1 Electronic Mail

Electronic mail, usually called e-mail, consists of simple text messages – a piece of text sent to a recipient via the internet.

E-mail Clients

To read e-mail, we use an e-mail client, which is software such as Outlook Express, Eudora, and pine. If you subscribe to a free e-mail service such as Hotmail or Yahoo, then to read your e-mail you use an e-mail client that appears in a webpage. With commercial providers such as AOL, you can use the provider’s e-mail reader or a web-browser. All e-mail clients typically provide four basic functions:

- View a list of the messages in your mailbox.
- Select a message from the list, and read the contents of the message.
- Create a new message and send it.
- Handle attachments – i.e. add attachments to a message that you send and save attachments to a message you receive.

E-mail Servers

To send and receive e-mail you need a special computer on the Internet for the client to connect to and hand over the e-mail message for delivery. This computer runs a software application to provide e-mail service, and so is called an e-mail server.

There are millions of computers on the Internet running software applications that act as Web servers, FTP servers, telnet servers, e-mail servers, etc. These applications run continuously on the server machine and they listen to specific ports, waiting for people or programs to attach to the port. The SMTP (Simple Mail Transfer Protocol) server handles outgoing e-mail. This server listens on port number 25 for anyone who wants to send a piece of e-mail. Your e-mail client interacts with the SMTP server to send mail.

The following is an example of how your STAT 133 instructor might send an e-mail to you. might
1. The instructor logs on to the Statistical Computing Facility computers and opens an e-mail client, say pine. She composes a message in the e-mail client to you that says:

   Dear Student,
   You received an A for your project.
   Sincerely,
   Your instructor

2. The instructor hits the send button, Ctrl-X in pine, to send the message to the student with email address student@hotmail.com.

3. Pine, the e-mail client, connects to the SMTP server at stat.berkeley.edu using port 25.

4. Pine gives the SMTP server the address of the sender, the address of the recipient, and the body of the message.

5. The SMTP server takes the ”to” address (student@hotmail.com) and breaks it into two parts:

   (a) The recipient name (student)
   (b) The domain name (hotmail.com)

   Note that if the instructor had sent the e-mail to the student’s class account, eg. s133cs@stat.berkeley.edu, then the SMTP server would simply hand the message to the server that handles incoming mail at stat.berkeley.edu. The incoming mail is handled by an IMAP server (more on it later).

6. Since the recipient is at another domain, SMTP needs to communicate with that domain. The SMTP server has a conversation with a Domain Name Server, or DNS, to find the IP of the SMTP server for hotmail.com. The DNS replies with the one or more IP addresses for the SMTP server(s) that Hotmail operates.

7. The SMTP server at stat.berkeley.edu connects with the SMTP server at Hotmail using port 25. It has a very simple text conversation with the other SMTP, and gives the message to it. The Hotmail server recognizes that the domain name for student is at Hotmail, so it hands the message to Hotmail’s IMAP server, which puts the message in student’s mailbox.
What does this conversation between the two SMTP servers look like? If is specified in public documents called Requests For Comments (RFC). The SMTP is defined in RFC 821 and RFC 1123. It is very simple; it demands an ordered data stream of 7-bit US-ASCII characters. The conversation is initiated by the sender issuing SMTP commands to the receiver. The receiver replies to the sender with numeric reply codes, followed by a text string with additional information about the reply code.

The SMTP server understands very simple text commands like HELO, MAIL, RCPT and DATA. The most common commands are:

- **HELO** - introduce yourself
- **EHLO** - introduce yourself and request extended mode
- **MAIL FROM:** - specify the sender
- **RCPT TO:** - specify the recipient
- **DATA** - specify the body of the message
- **RSET** - reset
- **QUIT** - quit the session

The following are some reply codes sent by the receiver SMTP server.

- **211** System Status or system help reply
- **220** domain Service ready
- **221** domain Service closing transmission channel
- **250** Requested action OK and completed
- **354** Start mail input; end with .
- **421** Domain service not available, closing connection
- **450** Mailbox unavailable, requested mail action not taken
In a typical exchange, after the sender contacts the receiver, the receiver replies with code 220 indicating that it is ready. The sender then sends the HELO command with the client host as an argument. The HELO command identifies the sender to the receiver, and the receiver will respond with a reply code 250. This tells the sender that the connection is open and ready to go. The next step in the transaction identifies and confirms host addresses for both the sender and receiver servers. After introducing itself, the e-mail client indicates the “from” and “to” addresses, and then issues the command DATA to asks the receiver if it is ready to receive the message. The receiver replies with a 354 code to indicate that the sender can deliver the message. This transmission ends with a lone period ‘.’ on a line. After sending the message, the sender issues the command QUIT, and the receiver responds with a 221 reply code. You can, in fact, telnet to a mail server machine at port 25 and have one of these dialogs yourself – this is how people “spoof” e-mail.

**The Anatomy of an E-mail message**

An e-mail consists of two parts, the header and the body. The header contains information about the e-mail such as the sender’s address, the recipient’s address, and the date of transmission. This information is relayed in a special format of KEY: VALUE pairs that conform to RFC 822. Below is a sample header from a message found in the SpamAssassin website.

```
Return-Path: whisper@oz.net
Delivery-Date: Fri Sep 6 20:53:36 2002
From: whisper@oz.net (David LeBlanc)
Date: Fri, 6 Sep 2002 12:53:36 -0700
Subject: [Spambayes] Deployment
In-Reply-To: <LNBBLJKPEHFDALKOLCIEJABCAB.tim.one@comcast.net>
Message-ID: <GCEDKONBLEFPPADDJCOECEHJENAA.whisper@oz.net>
```

Notice the keys are Return-Path, Delivery-Date, From, Date, Subject, In-Reply-to, and Message-ID. The value follows the keyword. For example, in the above header, the value of the From key is whisperoz.net (David LeBlanc).

