

CPSC 317 – Winter 1 2025

Introduction to Computer Networking

# **Application Layer Protocols - E-mail**

## **Module 3.4**

Norm & Ibtissem

# ADMINISTRATION

- Quiz 1 starting today
- PA1 due Sunday
- PA2 starting later this week
- iClicker today

# READING

- Reading: 2.3

## Learning goals – Email

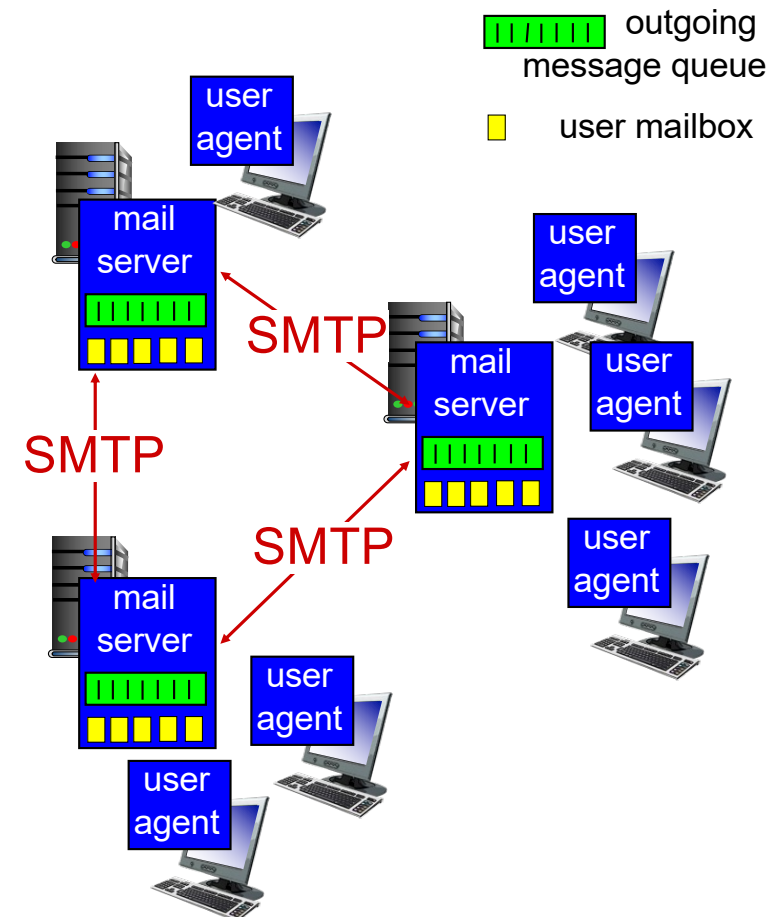
- Identify and describe the roles of various components and protocols of email
- Trace what needs to happen for an email to be sent, delivered, and read
- Describe the functionality of SMTP, POP3, IMAP
- Explain how a user agent would use IMAP, POP3, and SMTP to deliver a typical user email experience
- Describe and use the basic SMTP/POP3 commands to send/manage email
- Compare and contrast IMAP's functionality to that of POP3

## Case Study: Electronic Mail

- P.O. Box system:
  - Sender sends package to its local postal office
  - Postal system forwards the package to final postal office
  - Recipient must pick up package at its local office
- Email works the same way:
  - Sender sends message to its local server
  - The local server relays the message to the destination server
    - This can go through several hops
  - Recipient must connect to its server to retrieve (“pull”) message

# Email Components

- User agents (UAs)
  - Direct user interaction
  - Compose, edit, read mail messages
  - E.g.: Outlook, Thunderbird, Gmail, apps
- Mail servers
  - Maintain mailboxes (incoming messages) for each user
  - Queue outgoing messages to be sent to other servers

