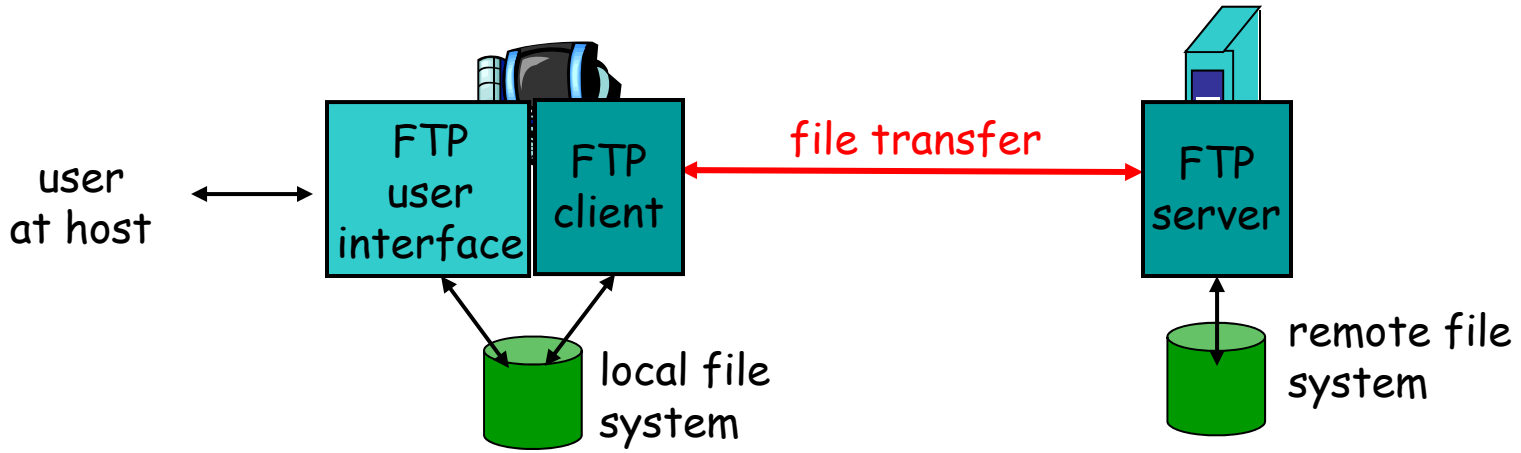
CS 372 Lecture #12

The Application Layer:

- File Transfer Protocol (ftp)
- Email protocols

Note: Many of the lecture slides are based on presentations that accompany *Computer Networking: A Top Down Approach,* 6th edition, by Jim Kurose & Keith Ross, Addison-Wesley, 2013.

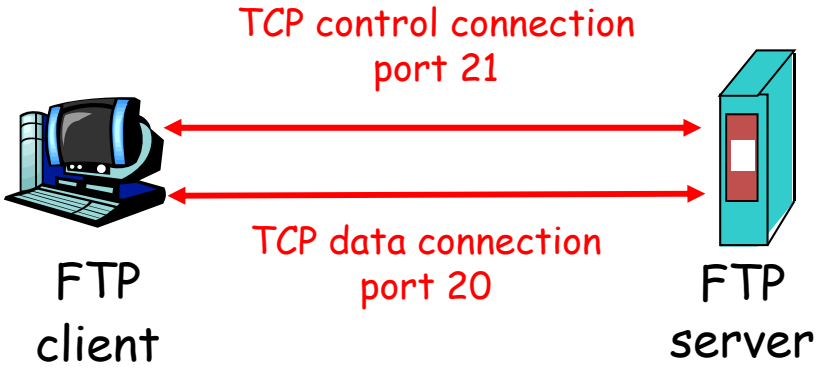
FTP: the file transfer protocol (RFC 959)



- transfer file to/from remote host
- client/server model
  - *client:* that initiates transfer (to or from remote)
  - *server:* remote host

FTP: separate control, data connections

- FTP client contacts FTP server to open TCP control connection
- Client browses remote directory by sending commands over the TCP control connection.



- When server receives file transfer command, server opens *2nd* TCP data connection (for file) to client
- After transferring one file, server closes TCP data connection. TCP control stays on.
- FTP server maintains "state":
  - current directory, earlier authentication
  - limit on concurrent connections

FTP commands, responses

A few sample commands:

sent as ASCII text over control channel

- **`ls`**
  - lists files in remote directory
- **`get filename`**
  - retrieves file from remote site to current local directory
- **`put filename`**
  - stores file onto remote host in current directory (if permitted)
- **`cd directory`**
  - changes remote directory
- **`help`**
  - lists all ftp commands

NOTE: Your textbook shows the "raw-ftp" commands (RETR, STOR, etc.). Most FTP applications use the commands as shown above … and these are translated to "raw-ftp".

