Electronic Mail: SMTP [RFC 2821]

- uses TCP to reliably transfer email message from client to server, port 25
- direct transfer: sending server to receiving server
- three phases of transfer
  - handshaking (greeting)
  - transfer of messages
  - closure
- command/response interaction
  - commands: ASCII text
  - response: status code and phrase
- messages must be in 7-bit ASCII (a historical artifact)

E-Mail Message Format (RFC 822)

- E-mail messages have two parts
  - A header, in 7-bit U.S. ASCII text
  - A body, also represented in 7-bit U.S. ASCII text
- Header
  - Series of lines ending in carriage return and line feed
  - Each line contains a type and value, separated by "":"n'
  - E.g., "To: prof@cs.edu" and "Subject: My grade"
  - Additional blank line before the body begins
- Body
  - Series of text lines with no additional structure/meaning
  - Conventions arose over time (e.g., e-mail signatures)
**Mail Message Format**

RFC 822: standard for text message format:

- header lines, e.g.,
  - To:
  - From:
  - Subject: different from SMTP commands!
- body
  - the "message", ASCII characters only
  - so how do you email pictures?

**Limitation: Sending Non-Text Data**

- E-mail body is 7-bit U.S. ASCII
  - What about non-English text?
  - What about binary files (e.g., images and executables)?
- Solution: convert non-ASCII data to ASCII
  - Base64 encoding: map each group of three bytes into four printable U.S.-ASCII characters
  - Uuencode (Unix-to-Unix Encoding) was widely used

**Limitation: Sending Multiple Items**

- Users often want to send multiple pieces of data
  - Multiple images, PowerPoint files, or e-mail messages
  - Yet, e-mail body is a single, un-interpreted data chunk
- Example: e-mail digests
  - Encapsulating several e-mail messages into one aggregate message (i.e., a digest)
  - Commonly used on high-volume mailing lists
- Conventions arose for how to delimit the parts
  - E.g., well-known separator strings between the parts
  - Yet, having a standard way to handle this is better
Multipurpose Internet Mail Extensions (MIME)

- Additional headers to describe the message body
  - MIME-Version: the version of MIME being used
  - Content-Type: the type of data contained in the message
  - Content-Transfer-Encoding: how the data are encoded
- Definitions for a set of content types and subtypes
  - E.g., image with subtypes gif and jpeg
  - E.g., text with subtypes plain, html, and rich text
  - E.g., application with subtypes postscript and msword
  - E.g., multipart for messages with multiple data types
- A way to encode the data in ASCII format
  - Base64 encoding, as in uuencode/uudecode

Example: E-Mail Message Using MIME

```
From: prof@cs.edu
To: student@cc.edu
Subject: picture of Thomas Edison
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

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

E-Mail Addresses

- Components of an e-mail address
  - Local mailbox (e.g., prof)
  - Domain name (e.g., cs.edu)
- Domain name is not necessarily the mail server
  - Mail server may have longer/cryptic name
    - E.g., cs.edu vs. mail.cs.edu
  - Multiple servers may exist to tolerate failures
    - E.g., cnn.com vs. altmail3.turner.com and nycmail2.turner.com
- Identifying the mail server for a domain
  - DNS query asking for MX records (Mail eXchange)
    - E.g., $dig colostate.edu MX
  - Then, a regular DNS query to learn the IP address
Identifying Mail Server for Domain

```
; $<<< DIX 8.3-PI $<<< ca.colostate.edu MX
; global options: <cmd
; \ Get answer:
; $->>HEADER<<- encode: QUERY, status: NOERROR, id: 48179
; $flags: QR ID RA QUERY: 1, ANSWER: 2, AUTHORITY: 8, ADDITIONAL: 0

; QUESTION SECTION:
; @ca.colostate.edu. IN MX

; ANSWER SECTION:
ca.colostate.edu. 3600 IN MX 20 chco.ca.colostate.edu.
cs.colostate.edu. 3600 IN MX cs.colostate-edu@mail.protecti
on.outlook.com.

