CHAPTER

23

Electronic Mail: SMTP, POP, IMAP, and MIME

One of the most popular Internet services is electronic mail (e-mail). The designers of the Internet probably never imagined the popularity of this application program. Its architecture consists of several components that we will discuss in this chapter.

At the beginning of the Internet era, the messages sent by electronic mail were short and consisted of text only; they let people exchange quick memos. Today, electronic mail is much more complex. It allows a message to include text, audio, and video. It also allows one message to be sent to one or more recipients.

In this chapter, we first study the general architecture of an e-mail system including the three main components: user agent, message transfer agent, and message access agent. We then describe the protocols that implement these components.

OBJECTIVES

The chapter has several objectives:

- To explain the architecture of electronic mail using four scenarios.
- To explain the user agent (UA), services provided by it, and two types of user agents.
- To explain the mechanism of sending and receiving e-mails.
- To introduce the role of a message transfer agent and Simple Mail Transfer Protocol (SMTP) as the formal protocol that handles MTA.
- To explain e-mail transfer phases.
- To discuss two message access agents (MAAs): POP and IMAP.
- To discuss MIME as a set of software functions that transforms non-ASCII data to ASCII data and vice versa.
- To discuss the idea of Web-based e-mail.
- To explain the security of the e-mail system.

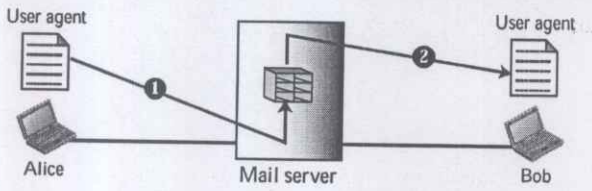
23.1 ARCHITECTURE

To explain the architecture of e-mail, we give four scenarios. We begin with the simplest situation and add complexity as we proceed. The fourth scenario is the most common in the exchange of e-mail.

First Scenario

In the first scenario, the sender and the receiver of the e-mail are users (or application programs) on the same mail server; they are directly connected to a shared mail server. The administrator has created one mailbox for each user where the received messages are stored. A *mailbox* is part of a local hard drive, a special file with permission restrictions. Only the owner of the mailbox has access to it. When Alice needs to send a message to Bob, she runs a *user agent* (*UA*) program to prepare the message and store it in Bob's mailbox. The message has the sender and recipient mailbox addresses (names of files). Bob can retrieve and read the contents of his mailbox at his convenience using a user agent. Figure 23.1 shows the concept.

Figure 23.1 *First scenario*



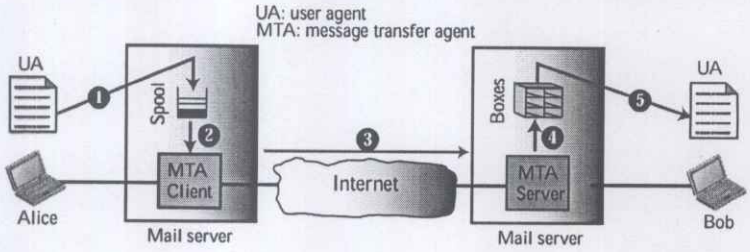
This is similar to the traditional memo exchange between employees in an office. There is a mail room where each employee has a mailbox with his or her name on it. When Alice needs to send a memo to Bob, she writes the memo and inserts it into Bob's mailbox. When Bob checks his mailbox, he finds Alice's memo and reads it.

When the sender and the receiver of an e-mail are on the same mail server, we need only two user agents.

Second Scenario

In the second scenario, the sender and the receiver of the e-mail are users (or application programs) on two different mail servers. The message needs to be sent over the Internet. Here we need **user agents (UAs)** and **message transfer agents (MTAs)** as shown in Figure 23.2.

Figure 23.2 *Second scenario*



Alice needs to use a user agent program to send her message to the mail server at her own site. The mail server at her site uses a queue (spool) to store messages waiting to be sent. Bob also needs a user agent program to retrieve messages stored in the mailbox of the system at his site. The message, however, needs to be sent through the Internet from Alice's site to Bob's site. Here two message transfer agents are needed: one client and one server. Like most client-server programs on the Internet, the server needs to run all of the time because it does not know when a client will ask for a connection. The client, on the other hand, can be triggered by the system when there is a message in the queue to be sent.

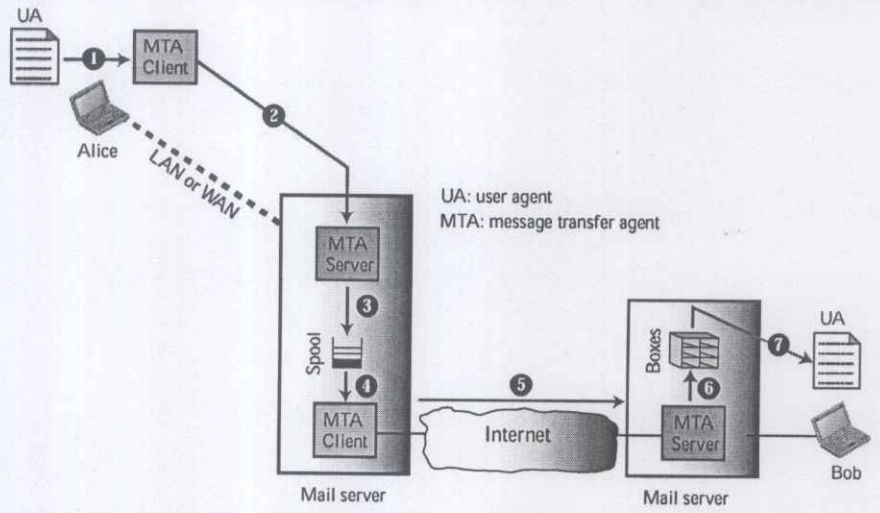
When the sender and the receiver of an e-mail are on different mail servers, we need two UAs and a pair of MTAs (client and server).