A few sample return codes:

status code and phrase

- **`200 Command successful`**
- **`331 Username OK, password required`**
- **`425 Can't open data connection`**
- **`452 Error writing file`**
- **`550 Failed to open file`**

Try it! Example:
`ftp gaia.cs.umass.edu`
Log in as `anonymous` and use your email address as a password.
Experiment with some of the commands.
Most files/directories are protected.

Electronic Mail

Three major components:

User Agent

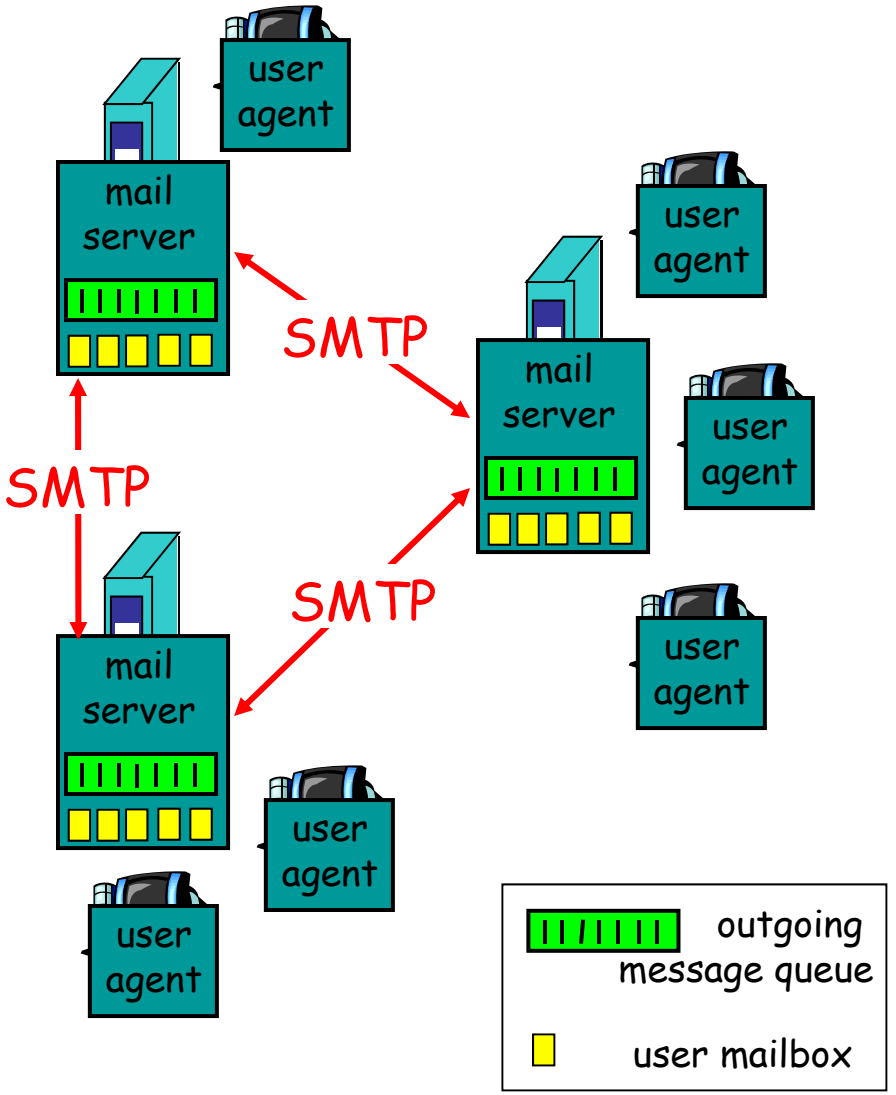
- email client: compose, read, edit, send
- e.g., Eudora, Outlook, elm, Mozilla Thunderbird
- outgoing, incoming messages stored on server

Mail Server

- mailbox contains incoming messages for user
- message queue of outgoing mail messages

SMTP [RFC 2821]

- Simple Mail Transfer Protocol
- defines message transfer rules & formats between mail servers
  - Client: sending mail
  - Server: receiving mail



Electronic Mail: SMTP [RFC 2821]

