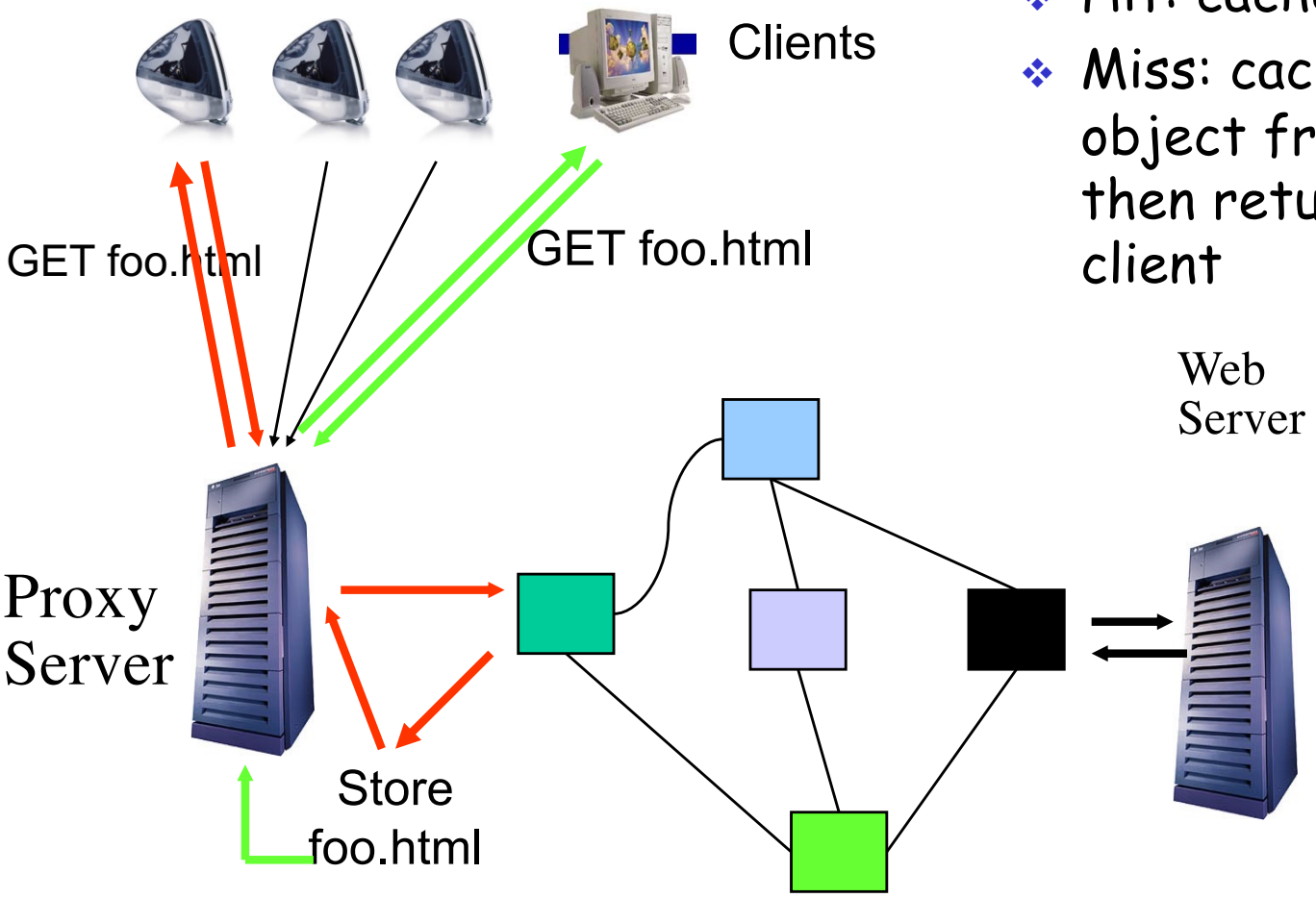
Web caches (proxy server)

Why?

- Reduce response time for client request.
- Reduce traffic on an institution's access link.

