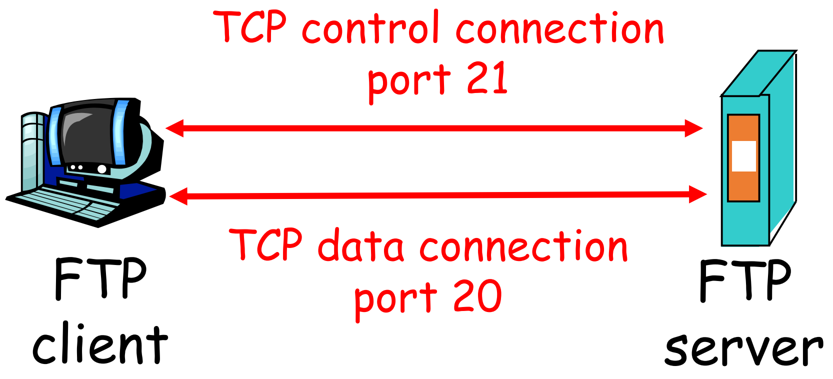
Application Layer Protocols

FTP, SMTP, HTTP

File Transfer Protocol (FTP)

- 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: through out the connetion
- 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

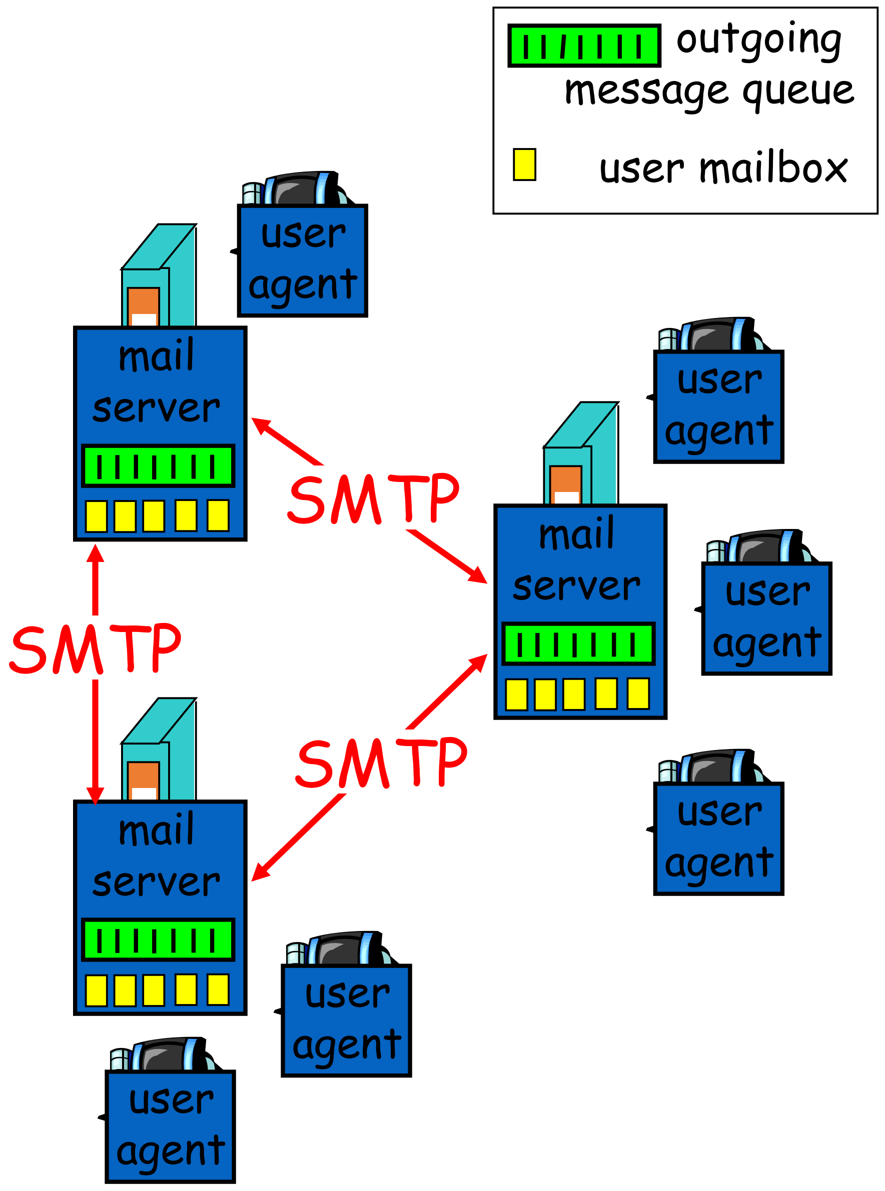
Sample return codes

- 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
  - e.g., Eudora, Outlook, Pine, Netscape Messenger
- mail servers
  - Incoming, outgoing messages
- Simple Mail Transfer Protocol: SMTP



Electronic Mail: SMTP [RFC 2821]

- Client’s SMTP mail server establishes a TCP connection to the recipients SMTP server using Port 25
- three phases in message 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

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
```

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
- SMPT is a “chatty” protocol

Comparison with HTTP:

- HTTP: pull
- SMTP: push

- both have ASCII command/response interaction, status codes

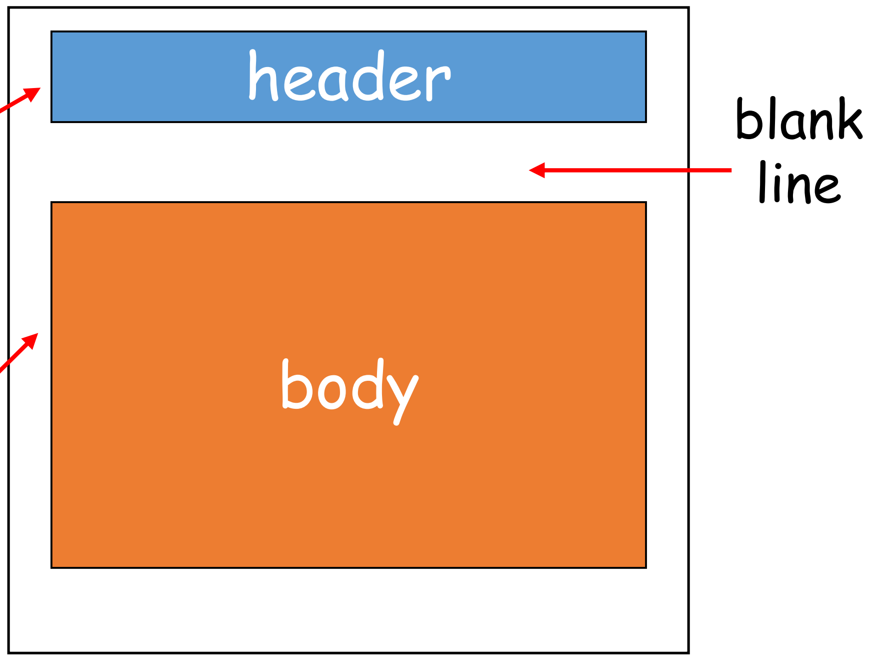
- HTTP: each object encapsulated in its own response msg
- SMTP: multiple objects sent in multipart msg

Mail message format

SMTP: protocol for exchanging email msgs

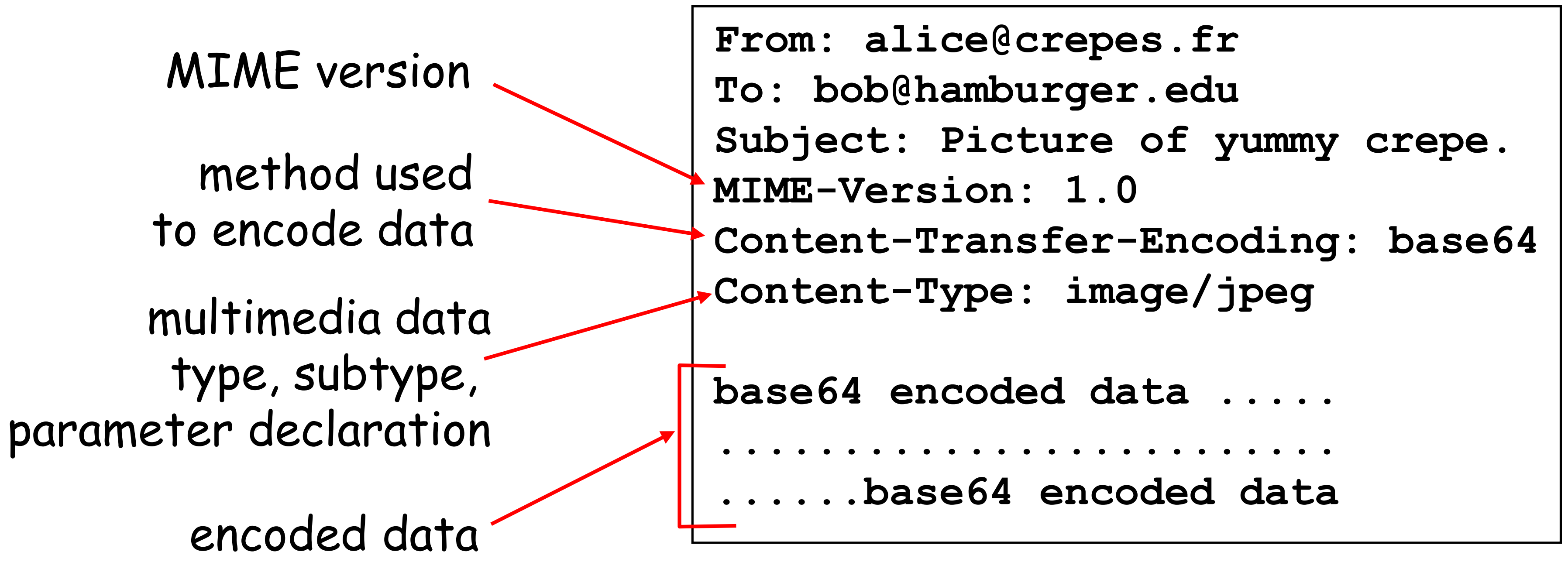
RFC 822: standard for text message format:

- header lines, e.g.,
  - To:
  - From:
  - Subject:
- body
  - the "message", ASCII characters only



Message format: multimedia extensions

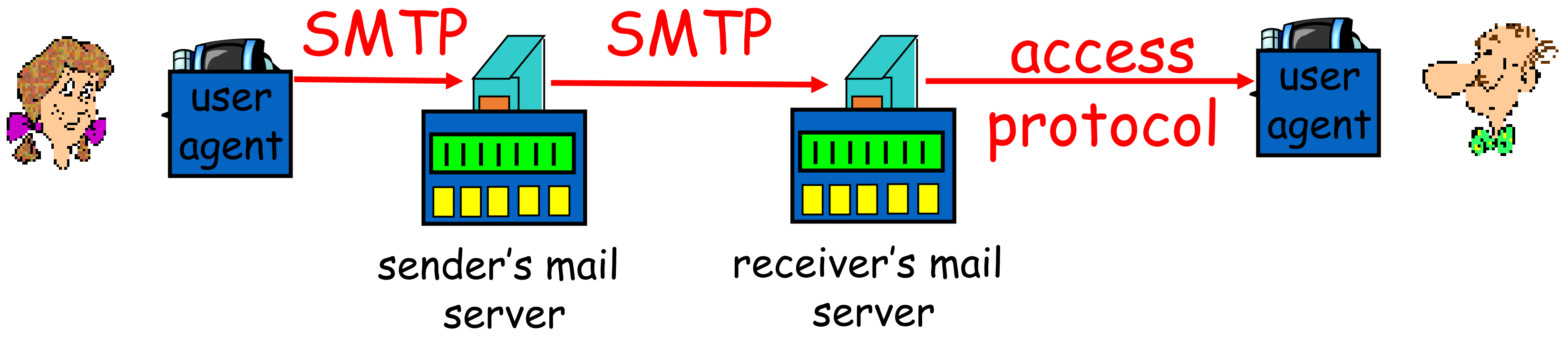
- MIME: multimedia mail extension, RFC 2045, 2056
- additional lines in msg header declare MIME content type


MIME version
method used to encode data
multimedia data type, subtype, parameter declaration
encoded data

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.........................
......base64 encoded data
```

Mail access protocols

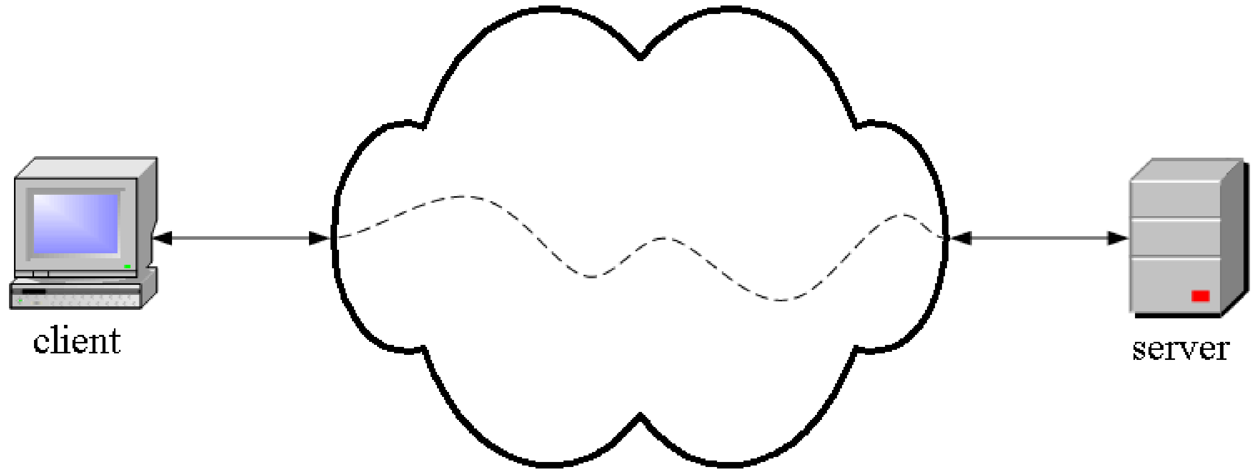


- SMTP is a push protocol
- Mail access protocol: retrieval from server
  - POP: Post Office Protocol [RFC 1939]
    - Users can't create folders on mail server
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - more features (more complex)
    - manipulation of stored msgs on server
  - HTTP: Hotmail , Yahoo! Mail, etc.

HTTP

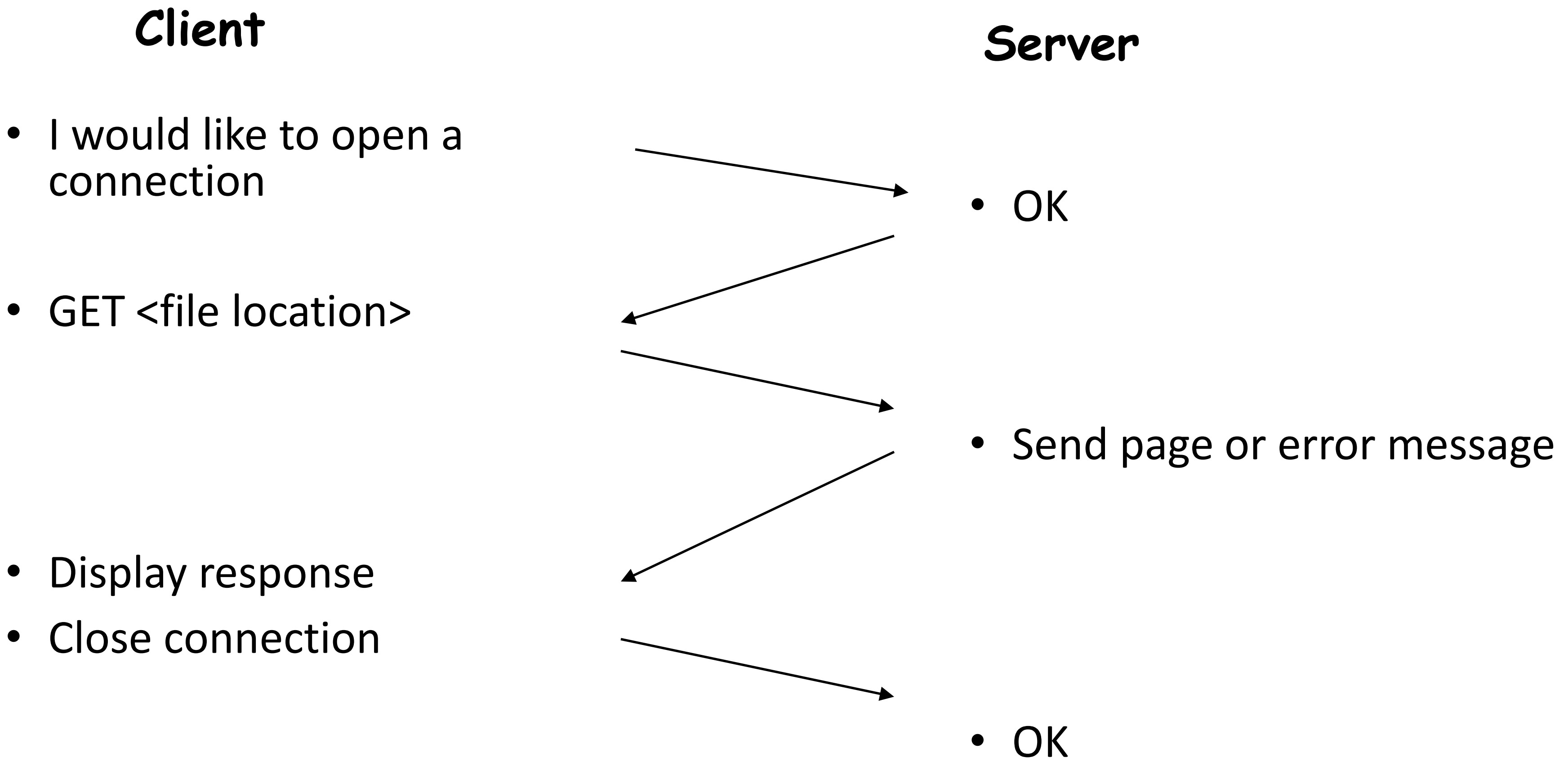
- HyperText Transfer Protocol
  - The rules governing the conversation between a Web client and a Web server
  - **HTTP** is the foundation of any data exchange on the Web. As are most application-layer protocols, HTTP is implemented in two programs: a client program: Web browser and server program: Web server that talk to each other by exchanging HTTP messages.
  - HTTP defines the structure of these messages and how the client and server exchange the messages.
  - Port 80 is the default port of HTTP.

HTTP is an application layer protocol



- The Web client and the Web server are application programs
- Application layer programs do useful work like retrieving Web pages, sending and receiving email or transferring files
- Lower layers take care of the communication details
- The client and server send messages and data without knowing anything about the communication network
- HTTP utilizes TCP connections to send client requests and server replies.

An HTTP conversation



Client

- I would like to open a connection
- GET <file location>
- Display response
- Close connection

Server

- OK
- Send page or error message
- OK

HTTP is the set of rules governing the format and content of the conversation between a Web client and server

An HTTP example

The message requesting a Web page must begin with the work "GET" and be followed by a space and the location of a file on the server, like this:

```
GET /fac/lpress/shortbio.htm
```

The protocol spells out the exact message format, so any Web client can retrieve pages from any Web server.

HTTP Message Examples

Typical Request Message From A Client:

```
GET /eecc694-spring2000/index.html HTTP/1.0
Connection: close
User-agent: Mozilla/4.72 [en] (Win98; I)
Accept: text/html, image/gif, image/jpeg
Accept-language:en
 (extra carriage return, line feed)
```

Typical Response Message From A Server:

```
HTTP/1.0 200 OK
Connection: close
Date: Wed, 05 April 2018 12:00:15 GMT
Server: NCSA/1.5.2
Last-Modified: Tue, 25 April 2018 11:23:24 GMT
Content-Length: 20419
Content-Type: text/html
and more data ...
```

Status Codes in general

1xx	Informational
2xx	Success
3xx	Redirection
4xx	Client Error
5xx	Server Error

HTTP Methods

Method	Description
GET	Request to read a Web page
HEAD	Request to read a Web page's header
PUT	Request to store a Web page
POST	Append to a named resource (e.g., a Web page)
DELETE	Remove the Web page
TRACE	Echo the incoming request
CONNECT	Reserved for future use
OPTIONS	Query certain options

TRACE method is used for debugging. It instructs the server to send back the request. This is useful when requests are not processed correctly and the client wants to know it sent the proper request.

CONNECT – is not used at the moment

OPTIONS – provides a way for the client to query the server about its properties or those of a specified file

The End