Lecture 6: Application Layer

Web proxies, Email, and SMTP

COMP 332, Spring 2018

Victoria Manfredi

WESLEYAN UNIVERSITY

Acknowledgements: materials adapted from Computer Networking: A Top Down Approach 7th edition: ©1996-2016, J.F Kurose and K.W. Ross, All Rights Reserved as well as from slides by Abraham Matta at Boston University and some material from Computer Networks by Tannenbaum and Wetherall.

Today

Announcements

- homework 2 due today, homework 3 posted
- Q: how do we run distributed tic-tac-toe game on different hosts?

Web and HTTP

- web caching
  - homework 3 and 4 will implement a version of this

Electronic mail

- SMTP
  - sending mail and communicating between mail servers
- mail access protocols
  - downloading mail from server
- testing it out

Web and HTTP

CACHING

Web caches (proxy server)

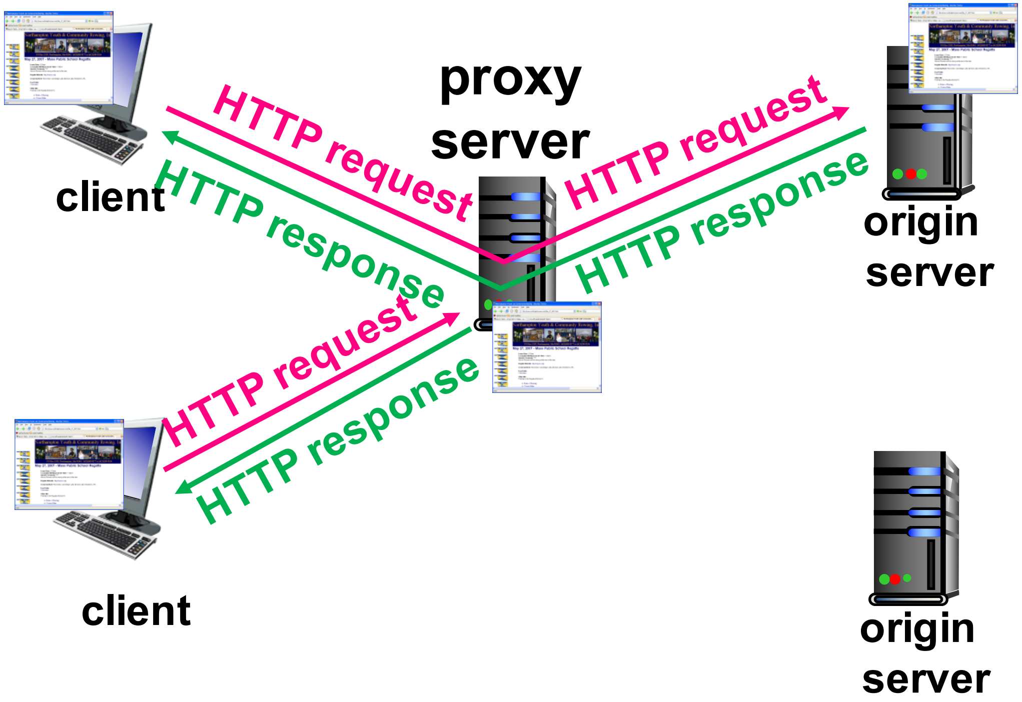
Goal: satisfy client request without (really) involving origin server

User sets browser

- perform web accesses via cache

Browser sends all HTTP requests to cache

- if object in cache
  - cache returns object
- else
  - cache requests object from origin server, then returns object to client



More about Web caching

Cache acts as both a client and server

- server for original requesting client
- client to origin server

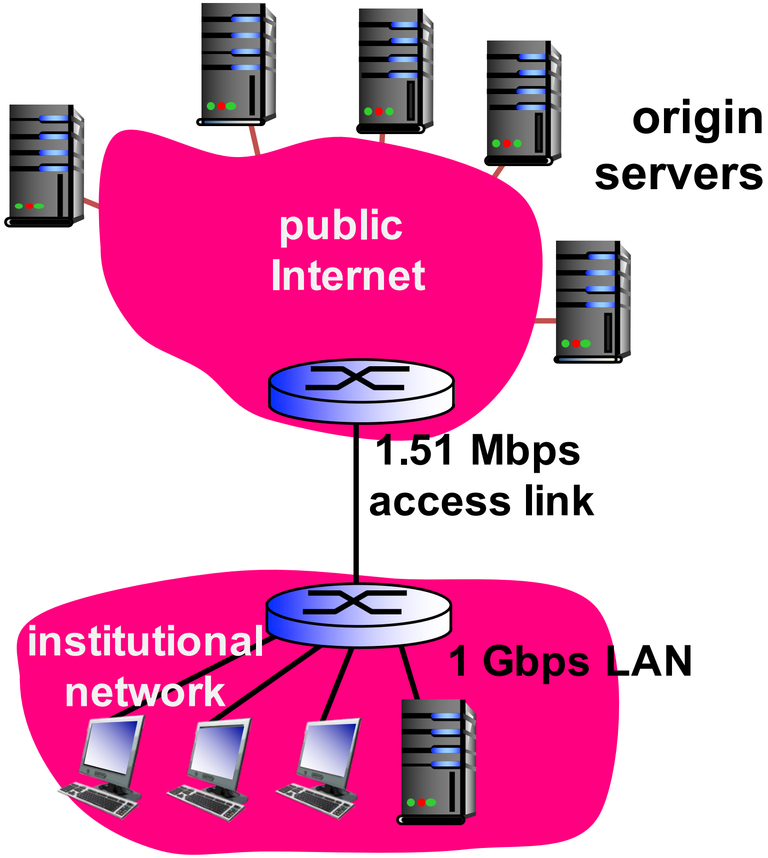
Typically cache is installed by ISP

- university, company, residential ISP

Q: why use web caching?

- reduce response time for client request
- reduce traffic on institution’s access link
- reduce load on origin servers
- Internet dense with caches
  - enables “poor” content providers to effectively deliver content
    - so too does P2P file sharing

Caching example

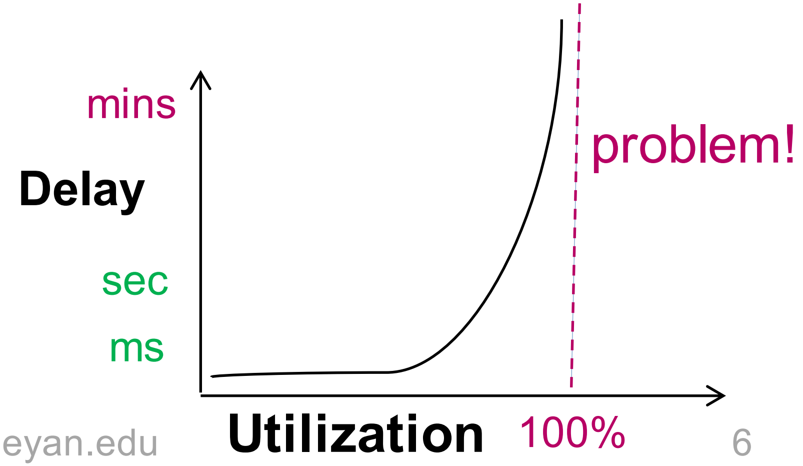


Assumptions

- avg object size: 100 Kbits
- avg request rate from browsers to origin servers: 15 requests / sec
- avg data rate to browsers: 1.50 Mbps
- RTT from institutional router to any origin server: 2 sec
- access link rate: 1.51 Mbps

Consequences

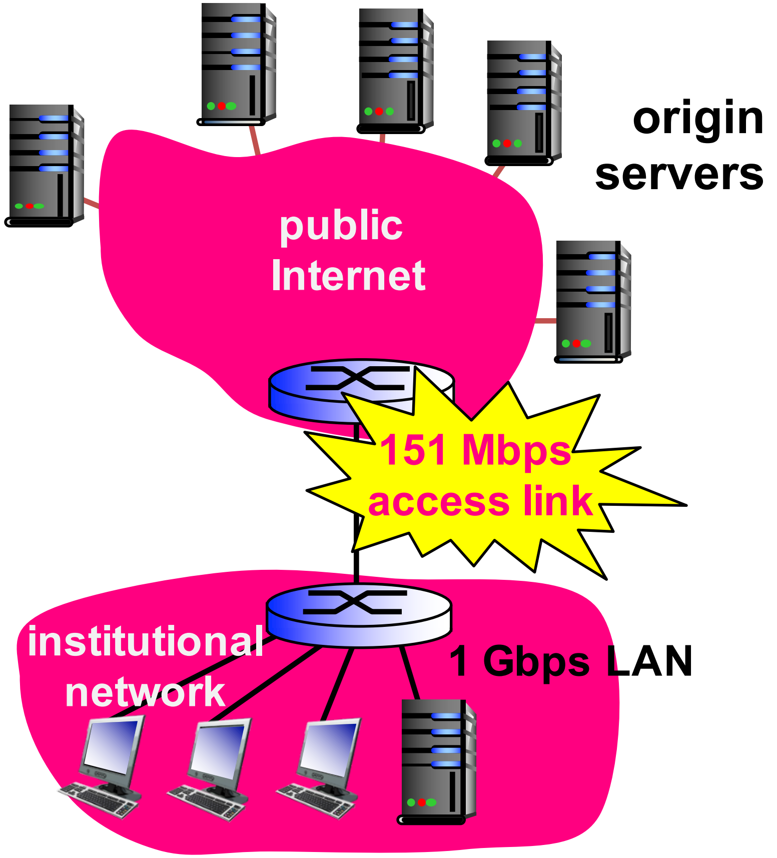
- LAN utilization: 1.5Mbps/1Gbps = 0.15%
- assume LAN delay: ~ µsec
- access link utilization: 1.50/1.51 = 99%



Total delay

= LAN delay + access delay + Internet delay
= µsec + minutes + 2 sec

Increase access link rate



Assumptions

- avg object size: 100 Kbits
- 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: 151 Mbps

Consequences

- LAN utilization: 1.5Mbps/1Gbps = 0.15%
- assume LAN delay: ~ µsec
- access link utilization: 1.50/1510 = 0.99%

Total delay

= LAN delay + access delay + Internet delay

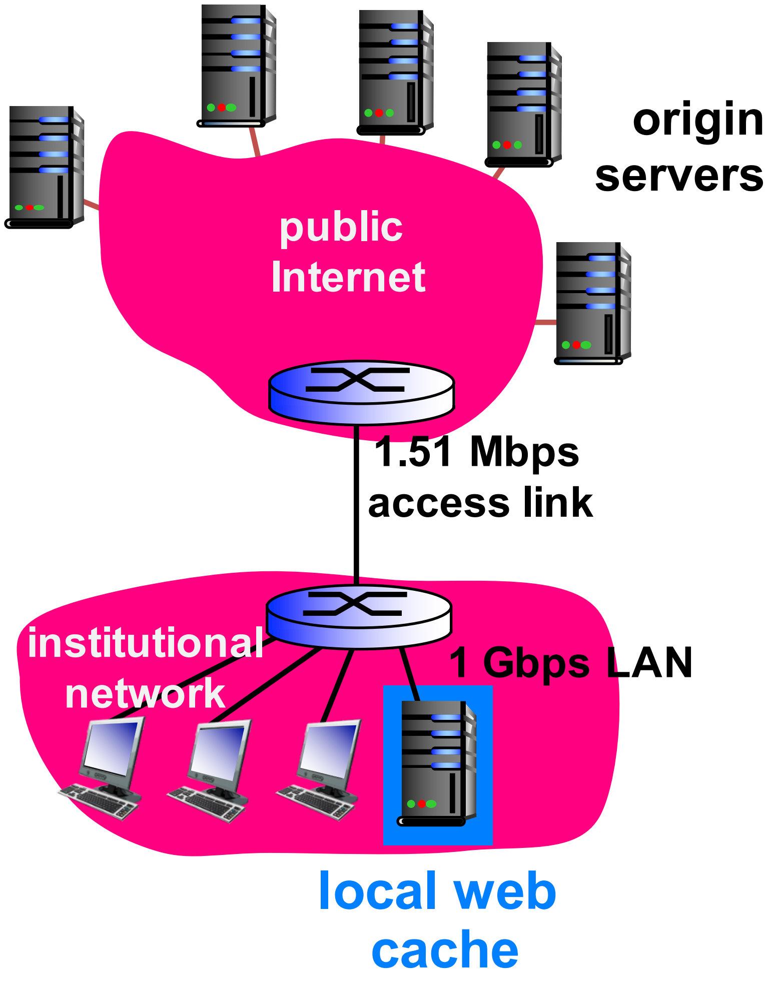
= µsec + msec + 2 sec

But, increasing access link rate is expensive!

Install local cache

Assumptions

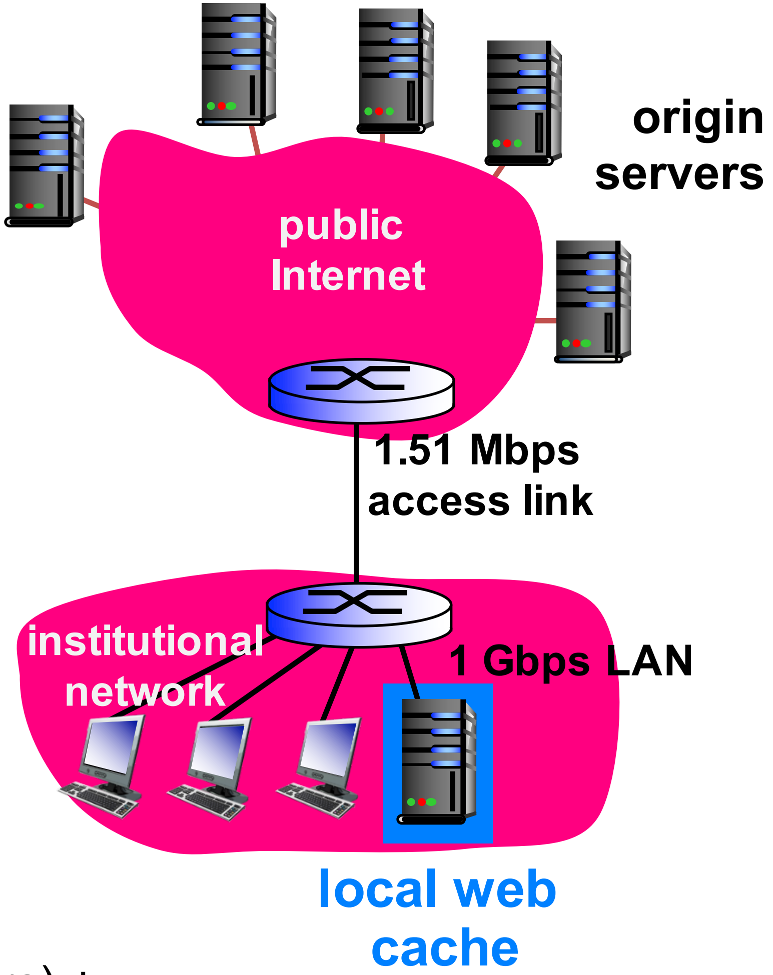
- avg object size: 100 Kbits
- avg request rate from browsers to origin servers: 15 / sec
- avg data rate to browsers: 1.50 Mbps
- RTT from institutional router to any origin server: 2 sec
- access link rate: 1.51 Mbps



How to compute access link utilization and delay?

Web cache is cheap!

Install local cache



Access link utilization, delay with cache

- suppose cache hit rate is: 0.4
  - 40% requests satisfied at cache
  - 60% requests satisfied at origin server
- 60% of requests use access link
  - data rate to browsers over access link
    - 0.6 x 1.50 Mbps = 0.9 Mbps
  - access link utilization
    - 0.9 Mbps /1.51 Mbps = 60%
  - assume access delay: ~700 msec

Total delay

= 0.6 x (delay when satisfied by origin servers) +
0.4 x (delay when satisfied by cache)

= 0.6 x (LAN delay + access delay + Internet delay) +
0.4 x (LAN delay)

= 0.6 (μsec + 700 msec + 2 sec) + 0.4 (μsec)

= 0.6 (2.7 sec) + 0.4 (μsec) = ~1.6 sec

Lower delay than with 151 Mbps link and cheaper too!

Conditional GET

Goal

- don't send object if cache has up-to-date version
- no object transmission delay
- lower link utilization

Cache

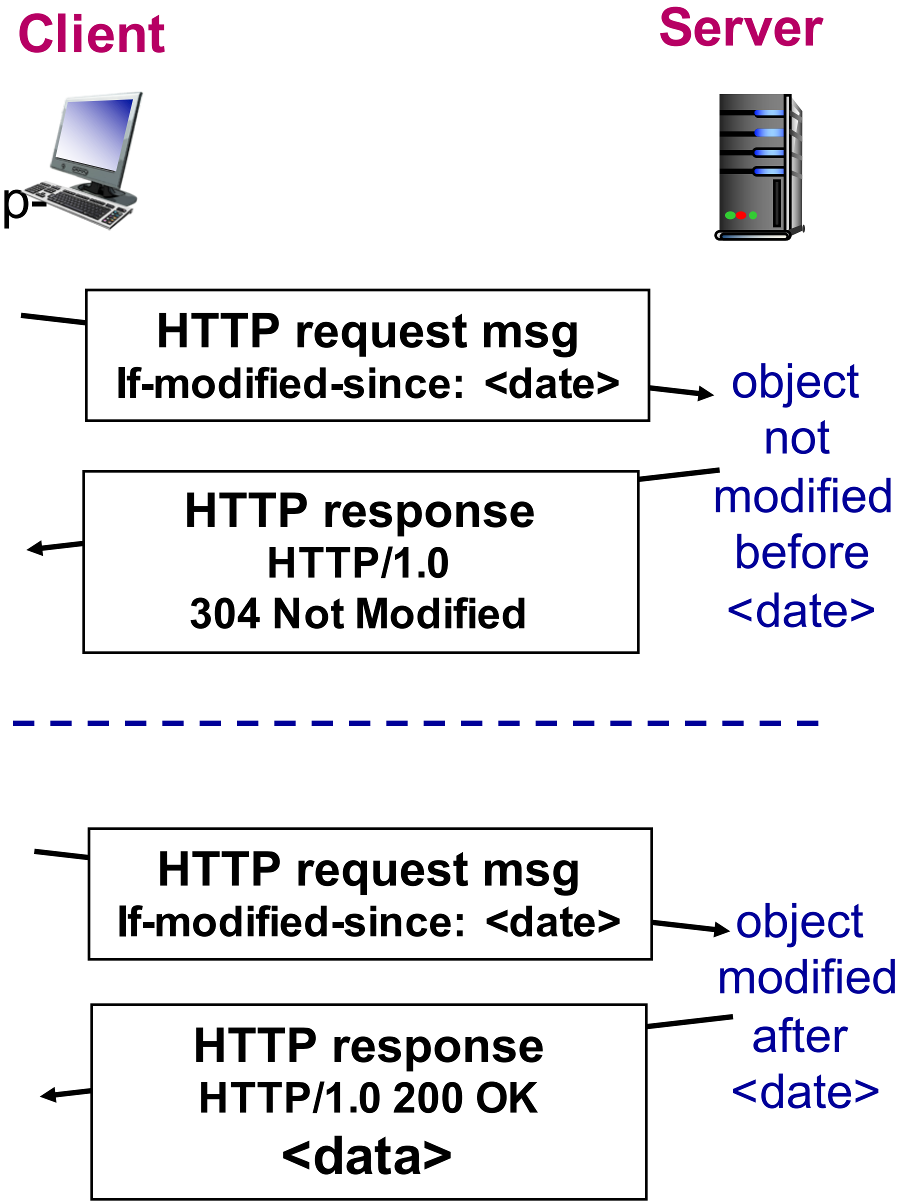
- specify date of cached copy in HTTP request

`If-modified-since:<date>`

Server

- response contains no object if cached copy is up-to-date:

`HTTP/1.0 304 Not Modified`



Client

Server

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>

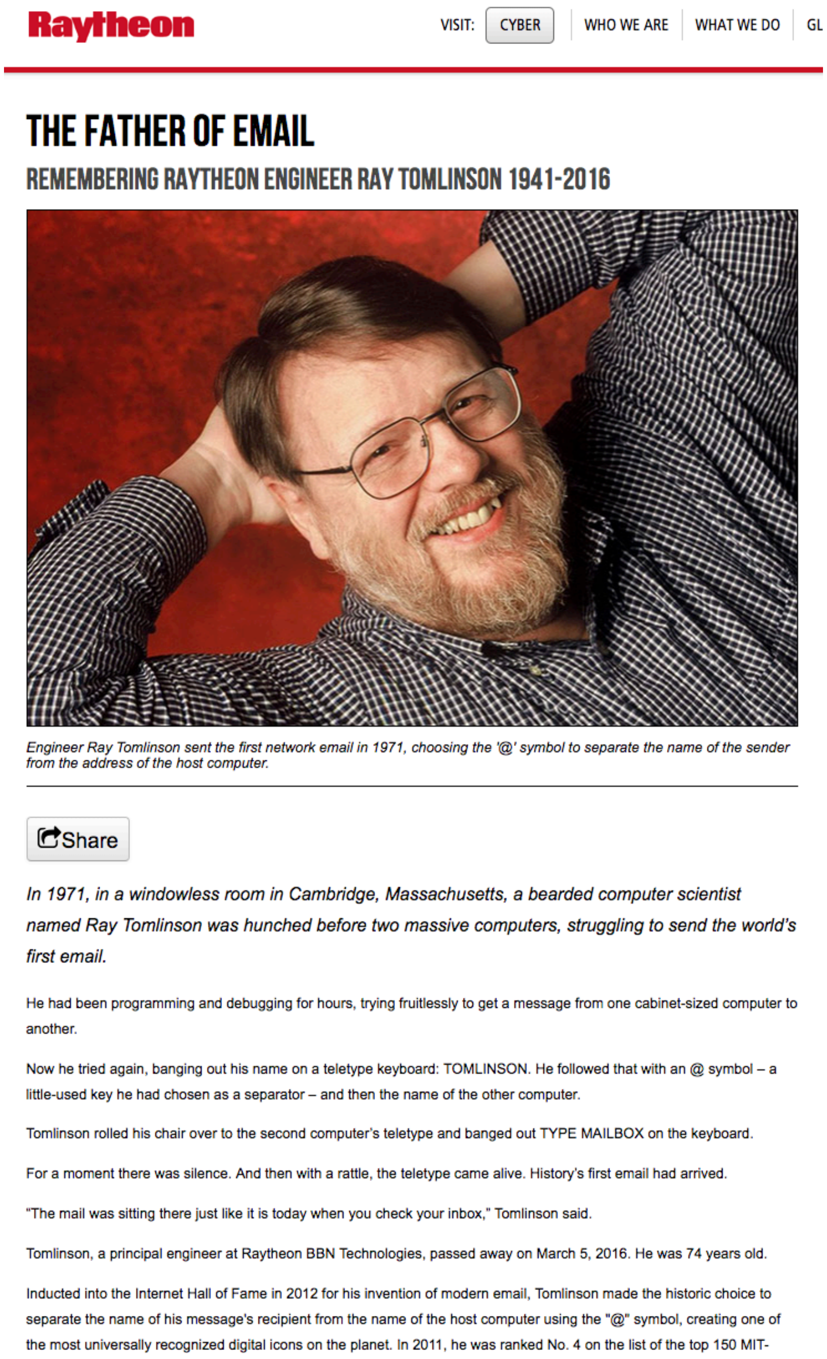
HTTP response
HTTP/1.0 200 OK
<data>

Electronic Mail

COMPONENTS

Inventor of Email

Ray Tomlinson at Raytheon BBN Technologies



THE FATHER OF EMAIL

REMEMBERING RAYTHEON ENGINEER RAY TOMLINSON 1941-2016

Engineer Ray Tomlinson sent the first network email in 1971, choosing the '@' symbol to separate the name of the sender from the address of the host computer.

In 1971, in a windowless room in Cambridge, Massachusetts, a bearded computer scientist named Ray Tomlinson was hunched before two massive computers, struggling to send the world's first email.

He had been programming and debugging for hours, trying fruitlessly to get a message from one cabinet-sized computer to another.

Now he tried again, banging out his name on a teletype keyboard: TOMLINSON. He followed that with an @ symbol – a little-used key he had chosen as a separator – and then the name of the other computer.

Tomlinson rolled his chair over to the second computer's teletype and banged out TYPE MAILBOX on the keyboard.

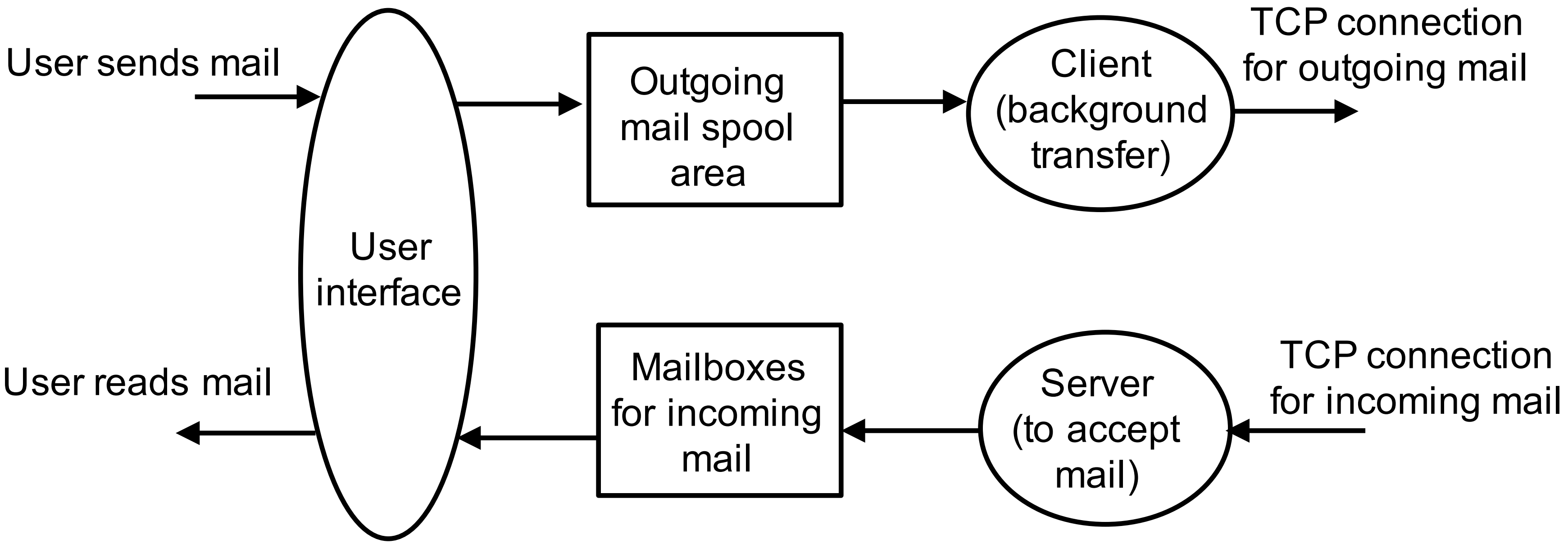
For a moment there was silence. And then with a rattle, the teletype came alive. History's first email had arrived.

"The mail was sitting there just like it is today when you check your inbox," Tomlinson said.

Tomlinson, a principal engineer at Raytheon BBN Technologies, passed away on March 5, 2016. He was 74 years old.

Inducted into the Internet Hall of Fame in 2012 for his invention of modern email, Tomlinson made the historic choice to separate the name of his message's recipient from the name of the host computer using the "@" symbol, creating one of the most universally recognized digital icons on the planet. In 2011, he was ranked No. 4 on the list of the top 150 MIT-

Overview


User sends mail
User interface
User reads mail
Outgoing mail spool area
Client (background transfer)
TCP connection for outgoing mail
Mailboxes for incoming mail
Server (to accept mail)
TCP connection for incoming mail

Uses client-server communication

– not interactive: transfer of msgs occurs in background ("spooling"")

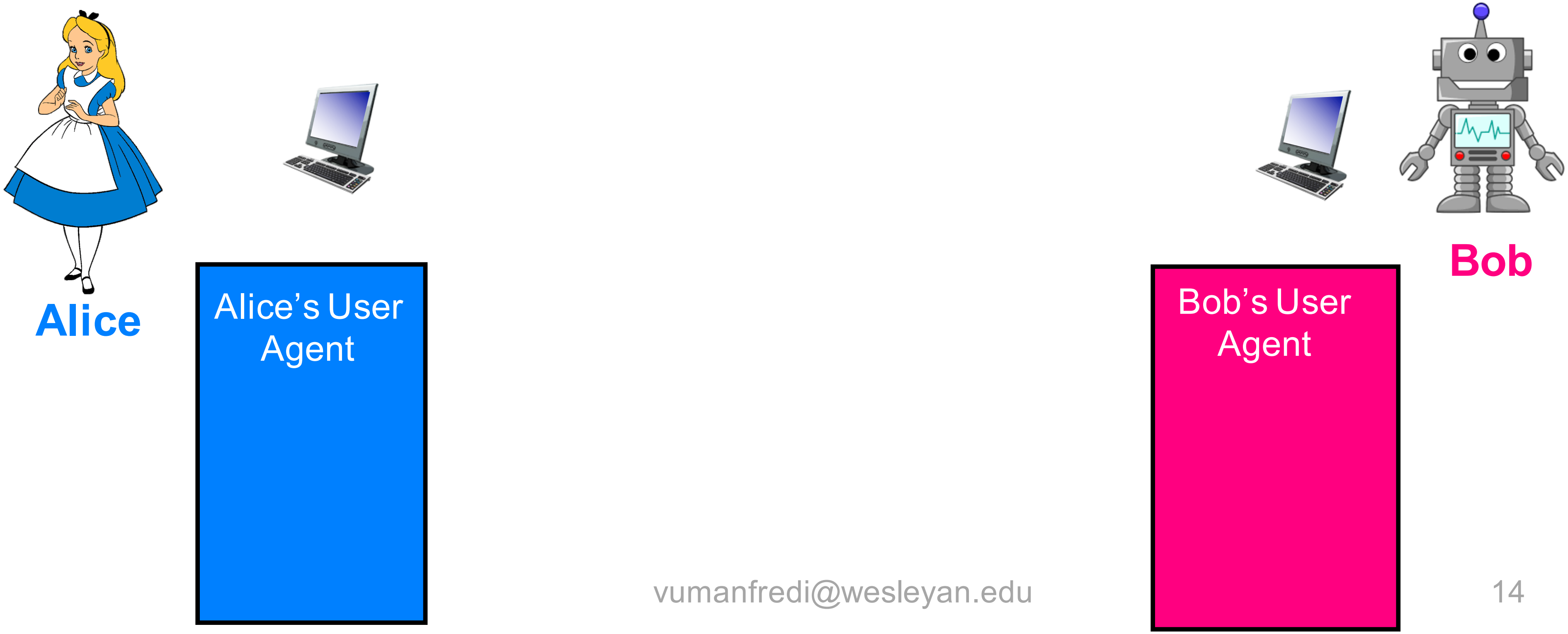
Reliable service

– uses TCP: server port 25

Major components of electronic mail (email)

User-agents

- aka mail reader (what you use)
- composing, editing, reading mail messages
- e.g., Outlook, Thunderbird, iPhone mail client, Gmail
- incoming/outgoing messages stored on mail server
  - client-server communication with mail server



Alice

Alice’s User Agent

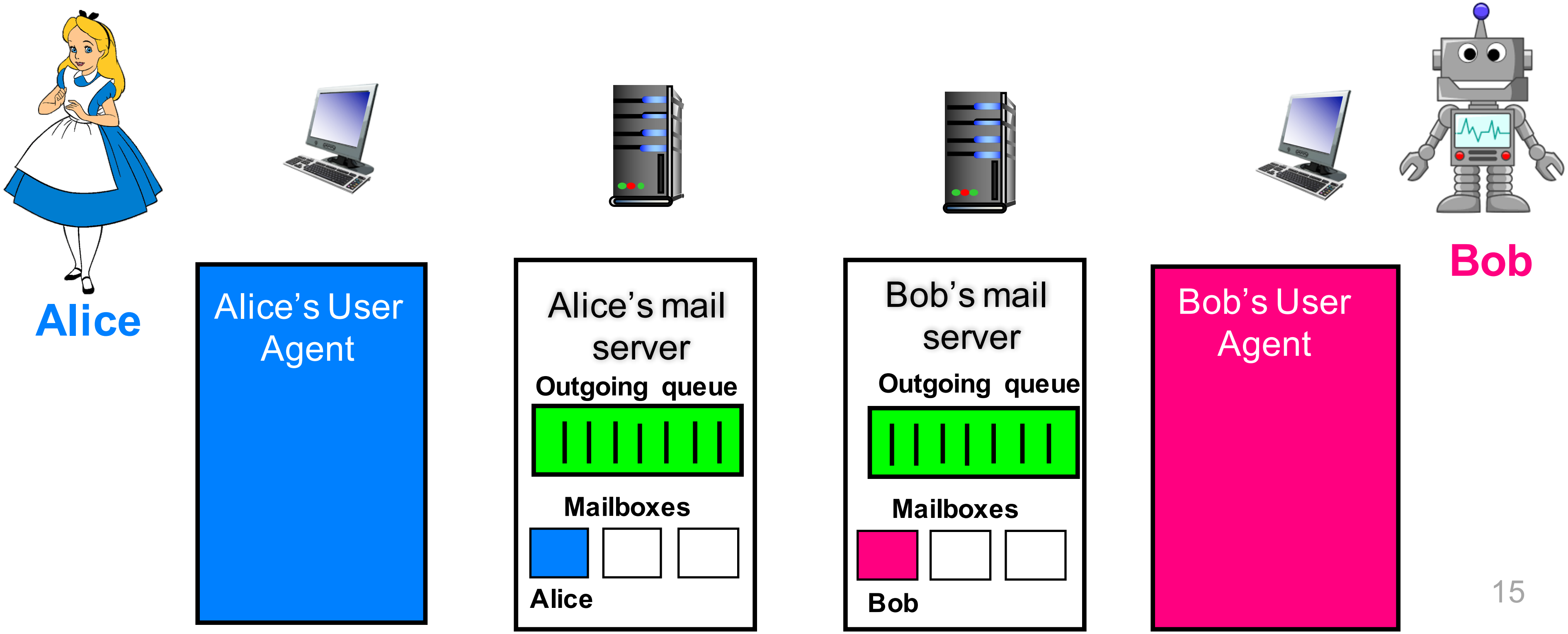
Bob’s User Agent

Bob

Major components of electronic mail (email)

Mail servers

- mailbox for each user: holds user's incoming messages
- outgoing message queue: holds messages to be sent
  - messages held in queue until successfully delivered
  - reattempts done every 30 min or so. If undeliverable, user notified



Alice

Alice's User Agent

Alice's mail server

Outgoing queue

Mailboxes

Alice

Bob's mail server

Outgoing queue

Mailboxes

Bob

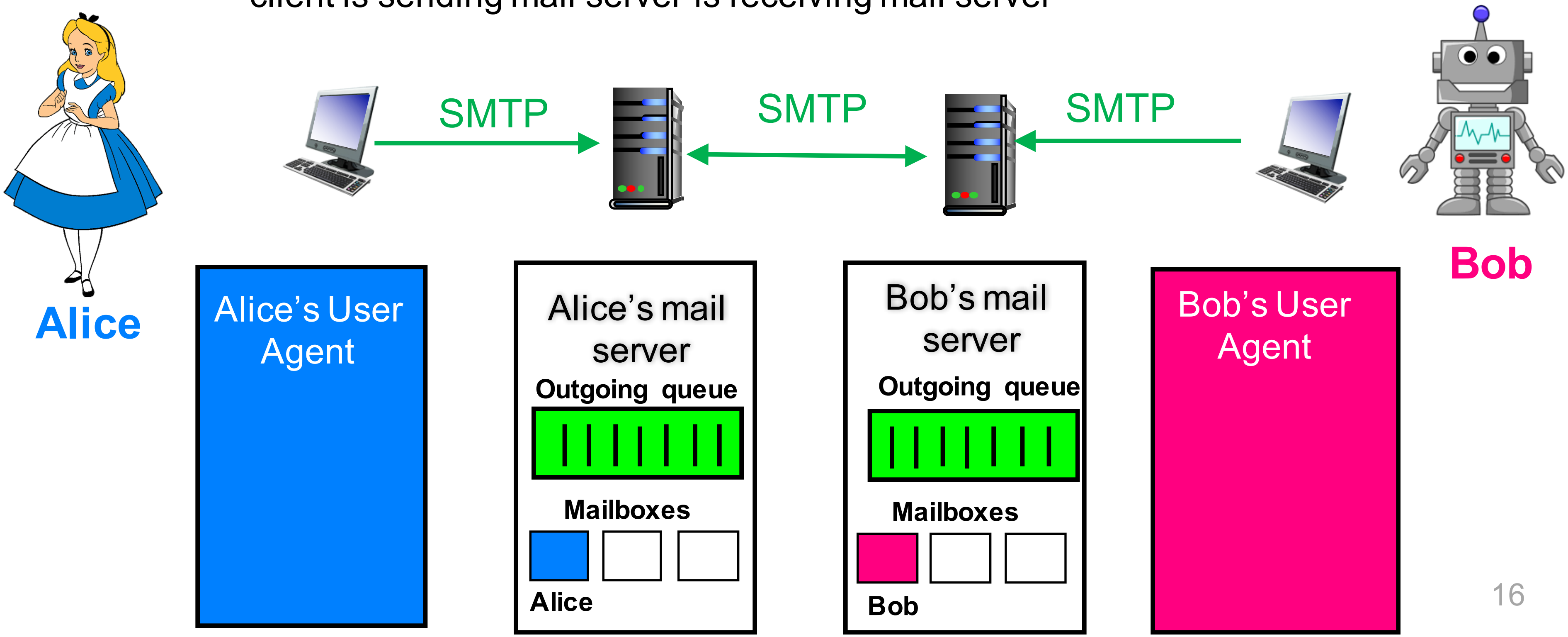
Bob's User Agent

Bob

Major components of electronic mail (email)

SMTP (simple mail transfer protocol)

- transfers messages from user agent to mail server and between mail servers
- persistent connection, TCP port 25, SSL encrypted uses port 465
- p2p comm among mail servers, client-server with user-agents
  - user agent does not run server side of SMTP (would need to always be on)
  - mail server runs both client and server sides
  - client is sending mail server is receiving mail server



SMTP

SMTP

SMTP

Alice

Bob

Alice's User Agent

Alice's mail server

Outgoing queue

Mailboxes

Alice

Bob's mail server

Outgoing queue

Mailboxes

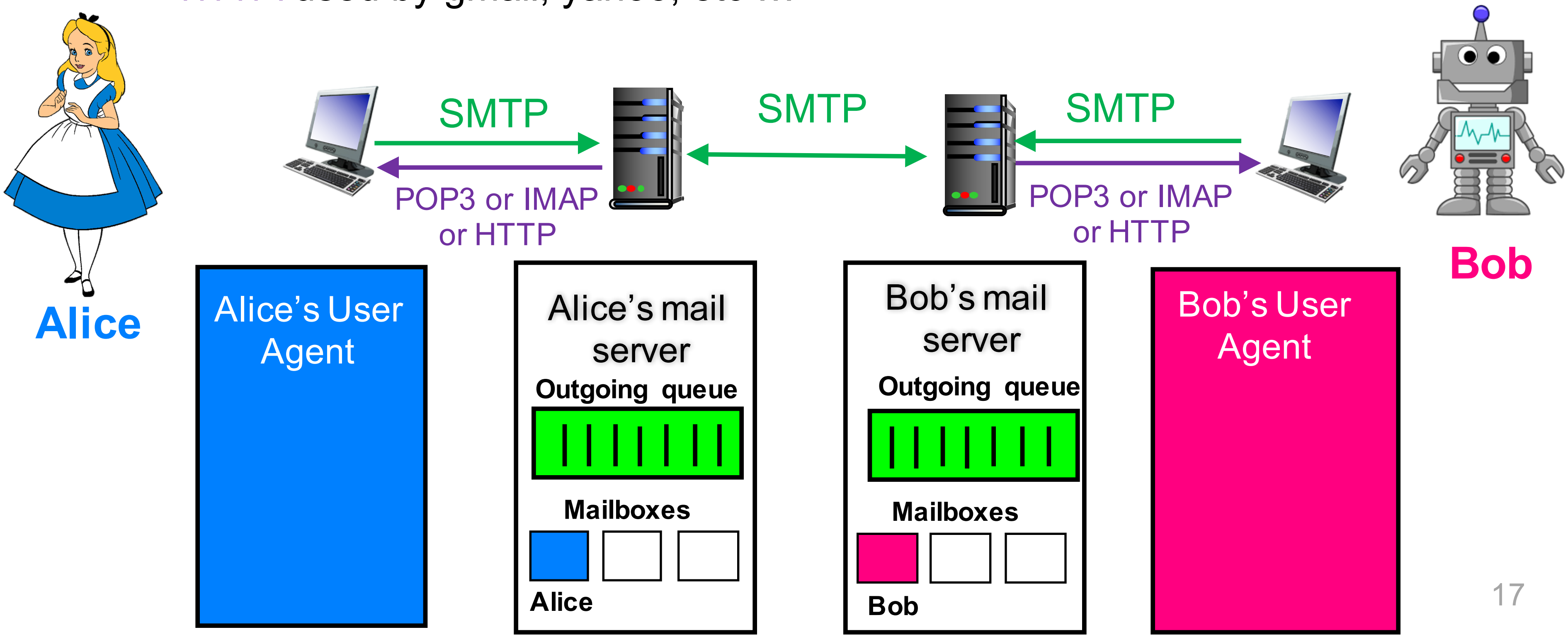
Bob

Bob's User Agent

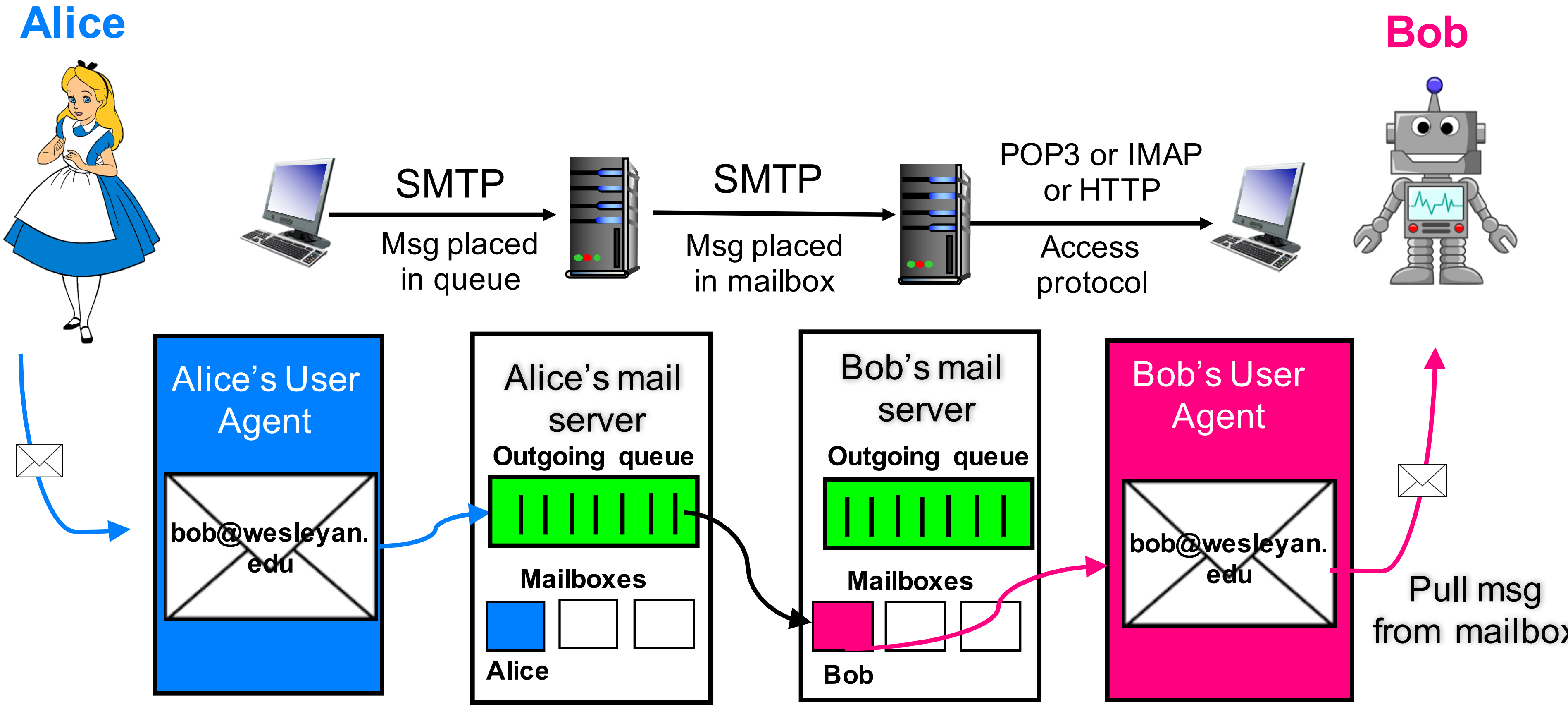
Major components of electronic mail (email)

Mail access protocols for user agent to retrieve mail

- POP3: Post Office Protocol
  - basic: downloads email, deletes from server, emails stored on computer
- IMAP: Internet Mail Access Protocol
  - more complex, recommended over POP3
    - manipulate msgs stored on server, email stored on server, use multiple computers
- HTTP: used by gmail, yahoo, etc …



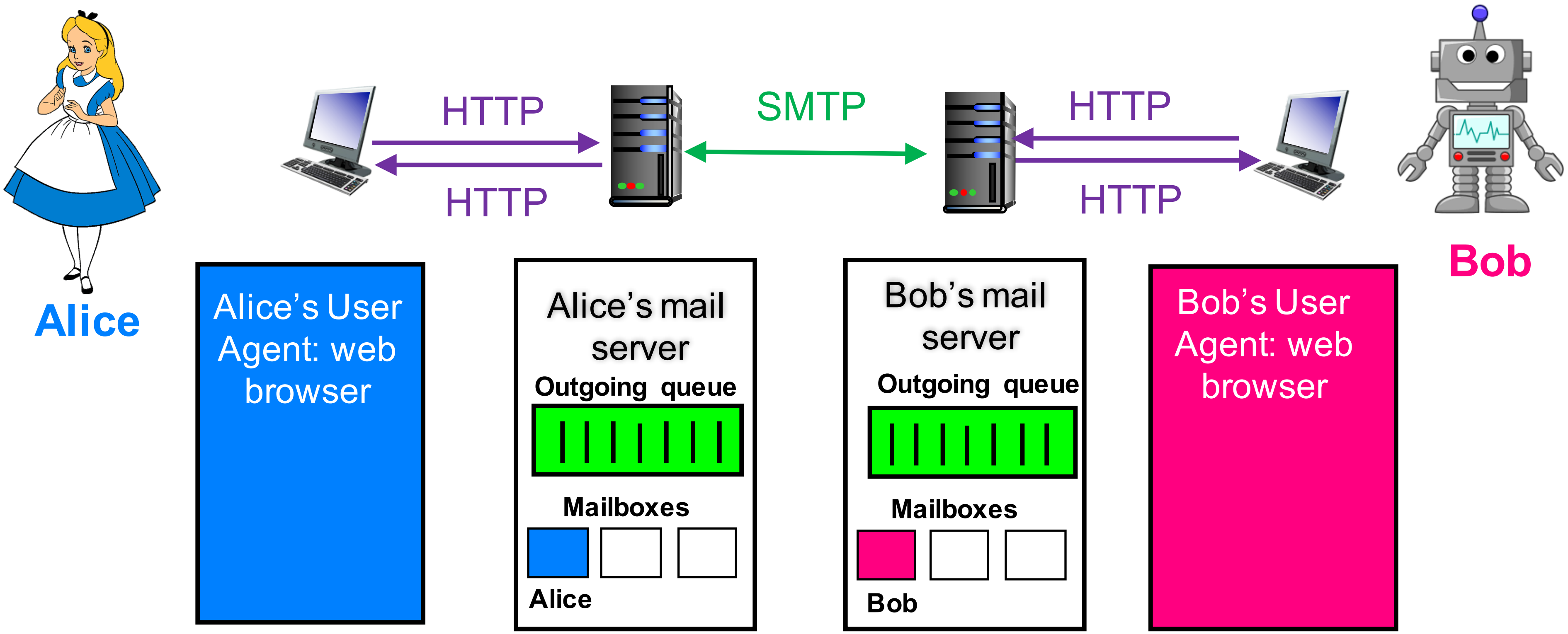
What happens when Alice sends email to Bob?



Q: What happens before any mail protocol communication?

TCP handshake

Webmail



Alice's User Agent: web browser

Alice's mail server

Outgoing queue

Mailboxes

Alice

Bob's mail server

Outgoing queue

Mailboxes

Bob

Bob's User Agent: web browser

HTTP is used for communication between Client and mail server

SMTP is used for communication between mail servers

Electronic Mail

SMTP

SMTP [RFC 2821]

Simple Mail Transfer Protocol

- defines exchange of mail from client to server and between servers
- uses TCP: to reliably transfer email message from client to server

Direct transfer

- sending server to receiving server
- 3 phases of transfer
  - handshaking (greeting)
  - transfer of messages
  - closure

Command/response interaction (like HTTP)

- commands: ASCII text
- response: status code and phrase

Testing out SMTP

Logon to an SMTP server

- use nc or telnet to open insecure connection
  - nc exchange2010.wesleyan.edu 25
- use opensssl to open secure connection
  - openssl s_client -crlf -connect exchange2010.wesleyan.edu:465
  - Aside
    - can use openssl to connect to https sites as well:
    - openssl s_client -crlf -connect www.bankofamerica.com:443

See 220 reply from server

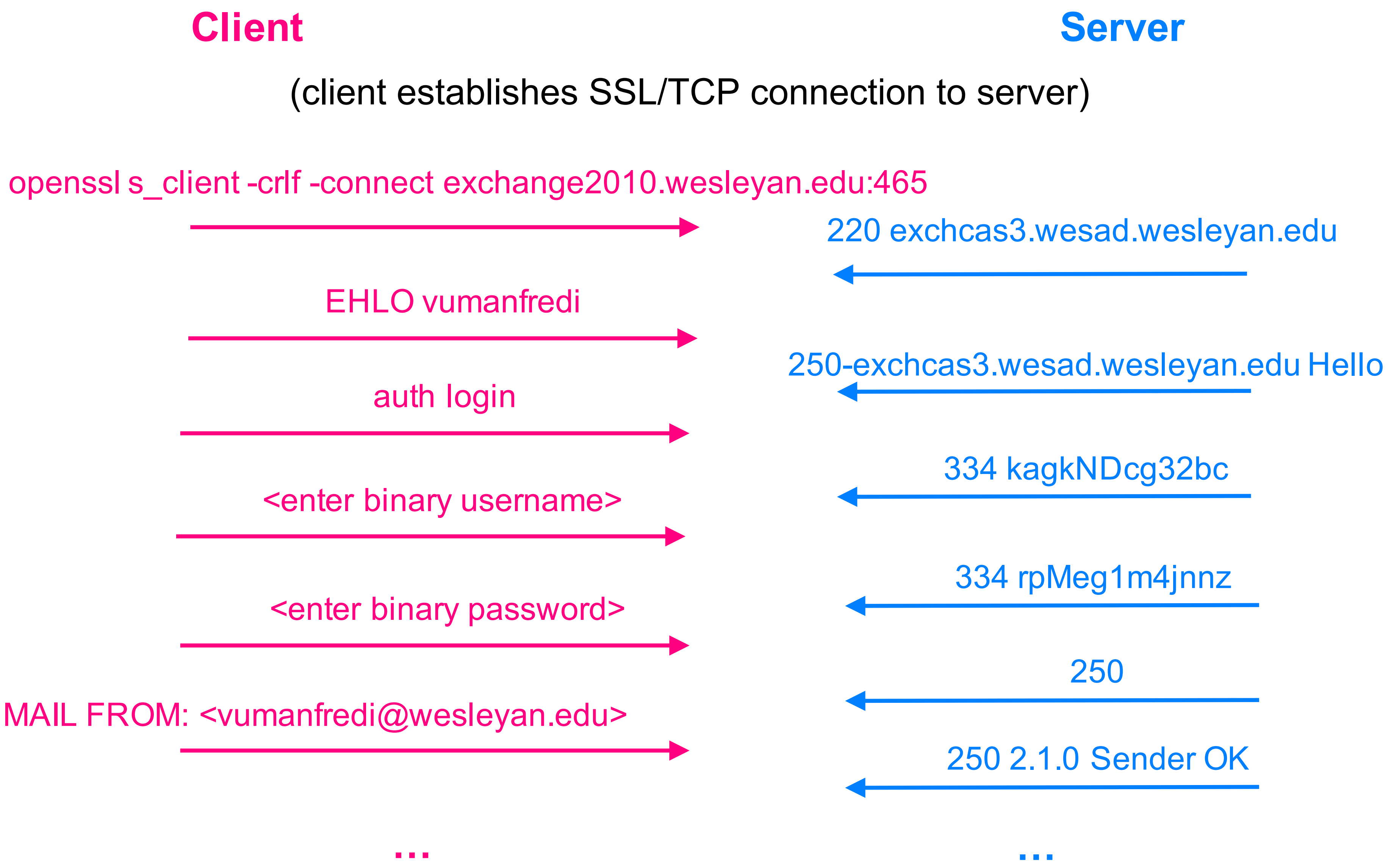
- enter HELO, MAIL FROM, RCPT TO, DATA, QUIT commands
- above lets you send email without using email client (reader)
  - you're directly logged onto mail server

Sample SMTP interaction once logged on

```
C: nc hamburger.edu 25
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
```

To really try this in practice, we need to encrypt…

SMTP client-server commands



Client

Server

(client establishes SSL/TCP connection to server)

openssl s_client -crlf -connect exchange2010.wesleyan.edu:465

220 exchcas3.wesad.wesleyan.edu

EHLO vumanfredi

250-exchcas3.wesad.wesleyan.edu Hello

auth login

334 kagkNDcg32bc

<enter binary username>

334 rpMeg1m4jnnz

<enter binary password>

250

MAIL FROM: <vumanfredi@wesleyan.edu>

250 2.1.0 Sender OK

...

...

See smtp.txt on schedule for full example and try yourself

Look at smtp.txt handout

Walkthrough how to logon to mail server and send email

SMTP details

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

Q: How do you send images in email?

HTTP vs SMTP

HTTP

- pull
- each object encapsulated in its own response message

SMTP

- push
- multiple objects sent in multipart message

Both have

- ASCII command/response interaction, status codes

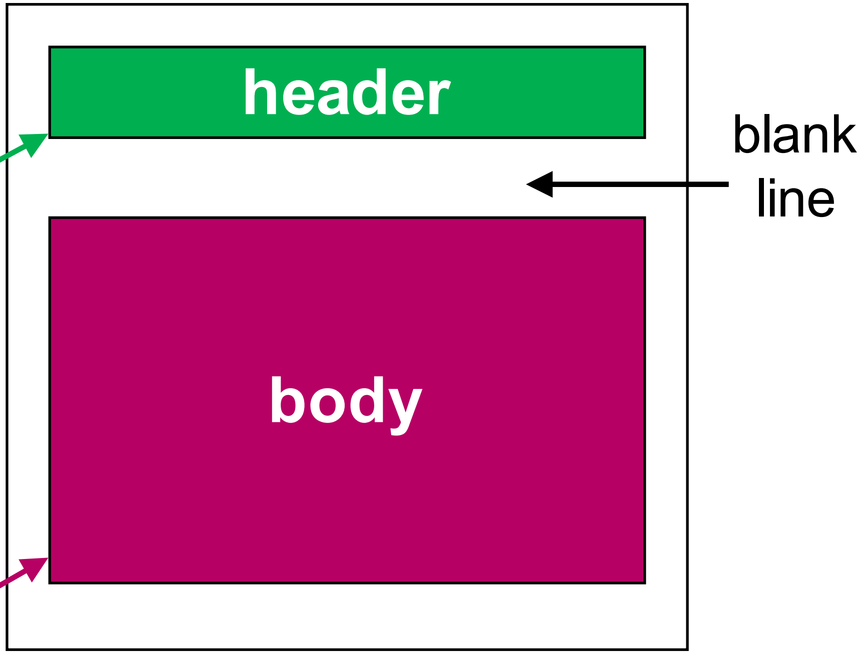
Message format

SMTP

- protocol for exchanging (ASCII only) email messages

RFC 822

- specifies format of e-mail message
- header lines
  - To:
  - From:
  - Subject:
  - **different from SMTP MAIL FROM, RCPT TO!**
- body: the "message"
  - ASCII characters only



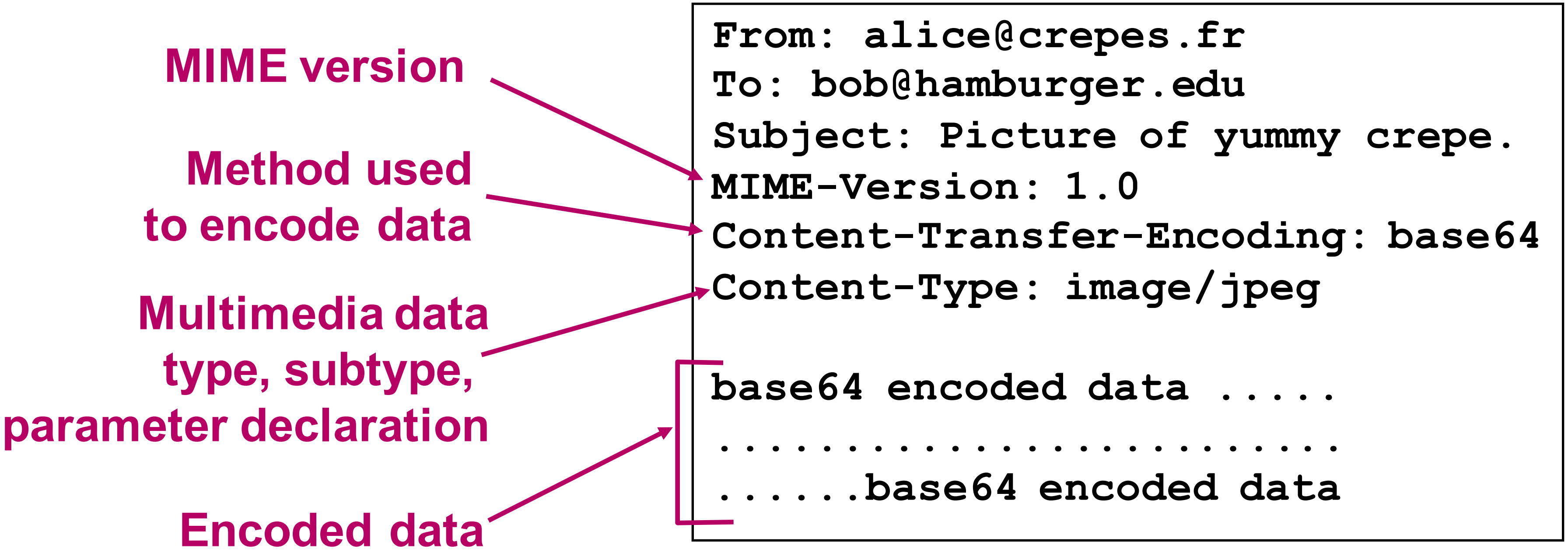
Q: How to send images?

MIME (Multipurpose Internet Mail Extensions) encodes arbitrary data (e.g. binary image) in plain ASCII text. SMTP supports only ASCII messages

Message format: MIME extension

MIME: Multipurpose Internet Mail Extensions, RFC 2045, 2056

- additional lines in message header declare MIME content type
- message can have multiple parts, e.g., text, image, etc.



MIME version

Method used to encode data

Multimedia data type, subtype, parameter declaration

Encoded data

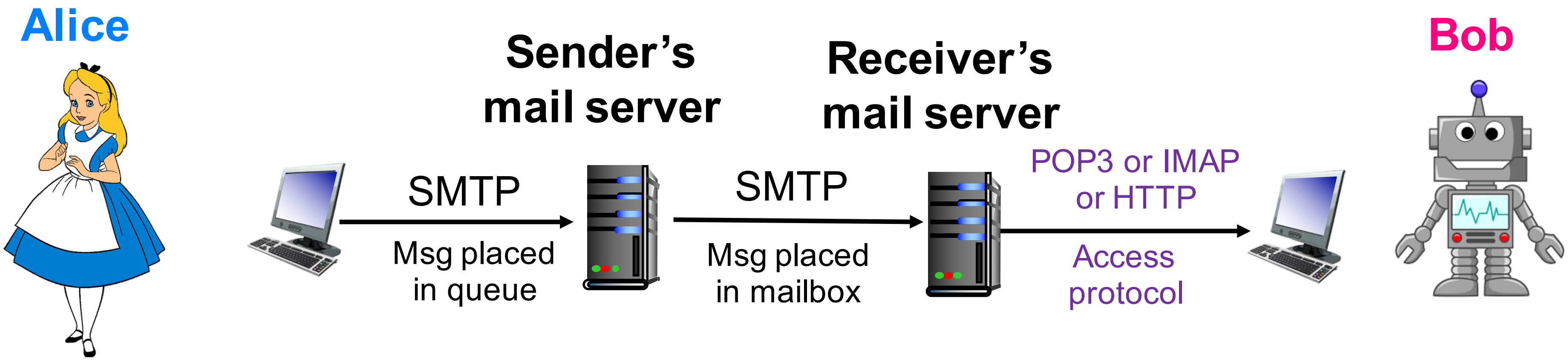
```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

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

Electronic Mail

MAIL ACCESS PROTOCOLS

Mail access protocols



Delivery/storage to receiver's server

- SMTP

Mail access protocol: retrieval from server

- POP3: Post Office Protocol [RFC 1939]
  - Authorization (agent <-> server) and download
- IMAP: Internet Mail Access Protocol [RFC 1730]
  - more features (more complex)
  - manipulation of stored messages on server
- HTTP: gmail, Hotmail, Yahoo! Mail, etc.

POP3 protocol

authorization phase

- client commands:
  - user: declare username
  - pass: password
- server responses
  - +OK
  - -ERR

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

transaction phase, client:

- list: list message numbers
- retr: retrieve message by number
- dele: delete
- quit

```
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 2 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

POP3 (more) and IMAP

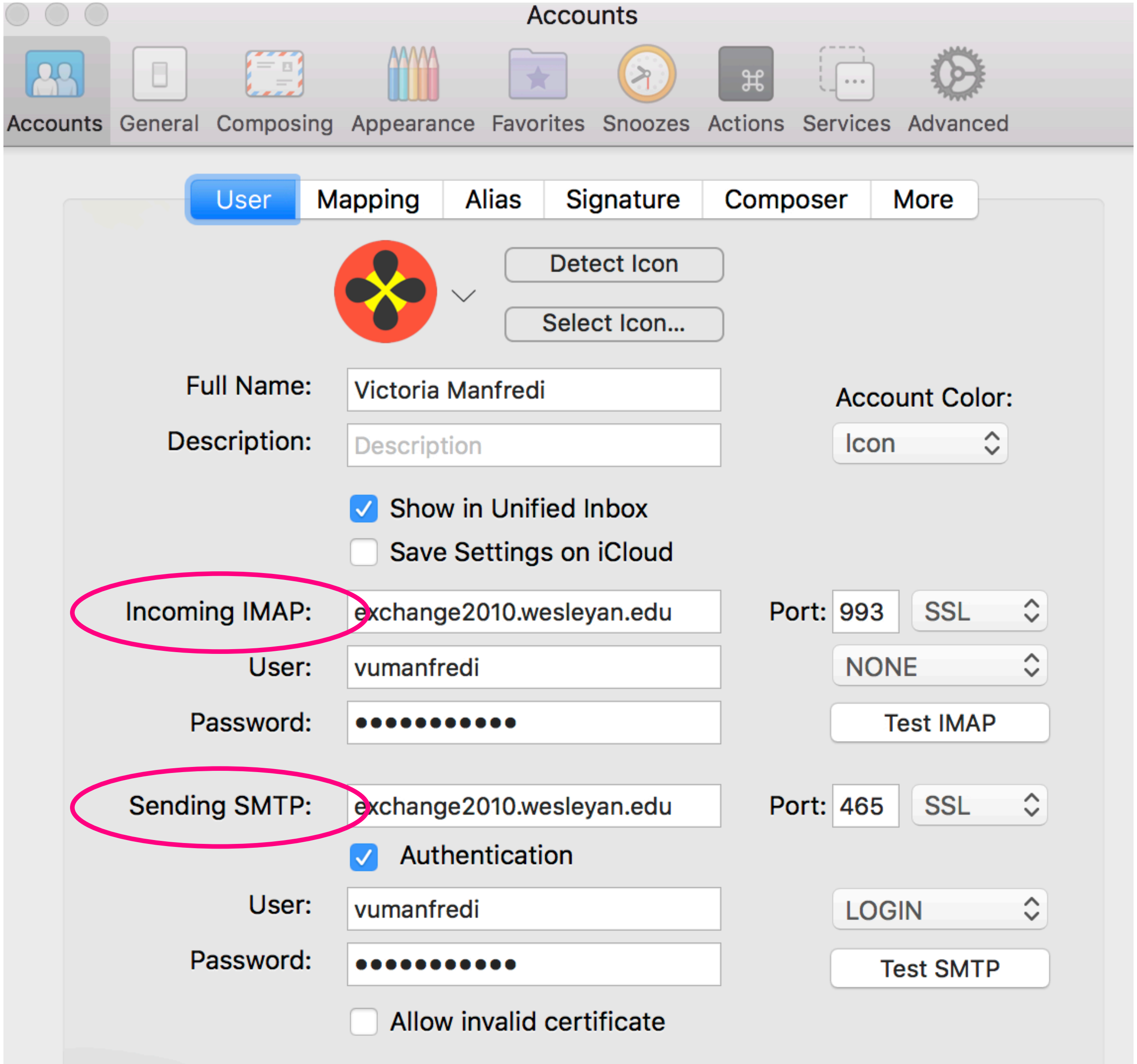
More about POP3

- “download and delete” mode
  - previous example
  - Bob cannot re-read e-mail if he changes client
- “download-and-keep” mode
  - copies of messages on different clients
- stateless across sessions

IMAP

- keeps all messages 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

Setting up your user agent



Full Name: Victoria Manfredi

Description: Description

Account Color: Icon

☑ Show in Unified Inbox

□ Save Settings on iCloud

Incoming IMAP: exchange2010.wesleyan.edu Port: 993 SSL

User: vumanfredi NONE

Password: ●●●●●●●●●●●

Sending SMTP: exchange2010.wesleyan.edu Port: 465 SSL

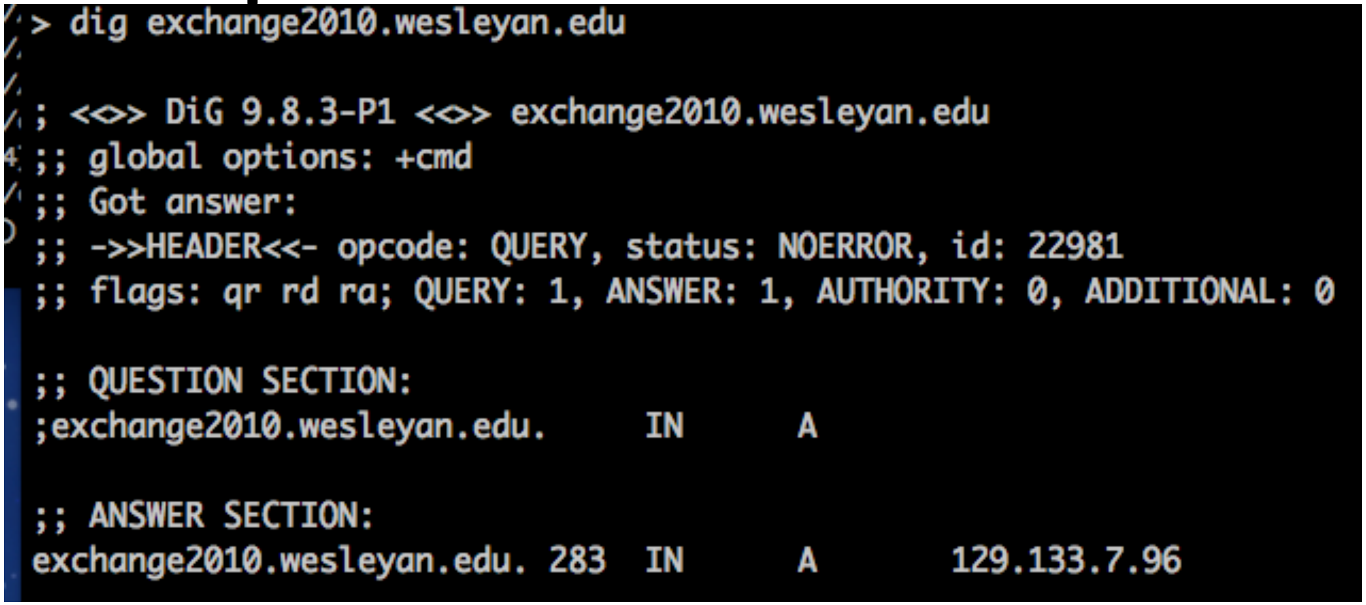
☑ Authentication

User: vumanfredi LOGIN

Password: ●●●●●●●●●●●

□ Allow invalid certificate

Mail server ip address

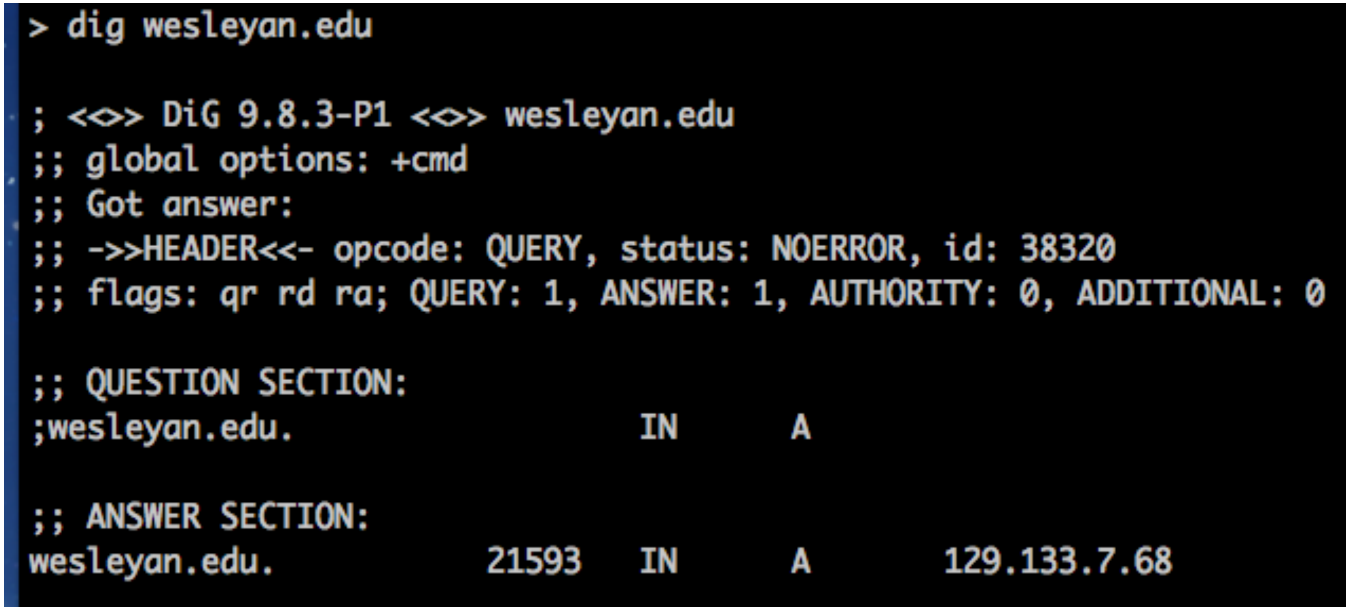


```
> dig exchange2010.wesleyan.edu

; <<>> DiG 9.8.3-P1 <<>> exchange2010.wesleyan.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22981
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;exchange2010.wesleyan.edu.     IN      A

;; ANSWER SECTION:
exchange2010.wesleyan.edu. 283  IN      A       129.133.7.96
```



```
> dig wesleyan.edu

; <<>> DiG 9.8.3-P1 <<>> wesleyan.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 38320
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;wesleyan.edu.                  IN      A

;; ANSWER SECTION:
wesleyan.edu.           21593   IN      A       129.133.7.68
```

Look at complete email header

Show raw source in gmail or wesleyan email