The Web

Session 4

INST 346

Technologies, Infrastructure and Architecture

Quiz

- Start: 5:00 sharp
- End: 5:05 sharp (pencil down or zero credit)
- Include your name!
- No communication with anyone till 5:05
  - No email, no talking, no SMS, no chat, …
  - Even if you finish early!
- Open book, open notes, open Web, open mind

Goals for Today

- Finish HTTP

- Getahead: Email

- Wireshark preview

HTTP request message

- two types of HTTP messages: *request, response*
- HTTP request message:
  - ASCII (human-readable format)
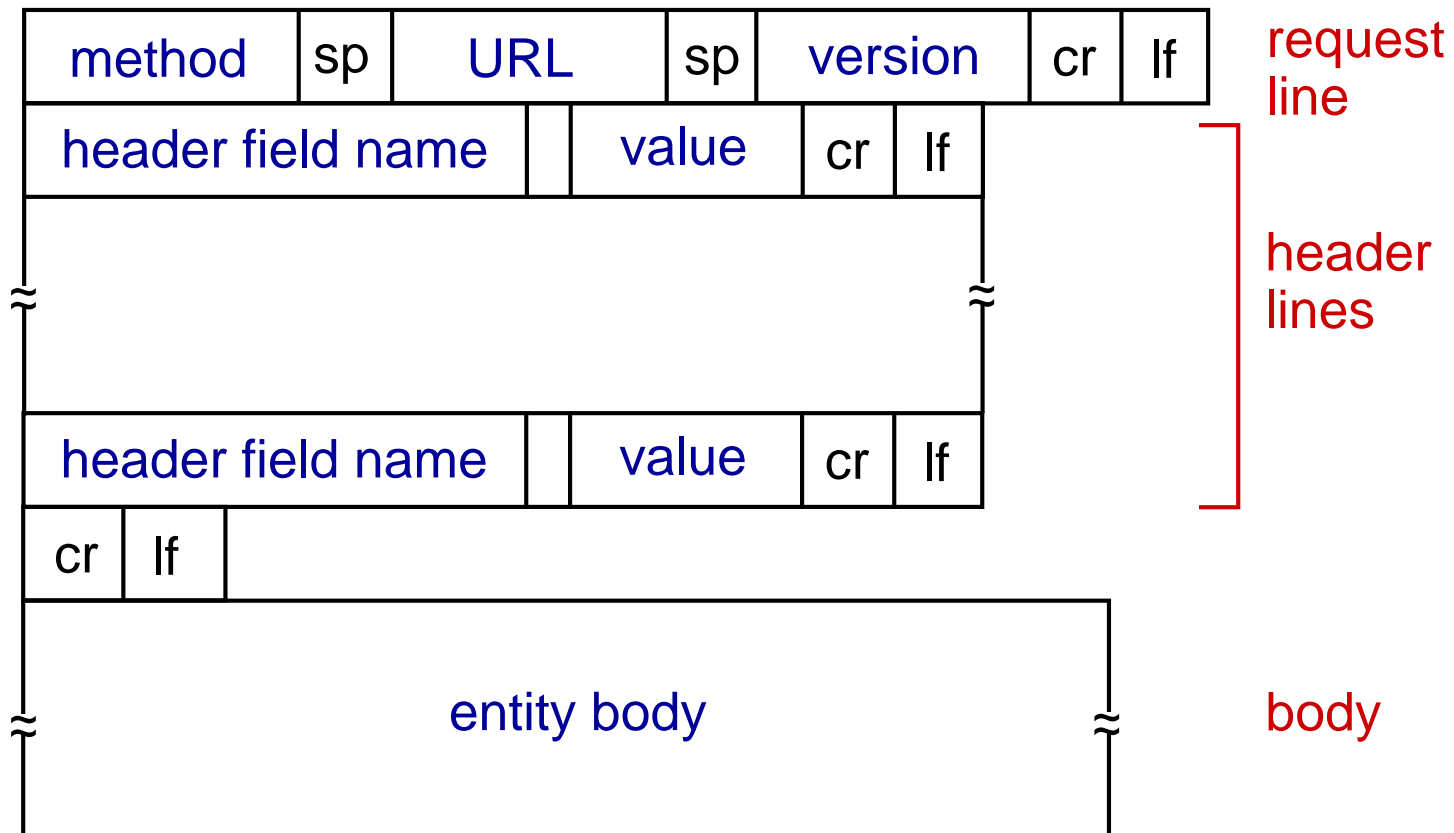
request line
(GET, POST, HEAD
commands)