Third Scenario

Figure 23.3 shows the third scenario. Bob, as in the second scenario, is directly connected to his mail server. Alice, however, is separated from her mail server. Alice is either connected to the mail server via a point-to-point WAN—such as a dial-up modem, a DSL, or a cable modem—or she is connected to a LAN in an organization that uses one mail server for handling e-mails; all users need to send their messages to this mail server.

Alice still needs a user agent to prepare her message. She then needs to send the message through the LAN or WAN. This can be done through a pair of message transfer agents (client and server). Whenever Alice has a message to send, she calls the user agent which, in turn, calls the MTA client. The MTA client establishes a connection with the MTA server on the system, which is running all the time. The system at Alice's site queues all messages received. It then uses an MTA client to send the messages to the system at Bob's site; the system receives the message and stores it in Bob's mailbox.

Figure 23.3 *Third scenario*



At his convenience, Bob uses his user agent to retrieve the message and reads it. Note that we need two pairs of MTA client-server programs.

When the sender is connected to the mail server via a LAN or a WAN, we need two UAs and two pairs of MTAs (client and server).

Fourth Scenario

In the fourth and most common scenario, Bob is also connected to his mail server by a WAN or a LAN. After the message has arrived at Bob's mail server, Bob needs to retrieve it. Here, we need another set of client-server agents, which we call **message access agents (MAAs).** Bob uses an MAA client to retrieve his messages. The client sends a request to the MAA server, which is running all the time, and requests the transfer of the messages. The situation is shown in Figure 23.4.

There are two important points we need to emphasize here. First, Bob cannot bypass the mail server and use the MTA server directly. To use the MTA server directly, Bob would need to run the MTA server all the time because he does not know when a message will arrive. This implies that Bob must keep his computer on all the time if he is connected to his system through a LAN. If he is connected through a WAN, he must keep the connection up all the time. Neither of these situations is feasible today.

Second, note that Bob needs another pair of client-server programs: message access programs. This is because an MTA client-server program is a *push* program: the client pushes the message to the server. Bob needs a *pull* program. The client needs to pull the message from the server. Figure 23.5 shows the difference.

Figure 23.4 *Fourth scenario*

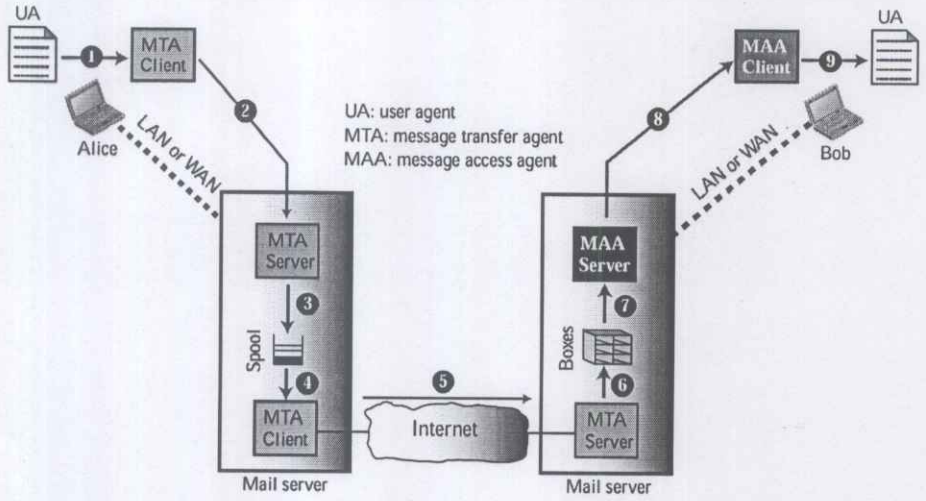


Figure 23.5 *Push vs. pull*

When both sender and receiver are connected to the mail server via a LAN or a WAN, we need two UAs, two pairs of MTAs (client and server), and a pair of MAAs (client and server). This is the most common situation today.

23.2 USER AGENT

The first component of an electronic mail system is the **user agent (UA).** It provides service to the user to make the process of sending and receiving a message easier.

Services Provided by a User Agent

A user agent is a software package (program) that composes, reads, replies to, and forwards messages. It also handles local mailboxes on the user computers.

User Agent Types

There are two types of user agents: command-driven and GUI-based. Command-driven user agents belong to the early days of electronic mail. They are still present as the underlying user agents in servers. A command-driven user agent normally accepts a one-character command from the keyboard to perform its task. For example, a user can type the character *r*, at the command prompt, to reply to the sender of the message, or type the character *R* to reply to the sender and all recipients.

Some examples of command-driven user agents are *mail*, *pine*, and *elm*.