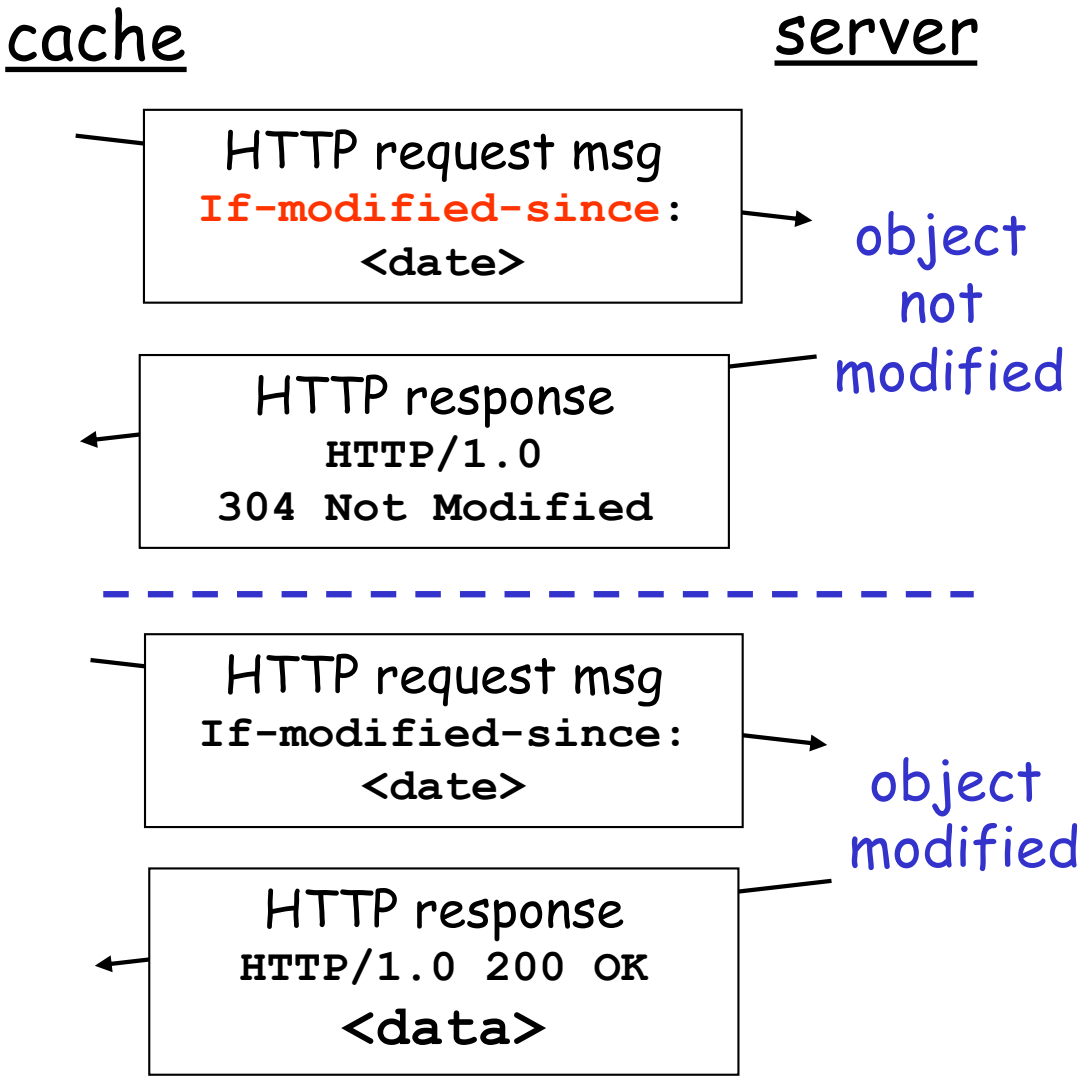
Web caches (proxy server)

- browser sends all HTTP requests to cache
  - Hit: cache returns object
  - Miss: cache requests object from origin server, then returns object to client



Web caches: implementation

- guarantees cache content is up-to-date
- saves traffic and response time whenever posssible



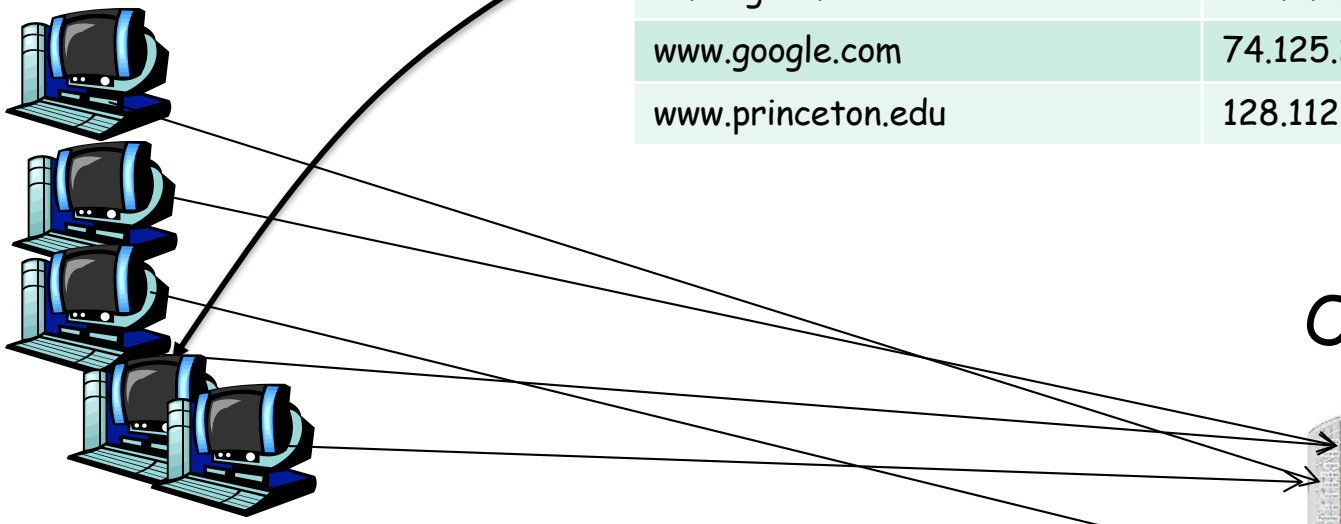
Content Distribution Networks (CDN)

Why?

- Reduce bandwidth requirements of content provider
- Reduce $$ of maintaining Servers
- Cache for server content
- Reduce traffic on the link to the content provider.
- Improve response time to user

Without CDN

DOMAIN NAME	IP ADDRESS
WWW.YAHOO.COM	98.138.253.109
cs.rutgers.edu	128.6.4.2
www.google.com	74.125.225.243
www.princeton.edu	128.112.132.86



Origin Server

98.138.253.109

- Huge B/W requirements
- Does not scale
- So, Distribute content to geographically distributed servers
- Use DNS to redirect request to copies user content