; Query time: 32 m sec
; SERVER: 192.168.8.198/192.168.8.1
; WHEN: Mon Nov 24 20:03:03 2014
; MSG SIZE rcvd: 116
```

Electronic Mail

Three major components:
- user agents
- mail servers
- simple mail transfer protocol: SMTP

User Agent
- a.k.a. "mail reader" – Eudora, Outlook, elm, etc.
- composing, editing, reading mail messages
- outgoing, incoming messages stored on server

Electronic Mail: Mail Servers

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
  - client: sending mail server
  - "server": receiving mail server

- outgoing message queue
- user mailbox
Simple Mail Transfer Protocol (RFC 2821)

- Principal application layer protocol for Internet electronic mail.
- Runs over TCP (port 25)
- It is used to "push" email messages from one mail server to another or from an user agent to a mail server

Alice sends message to Bob

- Alice composes email message
- Provides Bob's email address to her user agent
- Alice's user-agent uses SMTP client connection to push message to a SMTP server on Alice's mail server

Email header

- Every received email message will have a header
- Header lines are added by entities (email tool, user-agents, email servers) as they store and forward and email message
- The header lines are a series of text lines
  - Syntax: Header-Name: Header-Value
  - If a line starts with a "tab" character or a "space" then that line is a continuation of previous header-value
Example email header

From: Marta Oliva <oliva@eps.udl.es>
Received: from mailr3.udl.es (mailr3.udl.es [193.144.10.36]) by chico.cs.colostate.edu (8.12.10/8.12.9) with ESMTP id i5GAYmvN008288 for <indrajit@CS.ColoState.EDU>; Wed, 16 Jun 2004 04:34:50 -0600 (MDT)
Received: from eps.udl.es (193.144.10.36) by mailr3.udl.es (8.11.6/8.11.6) with ESMTP id i5GAYga31371 for <indrajit@CS.ColoState.EDU>; Wed, 16 Jun 2004 12:34:42 +0200
Received: from eps.udl.es by eps.udl.es (8.8.8+Sun/SMI-SVR4) id MAA22736; Wed, 16 Jun 2004 12:34:40 +0200 (MET DST)
Message-ID: <40D02249.6090105@eps.udl.es>
Date: Wed, 16 Jun 2004 12:34:49 +0200
From: Marta Oliva <oliva@eps.udl.es>
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.4) Gecko/20030624 Netscape/7.1 (ax)
X-Spam-Level: X-Spam-Checker-Version: 2.60 (1.212-2003-09-23-exp) on chico.cs.colostate.edu
Generation of email headers (1)

salieri.cs.colostate.edu
chico.cs.colostate.edu
mailhost.isse.gmu.edu
pinky.isse.gmu.edu

From: alice@cs.colostate.edu (Alice The Great)
To: bob@isse.gmu.edu
Date: Fri, 18 Jun 2004 10:22:55 -0600 (MDT)
X-Mailer: Pine v2.32
Subject: Conference call today?
Header generated by Alice's user agent and handed off to chico.cs.colostate.edu

Generation of email headers (2)

salieri.cs.colostate.edu
chico.cs.colostate.edu
mailhost.isse.gmu.edu
pinky.isse.gmu.edu

Received: from salieri.cs.colostate.edu (salieri.cs.colostate.edu [129.82.45.76] by chico.cs.colostate.edu (8.12.10/8.12.9) id MAA22736; Wed, 16 Jun 2004 12:34:40 +0200 (MET DST)
From: alice@cs.colostate.edu (Alice The Great)
To: bob@isse.gmu.edu
Date: Fri, 18 Jun 2004 10:22:55 -0600 (MDT)
Message-ID: <Pine.GS0.4.58.0406181022460@salieri.cs.colostate.edu>
X-Mailer: Pine v2.32
Subject: Conference call today?
Header fields added by chico.cs.colostate.edu as it transmits the message to mailhost.isse.gmu.edu
Understanding SMTP commands

- **HELO**
  - Identifies the sending machine
  - The sender can lie
    - Nothing, in principle, prevents chico.cs.colostate.edu from saying "HELO abc.freebie.com"
    - Receiver can find out the sending machine’s real identity, using reverse DNS lookup, for example
Understanding SMTP commands

• MAIL FROM
  – Initiates email processing
  – Address need not be the same as the sender’s own address
  – Turns into the from address in the Received header

Understanding SMTP commands

• RCPT TO
  – Dual of MAIL FROM
  – Specifies the intended recipient (the one to which the email will be delivered regardless of whatever is specified in the To: line in the message)
  – One mail can be sent to multiple recipients by including multiple RCPT TO command
  – Turns into the for address in the Received header

Understanding SMTP commands

• DATA
  – Starts the actual mail entry. Everything following it is considered the message
  – No restrictions on its form
  – Lines at the beginning of the message that start with a single word followed by a colon is considered part of message header
  – Line consisting only of a period terminates the message
• QUIT
  – Terminates the SMTP connection
Try SMTP interaction for yourself:

- `telnet mail.cs.colostate.edu 25`
- see 220 reply from server
- enter HELO, MAIL FROM, RCPT TO, DATA, QUIT commands
above lets you send email without using user agent

Mail Access Protocols

- **SMTP**: delivery/storage to receiver’s server
- Mail access protocol: retrieval from server
  - POP: Post Office Protocol (RFC 1939)
    - authorization (agent <-> server) and download
  - IMAP: Internet Mail Access Protocol (RFC 1730)
    - more features (more complex)
    - manipulation of stored msgs on server
  - HTTP: Hotmail, Yahoo! Mail, etc.

Retrieving E-Mail From the Server

- Server stores incoming e-mail by mailbox
  - Based on the “From” field in the message
- Users need to retrieve e-mail
  - Asynchronous from when the message was sent
  - With a way to view the message and reply
  - With a way to organize and store the messages
- In the olden days...
  - User logged on to the machine where mail was delivered
  - Users received e-mail on their main work machine
Then Users Bought PCs

- Separate machine for personal use
  - Users did not want to log in to remote machines
- Resource limitations
  - Most PCs did not have enough resources to act as a full-fledged e-mail server
- Intermittent connectivity
  - PCs only sporadically connected to the network
    - ... due to dial-up connections, and shutting down of PC
  - Too unwieldy to have sending server keep trying
- Led to the creation of Post Office Protocol (POP)

POP3 protocol

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

**Transaction phase**
- **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 1 contents>
S: .
```

```
C: dele 2
```

```
C: quit
S: +OK POP3 server signing off
```

Limitations of POP

- Does not handle multiple mailboxes easily
  - Designed to put user's incoming e-mail in one folder
- Not designed to keep messages on the server
  - Instead, designed to download messages to the client
- Poor handling of multiple-client access to mailbox
  - Increasingly important as users have home PC, work PC, laptop, Smartphone, etc.
- High network bandwidth overhead
  - Transfers all of the e-mail messages, often well before they are read (and they might not be read at all!)
**Interactive Mail Access Protocol (IMAP)**

- Supports connected and disconnected operation
  - Download message contents on demand
- Multiple clients can connect to mailbox at once
  - Detects changes made to the mailbox by other clients
  - Server keeps state about message (e.g., read, replied to)
- Access to MIME parts of messages & partial fetch
  - Clients can retrieve individual parts separately
  - E.g., text of a message without downloading attachments
- Multiple mailboxes on the server
  - Client can create, rename, and delete mailboxes
  - Client can move messages from one folder to another
- Server-side searches
  - Search on server before downloading messages

**POP3 and IMAP**

**More about POP3**

- Previous example uses “download and delete” mode.
- Bob cannot re-read e-mail if he changes client
- “Download-and-keep”: copies of messages on different clients
- POP3 is stateless across sessions

**IMAP**

- Keep all messages in one place: the server
- Allows user to organize messages in folders
- IMAP keeps user state across sessions:
  - names of folders and mappings between message IDs and folder name

**Web-Based E-Mail**

- User agent is an ordinary Web browser
  - User communicates with server via HTTP
    - E.g., Gmail, Yahoo mail, Hotmail
- Reading e-mail
  - Web pages display the contents of folders
    - … and allow users to download and view messages
    - “GET” request to retrieve the various Web pages
- Sending e-mail
  - User types the text into a form and submits to the server
    - “POST” request to upload data to the server
    - Server uses SMTP to deliver message to other servers
- Easy to send anonymous e-mail (e.g., spam)