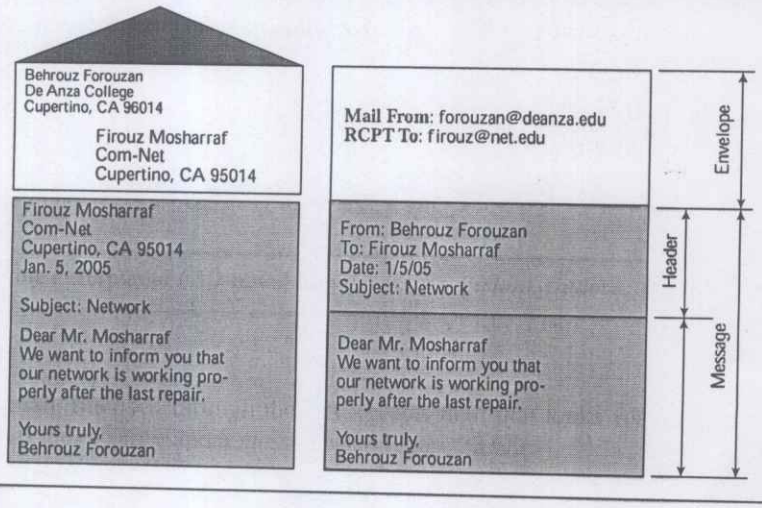
Modern user agents are GUI-based. They contain graphical user interface (GUI) components that allow the user to interact with the software by using both the keyboard and the mouse. They have graphical components such as icons, menu bars, and windows that make the services easy to access.

Some examples of GUI-based user agents are *Eudora*, *Outlook*, and *Netscape*.

Sending Mail

To send mail, the user, through the UA, creates mail that looks very similar to postal mail. It has an *envelope* and a *message* (see Figure 23.6).

Figure 23.6 *Format of an e-mail*



Envelope

The **envelope** usually contains the sender address, the receiver address, and other information.

Message

The message contains the **header** and the **body.** The header of the message defines the sender, the receiver, the subject of the message, and some other information. The body of the message contains the actual information to be read by the recipient.

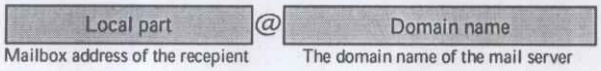
Receiving Mail

The user agent is triggered by the user (or a timer). If a user has mail, the UA informs the user with a notice. If the user is ready to read the mail, a list is displayed in which each line contains a summary of the information about a particular message in the mailbox. The summary usually includes the sender mail address, the subject, and the time the mail was sent or received. The user can select any of the messages and display its contents on the screen.

Addresses

To deliver mail, a mail handling system must use an addressing system with unique addresses. In the Internet, the address consists of two parts: a **local part** and a **domain name,** separated by an @ sign (see Figure 23.7).

Figure 23.7 *E-mail address*



Local Part

The local part defines the name of a special file, called the user mailbox, where all of the mail received for a user is stored for retrieval by the message access agent.

Domain Name

The second part of the address is the domain name. An organization usually selects one or more hosts to receive and send e-mail; they are sometimes called *mail servers* or *exchangers.* The domain name assigned to each mail exchanger either comes from the DNS database or is a logical name (for example, the name of the organization).

Mailing List or Group List

Electronic mail allows one name, an **alias,** to represent several different e-mail addresses; this is called a mailing list. Every time a message is to be sent, the system checks the recipient's name against the alias database; if there is a mailing list for the defined alias, separate messages, one for each entry in the list, must be prepared and handed to the MTA. If there is no mailing list for the alias, the name itself is the receiving address and a single message is delivered to the mail transfer entity.

Commands

Commands are sent from the client to the server. The format of a command is shown below:

```
Keyword: argument(s)
```

It consists of a keyword followed by zero or more arguments. SMTP defines 14 commands listed in Table 23.1 and described in more detail below.

Table 23.1 *Commands*

Keyword	*Argument(s)*	*Keyword*	*Argument(s)*
HELO	Sender's host name	NOOP	
MAIL FROM	Sender of the message	TURN	
RCPT TO	Intended recipient	EXPN	Mailing list
DATA	Body of the mail	HELP	Command name
QUIT		SEND FROM	Intended recipient
RSET		SMOL FROM	Intended recipient
VRFY	Name of recipient	SMAL FROM	Intended recipient

- **HELO.** This command is used by the client to identify itself. The argument is the domain name of the client host. The format is

```
HELO: challenger.atc.fhda.edu
```

- **MAIL FROM.** This command is used by the client to identify the sender of the message. The argument is the e-mail address of the sender (local part plus the domain name). The format is

```
MAIL FROM: forouzan@challenger.atc.fhda.edu
```

- **RCPT TO.** This command is used by the client to identify the intended recipient of the message. The argument is the e-mail address of the recipient. If there are multiple recipients, the command is repeated. The format is

```
RCPT TO: betsy@mcgraw-hill.com
```

- **DATA.** This command is used to send the actual message. All lines that follow the DATA command are treated as the mail message. The message is terminated by a line containing just one period. The format is

```
DATA
This is the message
to be sent to the McGraw-Hill
Company.
.
```

- **QUIT.** This command terminates the message. The format is

```
QUIT
```

- **RSET.** This command aborts the current mail transaction. The stored information about the sender and recipient is deleted. The connection will be reset.

```
RSET
```

- **VRFY.** This command is used to verify the address of the recipient, which is sent as the argument. The sender can ask the receiver to confirm that a name identifies a valid recipient. Its format is

```
VRFY: betsy@mcgraw-hill.com
```

- **NOOP.** This command is used by the client to check the status of the recipient. It requires an answer from the recipient. Its format is

```
NOOP
```

- **TURN.** This command lets the sender and the recipient switch positions, whereby the sender becomes the recipient and vice versa. However, most SMTP implementations today do not support this feature. The format is

```
TURN
```

- **EXPN.** This command asks the receiving host to expand the mailing list sent as the arguments and to return the mailbox addresses of the recipients that comprise the list. The format is

```
EXPN: x y z
```

- **HELP.** This command asks the recipient to send information about the command sent as the argument. The format is

```
HELP: mail
```

