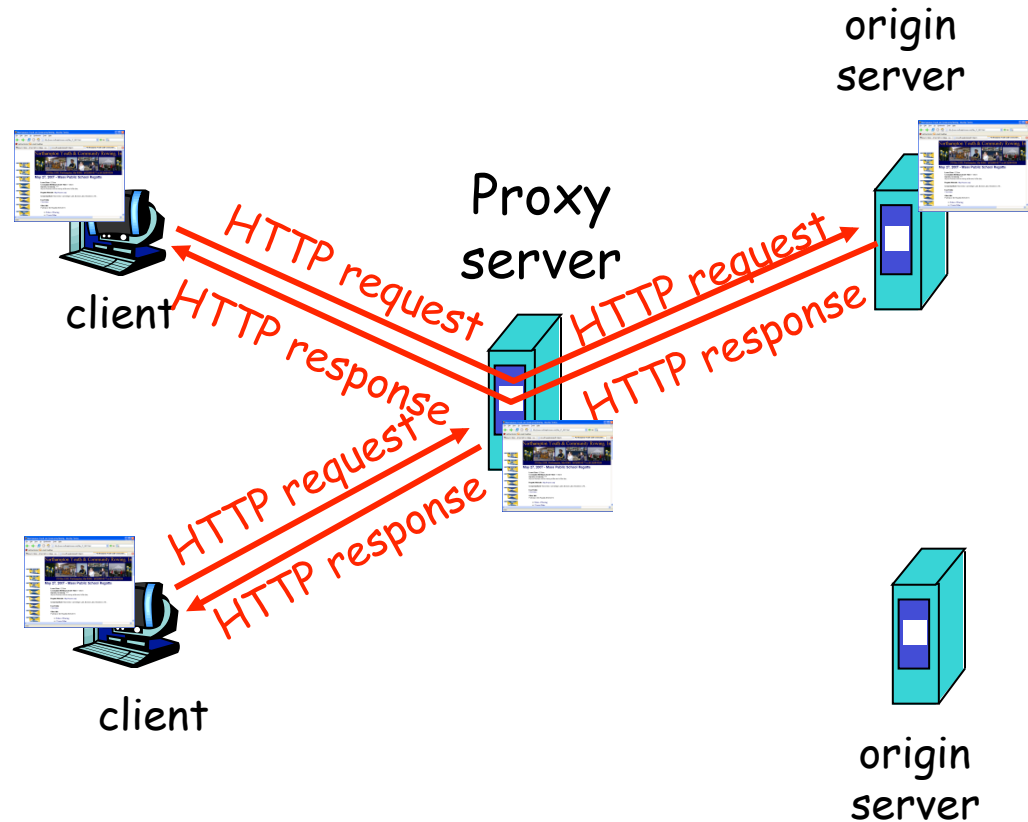
- ❖ Bonus:
a detour on CDN

- ❖ 2.6 P2P applications
- ❖ 2.7 Socket programming with TCP
- ❖ 2.8 Socket programming with UDP

Web caches (proxy server)

Goal: satisfy client request without involving origin server

- ❖ user sets browser: Web accesses via cache
- ❖ browser sends all HTTP requests to cache
 - object in cache: cache returns object
 - else cache requests object from origin server, then returns object to client