carriage return character

line-feed character

```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

header
lines

carriage return, line feed
at start of line indicates
end of header lines

* Check out the online interactive exercises for more
examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

HTTP request message: general format

| method | sp | URL | sp | version | cr | lf | request line |

| header field name | | value | cr | lf |

~ ~

| header field name | | value | cr | lf |

| cr | lf |

header lines

| entity body |

body

Method types

HTTP/1.0:

- GET
- POST
  - input is uploaded to server in entity body
- HEAD
  - asks server to leave requested object out of response

HTTP/1.1:

- GET, POST, HEAD
- PUT
  - uploads file in entity body to path specified in URL field
- DELETE
  - deletes file specified in the URL field

HTTP response message

status line
(protocol
status code
status phrase)

header
lines

data, e.g.,
requested
HTML file

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02
    GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-
    1\r\n
\r\n
data data data data data ...
```

* Check out the online interactive exercises for more
examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

HTTP response status codes

- status code appears in 1st line in server-to-client response message.

- some sample codes:

200 OK
- request succeeded, requested object later in this msg

301 Moved Permanently
- requested object moved, new location specified later in this msg (Location:)

400 Bad Request
- request msg not understood by server

404 Not Found
- requested document not found on this server

505 HTTP Version Not Supported

Cookies

Client → Server:
```
GET http://www.example.com/ HTTP/1.1
```

Server → Client:
```
HTTP/1.1 200 OK
Set-Cookie: session-id=12345;
```

Client → Server:
```
GET http://www.example.com/ HTTP/1.1
Cookie: session-id=12345;
```

Cookies

what cookies can be used for:

- authorization
- shopping carts
- recommendations
- user session state (Web e-mail)

how to keep "state":

- protocol endpoints: maintain state at sender/receiver over multiple transactions
- cookies: http messages carry state

cookies and privacy:

- cookies permit sites to learn a lot about you
- you may supply name and e-mail to sites

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
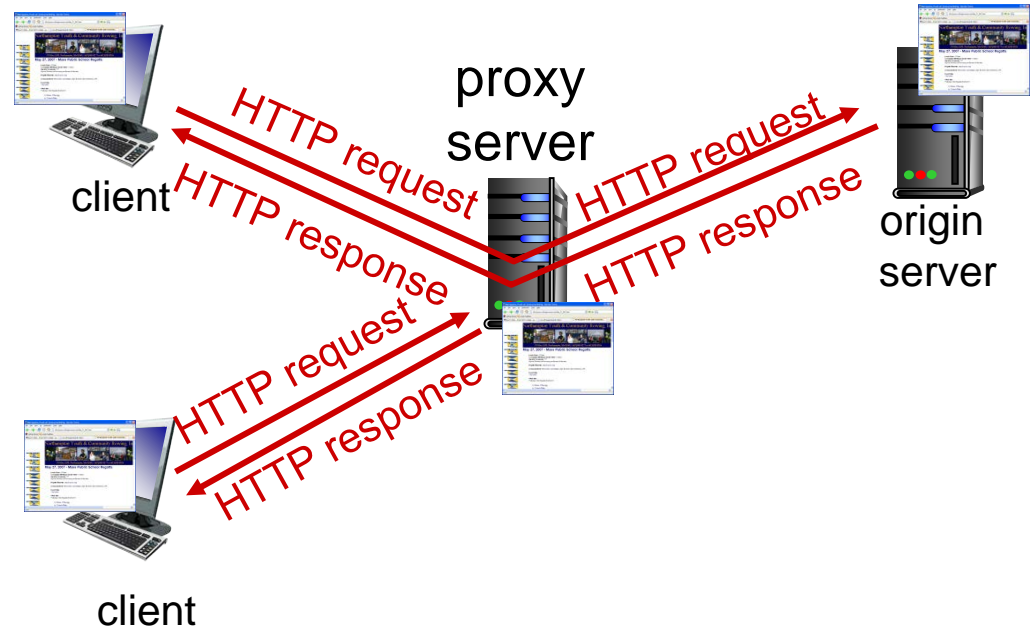
Why Web caching

- reduce response time for client request

- reduce traffic on an institution's access link

- Internet dense with caches enables low-bandwidth content providers to effectively deliver content
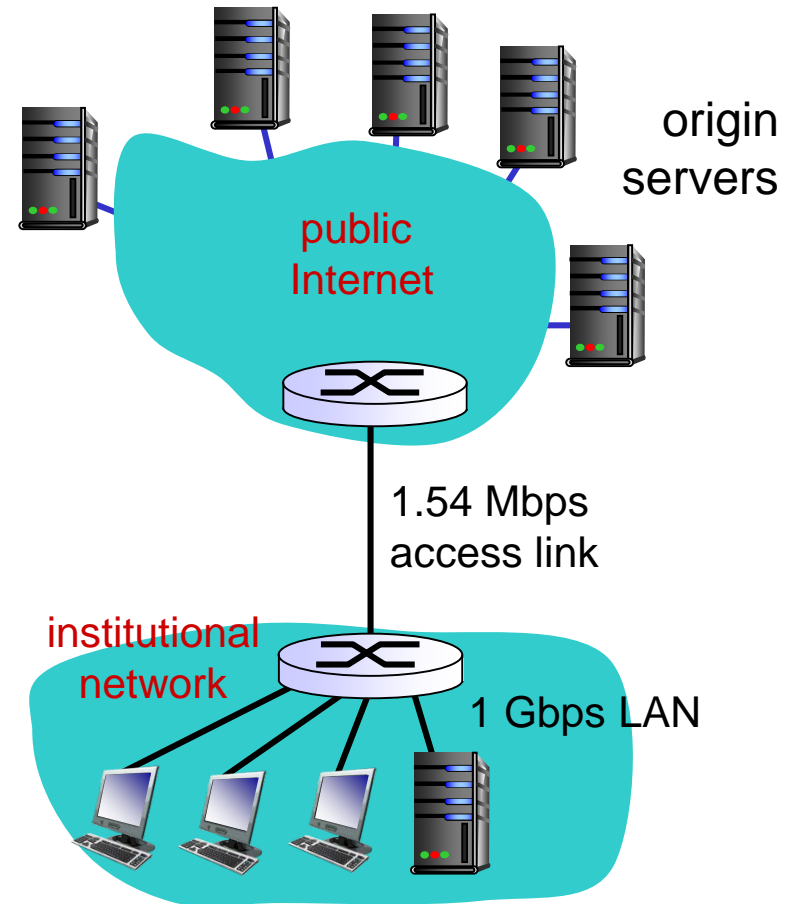
Caching example: without local cache

assumptions:
- avg object size: 100K bits
- avg request rate from browsers to origin servers:15/sec
- avg data rate to browsers: 1.50 Mbps
- RTT from institutional router to any origin server: 2 sec
- access link rate: 1.54 Mbps

consequences:
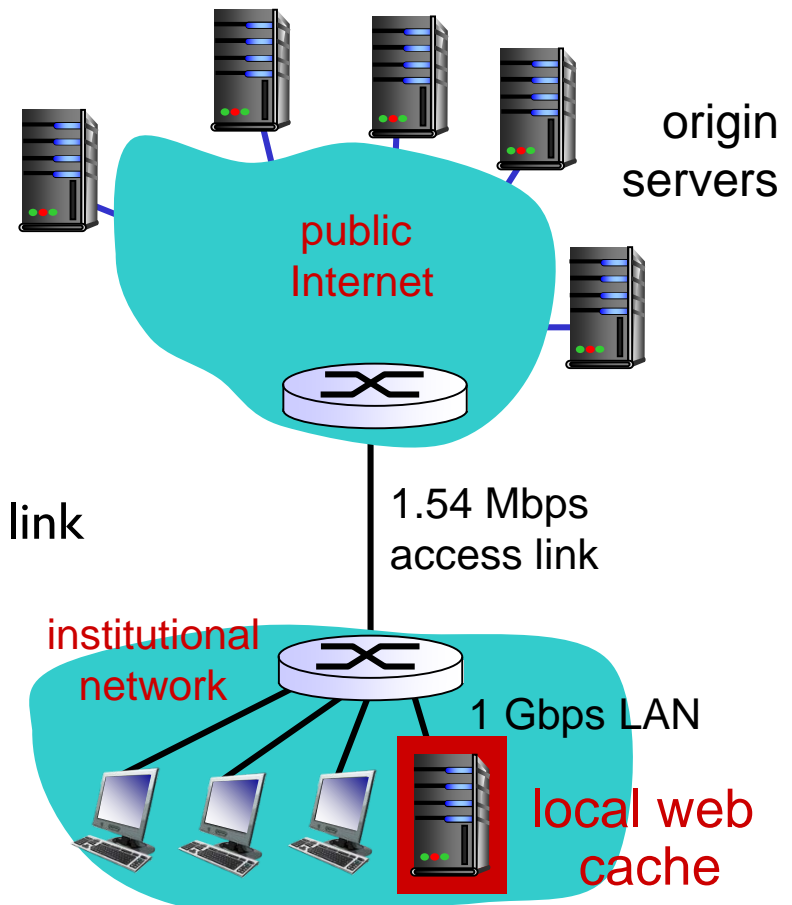- LAN utilization: 15%   *problem!*
- access link utilization = 99%
- total delay   = Internet delay + access delay + LAN delay
    = 2 sec + minutes + usecs

origin servers

public Internet

1.54 Mbps access link

institutional network

1 Gbps LAN

Caching example: install local cache

Calculating access link utilization, delay with cache:

- **suppose cache hit rate is 0.4**
  - 40% requests satisfied at cache, 60% requests satisfied at origin

- **access link utilization:**
  - 60% of requests use access link

- **data rate to browsers** over access link
  = 0.6*1.50 Mbps = .9 Mbps
  - utilization = 0.9/1.54 = .58

- **total delay**
  - = 0.6 * (delay from origin servers) +0.4 * (delay when satisfied at cache)
  - = 0.6 (2.01) + 0.4 (~msecs) = ~ 1.2 secs

origin servers

public Internet

1.54 Mbps access link

institutional network

1 Gbps LAN

local web cache

Conditional GET

client   server
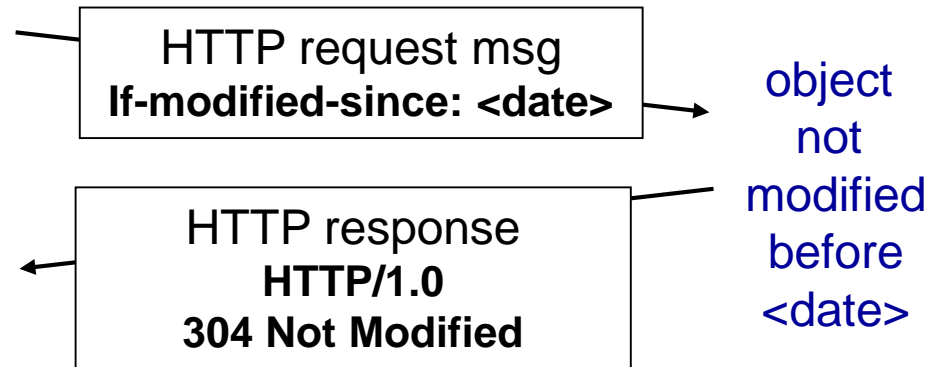
- *Goal:* don't send object if cache has up-to-date cached version
  - no object transmission delay
  - lower link utilization
- *cache:* specify date of cached copy in HTTP request

  If-modified-since: <date>

- *server:* response contains no object if cached copy is up-to-date:
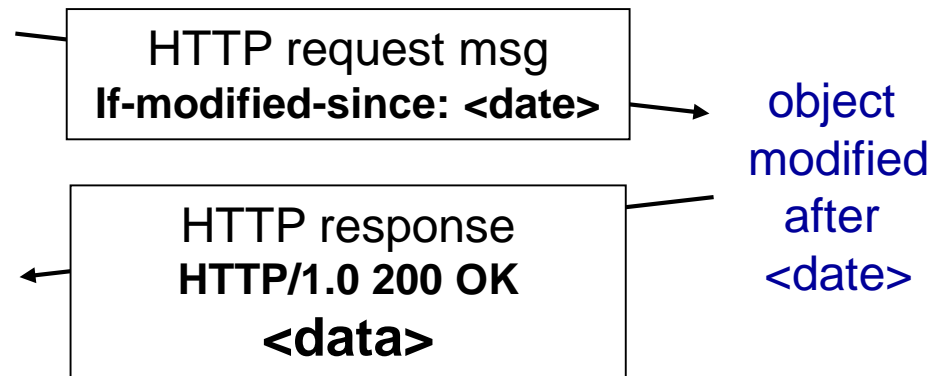
  HTTP/1.0 304 Not Modified

HTTP request msg
If-modified-since: <date>

object not modified before <date>

HTTP response
HTTP/1.0 304 Not Modified

- -

HTTP request msg
If-modified-since: <date>

object modified after <date>

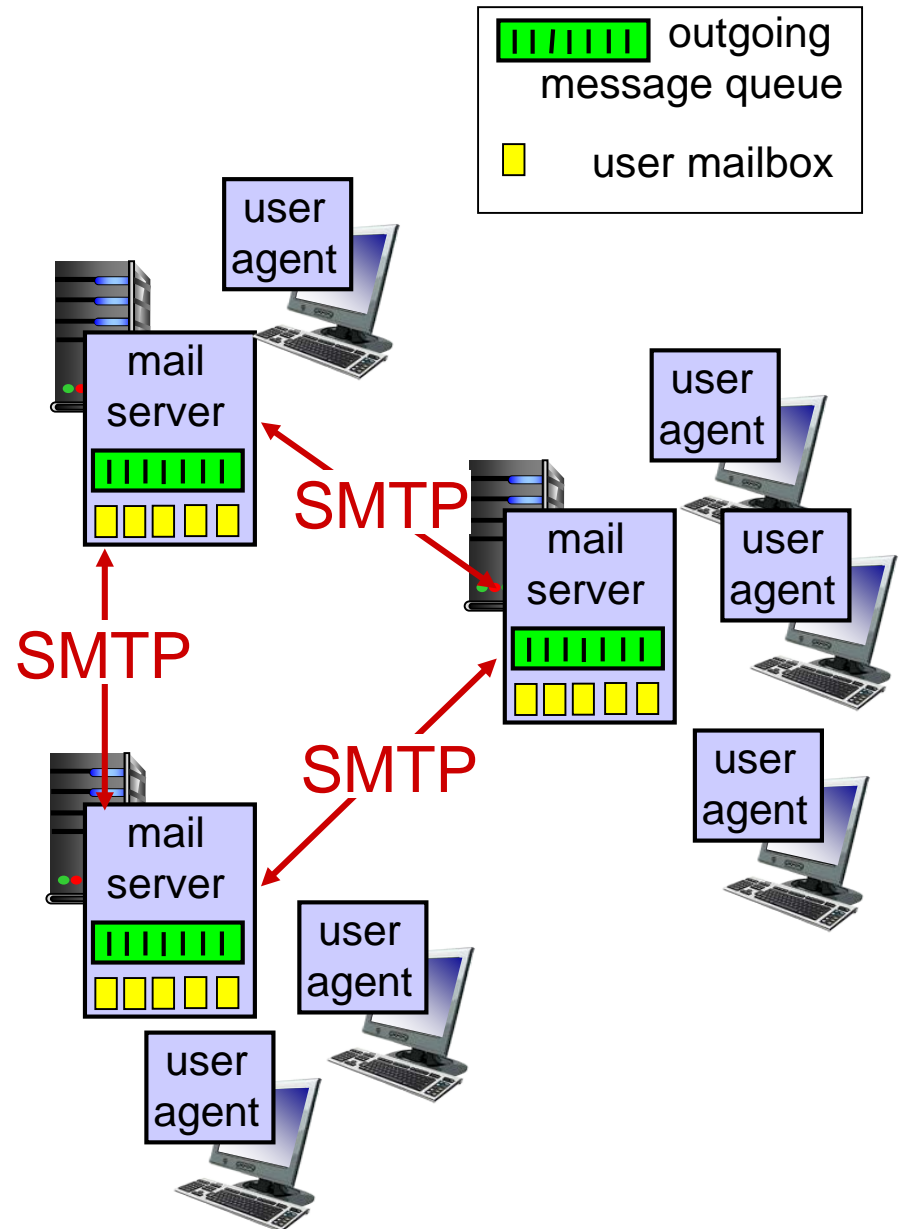HTTP response
HTTP/1.0 200 OK <data>

Getahead: Email

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, 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
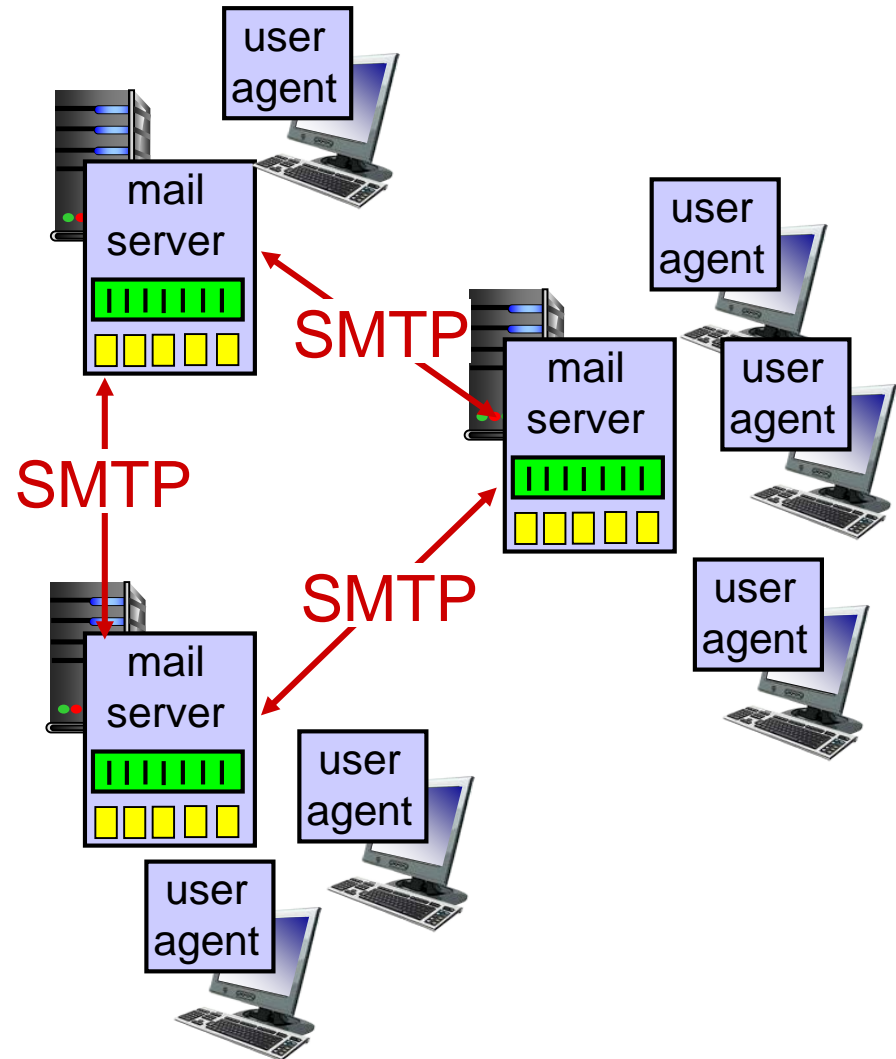
Electronic Mail: SMTP [RFC 2821]

- 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 (like HTTP)
  - commands: ASCII text
  - response: status code and phrase
- messages must be in 7-bit ASCI

Scenario: Alice sends message to Bob

1) Alice uses UA to compose message "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

Alice's mail server   Bob's mail server
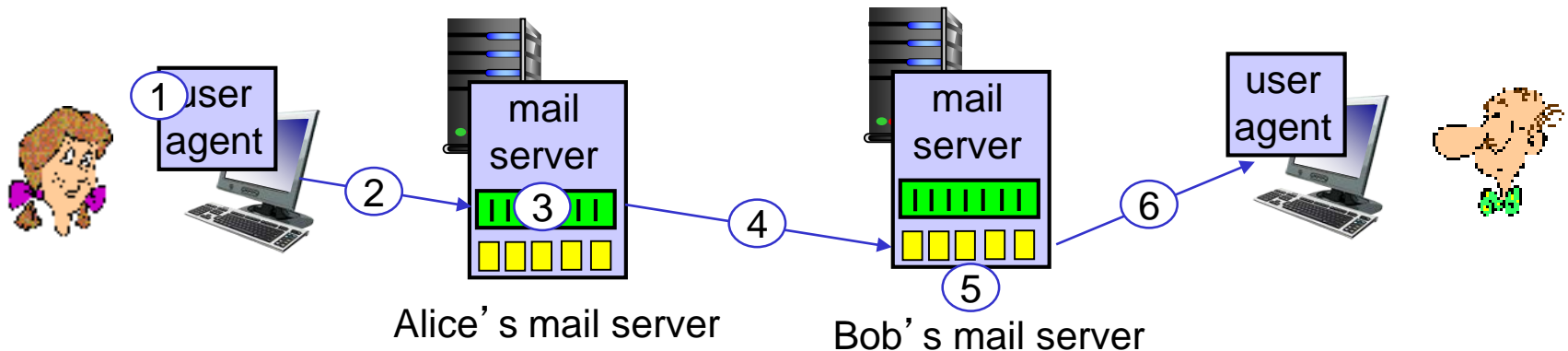
Sample SMTP interaction

```
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 message
- SMTP: multiple objects sent in multipart message

Before You Go

On a sheet of paper, answer the following (ungraded) question (no names, please):

What was the muddiest point in today's class?