- **SEND FROM.** This command specifies that the mail is to be delivered to the terminal of the recipient, and not the mailbox. If the recipient is not logged in, the mail is bounced back. The argument is the address of the sender. The format is

```
SEND FROM: forouzan@fhda.atc.edu
```

- **SMOL FROM.** This command specifies that the mail is to be delivered to the terminal or the mailbox of the recipient. This means that if the recipient is logged in, the

mail is delivered only to the terminal. If the recipient is not logged in, the mail is delivered to the mailbox. The argument is the address of the sender. The format is

SMOL FROM: forouzan@fhda.atc.edu

- **SMAL FROM.** This command specifies that the mail is to be delivered to the terminal and the mailbox of the recipient. This means that if the recipient is logged in, the mail is delivered to the terminal and the mailbox. If the recipient is not logged in, the mail is delivered only to the mailbox. The argument is the address of the sender. The format is

SMAL FROM: forouzan@fhda.atc.edu

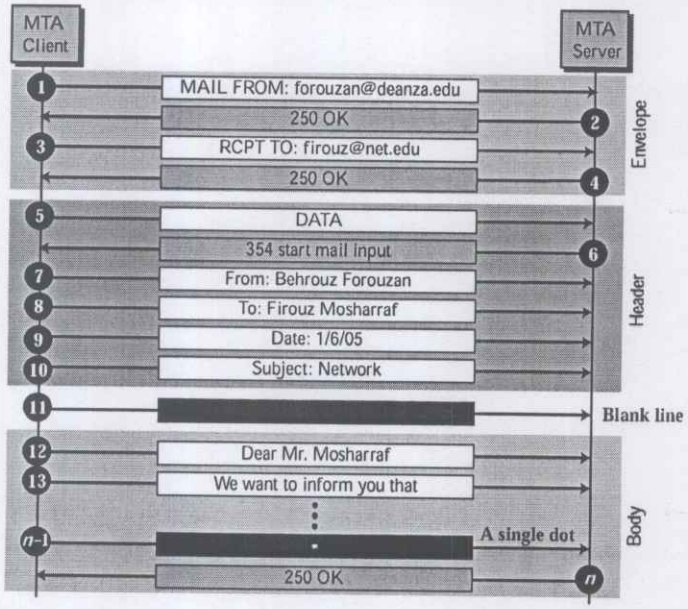
Responses

Responses are sent from the server to the client. A response is a three-digit code that may be followed by additional textual information. The idea is the same as discussed in the case of HTTP responses in Chapter 22. Table 23.2 lists some of the responses.

Table 23.2 *Responses*

Code	*Description*
	Positive Completion Reply
211	System status or help reply
214	Help message
220	Service ready
221	Service closing transmission channel
250	Request command completed
251	User not local; the message will be forwarded
	Positive Intermediate Reply
354	Start mail input
	Transient Negative Completion Reply
421	Service not available
450	Mailbox not available
451	Command aborted: local error
452	Command aborted; insufficient storage
	Permanent Negative Completion Reply
500	Syntax error; unrecognized command
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command temporarily not implemented
550	Command is not executed; mailbox unavailable
551	User not local
552	Requested action aborted; exceeded storage location
553	Requested action not taken; mailbox name not allowed
554	Transaction failed

Figure 23.11 *Message transfer*

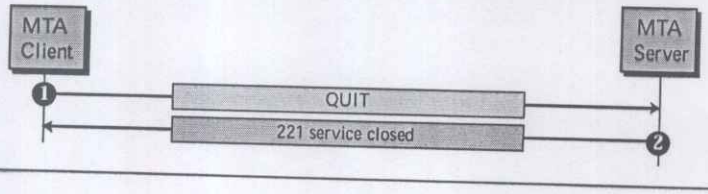


6. The server responds with code 354 (start mail input) or some other appropriate message.
7. The client sends the contents of the message in consecutive lines. Each line is terminated by a two-character end-of-line token (carriage return and line feed). The message is terminated by a line containing just one period.
8. The server responds with code 250 (OK) or some other appropriate code.

Connection Termination

After the message is transferred successfully, the client terminates the connection. This phase involves two steps (see Figure 23.12).

Figure 23.12 *Connection termination*



1. The client sends the QUIT command.
2. The server responds with code 221 or some other appropriate code.

After the connection termination phase, the TCP connection must be closed.

Example 23.1

Let us see how we can directly use SMTP to send an e-mail and simulate the commands and responses we described in this section. We use TELNET to log into port 25 (the well-known port for SMTP). We then use the commands directly to send an e-mail. In this example, forouzanb@adelphia.net is sending an e-mail to himself. The first few lines show TELNET trying to connect to the *adelphia* mail server.

```
$ telnet mail.adelphia.net 25
Trying 68.168.78.100...
Connected to mail.adelphia.net (68.168.78.100).
```

After connection, we can type the SMTP commands and then receive the responses as shown below. We have shown the commands in black and the responses in color. Note that we have added for clarification some comment lines, designated by the "=" sign. These lines are not part of the e-mail procedure.

```
================== Connection Establishment ===============
  220 mta13.adelphia.net SMTP server ready Fri, 6 Aug 2004 . . .
HELO mail.adelphia.net
  250 mta13.adelphia.net
=====================        Envelope          ==================
MAIL FROM: forouzanb@adelphia.net
  250 Sender <forouzanb@adelphia.net> Ok
RCPT TO: forouzanb@adelphia.net
  250 Recipient <forouzanb@adelphia.net> Ok
==================       Header and Body       ================
DATA
  354 Ok Send data ending with <CRLF>.<CRLF>
From: Forouzan
TO: Forouzan

This is a test message
to show SMTP in action.
.
```

```
=================== Connection Termination ================
  250 Message received: adelphia.net@mail.adelphia.net
QUIT
    221 mta13.adelphia.net SMTP server closing connection
Connection closed by foreign host.
```