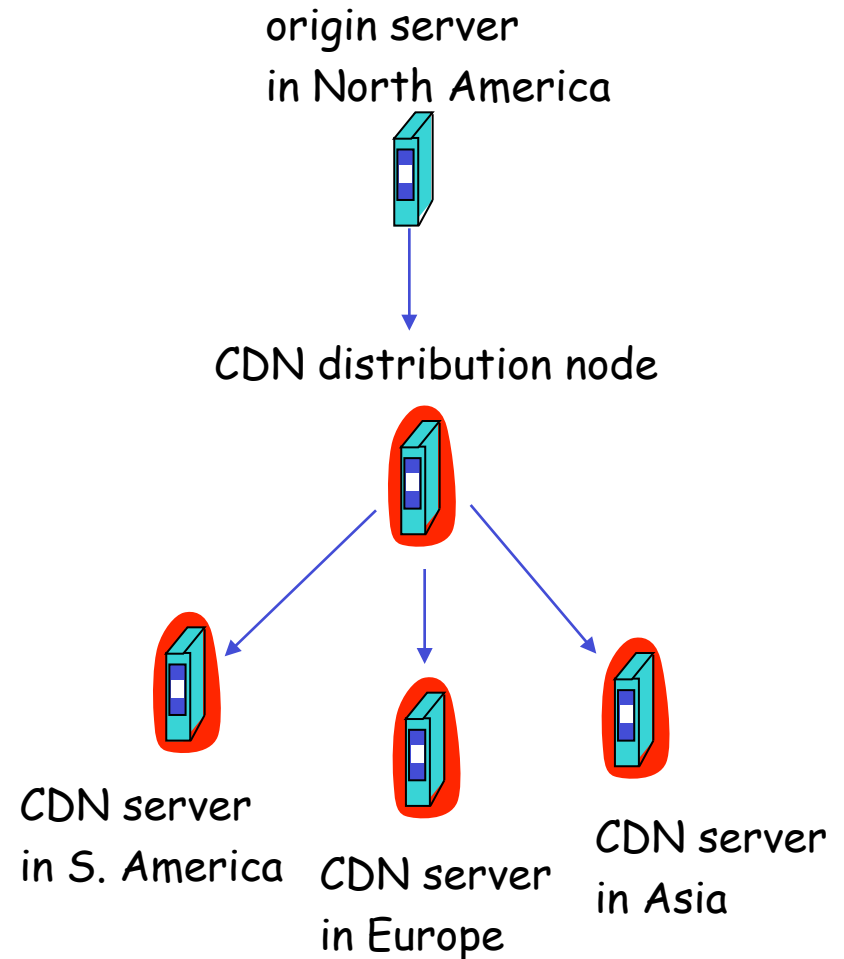


Content distribution networks (CDNs)

Content replication

- ❖ challenging to stream large files (e.g., video) from single origin server in real time
- ❖ *solution*: replicate content at hundreds of servers throughout Internet
 - content downloaded to CDN servers ahead of time
 - *placing content “close” to user avoids impairments (loss, delay) of sending content over long paths*
 - CDN server typically in edge/access network

