Where we are

- **Application**: the application (e.g., the Web, Email)
- **Transport**: end-to-end connections, reliability
- **Network**: routing
- **Link (data-link)**: framing, error detection
- **Physical**: 1’s and 0’s/bits across a medium (copper, the air, fiber)
Last class

- Identifiers and addressing

- Domain Name System
  - DNS Protocol, messages
  - Iterative vs. Recursive resolvers
  - Caching
  - DNS Attacks
Today

• Three main parts to email:
  – Mail User Agent
  – Mail Transfer Agent
  – SMTP protocol used to negotiate transfers

• SMTP Protocol

• Mail Access Protocols
  – POP3
  – IMAP
  – Webmail
Electronic mail

Three major components:
- mail user agent (MUA)
- mail transfer agent (MTA)
- simple mail transfer protocol
Mail User Agent a.k.a. “mail reader”

- composing, editing, reading mail messages
- Outlook, Thunderbird, iPhone mail client
Mail Transfer Agent. a.k.a 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 (one-way)
  - client: sending mail server
  - “server”: receiving mail server
Scenario: Alice sends message to Bob

1) Alice uses a MUA to compose message “to” bob@swarthmore.edu
2) Alice’s MUA 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 MUA to read message
Mail Servers: Ever Vigilant

• Always on, because they always need to be ready to accept mail.

• Usually owned by ISP
  – You use the email server for either Swarthmore College, or the CS department.
Simple Mail Transfer: SMTP [RFC 2821]

- TCP: reliably transfer email message from client to server, port 25
- Direct transfer: sending server to receiving server
- Messages must be in 7-bit ASCII
- Command/response interaction (like HTTP, FTP)
  - commands: ASCII text
  - response: status code and phrase
Simple Mail Transfer: SMTP [RFC 2821]

- **Direct transfer:** *sending server* to *receiving server*
- **TCP connection** on port 25
  - from *sending mail server* (client)
  - to *receiving mail server* (server)
- **Three phases** of transfer
  - handshaking (greeting), MAIL FROM:, RCPT TO:
  - transfer of messages
  - closure

![Diagram showing SMTP communication process](image-url)
SMTP Message Format

RFC 822: standard for text message format:
- header lines, e.g.,
  - To:
  - From:
  - Subject:
    different from SMTP MAIL FROM, RCPT TO: commands!
- Body: the “message”
  - ASCII characters only
  - Signal EOM with “\r\n.\r\n”
Try SMTP interaction for yourself:

- `telnet allspice.cs.swarthmore.edu 25`
- You should see a 220 reply from the server.
- enter HELO, MAIL FROM, RCPT TO, DATA, QUIT commands

(lets you send email without using email client (MUA))
Demo
Sample SMTP interaction

$ telnet allspice.cs.swarthmore.edu 25
Trying 130.58.68.9...
Connected to allspice.cs.swarthmore.edu
220 allspice.cs.swarthmore.edu ESMTP Postfix
HELO cs.swarthmore.edu
250 allspice.cs.swarthmore.edu
MAIL FROM:<chaganti@cs.swarthmore.edu>
250 2.1.0 OK
RCPT TO:<chaganti@cs.swarthmore.edu>
250 2.1.5 OK
DATA
354 End data with <CR><LF>.<CR><LF>
To: Vasanta Chaganti <chaganti@cs.swarthmore.edu>
From: Vasanta Chaganti <chaganti@cs.swarthmore.edu>
Subject: Telnet test message

This is a test message, via telnet, to myself.
.

Slide 16
Sample SMTP interaction

$ telnet allspice.cs.swarthmore.edu 25
Trying 130.58.68.9...
Connected to allspice.cs.swarthmore.edu
220 allspice.cs.swarthmore.edu ESMTP Postfix
HELO cs.swarthmore.edu
250 allspice.cs.swarthmore.edu
MAIL FROM:<chaganti@cs.swarthmore.edu>
250 2.1.0 OK
RCPT TO:<chaganti@cs.swarthmore.edu>
250 2.1.5 OK
DATA
354 End data with <CR><LF>.<CR><LF>
To: Vasanta Chaganti <chaganti@cs.swarthmore.edu>
From: Vasanta Chaganti <chaganti@cs.swarthmore.edu>
Subject: Telnet test message

This is a test message, via telnet, to myself.

End of message:
CRLF (Dot) CRLF
Fun Demo
Message Signing

1. Sender creates cryptographic **public/private key pair**, publishes public key to the world.

2. Sender uses private key to sign messages.

3. Receiver can verify*, using published public key, that only the holder of the corresponding private key could have sent the message.

* With very high probability.
Logging In / Passwords
Any mail server may need to send a message to Bob’s.

Tough for them all to share credentials...
SMTP versus 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
SMTP: final words

- SMTP uses persistent connections
  - Can send multiple emails in one session

- SMTP requires message (header & body) to be in 7-bit ASCII

- SMTP server uses CRLF.CRLF to determine end of message
Multipurpose Internet Mail Extensions (MIME)

• Special formatting instructions

• Indicated in the header portion of message (not SMTP)
  – SMTP does not care, just looks like message data

• Supports
  – Text in character sets other than ASCII
  – Non-text attachments
  – Message bodies with multiple parts
  – Header information in non-ASCII character sets
MIME

• Adds optional headers
  – Designed to be compatible with non-MIME email clients
  – Both clients must understand it to make sense of it

• Specifies content type, other necessary information

• Designates a boundary between email text and attachments
Mail access protocols

- **SMTP**: delivery/storage to receiver’s server
- mail access protocol: retrieval from server
  - **POP**: Post Office Protocol: authorization, download
  - **IMAP**: Internet Mail Access Protocol: more features, including 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

transaction phase, **client:**
- **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 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```
More about POP3

• Previous example uses “download and delete” mode
  – Bob cannot re-read e-mail if he changes client

• POP3 “download-and-keep”: copies of messages on different clients

• POP3 is stateless across sessions

• Limitations:
  – Can’t retrieve just the headers
  – Can’t impose structure on messages
IMAP

• Keeps all messages in one place: 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

• Can **request pieces of a message** (e.g., text parts without large attachments)
Webmail

• Uses a web browser

• Sends emails using HTTP rather than POP3 or IMAP

• Mail is stored on the 3rd party webmail company’s servers
Summary

• Three main parts to email:
  – Mail User Agent (mail client): read / write for humans
  – Mail Transfer Agent: server that accepts / sends messages
  – SMTP protocol used to negotiate transfers

• No SMTP support for fraud detection