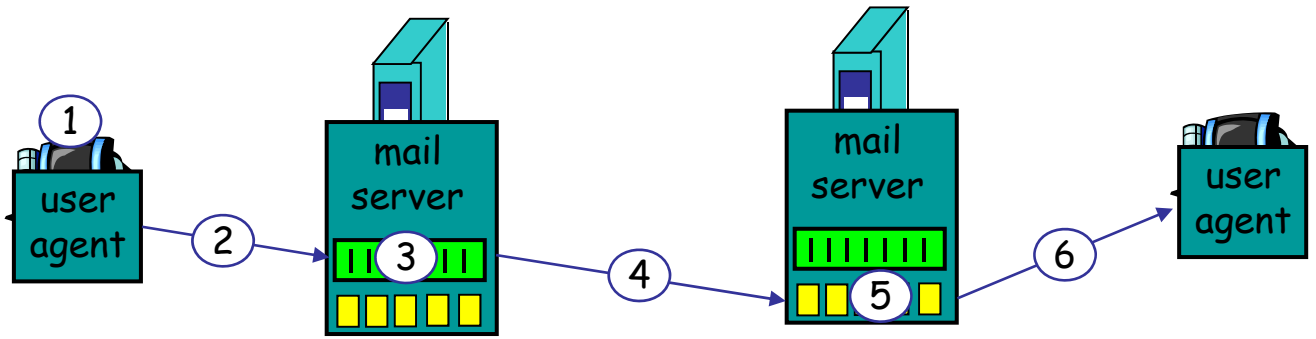
- application layer protocol uses TCP
  - client-server application for sending mail
  - reliable transfer
  - default port is 25
- command/response interaction
  - commands: plain text
  - response: status code and phrase
- messages must be in 7-bit ASCI

Sample SMTP interaction (Lab #1)

```
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@someschool.edu>
S: 250 bob@someschool.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Bon jour from the Far Side.
C: Have you scheduled your flight yet?
C: .
S: 250 Message accepted for delivery
C: QUIT
```

Scenario: Alice sends message to Bob

1) Alice uses her user agent to compose message and send to `bob@someschool.edu`
2) Alice's user agent 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 uses his user agent to read message



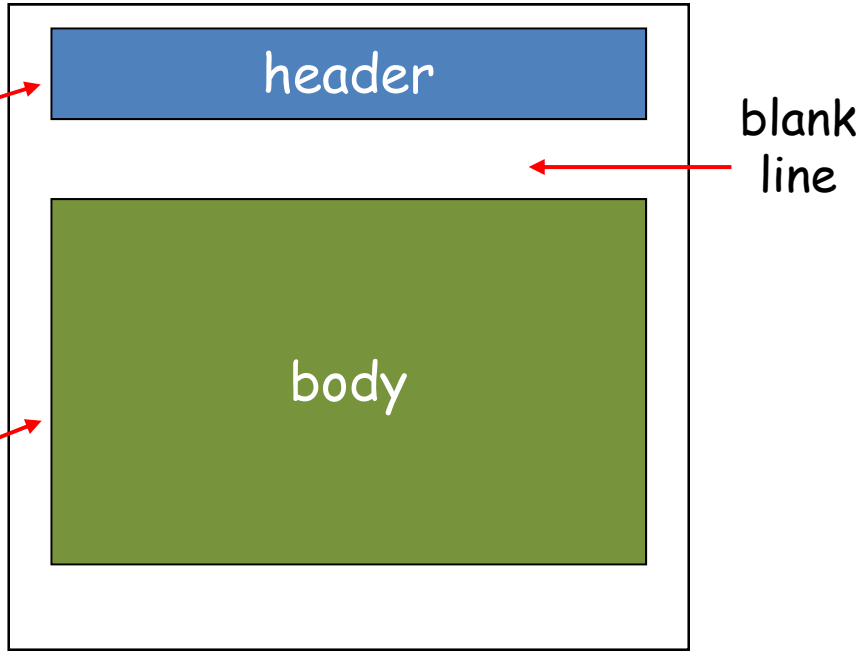
Mail message format

SMTP: protocol for exchanging email messages

RFC 822: standard for text message format:

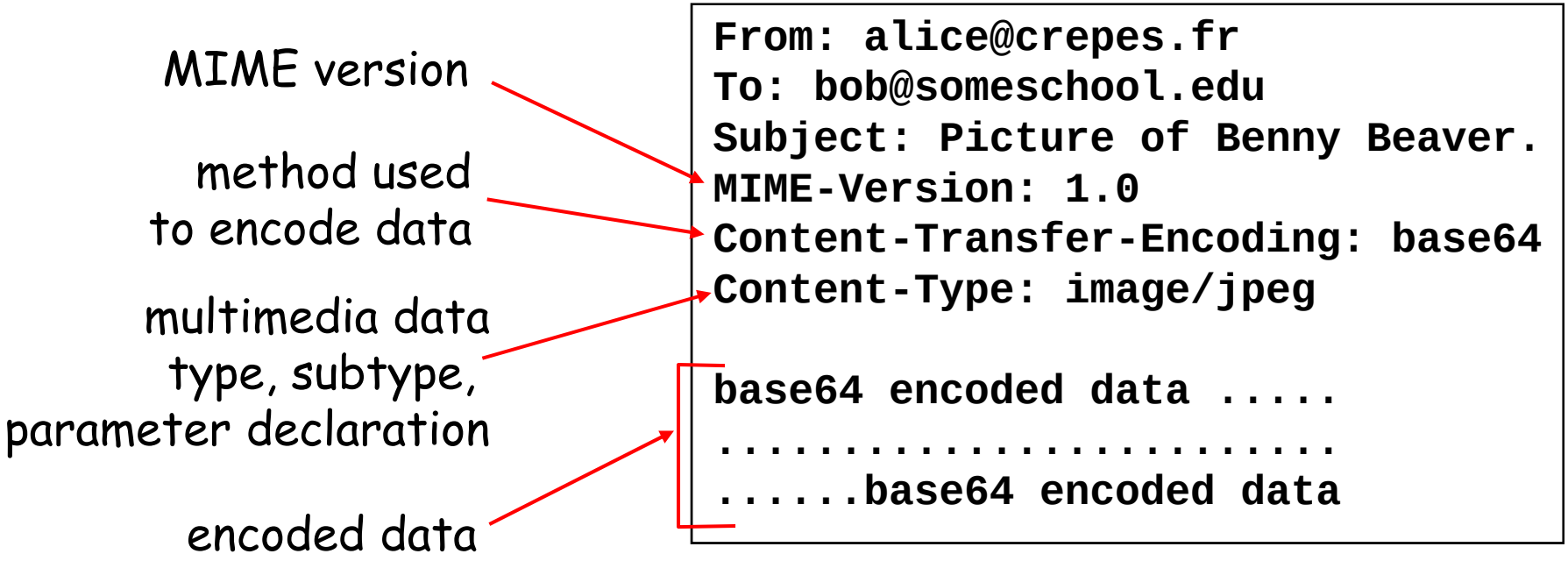
- header lines (entered by user, formatted by local user agent)
  - To:
  - From:
  - Subject:

- body
  - the “message”, ASCII characters only

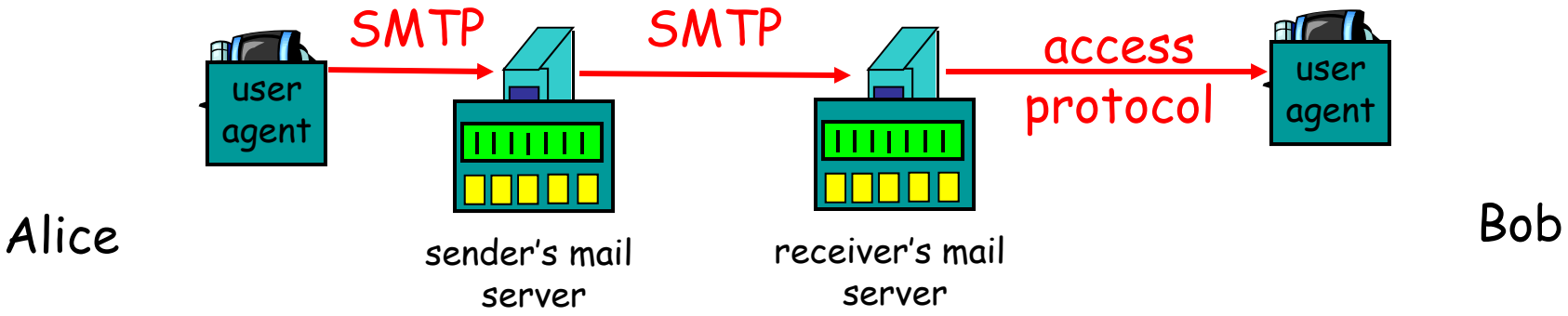


Message format: multimedia extensions

- MIME: Multipurpose Internet Mail Extension, RFC 2045, 2056
  - Allows SMTP to handle foreign characters and images
  - additional lines in message header declare MIME content type



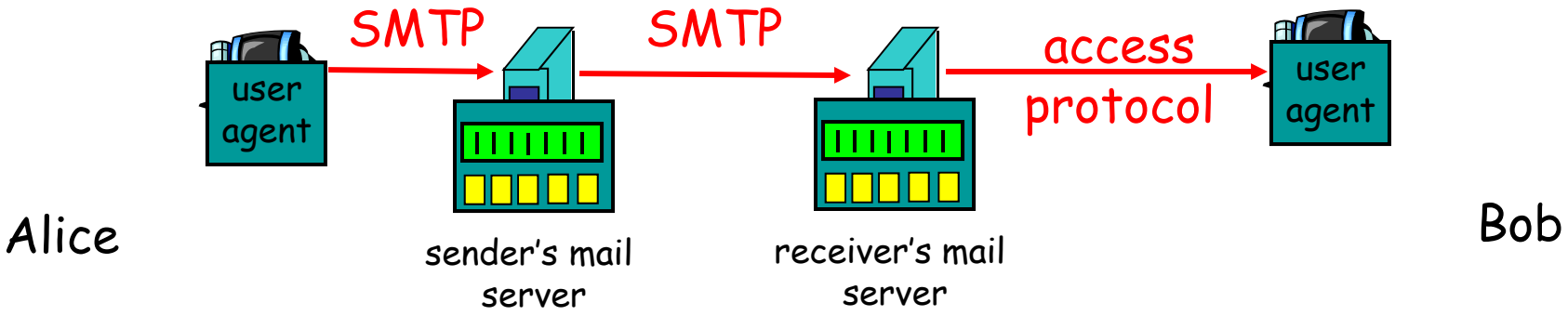
Mail access protocols



- Q: Why do we need mail servers?
  - Why can't Alice's user agent push message to Bob's user agent without going via mail servers?

- A: Potential problems:
  - first attempt fails (who tries again?)
  - Bob's user agent is off (who stores incoming messages for him?)

Mail access protocols

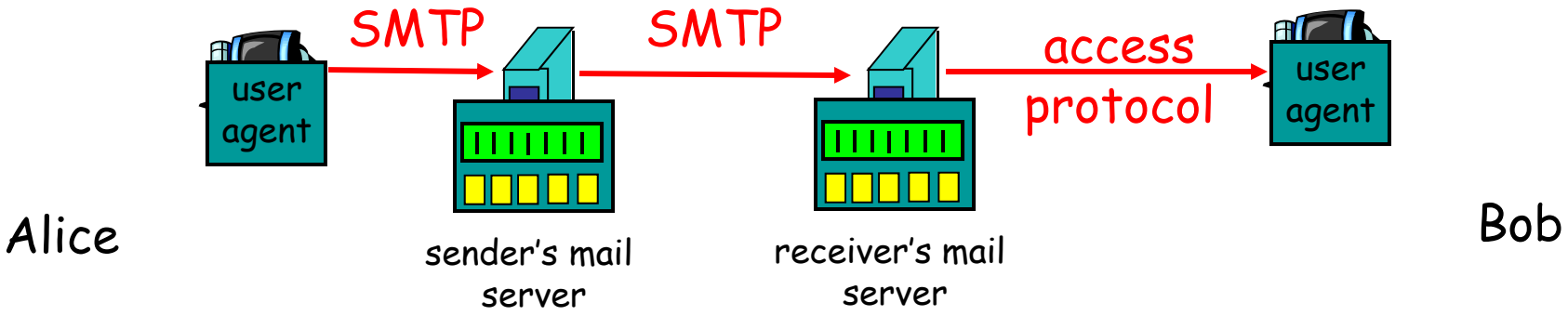


- Now because messages are to be stored at the server, Bob's user agent must be able to retrieve them

- Q: why can't Bob use SMTP to retrieve messages ?

- A: SMTP is a "push" protocol. Hence, we need "pull" protocols: called "mail access protocols"

Mail access protocols



3 mail access protocols:

- POP: Post Office Protocol [RFC 1939]
  - authorization (agent <-->server) and download
- IMAP: Internet Mail Access Protocol [RFC 1730]
  - manipulation of stored messages on server, more complex
- HTTP: gmail, Hotmail, Yahoo! Mail, etc.

POP3 and IMAP

POP3

- 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

Summary Lecture #12

- FTP
  - control connection, data connection
- SMTP
  - MIME
- Mail access protocols
  - POP3
  - IMAP
  - HTTP
- Definitions
  - user agent
  - push protocol
  - pull protocol