CDN terms

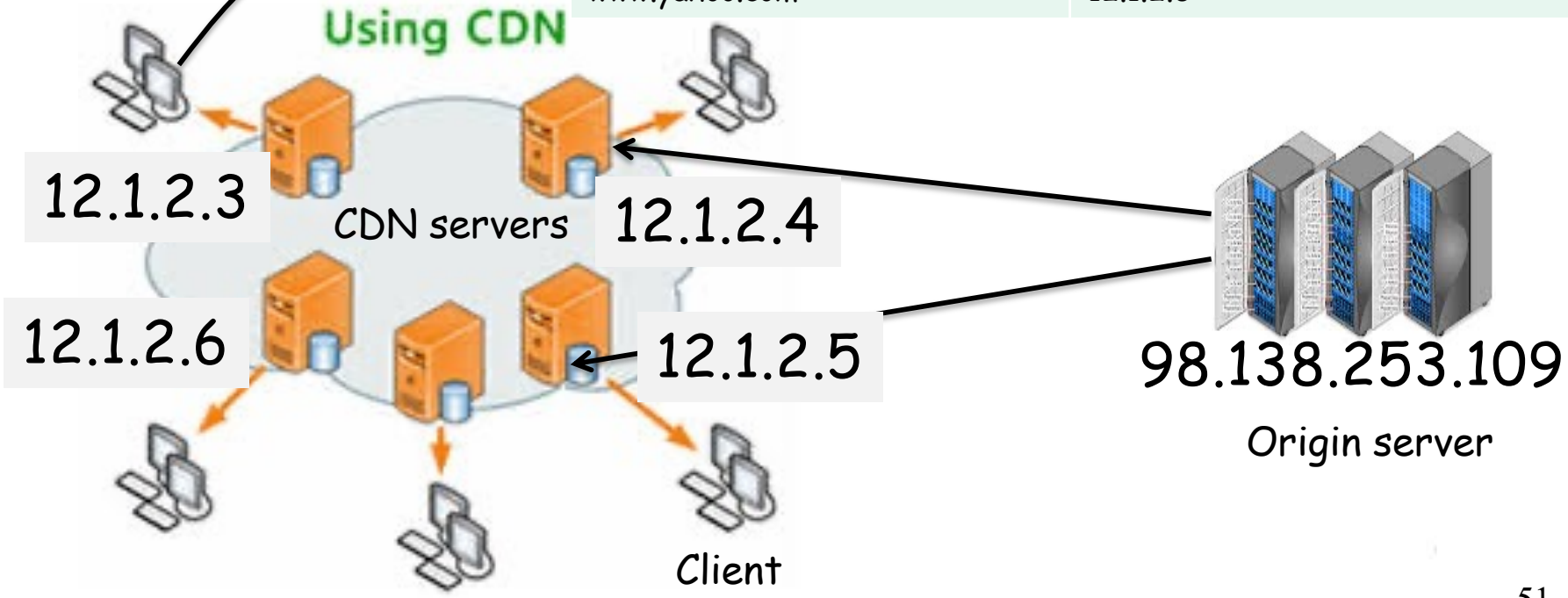
- Origin server
  - Server that holds the authoritative copy of the content
- CDN server
  - A replica server owned by the CDN provider
- CDN name server
  - A DNS like name server used for redirection
- Client

With CDN

DOMAIN NAME	IP ADDRESS
WWW.YAHOO.COM	124.8.9.8 (NS of CDN)
cs.rutgers.edu	128.6.4.2
www.google.com	74.125.225.243
www.princeton.edu	128.112.132.86

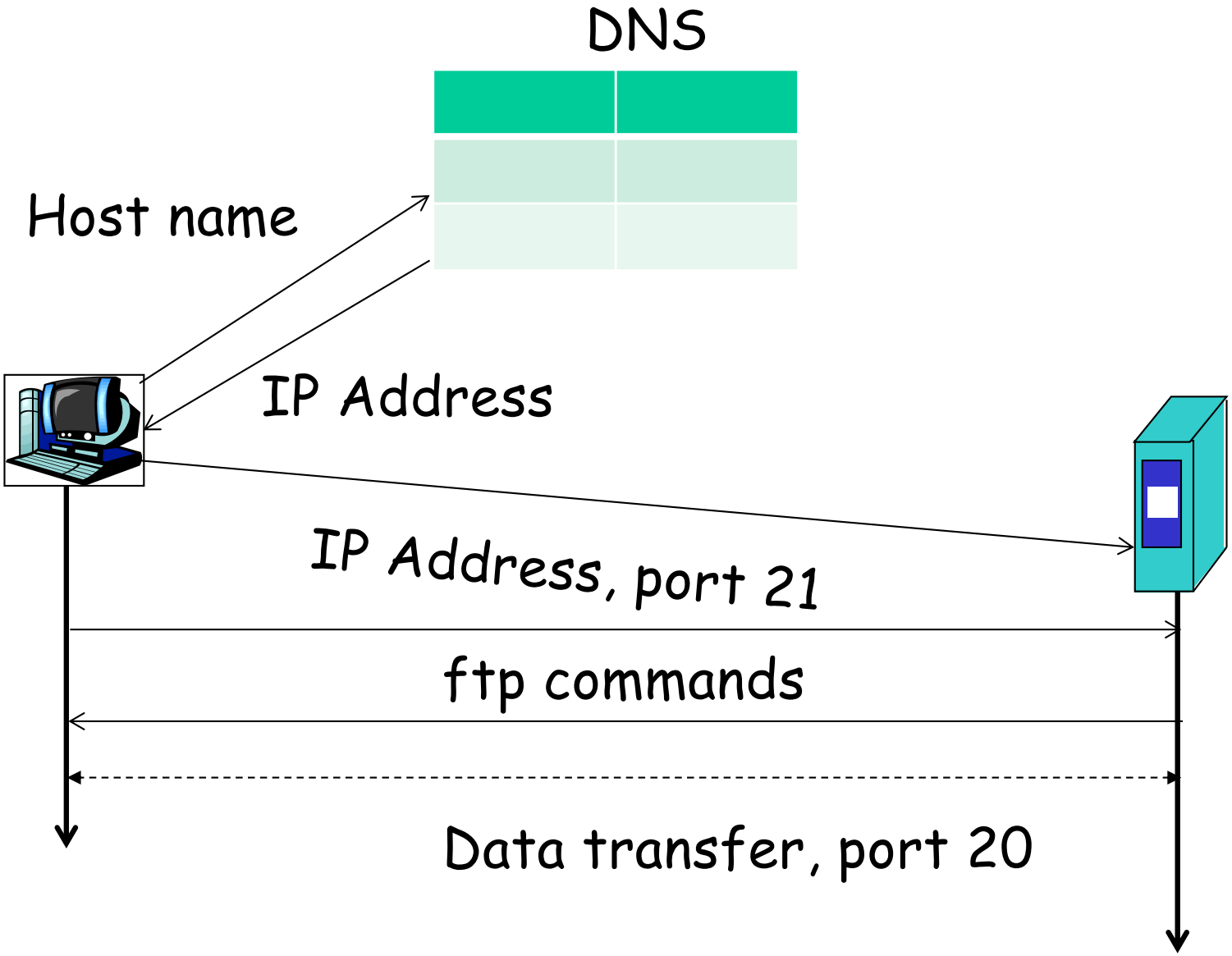
CDN Name Server (124.8.9.8)