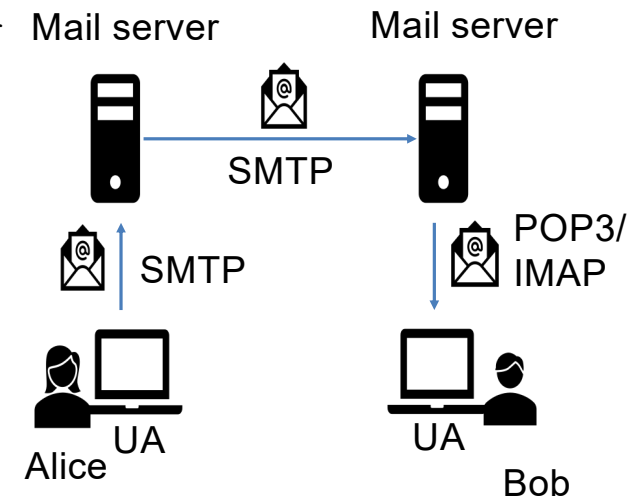


# Email Protocols

- Simple Mail Transfer Protocol (SMTP)
  - Used to send messages from a user agent to a server (submission)
  - Used to send messages from server to server (relay)
- Post Office Protocol 3 (POP3)
  - Used by a user agent to retrieve messages from a server
- Internet Message Access Protocol (IMAP)
  - Used by a user agent to retrieve messages from a server
- Proprietary protocols
  - Microsoft Exchange (MAPI, EAS)
  - Web interfaces

# Email Transfer Example

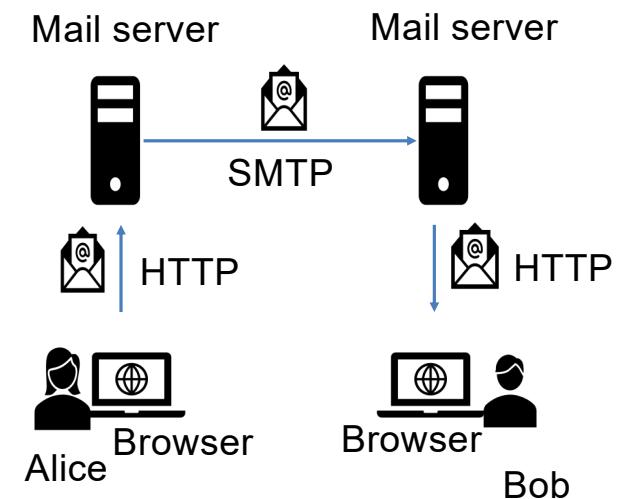
- Alice uses a user agent (UA) to compose a message to Bob
- Alice's UA sends the message (via SMTP) to her mail server
- Alice's server connects (as a client) to Bob's server using SMTP
  - Message is relayed using this connection
  - In some cases additional relaying servers may be needed
- Bob's server places the message in Bob's mailbox
- Bob invokes his UA to retrieve the message from Bob's server using POP3, IMAP





## Web-Based Email

- Alice uses a browser to compose a message to Bob
- Alice's browser sends the message (via HTTP) to her mail server
- Alice's server connects (as a client) to Bob's server using SMTP
- Bob's server places the message in Bob's mailbox
- Bob invokes his browser to retrieve the message from Bob's server using HTTP



# SMTP

- Uses TCP to transfer messages
  - Port 25 for message relay
  - Port 587 for message submission
    - Other ports may be used if 587 is blocked (e.g., 465, 2525)
- Command-response interaction (similar to DICT)
  - Command: ASCII text, single line
  - Response: 3-digit status code followed by text
  - In-band message transfer (in ASCII only)

## SMTP commands

- HELO (EHLO)
  - MAIL FROM:
  - RCPT TO:
  - DATA (Data ends with a . on a line by itself)
  - QUIT
- 
- NOTE: no commands for subject and other mail header fields (bcc, cc, etc)