Some of these keys are mandatory such as Date, From, and To (or In-Reply-To, or BCC). Others keys are optional but widely used, such as Subject, Cc, Received, and Message-ID. Many keys are ignored by the mail system, but the entire header is relayed on to the recipient’s server whether or not it is recognized. For example, headers starting with ’X-’ are for personal application or institution use.
The "Received" header lines are important because they allow the message to be tracked. As a message makes its way to the intended recipient, servers add additional "Received" lines to the header. These lines are added to the top of the header so the first server to receive the message will be at the bottom of the list of Received lines. Note that spammers often add fake Received lines to their e-mail headers.

Below are some typical header keys:

- **Message-Id** - a unique identifier for the e-mail, assigned by the originating server
- **Return-Path** - specifies the ENVELOPE sender’s address; BOUNCED mail gets sent to this address.
- **Date** - added by the e-mail client
- **CC** - the recipients of a “carbon copied” email
- **Reply-To** - the address set by the sender to which the recipient can reply
- **MIME-Version** - used for encoding binary content as attachments.
- **X-header key** - for personal application or institution use.

A value may be continued on a second line of the header, in which case the line will be indented and begin with a tab character or blank spaces.

**Attachments**

The body of the e-mail is separated from the header by a single blank line. When an attachment is added to an e-mail message, the attachment is included in the body of the message. Even with attachments, e-mail messages are still only text messages.

An Internet standard called MIME, Multipurpose Internet Mail Extensions, specifies how messages may be formatted and how to separate the attachments from the message. Information about the MIME encoding is provided through header fields, which are specified in an RFC.

The Content-Type key is used to describe the content of a component or of the entire body. The value provides the top-level type and subtype using the the syntax: top-level/subtype; parameter. Parameters may be required or optional.
Below is an example of a content-type where the top-level is “multipart” which indicates there will be several documents in the body of the message, and the “mixed” subtype tells us that each document may be of a different type.

```
Content-Type: multipart/mixed;
    boundary="----=_NextPart_000_00DE_01511A02.DB1A02A0"
```

In this example, the Content-Type field tells the receiving e-mail program that this message has more than one component, and each component will be separated by the string of characters

```
"----=_NextPart_000_00DE_01511A02.DB1A02A0"
```

The boundary string marks the beginning of each component. It is prefaced with two hyphens in all instances. The boundary string is also used to denote the end of the message, where it is both prefaced by two hyphens and followed immediately by two hyphens. The receiving email program knows when the last component of the message has been read when it reads the boundary string followed by two hyphens.

Each component of an e-mail must be prefaced by this boundary string, optional MIME information, and a mandatory blank line. If the blank line is missing, the recipient’s email program may have difficulty telling where the header information stops and the text of the message begins.

There are seven top-level types: text, image, audio, video, application, multipart, and message. Other examples of Content-Type follow:

```
Content-type: text/html; charset=euc-kr;

Content-Type: application/zip; name="testFile.zip"
```

The first indicates that the message will be in HTML format using a Korean character set. The second indicates that the component will is a zip file, and it sender indicates that it should be saved as “testFile.zip”. Binary files (such as a compressed archive) can be sent as attachments. In such cases, the sender must first encode the binary file so that it can be sent over the Internet. One common encoding scheme is known as base64.

We conclude with two sample e-mail messages. The first is a plain text e-mail with no attachments. It consists of an instructor’s response to an e-mail inquiry sent by a student.
From nolan@stat.Berkeley.EDU Mon Feb 2 22:16:19 2004 -0800
Date: Mon, 2 Feb 2004 22:16:19 -0800 (PST)
From: nolan@stat.Berkeley.EDU
X-X-Sender: nolan@kestrel.Berkeley.EDU
To: Txxxxx Uxxx <txxxxx@uclink.berkeley.edu>
Subject: Re: prof: did you receive my hw?
In-Reply-To: <web-569552@calmail-st.berkeley.edu>
Message-ID: <Pine.SOL.4.50.0402022216120.2296-100000@kestrel.Berkeley.EDU>
References: <web-569552@calmail-st.berkeley.edu>
MIME-Version: 1.0
Content-Type: TEXT/PLAIN; charset=US-ASCII
Status: O
X-Status:
X-Keywords:
X-UID: 9079

Yes it was received.

On Mon, 2 Feb 2004, txxxxx wrote:

> hey prof .nolan,
> 
> i sent out my hw on sunday night. i just wonder did you receive it.
> because i am kinda scared that you didn't receive it.
> like i just wonder how do i know if you got it or not, since the cal
> mail system is kinda weird sometimes. thanks
> 
> txxxxx
>

The second e-mail consists of a message and two attachments from a student to the instructor that has been forwarded on by the instructor to the teaching assistant. The three periods at the end of each attachment indicates that the attachment continues quite a bit longer. The first attachment is a PDF file and the second is an HTML file. The forwarded message is a plain text file.
I am sorry to send this email again, but my outbox told me that the last email only send 1 attached file. I am sending this again to make sure you receive all the necessary files. You and sorry for the inconvenience.