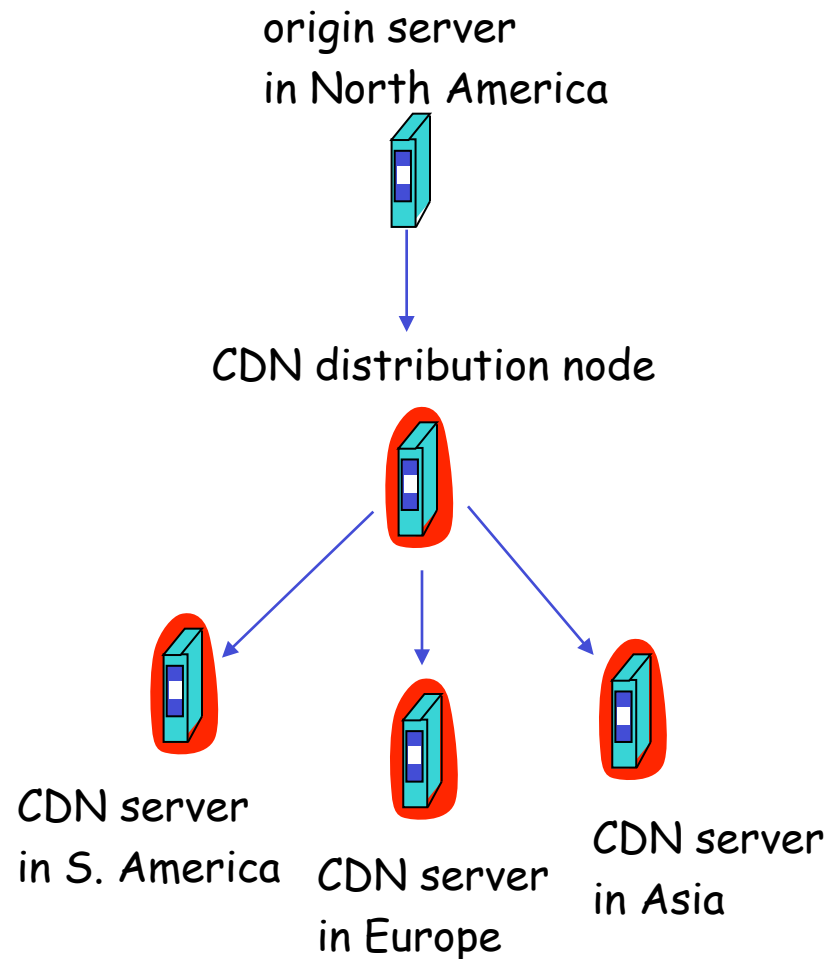
Multimedia Networking



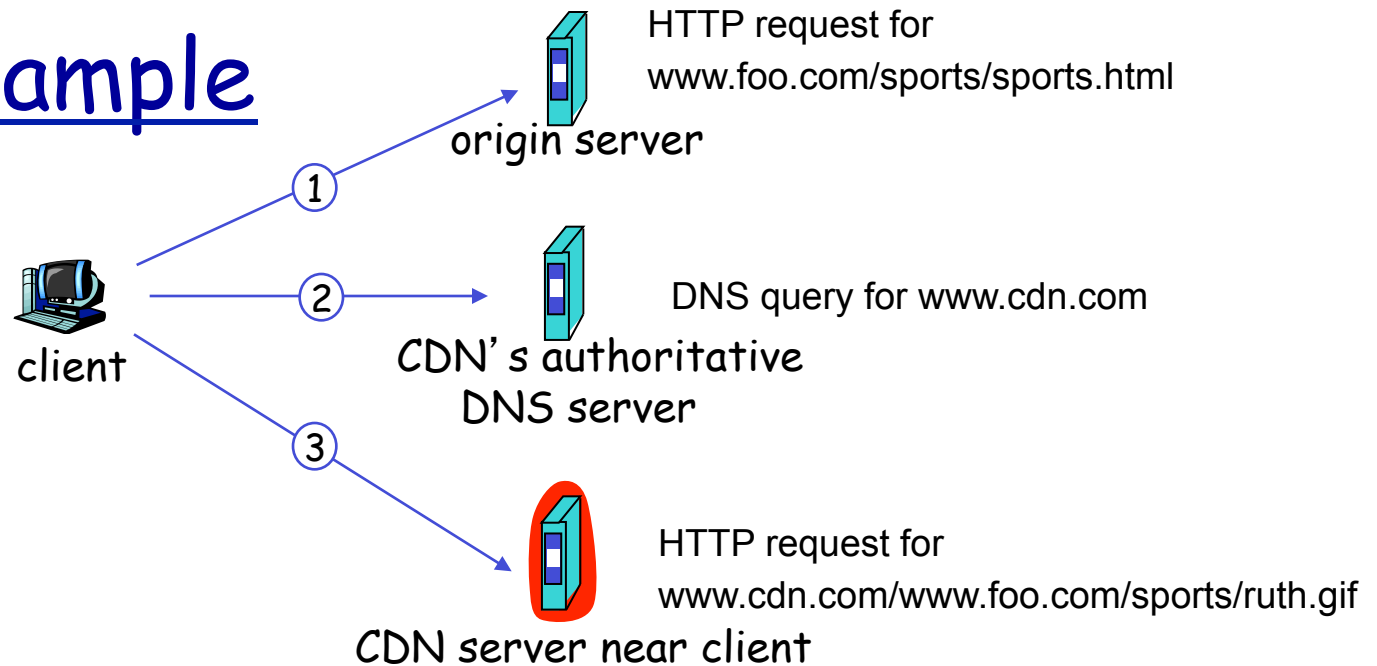
Content distribution networks (CDNs)

Content replication

- ❖ CDN (e.g., Akamai) customer is the content provider (e.g., CNN)
- ❖ CDN replicates customers' content in CDN servers.
- ❖ when provider updates content, CDN updates servers



CDN example



origin server (www.foo.com)

❖ distributes HTML

❖ replaces:

`http://www.foo.com/sports.ruth.gif`

with

`sports/ruth.gif`

`http://www.cdn.com/www.foo.com/`

CDN company (cdn.com)

❖ distributes gif files

❖ uses its authoritative DNS server to route redirect requests

More about CDNs

routing requests

- ❖ CDN creates a “map”, indicating distances from leaf ISPs and CDN nodes
- ❖ when query arrives at authoritative DNS server:
 - server determines ISP from which query originates
 - uses “map” to determine best CDN server
- ❖ CDN nodes create application-layer overlay network