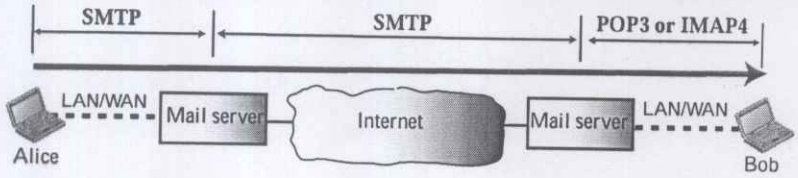
23.4 MESSAGE ACCESS AGENT: POP AND IMAP

The first and the second stages of mail delivery use SMTP. However, SMTP is not involved in the third stage because SMTP is a *push* protocol; it pushes the message from the client to the server. In other words, the direction of the bulk data (messages) is

from the client to the server. On the other hand, the third stage needs a *pull* protocol; the client must pull messages from the server. The direction of the bulk data are from the server to the client. The third stage uses a message access agent.

Currently two message access protocols are available: Post Office Protocol, version 3 (POP3) and Internet Mail Access Protocol, version 4 (IMAP4). Figure 23.13 shows the position of these two protocols in the most common situation (fourth scenario).

Figure 23.13 *POP3 and IMAP4*

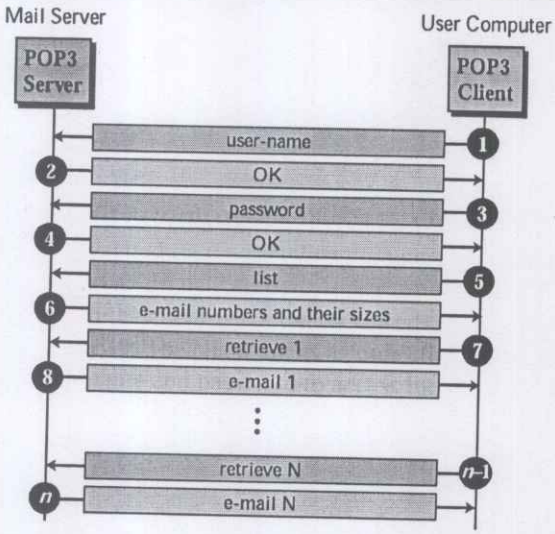


POP3

Post Office Protocol, version 3 (POP3) is simple and limited in functionality. The client POP3 software is installed on the recipient computer; the server POP3 software is installed on the mail server.

Mail access starts with the client when the user needs to download its e-mail from the mailbox on the mail server. The client opens a connection to the server on TCP port 110. It then sends its user name and password to access the mailbox. The user can then list and retrieve the mail messages, one by one. Figure 23.14 shows an example of downloading using POP3.

Figure 23.14 *POP3*



POP3 has two modes: the delete mode and the keep mode. In the delete mode, the mail is deleted from the mailbox after each retrieval. In the keep mode, the mail remains in the mailbox after retrieval. The delete mode is normally used when the user is working at her permanent computer and can save and organize the received mail after reading or replying. The keep mode is normally used when the user accesses her mail away from her primary computer (e.g., a laptop). The mail is read but kept in the system for later retrieval and organizing.

IMAP4

Another mail access protocol is **Internet Mail Access Protocol, version 4 (IMAP4).** IMAP4 is similar to POP3, but it has more features; IMAP4 is more powerful and more complex.

POP3 is deficient in several ways. It does not allow the user to organize her mail on the server; the user cannot have different folders on the server. (Of course, the user can create folders on her own computer.) In addition, POP3 does not allow the user to partially check the contents of the mail before downloading.

IMAP4 provides the following extra functions:

- A user can check the e-mail header prior to downloading.
- A user can search the contents of the e-mail for a specific string of characters prior to downloading.
- A user can partially download e-mail. This is especially useful if bandwidth is limited and the e-mail contains multimedia with high bandwidth requirements.
- A user can create, delete, or rename mailboxes on the mail server.
- A user can create a hierarchy of mailboxes in a folder for e-mail storage.

23.5 MIME

Electronic mail has a simple structure. Its simplicity, however, comes with a price. It can send messages only in NVT 7-bit ASCII format. In other words, it has some limitations. It cannot be used for languages other than English (such as French, German, Hebrew, Russian, Chinese, and Japanese). Also, it cannot be used to send binary files or video or audio data.

Multipurpose Internet Mail Extensions (MIME) is a supplementary protocol that allows non-ASCII data to be sent through e-mail. MIME transforms non-ASCII data at the sender site to NVT ASCII data and delivers it to the client MTA to be sent through the Internet. The message at the receiving site is transformed back to the original data.

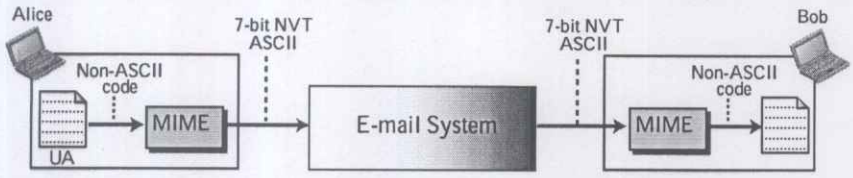
We can think of MIME as a set of software functions that transforms non-ASCII data to ASCII data and vice versa, as shown in Figure 23.15.

MIME Headers

MIME defines five headers that can be added to the original e-mail header section to define the transformation parameters:

1. MIME-Version
2. Content-Type

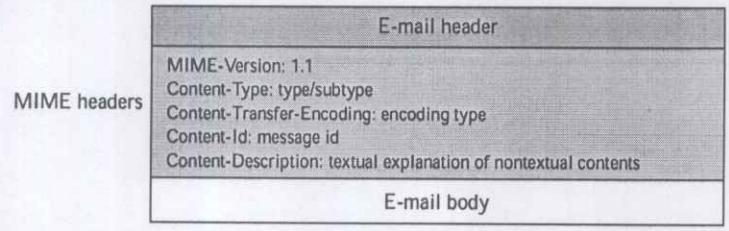
Figure 23.15 *MIME*



3. Content-Transfer-Encoding
4. Content-Id
5. Content-Description

Figure 23.16 shows the MIME headers. We will describe each header in detail.

Figure 23.16 *MIME header*



MIME-Version

This header defines the version of MIME used. The current version is 1.1.

Content-Type

This header defines the type of data used in the body of the message. The content type and the content subtype are separated by a slash. Depending on the subtype, the header may contain other parameters. MIME allows seven different types of data, listed in Table 23.3.

- **Text.** The original message is in 7-bit ASCII format and no transformation by MIME is needed. There are two subtypes currently used, *plain* and *HTML*.
- **Multipart.** The body contains multiple, independent parts. The multipart header needs to define the boundary between each part. A parameter is used for this purpose. The parameter is a string token that comes before each part; it is on a separate line by itself and is preceded by two hyphens. The body is terminated using the boundary token, again preceded by two hyphens and then terminated with two hyphens. Four subtypes are defined for this type: *mixed, parallel, digest,* and *alternative*. In the mixed subtype, the parts must be presented to the recipient in the exact order as in the message. Each part has a different type and is defined at the boundary. The parallel subtype is similar to the mixed subtype, except that the order of the parts is

Table 23.3 *Data Types and Subtypes in MIME*

Type	*Subtype*	*Description*
Text	Plain	Unformatted
	HTML	HTML format (see Appendix E)
Multipart	Mixed	Body contains ordered parts of different data types
	Parallel	Same as above, but no order
	Digest	Similar to Mixed, but the default is message/RFC822
	Alternative	Parts are different versions of the same message
Message	RFC822	Body is an encapsulated message
	Partial	Body is a fragment of a bigger message
	External-Body	Body is a reference to another message
Image	JPEG	Image is in JPEG format
	GIF	Image is in GIF format
Video	MPEG	Video is in MPEG format
Audio	Basic	Single channel encoding of voice at 8 KHz
Application	PostScript	Adobe PostScript
	Octet-stream	General binary data (eight-bit bytes)

unimportant. The digest subtype is also similar to the mixed subtype except that the default type/subtype is message/RFC822 as defined below. In the alternative subtype, the same message is repeated using different formats. The following is an example of a multipart message using a mixed subtype:

```
Content-Type: multipart/mixed; boundary=xxxx

      --xxxx
      Content-Type: text/plain;
      ...
      --xxxx
      Content-Type: image/gif;
      ...
      --xxxx--
```

- **Message.** In the message type, the body is itself an entire mail message, a part of a mail message, or a pointer to a message. Three subtypes are currently used: *RFC822, partial,* and *external-body.* The subtype RFC822 is used if the body is encapsulating another message (including header and the body). The partial subtype is used if the original message has been fragmented into different mail messages and this mail message is one of the fragments. The fragments must be reassembled at the destination by MIME. Three parameters must be added: *id, number,* and the *total.* The id identifies the message and is present in all the fragments. The number defines the sequence order of the fragment. The total defines the number of fragments that comprise the original message. The following is an example of a message with three fragments:

```
Content-Type: message/partial;
id="forouzan@challenger.atc.fhda.edu";
number=1;
total=3;
...
```

The subtype external-body indicates that the body does not contain the actual message but is only a reference (pointer) to the original message. The parameters following the subtype define how to access the original message. The following is an example:

```
Content-Type: message/external-body;
name="report.txt";
site="fhda.edu";
access-type="ftp";
...
```

- **Image.** The original message is a stationary image, indicating that there is no animation. The two currently used subtypes are *Joint Photographic Experts Group (JPEG)*, which uses image compression, and *Graphics Interchange Format (GIF)*.
- **Video.** The original message is a time-varying image (animation). The only subtype is *Moving Picture Experts Group (MPEG)*. If the animated image contains sounds, it must be sent separately using the audio content type.
- **Audio.** The original message is sound. The only subtype is basic, which uses 8-kHz standard audio data.
- **Application.** The original message is a type of data not previously defined. There are only two subtypes used currently: *PostScript* and *octet-stream.* PostScript is used when the data are in Adobe PostScript format. Octet-stream is used when the data must be interpreted as a sequence of 8-bit bytes (binary file).

Content-Transfer-Encoding

This header defines the method used to encode the messages into 0s and 1s for transport:

```
Content-Transfer-Encoding: <type>
```

The five types of encoding methods are listed in Table 23.4.

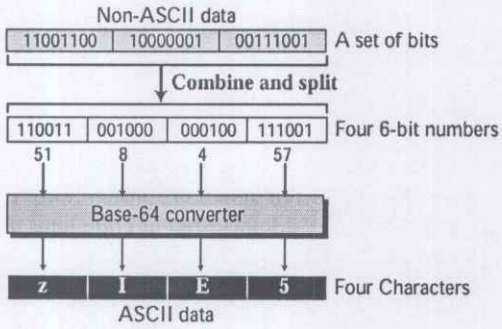
Table 23.4 *Content-Transfer-Encoding*

Type	*Description*
7bit	NVT ASCII characters and short lines
8bit	Non-ASCII characters and short lines
Binary	Non-ASCII characters with unlimited-length lines
Base64	6-bit blocks of data are encoded into 8-bit ASCII characters
Quoted-printable	Non-ASCII characters are encoded as an equal sign plus an ASCII code

- **7bit.** This is 7-bit NVT ASCII encoding. Although no special transformation is needed, the length of the line should not exceed 1,000 characters.

- **8bit.** This is 8-bit encoding. Non-ASCII characters can be sent, but the length of the line still should not exceed 1,000 characters. MIME does not do any encoding here; the underlying SMTP protocol must be able to transfer 8-bit non-ASCII characters. It is, therefore, not recommended. Base64 and quoted-printable types are preferable.
- **Binary.** This is 8-bit encoding. Non-ASCII characters can be sent, and the length of the line can exceed 1,000 characters. MIME does not do any encoding here; the underlying SMTP protocol must be able to transfer binary data. It is, therefore, not recommended. Base64 and quoted-printable types are preferable.
- **Base64.** This is a solution for sending data made of bytes when the highest bit is not necessarily zero. Base64 transforms this type of data to printable characters, which can then be sent as ASCII characters or any type of character set supported by the underlying mail transfer mechanism. Base64 divides the binary data (made of streams of bits) into 24-bit blocks. Each block is then divided into four sections, each made of 6 bits (see Figure 23.17).

Figure 23.17 *Base64*



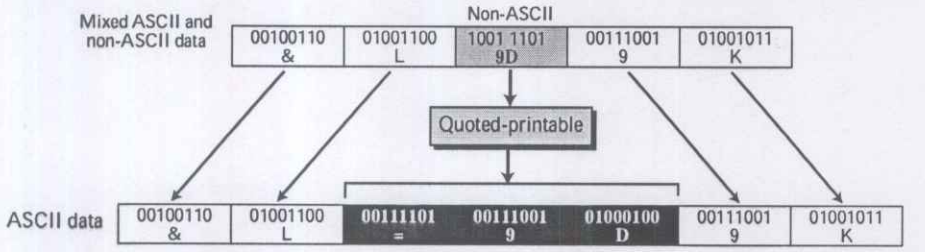
Each 6-bit section is interpreted as one character according to Table 23.5.

Table 23.5 *Base-64 Converting Table*

Value	*Code*	*Value*	*Code*	*Value*	*Code*	*Value*	*Code*	*Value*	*Code*	*Value*	*Code*
0	A	11	L	22	W	33	h	44	s	55	3
1	B	12	M	23	X	34	i	45	t	56	4
2	C	13	N	24	Y	35	j	46	u	57	5
3	D	14	O	25	Z	36	k	47	v	58	6
4	E	15	P	26	a	37	l	48	w	59	7
5	F	16	Q	27	b	38	m	49	x	60	8
6	G	17	R	28	c	39	n	50	y	61	9
7	H	18	S	29	d	40	o	51	z	62	+
8	I	19	T	30	e	41	p	52	0	63	/
9	J	20	U	31	f	42	q	53	1		
10	K	21	V	32	g	43	r	54	2		

- **Quoted-printable.** Base64 is a redundant encoding scheme; that is, 24 bits become four characters, and eventually are sent as 32 bits. We have an overhead of 25 percent. If the data consist mostly of ASCII characters with a small non-ASCII portion, we can use quoted-printable encoding. If a character is ASCII, it is sent as is. If a character is not ASCII, it is sent as three characters. The first character is the equal sign (=). The next two characters are the hexadecimal representations of the byte. Figure 23.18 shows an example.

Figure 23.18 *Quoted-printable*



Content-Id

This header uniquely identifies the whole message in a multiple message environment.

Content-Description

This header defines whether the body is image, audio, or video.

23.6 WEB-BASED MAIL

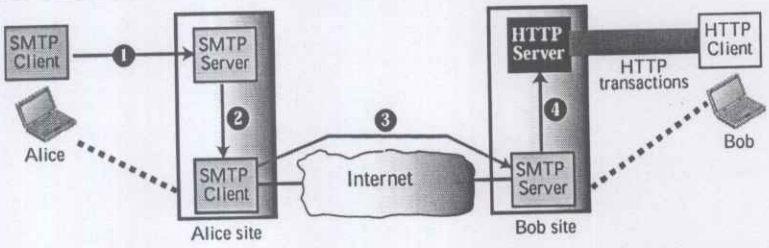
E-mail is such a common application that some websites today provide this service to anyone who accesses the site. Three common sites are Hotmail, Yahoo, and Google. The idea is very simple. Let us go through two cases:

Case I

In the first case, Alice, the sender, uses a traditional mail server; Bob, the receiver, has an account on a Web-based server. Mail transfer from Alice's browser to her mail server is done through SMTP. The transfer of the message from the sending mail server to the receiving mail server is still through SMTP. However, the message from the receiving server (the web server) to Bob's browser is done through HTTP. In other words, instead of using POP3 or IMAP4, HTTP is normally used. When Bob needs to retrieve his e-mails, he sends a request HTTP message to the website (Hotmail, for example). The website sends a form to be filled in by Bob, which includes the log-in name and the password. If the log-in name and password match, the list of e-mails is transferred from the Web server to Bob's browser in HTML format. Now Bob can

browse through his received e-mails and then, using more HTTP transactions, can get his e-mails one by one. This is shown in Figure 23.19.

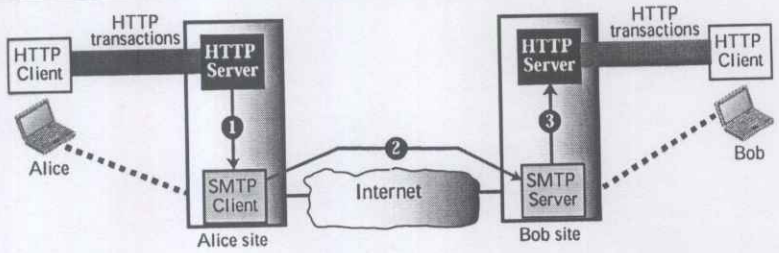
Figure 23.19 *Web-base e-mail, case I*



Case II

In the second case, both Alice and Bob use Web servers, but not necessarily the same server. Alice sends the message to the Web server using HTTP transactions. Alice sends an HTTP request message to her Web server using the name and address of Bob's mailbox as the URL. The server at the Alice site passes the message to the SMTP client and sends it to the server at the Bob site using SMTP protocol. Bob receives the message using HTTP transactions. However, the message from the server at the Alice site to the server at the Bob site still takes place using SMTP protocol. Figure 23.20 shows the idea.

Figure 23.20 *Web-based e-mail, case II*



23.7 E-MAIL SECURITY

The protocol discussed in this chapter does not provide any security provisions per se. However, e-mail exchanges can be secured using two application-layer securities designed in particular for e-mail systems. Two of these protocols, Pretty Good Privacy (PGP) and Secure MIME (SMIME) are discussed in Chapter 30 after we have discussed the basic network security.

23.8 FURTHER READING

For more details about subjects discussed in this chapter, we recommend the following books and RFCs. The items enclosed in brackets refer to the reference list at the end of the book.

Books

Several books give an easy but thorough coverage of electronic mail including [Com 06], [Ste 94], and [Tan 03], and [Kur & Ros 08].

RFCs

Several RFCs show updates on SMTP, including RFC 2821 and RFC 2822. POP3 is explained in RFC 1939. Several RFCs refer to MIME, including RFC 2046, RFC 2047, RFC 2048, and RFC 2049.

23.9 KEY TERMS

alias
body
connection establishment
connection termination
domain name
envelope
header
Internet Mail Access Protocol, version 4 (IMAP4)
local part
message access agent (MAA)
message transfer agent (MTA)
Multipurpose Internet Mail Extensions (MIME)
Post Office Protocol, version 3 (POP3)
Simple Mail Transfer Protocol (SMTP)
user agent (UA)

23.10 SUMMARY

- Electronic mail is one of the most common applications on the Internet. The e-mail architectures consists of several components such as user agent (UA), main transfer agent (MTA), and main access agent (MAA).
- The UA prepares the message, creates the envelope, and puts the message in the envelope. The mail address consists of two parts: a local part (user mailbox) and a domain name. The form is localpart@domainname. An alias allows the use of a mailing list.
- The MTA transfers the mail across the Internet, a LAN, or a WAN. The protocol that implements MTA is called Simple Main Transfer Protocol (SMTP). SMTP uses commands and responses to transfer messages between an MTA client and an MTA server. The steps in transferring a mail message are: connection establishment mail transfer, and connection termination.

- Two protocols are used to implement MAA: Post Office Protocol, version 3 (POP3) and Internet Mail Access Protocol, version 4 (IMAP4). These protocols are used by the receiver to pull messages from a mail server.
- Multipurpose Internet Mail Extension (MIME) allows the transfer of multimedia messages. MIME changes multimedia characters into ASCII characters that are transferable through the e-mail system.
- Web-based e-mails get popularity through sites that offer free e-mails for the users. In Web-based e-mail systems part of the data transfer is done through the SMTP protocol and part through the HTTP protocol.
- Secure e-mail is possible through two technologies: Pretty Good Privacy (PGP) and SMIME (Secure MIME).

23.11 PRACTICE SET

Exercises

1. A sender sends unformatted text. Show the MIME header.
2. A sender sends a JPEG message. Show the MIME header.
3. A non-ASCII message of 1,000 bytes is encoded using base64. How many bytes are in the encoded message? How many bytes are redundant? What is the ratio of redundant bytes to the total message?
4. A message of 1,000 bytes is encoded using quoted-printable. The message consists of 90 percent ASCII and 10 percent non-ASCII characters. How many bytes are in the encoded message? How many bytes are redundant? What is the ratio of redundant bytes to the total message?
5. Compare the results of Exercises 3 and 4. How much is the efficiency improved if the message is a combination of ASCII and non-ASCII characters?
6. Encode the following message in base64:

```
01010111 00001111 11110000 10101111 01110001 01010100
```

7. Encode the following message in quoted-printable:

```
01010111 00001111 11110000 10101111 01110001 01010100
```

8. Encode the following message in base64:

```
01010111 00001111 11110000 10101111 01110001
```

9. Encode the following message in quoted-printable:

```
01010111 00001111 11110000 10101111 01110001
```

10. Are the HELO and MAIL FROM commands both necessary? Why or why not?
11. In Figure 23.11 what is the difference between MAIL FROM in the envelope and the From in the header?
12. Why is a connection establishment for mail transfer needed if TCP has already established a connection?
13. Show the connection establishment phase from aaa@xxx.com to bbb@yyy.com.
14. Show the message transfer phase from aaa@xxx.com to bbb@yyy.com. The message is "Good morning my friend."
15. