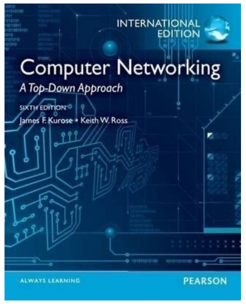
Mic!

Chapter 2.4

Email

SCC. 203 – Computer Networks

Geoff Coulson
Week 14 Lecture 1



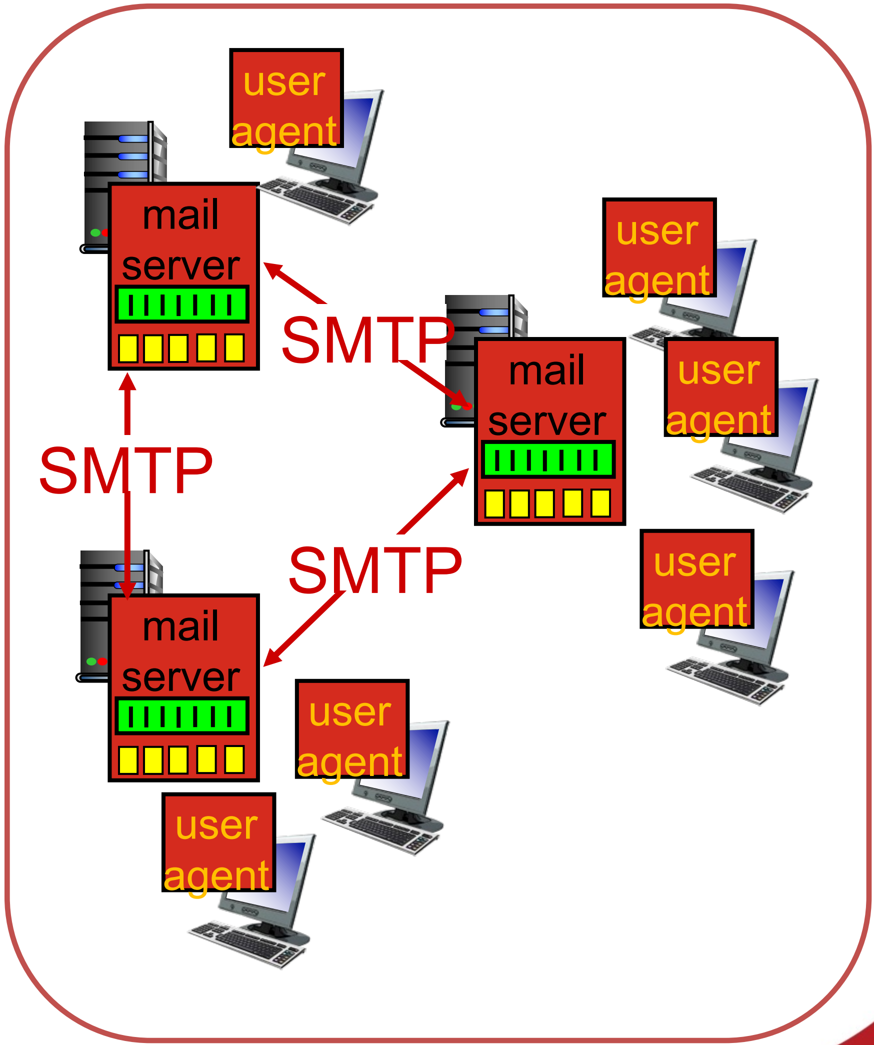
Email…

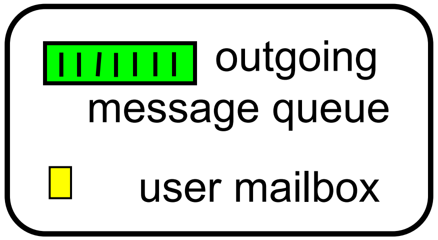
- Yet another network application
- Incredibly popular and widely used
  - In 2015, the numbers of emails sent and received per day totalled over 205 billion
  - In 2015, the average worker sent and received 122 emails a day
- Asynchronous in nature
  - People can send, receive and read messages at any time

Application architecture

Three essential components:

- Mail Servers
- User Agents
- "Simple Mail Transfer Protocol" (SMTP)





Mail Servers

- Composition of a mail server
  - *Mailboxes* (one per user) contain incoming messages for users
  - *Message queue* of outgoing (to be sent) mail messages
  - *Messages are sent* between mail servers using SMTP

- Mail servers play both a “client” role and a “server” role
  - “Server” role: receive emails from an upstream server (or UA)
  - “Client” role: sending emails to a downstream server

User Agent

- "Mail reader"
- Outlook, Thunderbird, iPhone mail client, etc.
- Used to compose, edit, send, read,… mail messages
- Both outgoing and incoming messages may stored in the UA itself and/or the associated server

Simple Mail Transfer Protocol [RFC 5321]

- Principal application-layer protocol for Internet electronic mail
- Older than HTTP!
  - RFC 5321 was published in 1982, but SMTP was used before that too
- Still in widespread use, despite its age
- Some archaic characteristics
  - Body **must** be simple 7-bit ASCII
  - This made sense in the 1980s when *throughput* was scarce
  - However, with the advent of the multimedia era, sending images, audio, etc. became more common
  - Binary multimedia data has to be encoded to ASCII before sending
    - Decoded on the receiver side too

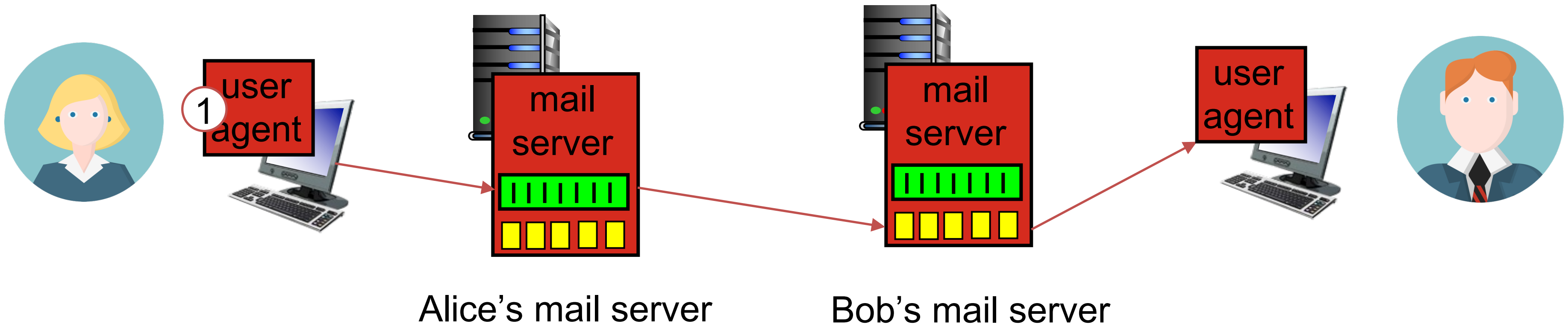
Simple Mail Transfer Protocol [RFC 5321]

- Uses TCP to reliably transfer email messages from mail server to mail server
  - Runs over TCP port 25
  - Direct transfer between sending server and receiving server
- Three phases of transfer:
  - Handshake (greeting)
  - Transfer of messages
  - Close
- Command/response style interaction (like HTTP)
  - Commands: ASCII text
  - Response: status code + text

Scenario

Alice wants to send a message to Bob

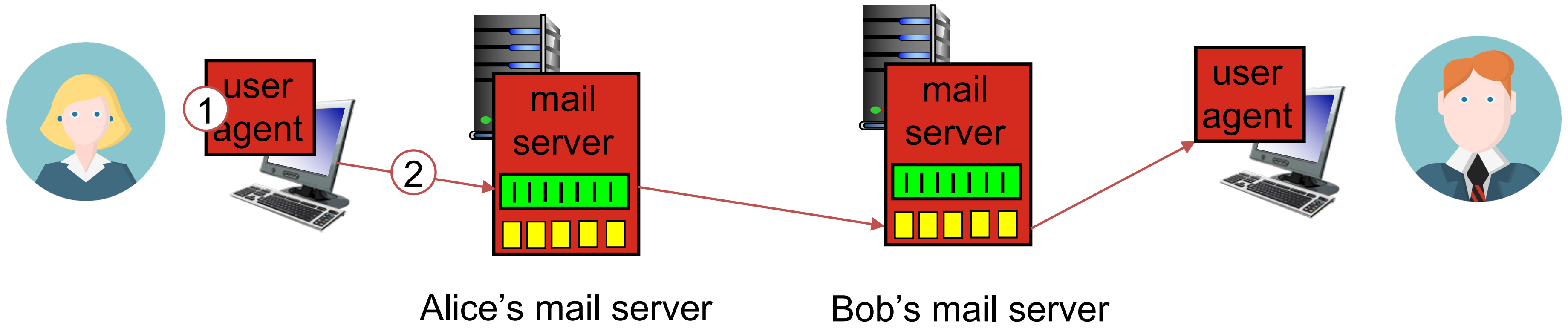
1) Alice uses UA to compose message "to" `bob@someschool.edu`



Scenario

Alice wants to send a message to Bob

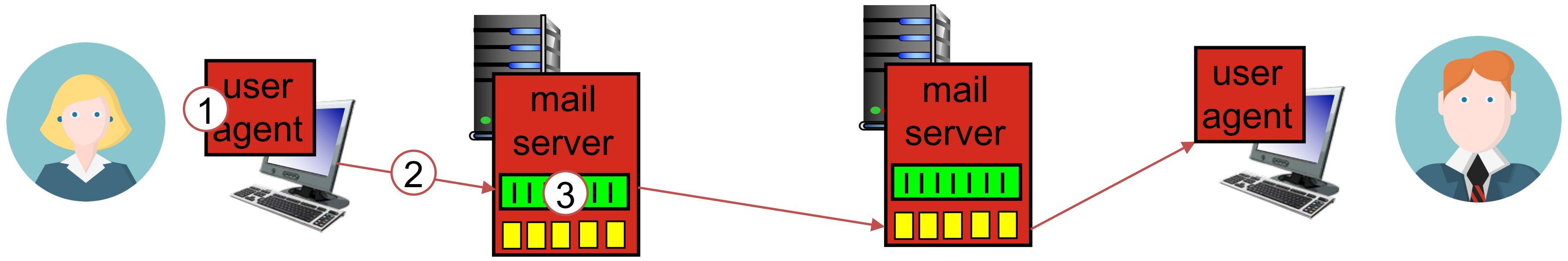
2) Alice's UA sends message to her mail server; message placed in message queue



Scenario

Alice wants to send a message to Bob

3) Client side of SMTP opens TCP connection with Bob's mail server



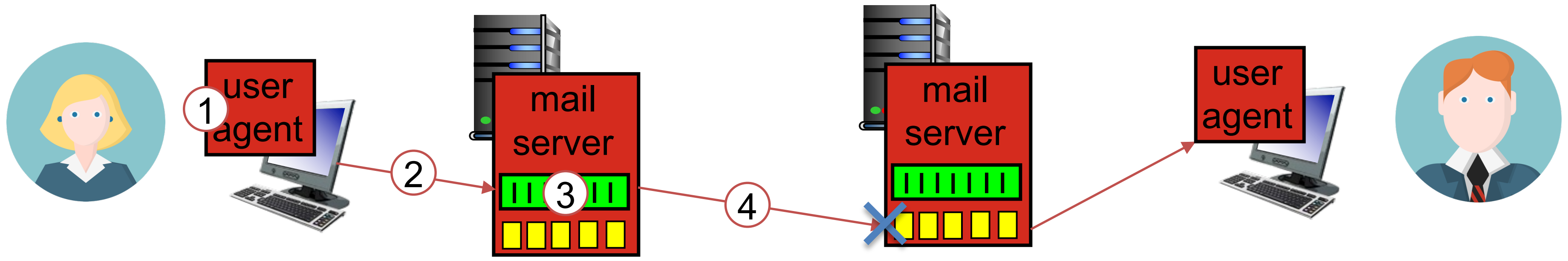
Alice's mail server

Bob's mail server

Scenario

Alice wants to send a message to Bob

4) SMTP client sends Alice's message over the TCP connection; let's assume that Bob's mail server queue is full and rejects the message

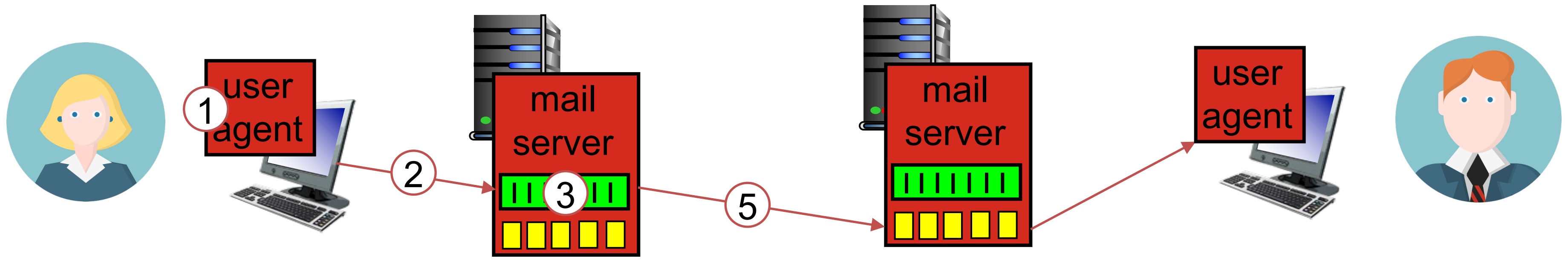


Alice's mail server   Bob's mail server

Scenario

Alice wants to send a message to Bob

5) Alice's mail server retries every 30 minutes until it's successful; the message is eventually delivered



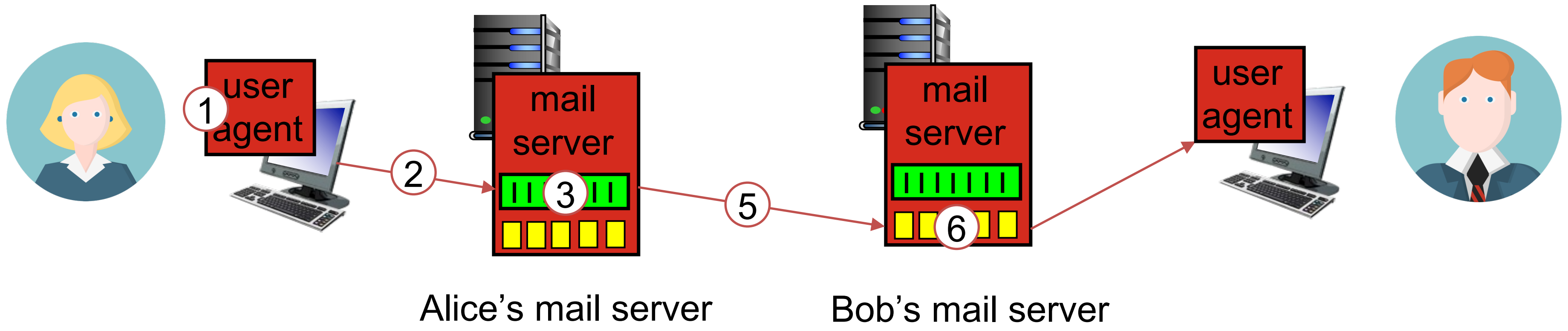
Alice's mail server

Bob's mail server

Scenario

Alice wants to send a message to Bob

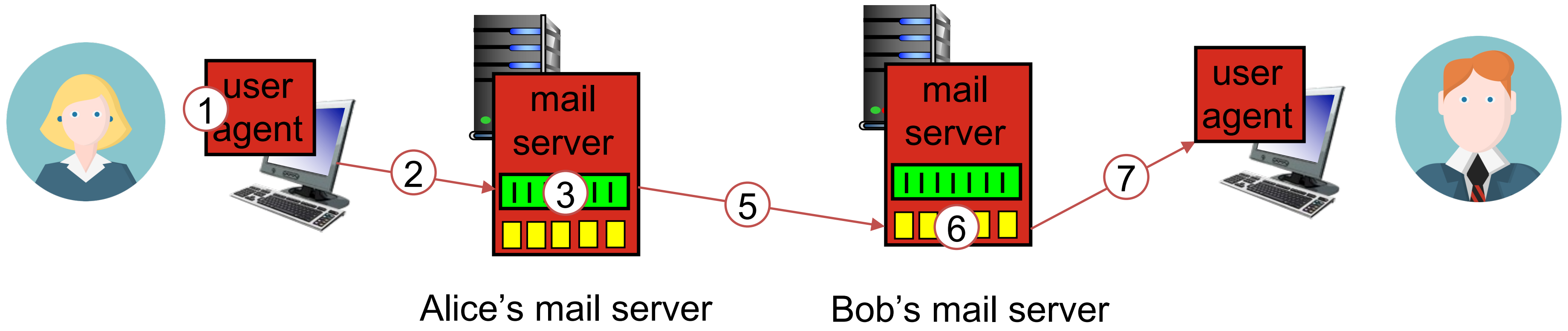
6) Bob's mail server places the message in Bob's mailbox



Scenario

Alice wants to send a message to Bob

7) Bob invokes his user agent to read message



Intermediate Servers

- Sending does not use intermediate mail servers
  - Even if the mail servers are on opposite sides of the world
- Example: Alice's server is in Lancaster, Bob's server is in New York: direct TCP connection between the two servers
- If Bob's mail server is not receiving additional messages, the message will remain on Alice's mail server in Lancaster
  - It does not get placed in an intermediate server

Sample SMTP Interaction

Much like face-to-face communication!

```
S: 220 nyu.edu
C: HELO lancs.uk
S: 250  Hello lancs.uk, pleased to meet you
C: MAIL FROM: <alice@lancs.uk>
S: 250 alice@lancs.uk… Sender ok
C: RCPT TO: <bob@nyu.edu>
S: 250 bob@nyu.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 nyu.edu closing connection
```

Try SMTP for yourself!

- `telnet smtp.lancaster.ac.uk 25`
- See 220 reply from server
- Enter HELO, MAIL FROM, RCPT TO, DATA, QUIT commands
  - See previous slide

- Lets you send email without using email client (UA)
- (Only works while attached to campus network - for authentication and security reasons)

Comparison to HTTP

- HTTP is “pull”; SMTP is "push"
- Both have ASCII command/response interaction, status codes

- HTTP: no inherent format restrictions
- SMTP: enforces 7-bit ASCII format

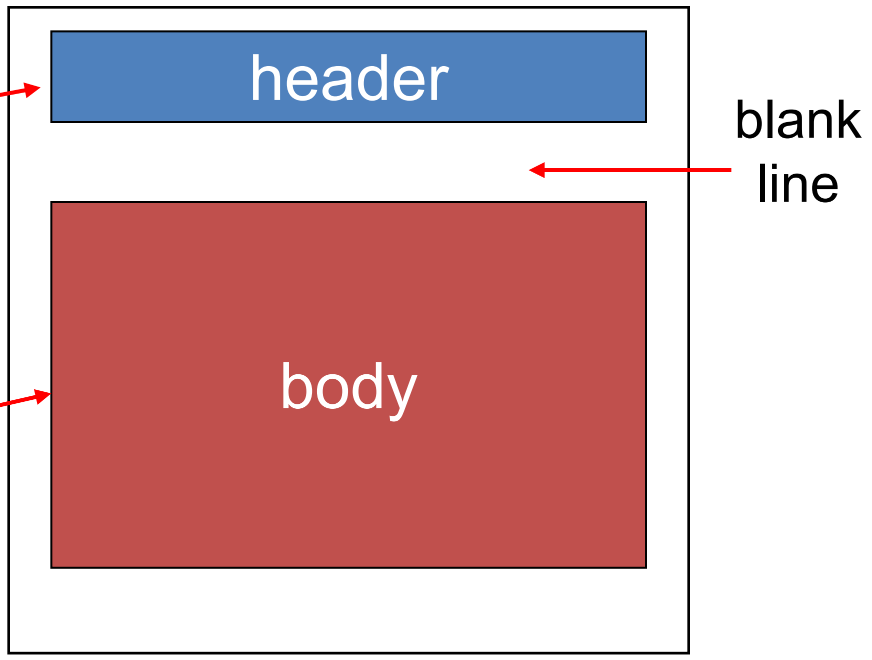
- HTTP: each “object” is encapsulated in its own response message
- SMTP: multiple objects sent in multipart message

Mail Message Format

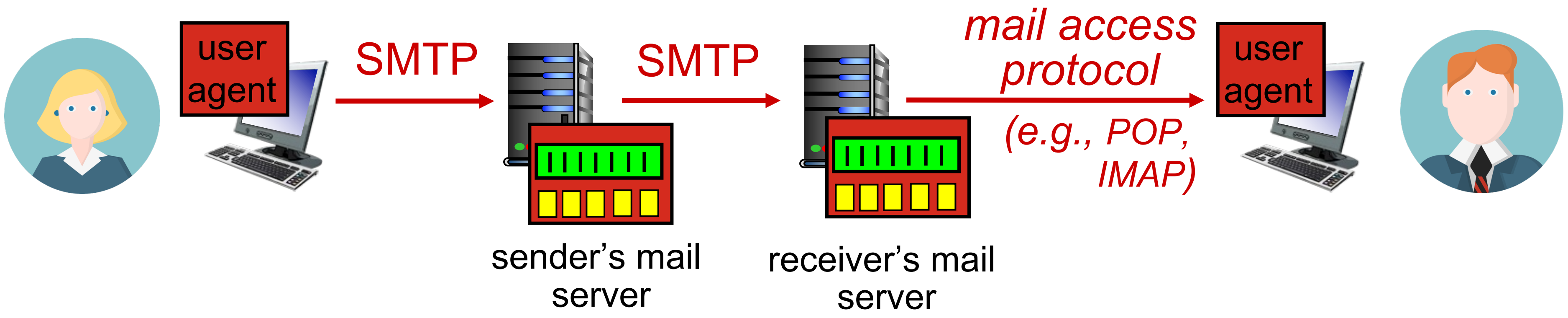
[RFC 822]

Standard for text message format

- Header:
  - Multiple lines:
    - To
    - From
    - Subject
- Body:
  - The “message”
  - 7 bit ASCII characters only



Mail Access Protocols



- SMTP: delivery to receiver's server
- Mail access protocol: subsequent retrieval from server
  - POP: Post Office Protocol [RFC 1939]: authorisation, download
  - IMAP: Internet Mail Access Protocol [RFC 1730]: more sophisticated, e.g. allows manipulation of stored messages on server
  - HTTP: GMail, Hotmail, Yahoo! Mail, etc.

POP3 example

authorisation phase

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

transaction phase,

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

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

```
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 contd.

- Previous example used “download and delete” mode
  - Bob cannot re-read e-mail if he changes client, as it has been deleted
- “download-and-keep” mode allows copies of messages to be kept on different clients
- POP3 is stateless across sessions
  - Simple protocol, but limited functionality
  - Still in widespread use

IMAP

- Can keep and manipulate all messages at the server
- Allows users to organise messages in folders
  - With POP3 this would be achieved locally, on the client-side
  - However, IMAP allows this to be done remotely
- Keeps user state across sessions
- Allows *parts* of a message to be retrieved
  - Useful with low bandwidth connections
  - Avoid fetching attachments, for example

Web-based Email

- Increasingly popular…
  - As provided by Google (and others - e.g. Lancaster University)
- Web browser acts as User Agent
  - When an email is accessed from a mailbox, it is retrieved using HTTP
- Importantly, the mail servers still send and receive messages using SMTP

Thanks for listening!
Any questions?

g.coulson@lancaster.ac.uk