DOMAIN NAME	IP ADDRESS
WWW.YAHOO.COM	12.1.2.3
www.yahoo.com	12.1.2.4
www.yahoo.com	12.1.2.5
www.yahoo.com	12.1.2.6

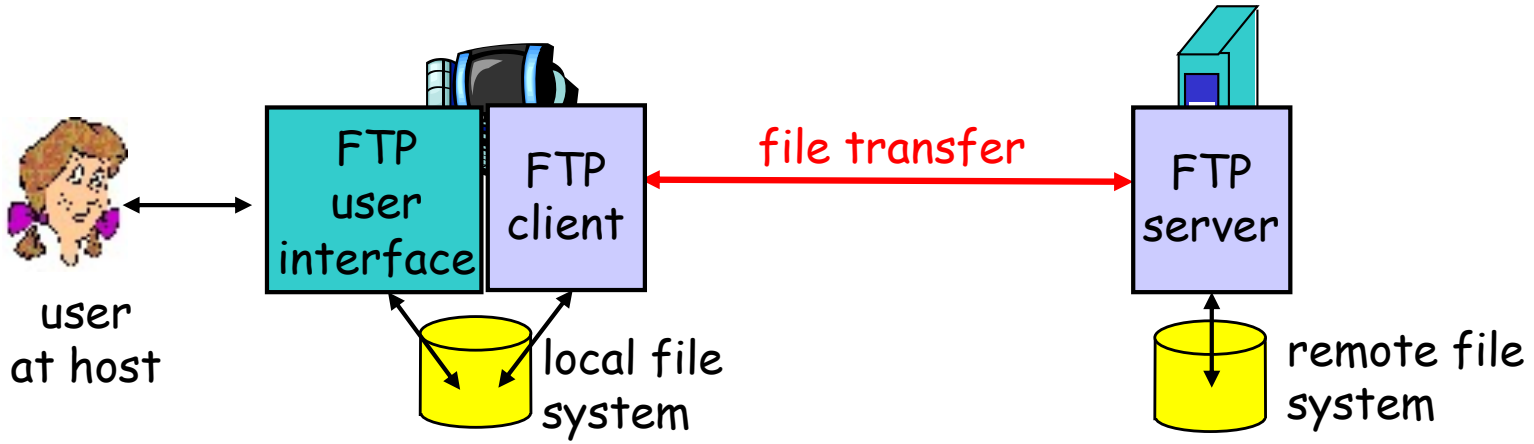


FTP

Client server connection



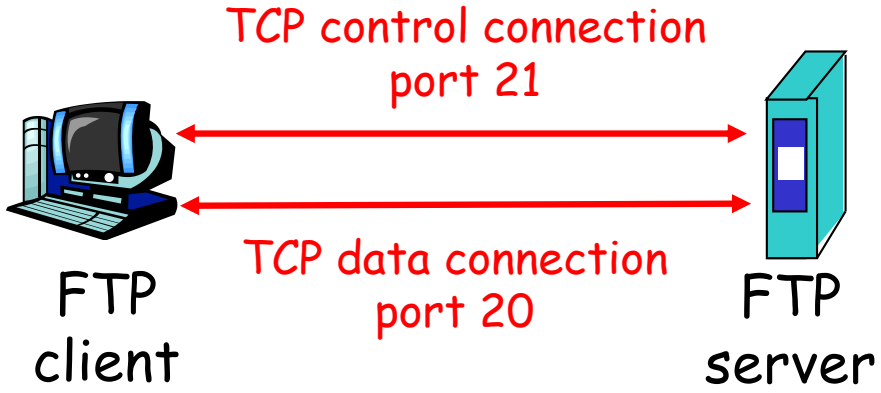
FTP: the file transfer protocol



- 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, port 20 (data connection)

FTP: separate control, data connections

- "out of band" control
  - Control connection:
    - Authorization
    - Directory browse
    - Commands
  - Data connection
    - Transfer files
- FTP server maintains "state":
  - current directory
  - earlier authentication