Chapter 2: Application layer

2.1 Principles of network applications

2.2 Web and HTTP

2.3 FTP

2.4 Electronic mail

- SMTP, POP3, IMAP

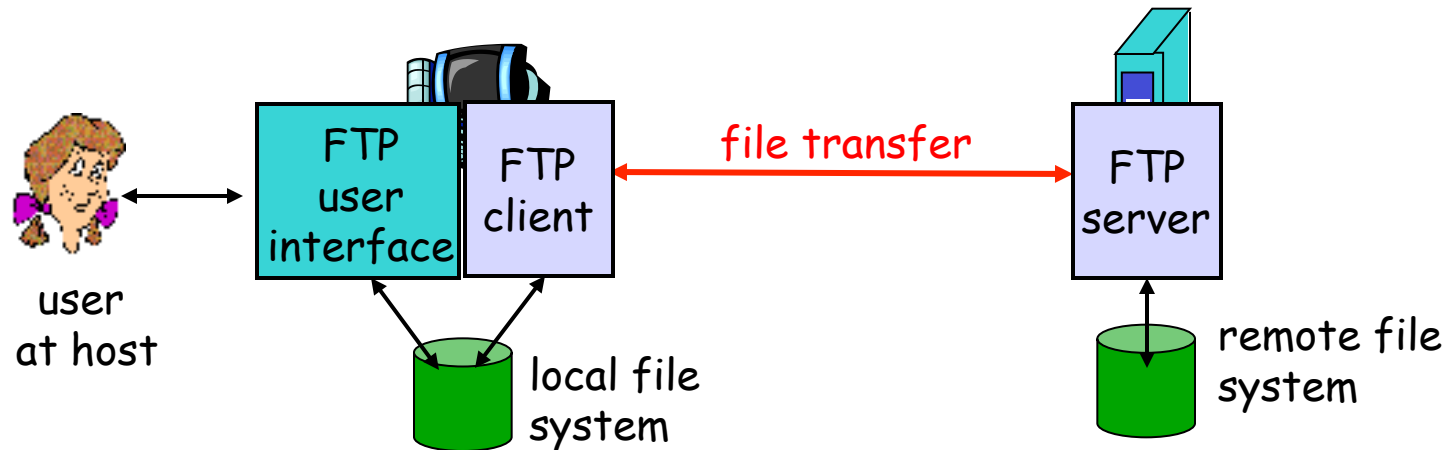
2.5 DNS

2.6 P2P applications

2.7 Socket programming with TCP

2.8 Socket programming with UDP

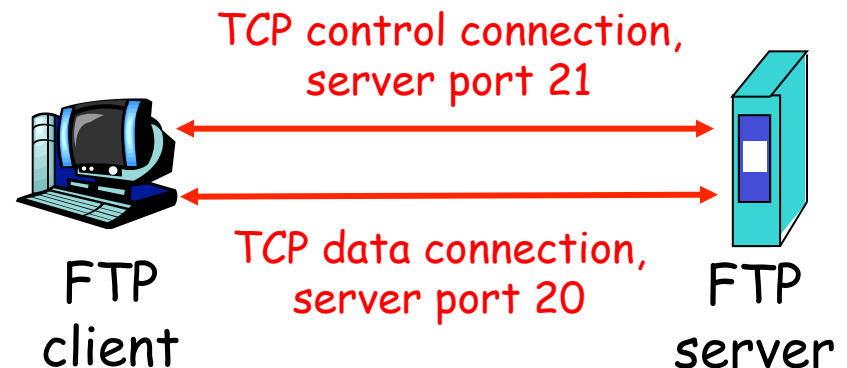
FTP: the file transfer protocol



- ❖ 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, TCP is transport protocol
- ❖ client authorized over control connection
- ❖ client browses remote directory by sending commands over control connection.
- ❖ when server receives file transfer command, server opens 2nd TCP connection (for file) to client
- ❖ after transferring one file, server closes data connection.



- ❖ server opens another TCP data connection to transfer another file.
- ❖ control connection: “out of band”
- ❖ 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

Chapter 2: Application layer

2.1 Principles of network applications

2.2 Web and HTTP

2.3 FTP

2.4 Electronic Mail

- SMTP, POP3, IMAP

2.5 DNS

2.6 P2P applications

2.7 Socket programming with TCP

2.8 Socket programming with UDP

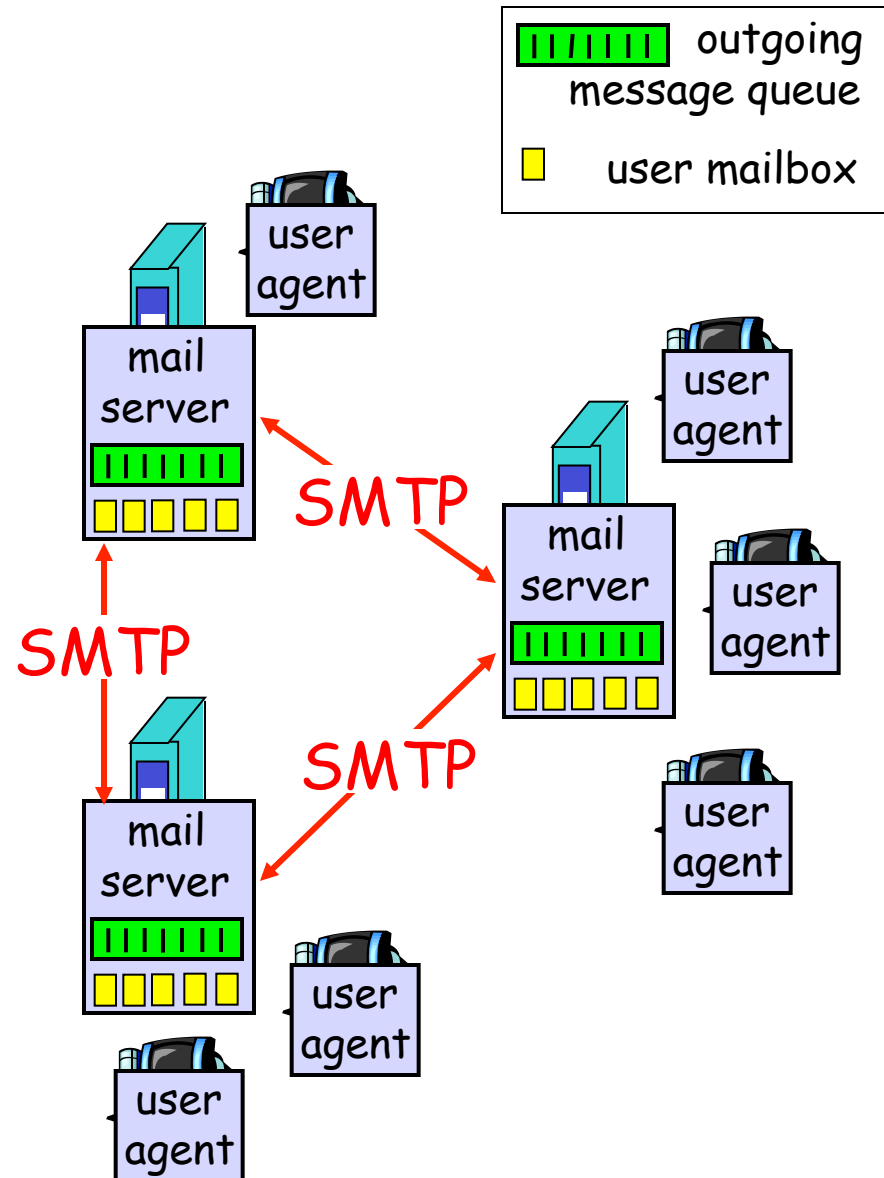
Electronic Mail

Three major components:

- ❖ user agents
- ❖ mail servers
- ❖ simple mail transfer protocol: SMTP

User Agent

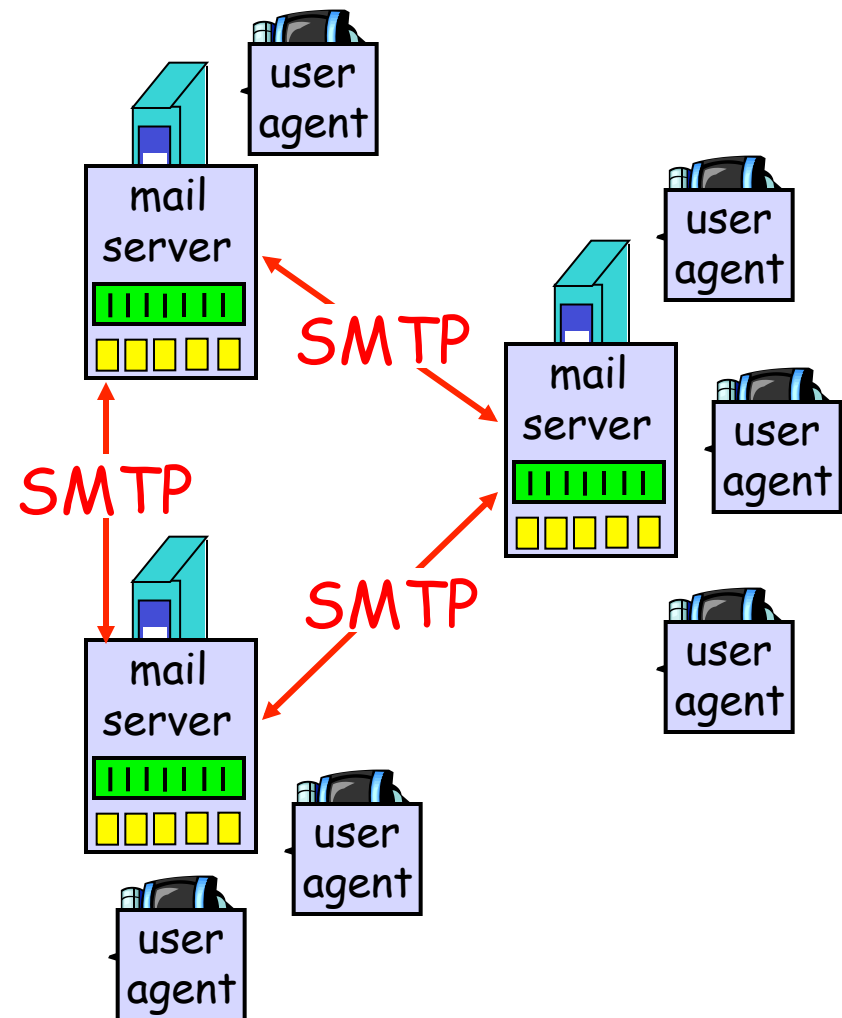
- ❖ a.k.a. “mail reader”
- ❖ composing, editing, reading mail messages
- ❖ e.g., Outlook, elm, Mozilla Thunderbird, iPhone mail client
- ❖ outgoing, incoming messages stored on server



Electronic Mail: mail servers

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

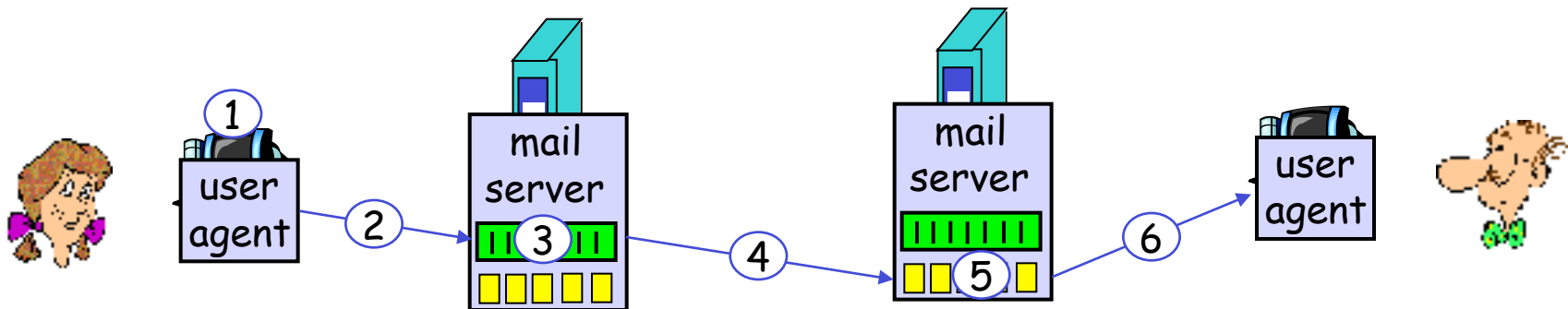


Electronic Mail: SMTP [RFC 2821]

- ❖ 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

Scenario: Alice sends message to Bob

- 1) Alice uses UA to compose message and "to" bob@some school.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



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 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)

SMTP: final words

- ❖ 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

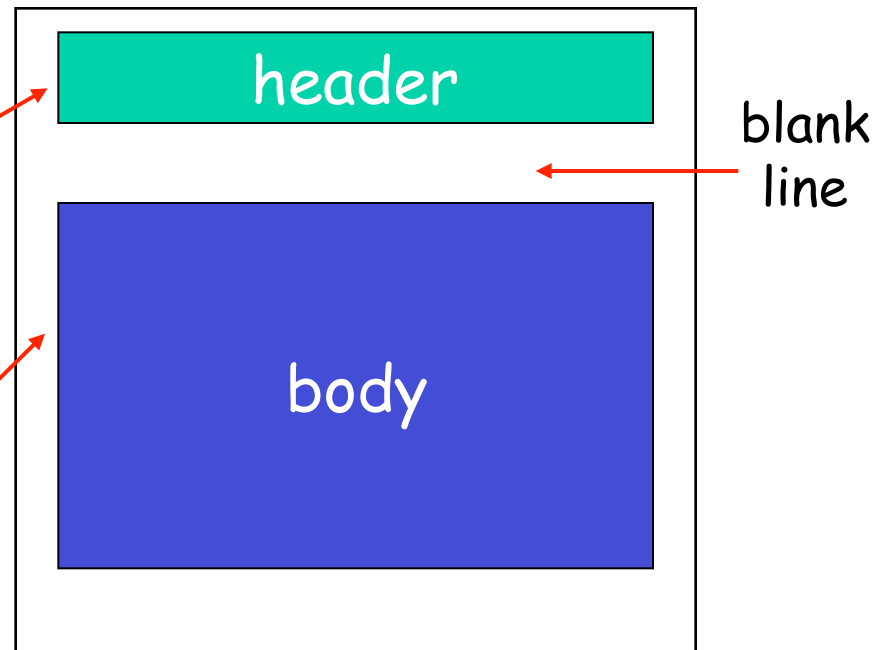
Mail message format

SMTP: protocol for exchanging email msgs

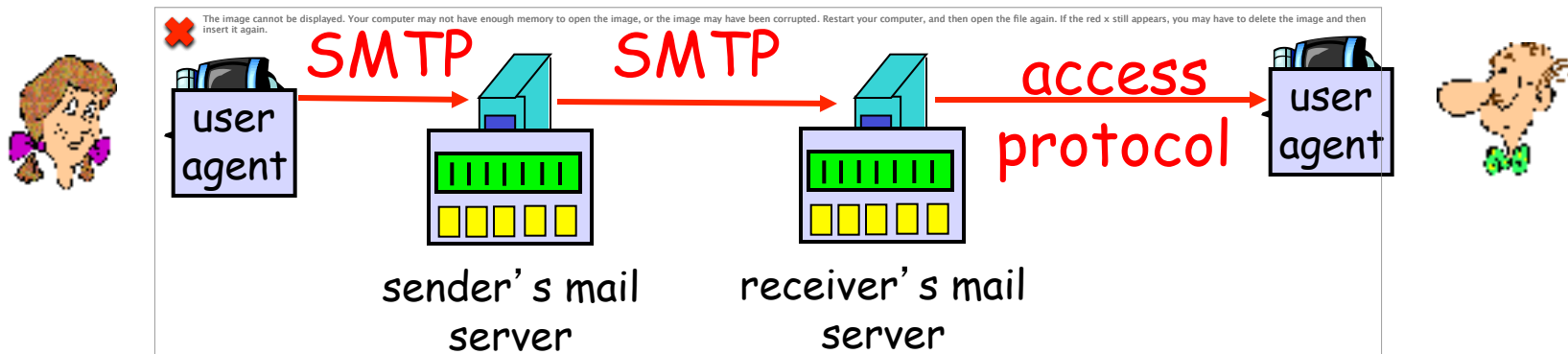
RFC 822: standard for text message format:

- ❖ header lines, e.g.,
 - To:
 - From:
 - Subject:

different from SMTP commands!
- ❖ body
 - the “message”, ASCII characters only



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.

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
```

POP3 (more) and IMAP

more about POP3

- ❖ 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

- ❖ keeps all messages in one place: at server
- ❖ allows user to organize messages in folders
- ❖ keeps user state across sessions:
 - names of folders and mappings between message IDs and folder name