## Mail Message Format (RFC 822)

- Header lines
  - Each line format “Name: value”
  - Examples: From, To, Cc, Subject
  - Ends with blank line (end of header)
- Message body
  - ASCII characters only
  - May have multiple parts: attachment, text vs HTML, etc.
  - Binary data (e.g., attachments) encoded using text-only format (e.g., Base64)

# SMTP Demo

## Sending messages

- Think of SMTP as manipulating an envelope
- The SMTP commands instruct an SMTP server to send a message to particular receivers
- The SMTP server does not care about the contents of the message
  - Except that it can add new headers
- The message and the envelope can disagree

## POP3

- TCP port 110
  - TCP port 995 for POP3S (typically)
- **Download and delete mode:** client retrieves messages and deletes them from the server
  - Changing clients requires transferring messages from client to client
- **Download and keep mode:** client leaves messages in the server
- Stateless across sessions

## Pop3 example

```
Connected to mailserver.net.  
+OK mailserver.net POP3 Service Ready  
user xxxxxxxx  
+OK password required for user xxxxxxxx  
pass password  
+OK Maildrop ready  
list  
+OK scan listing follows  
1 220  
2 2312  
3 30152  
4 37373  
5 11780  
6 896  
.
```



## Pop3 example (continued)

retr 1

+OK 220 octets

From: Norm <norm@cs.ubc.ca>

To: John Doe

<john.doe@example.com>

Subject: Some boring mail

Some data

..A line that starts with a .

... Another one

.

dele 1

+OK message 1 deleted

list

+OK scan listing follows

2 2312

3 30152

4 37373

5 11780

6 896

.

quit

+OK mailserver.net POP3 Service

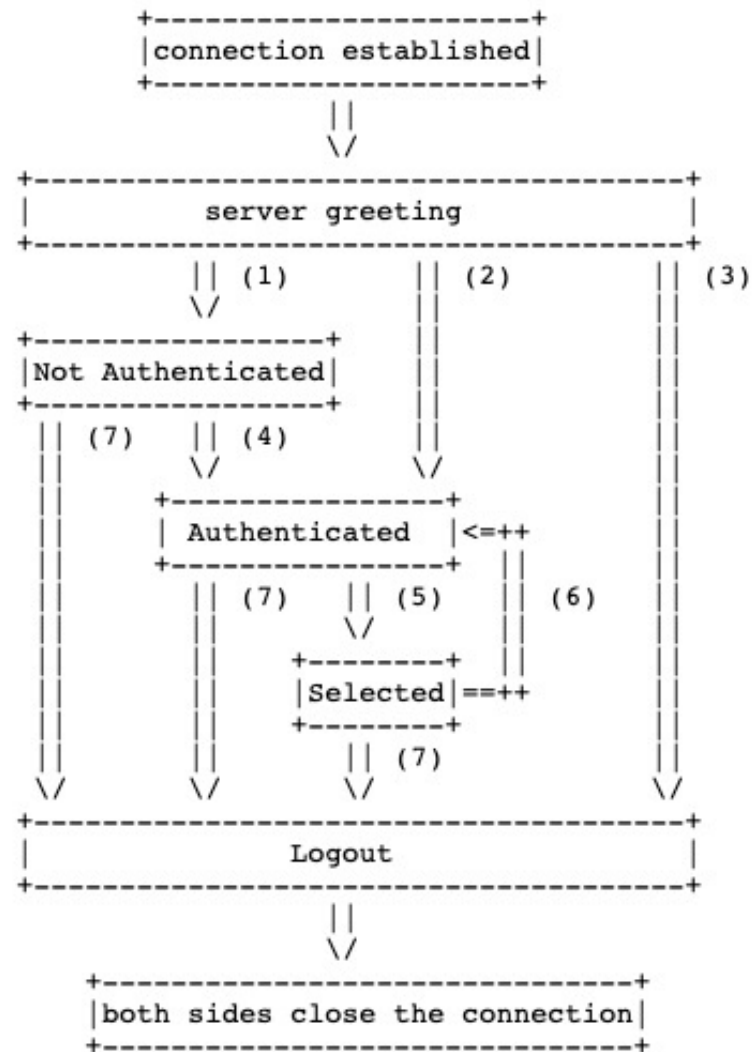
Signing off

# IMAP

- TCP port 143
  - TCP port 993 for IMAPS
- All messages are kept in the server and organized in folders
  - New messages stored in an INBOX folder
  - Users can move messages into user-created folders
- Message state, organization and deletion status is synchronized between server and all clients and across sessions

# IMAP COMMANDS

- Create folders, move messages between folders
- Search remote folders based on specific matching criteria
- Can request for a part of the messages (e.g., just the header)




# IMAP Session

```
S: * OK [CAPABILITY STARTTLS AUTH=SCRAM-SHA-256 LOGINDISABLED IMAP4rev2] IMAP4rev2 Service Ready
C: a000 starttls
S: a000 OK Proceed with TLS negotiation <TLS negotiation>
C: a001 AUTHENTICATE SCRAM-SHA-256 <Authentication>
S: a001 OK SCRAM-SHA-256 authentication successful
C: babc ENABLE IMAP4rev2
S: * ENABLED IMAP4rev2
S: babc OK Some capabilities enabled
C: a002 select inbox
S: * 18 EXISTS
S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S: * OK [UIDVALIDITY 3857529045] UIDs valid
S: * LIST () "/" INBOX ("OLDNAME" ("inbox"))
S: a002 OK [READ-WRITE] SELECT completed
```

# IMAP Session

C: a003 fetch 12 full

```
S: * 12 FETCH (FLAGS (\Seen) INTERNALDATE
  "17-Jul-1996 02:44:25 -0700" RFC822.SIZE 4286 ENVELOPE (
  "Wed, 17 Jul 1996 02:23:25 -0700 (PDT)"
  "IMAP4rev2 WG mtg summary and minutes"
  (("Terry Gray" NIL "gray" "cac.washington.edu"))
  (("Terry Gray" NIL "gray" "cac.washington.edu"))
  (("Terry Gray" NIL "gray" "cac.washington.edu"))
  ((NIL NIL "imap" "cac.washington.edu"))
  ((NIL NIL "minutes" "CNRI.Reston.VA.US")
  ("John Klensin" NIL "KLENSIN" "MIT.EDU")) NIL NIL
  "<B27397-01000000@cac.washington.ed>")
  BODY ("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT"
  3028 92))
S: a003 OK FETCH completed
```



One very  
long line

```
C: a004 fetch 12 body[header]
S: * 12 FETCH (BODY[HEADER] {342}
S: Date: Wed, 17 Jul 1996 02:23:25 -0700 (PDT)
S: From: Terry Gray <gray@cac.washington.edu>
S: Subject: IMAP4rev2 WG mtg summary and minutes
S: To: imap@cac.washington.edu
S: cc: minutes@CNRI.Reston.VA.US, John Klensin <KLENSIN@MIT.EDU>
S: Message-Id: <B27397-0100000@cac.washington.edu>
S: MIME-Version: 1.0
S: Content-Type: TEXT/PLAIN; CHARSET=US-ASCII
S:
S: )
S: a004 OK FETCH completed
C: a005 store 12 +flags \deleted
S: * 12 FETCH (FLAGS (\Seen \Deleted))
S: a005 OK +FLAGS completed
C: a006 logout
S: * BYE IMAP4rev2 server terminating connection
S: a006 OK LOGOUT completed
```

# Performance

- We spent a lot of time talking about the performance of HTTP (persistent connections, pipelining, caching) and DNS (caching, not allowing recursive resolution)
- We aren't going to talk about the performance of E-Mail protocols
- Why not?



## In-class activity

- ICA34

# **Next Topic: Application Layer Protocols – Peer-to-Peer**