Email: SMTP (Simple Mail Transfer Protocol)

Simple Mail Transfer Protocol (SMTP)

- Protocol defined in RFC 2821 (April 2001)
  - Original definition in RFC 821 (August 1982)
- Designed for:
  - Direct transfer of email from the sender to the receiver (rather than go through a set of relays)
  - Destination system is always up and connected
  - Use TCP to transfer email from client to destination server

Note: There are a lot of variations on email delivery, transmission, and routing. Our interest is in the app-layer protocol and we'll avoid the terminology of mail submission agents, mail transfer agents, mail exchangers, and mail delivery agents. In most cases one program serves the role all of these.

Small enhancements to the model

Mail delivery
- Alice will usually not be on the same machine as the local SMTP server
- Alice's UA is a mail app running on her phone or laptop.
- It sends the message to her email provider's SMTP server using SMTP
- The email provider uses SMTP to talk to the destination server

Mail receipt
- Bob is usually not on the same machine as his mail server
- His mail program (UA) cannot access his mailbox directly
- Bob's UA needs to use a network mailbox access protocol such as IMAP or POP to get the message

Simple Mail Transfer Protocol (SMTP)

Three phases:
1. Handshake (greeting)
2. Transfer of message
3. Close

Command/response interaction
- The SMTP protocol is conversational text
  - A sequence of one-line messages & one-line responses
  - Then the message followed by a single line containing a period (.)
  - Finally, a QUIT command
- All transactions in 7-bit ASCII text
- Responses contain a status code & phrase (like HTTP and FTP)
The basic protocol

The following sequence of commands are used to send email:

- **HELO cs.rutgers.edu:** The client identifies itself as cs.rutgers.edu. This is often ignored now since the server may do a reverse DNS lookup on the IP address.
- **MAIL FROM: <pxk@cs.rutgers.edu>** Identify the address of the mail sender. Note that the domain here may be different from that in the HELO message.
- **RCPT TO: <testuser@pk.org>** Identify the destination(s) for this message. You can have a list of these – one line per destination.
- **DATA** Now give the server the mail message. All the lines after this are the message. A line containing a single period ends it.
- **QUIT** We’re done!

Sample Interaction

This is a sample interaction with me connecting to a Rutgers SMTP server via the telnet program and typing in SMTP commands. My typing is in blue.

```
250 aramis.rutgers.edu ESMTP Sendmail 8.11.7p3+Sun/8.8.8; Tue, 9 Feb 2016 14:45:04 -0500 (EST)
HELO aramis.rutgers.edu
250 Hello aramis.rutgers.edu [128.6.4.2], pleased to meet you
MAIL FROM: <pxk@cs.rutgers.edu>
250 2.1.0 <pxk@cs.rutgers.edu>... Sender ok
RCPT TO: <testuser@pk.org>
250 2.1.5 <testuser@pk.org>... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
From: Paul Krzyzanowski <pxk@cs.rutgers.edu>
Subject: test message
Date: Tue, 9 Feb 2016 14:46:14 -0500
To: Whomever <testuser@pk.org>
Hi,
This is a test.
250 2.0.0 r1BLxln29829 Message accepted for delivery
QUIT
221 2.0.0 aramis.rutgers.edu closing connection
```

Mail message format

- **RFC 822** defines the basic format for text messages
  - **Header lines:**
    - **To:**
    - **From:**
    - **Subject:**
    - Each line contains **field_name**: **field_value**
    - Terminated by a blank line
  - **Body**
    - The actual message
  - **All this is treated as the message by SMTP**
  - It’s up to the user agents to interpret those headers

Mail headers versus data

- **SMTP** is interested in delivering the message
- The crucial data is in the “RCPT TO:” commands
- All mail headers are ignored by the SMTP protocol
  - From:, To:, Subject:, Cc:, Bcc:, etc.
  - These are strictly for the mail apps (user agents) to use
  - Mail is delivered exactly the same way whether a recipient is specified as a To, Cc, or Bcc: they will always end up as RCPT TO: commands in SMTP
- The User Agent can determine what to display

Problem: We want to send more than text

- Originally – we used email to just send plain text (ASCII)
- Later – we wanted to send:
  - Rich text or HTML text (formatted)
  - One or more Images
  - One or more attached files
Multimedia extensions: MIME

- MIME: multimedia mail extensions (RFC 2045, 2056)
- Lines in the message header define the content type
  
  **Type of data**
  
  **Method used to encode the data**
  
  **Version of the format**
  
  **Other stuff, such as how to present the data and what the filename is**
  
  **Base64 encoded data** (allows us to send arbitrary binary data using 7 bit ascii)

Mail access protocols: POP & IMAP

- SMTP deals with mail delivery (sending)
  
  - Sending messages to their destination
  
  - When people ran mail apps on machines other than the mail server:
    
    - The app didn’t have direct (file system) access to the mailbox
    
    - A protocol was needed to interact with the user’s mailbox on the server
  
  - Two protocols were developed
    
    - POP3: Post Office Protocol version 3 (RFC 1939)
    
    - IMAP: Internet Mail Access Protocol (RFC 1730)

POP3

- Client (mail application) connects to TCP port 110

- Three phases:
  1. Identification & authentication
     
       - Send user name and password
  2. Transaction
     
       - Mail access commands
  3. Update & exit

Mail access

- SMTP deals with mail delivery (sending)

  - Sending messages to their destination

  - When people ran mail apps on machines other than the mail server:
    
    - The app didn’t have direct (file system) access to the mailbox
    
    - A protocol was needed to interact with the user’s mailbox on the server

  - Two protocols were developed
    
    - POP3: Post Office Protocol version 3 (RFC 1939)
    
    - IMAP: Internet Mail Access Protocol (RFC 1730)

POP3: Authentication

- User login commands
  
  - user username
  
  - pass password

- All commands are sent in plain text
  
  - This is not secure since internet traffic can be intercepted
  
  - It is a fundamental weakness here and with other protocols (HTTP, telnet, ftp)
  
  - POP3 is usually run over an encrypted session
**Internet Technology**

**POP3: Transaction**

After authentication, the mail app (user agent) sends a series of commands to fetch or delete mail messages

- **stat** show the number of messages in the mailbox and total size

```
STAT 5 5477
```

- **list** show a list of messages with the size of each message. A line containing a period indicates end of data

```
STAT Mailbox scan listing follows
1 1823
2 1825
3 1819
```  

- **retr** retrieve a specific message

```
RETR 2
```

- **dele** delete a specific message

```
delete Message deleted
```  

- **rset** reset the session; undo all deletes

```
rset
```  

**POP3: Update & exit**

When done with the session, make changes permanent

- **quit** commit changes and exit

```
QUIT
```  

**POP3 behaviors**

- "download-and-delete" behavior
  - A mail client connects to a server
  - Downloads messages into its local mailbox
  - Delete them from the server
  - This isn’t useful when you access mail from multiple devices
    - Once a message goes to a device, it’s no longer on the server

- "download-and-keep" behavior
  - A mail client connects to a server
  - Downloads messages but does not delete them
  - User may delete specific messages
  - A user can access messages from another device
  - But POP3 does not keep session state
    - It does not know if a user marked a message for deletion during a previous session

**IMAP: Internet Message Access Protocol**

- **With POP3**
  - Folders, moving messages, & search are all handled at the client with client downloads of the messages
  - IMAP was created for users who access mail from multiple clients
  - IMAP keeps all messages on the server
  - User can organize messages in folders
  - State is stored on the server & available across sessions
    - Names of folders
    - Message ID mappings
    - Mark for deletion
  - Messages reside in folders
  - Commands allow users to
    - Create/delete folders, move messages between folders, search for specific messages, mark messages for deletion, delete messages, fetch headers only

**IMAP Commands**

- We won’t cover the IMAP protocol
  - It’s a lot uglier and a lot more verbose
    - See RFC 3501

- TCP connection
  - All commands are sent as lines of ASCII text
    - Commands are more verbose – spelled out, not abbreviated
      - Each command is prefixed with a unique tag (unique per session)
      - A sequence of commands can be sent without waiting for a response before sending the next one
IMAP Protocol Example

- Login
  ```
  LOGIN  user
  RESPONSE: A01 OK User logged in.
  ```
- List folders
  ```
  A02 LIST (Marked) "Mail" Mail/Trash
  A02 LIST (Unmarked) "Mail" Mail/Deleted
  A02 LIST (Unmarked) "Mail" Mail/Outbox
  A02 LIST (Unmarked) "Mail" Mail/Sent
  A02 LIST (Unmarked) "Mail" Mail/Trash
  A02 LIST (Unmarked) "Mail" Mail/Inbox
  A02 LIST (Unmarked) "Mail" Mail/Drafts
  ```
- Fetch message 1 with full headers
  ```
  A03 FETCH 1 (FLAGS INTERNALDATE SIZE RFC822.HEADER)
  RESPONSE: A03 OK FETCH completed.
  ```

The end