FTP commands, responses

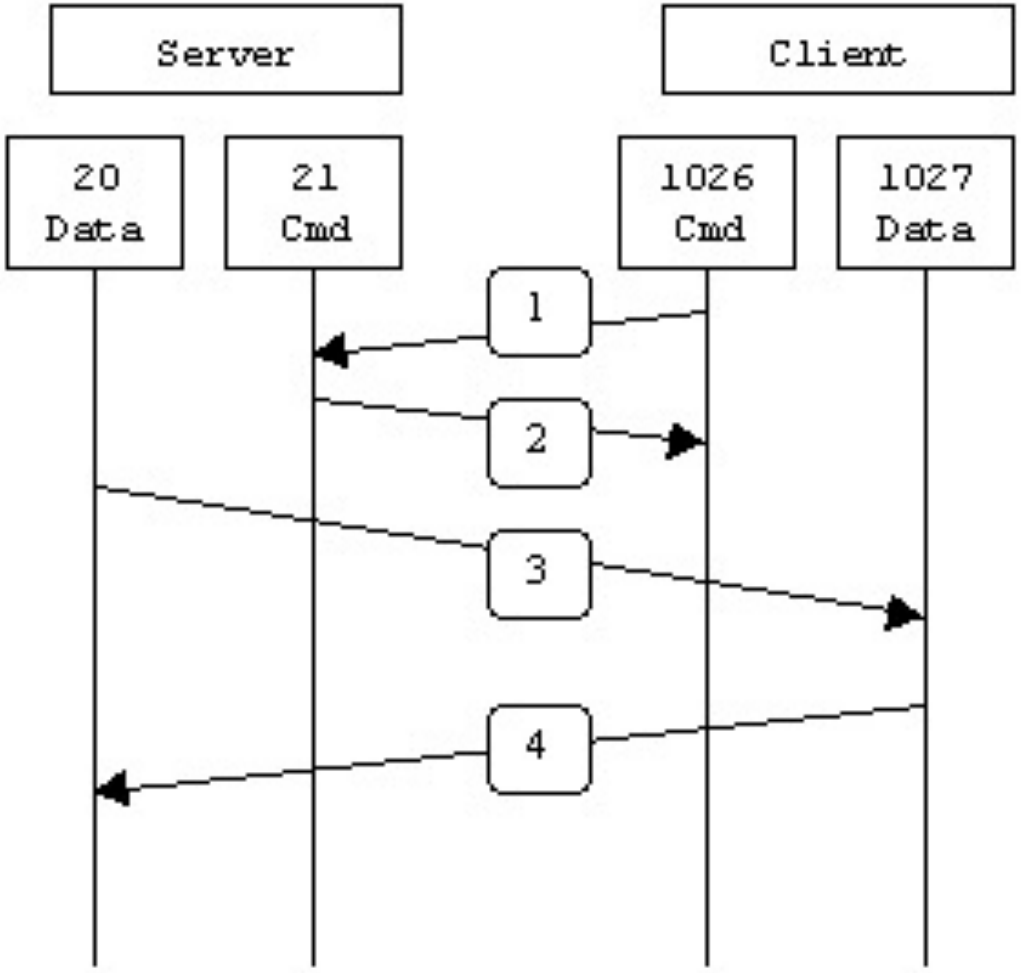
Sample commands:

- 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

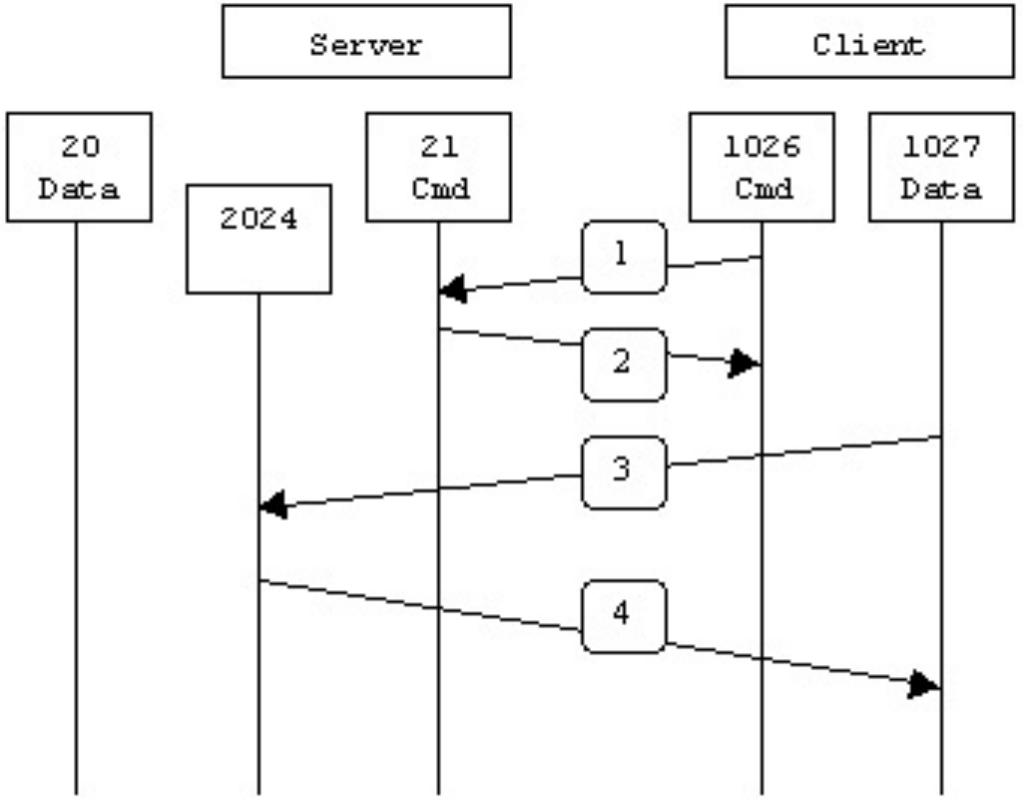
- status code and phrase (as in HTTP)
- **331 Username OK, password required**
- **125 data connection already open; transfer starting**
- **425 Can't open data connection**
- **452 Error writing file**

FTP Active connection (Data connection initiated from server)



- Client opens a connection from port x for sending commands to server port 21
- Server opens a connection from port 20 to send data at port x+1

FTP passive connection (always client initiated)



- Client opens a connection from port x for sending commands to server port 21
- Client sends a request for PASSIVE connection with PASV command
- Server replies with a new port number Sp on which it is listening
- Client opens a connection from port x+1 to server port Sp

FTP

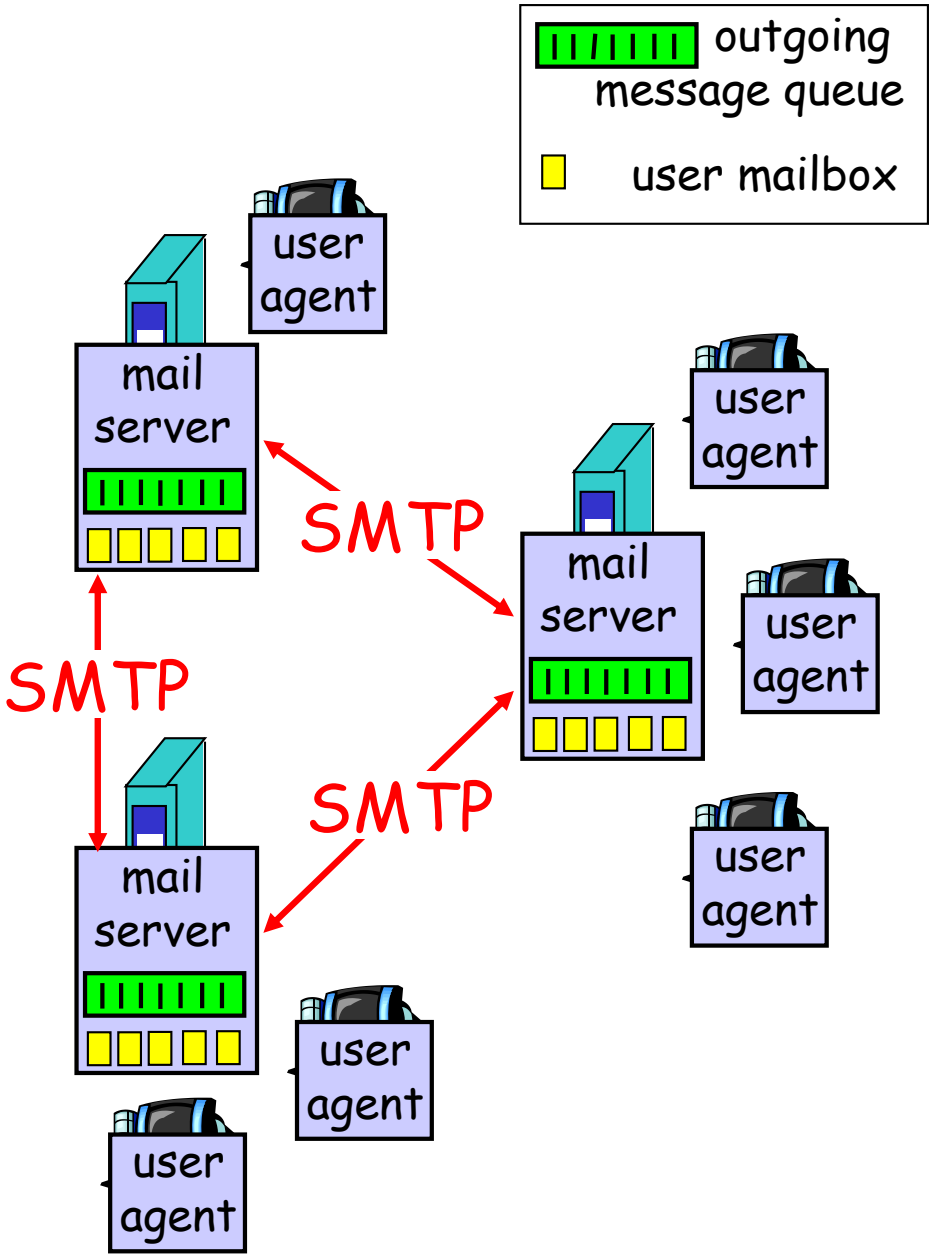
- Sends passwords in plain ASCII text
  - Eavesdropper can recover passwords
  - Fatal flaw, turned off at a lot of sites
  - Replaced with scp, sftp instead

SMTP

Electronic Mail

Three major components:

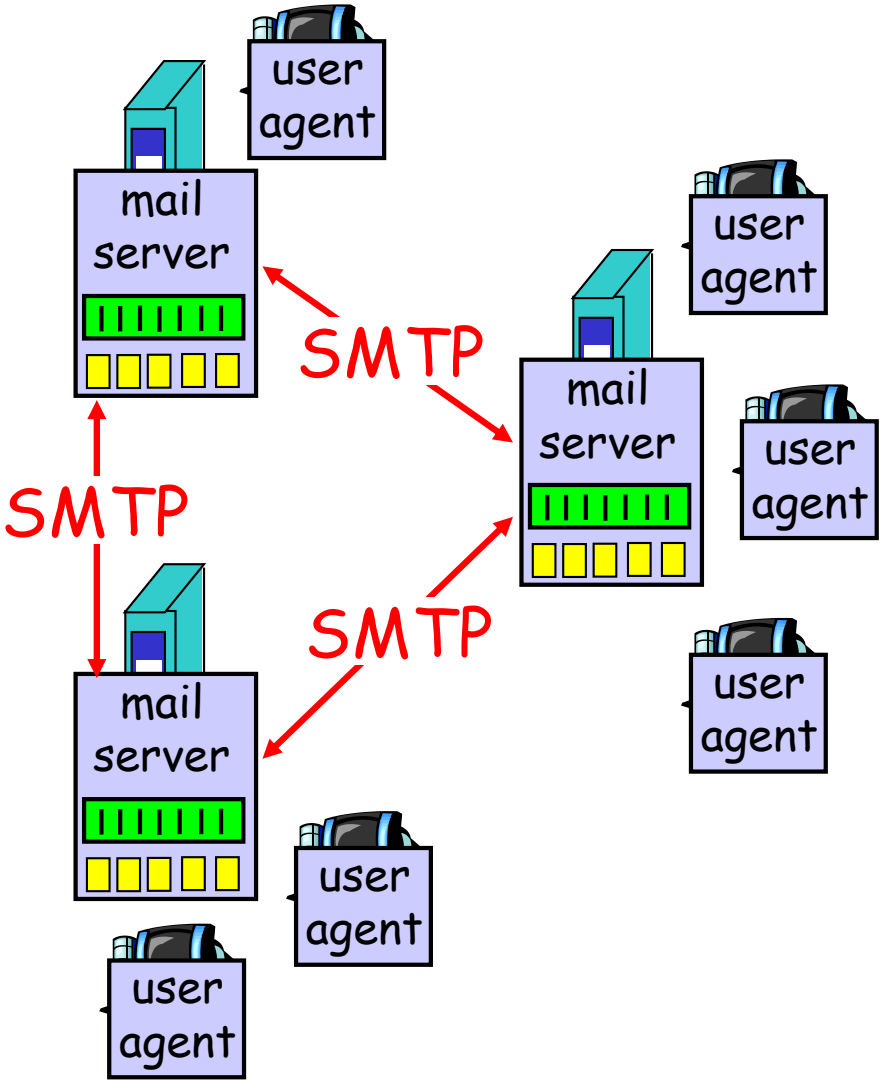
1. user agents
   - a.k.a. "mail reader"
   - e.g., gmail, Outlook, yahoo



Electronic Mail: mail servers

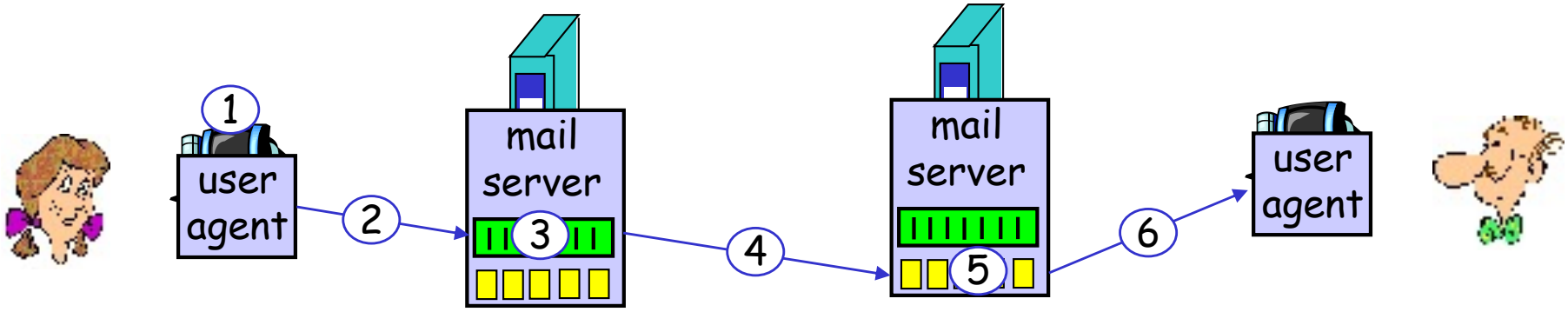
2. Mail Servers
   - mailbox contains incoming messages for user
   - message queue of outgoing (to be sent) mail messages
- Sender mail server makes connection to Receiver mail server
  - IP address, port 25

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 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



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

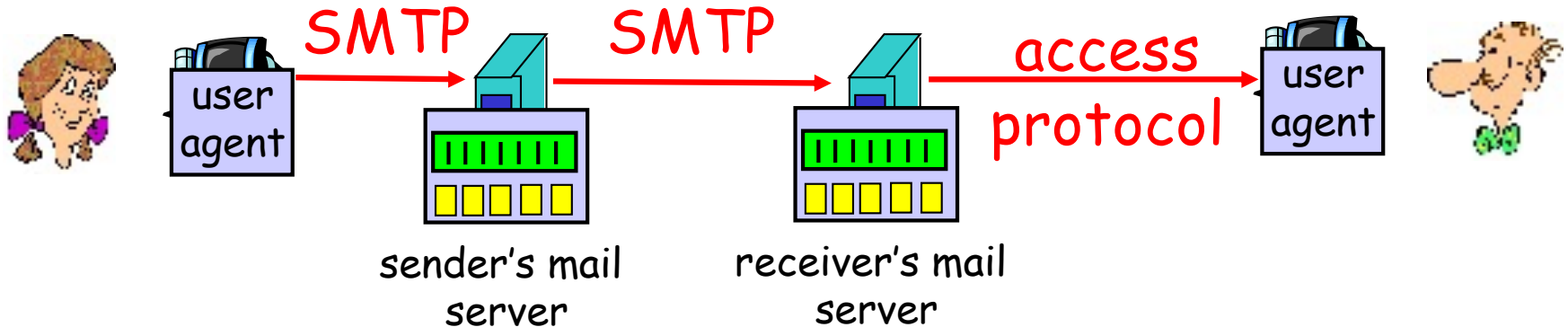
221 hill.com closing connection

MAIL

Table 23.2 *Responses*

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

Mail access protocols



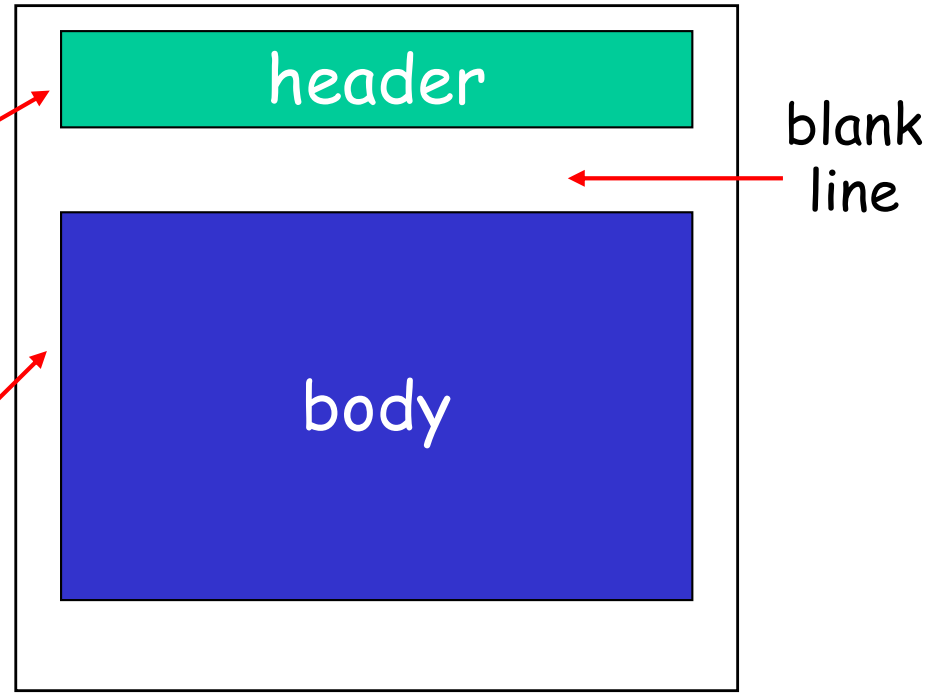
- SMTP: delivery/storage to receiver's server
- Mail access protocol: retrieval from server
  - POP: Post Office Protocol [RFC 1939]
  - IMAP: Internet Mail Access Protocol [RFC 1730]
  - HTTP: Hotmail , Yahoo! Mail, etc.

Mail message (stored on server) format

SMTP: protocol for exchanging email msgs

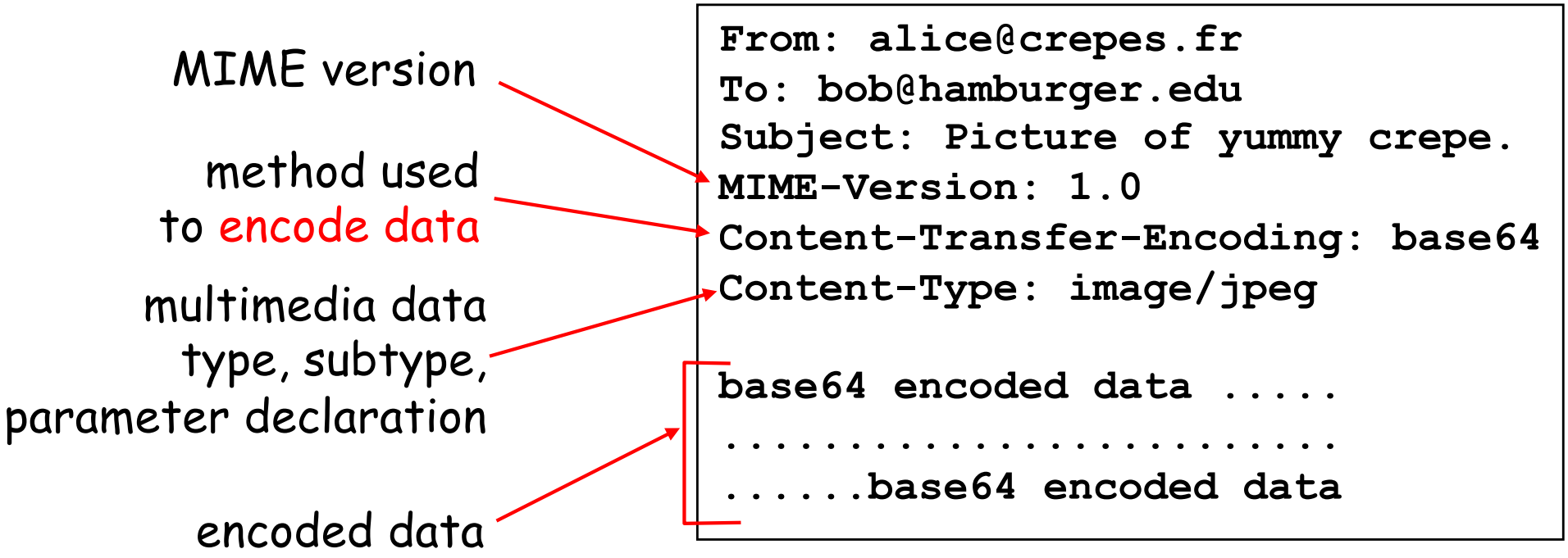
RFC 822: standard for text message format:

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



SMTP: final words

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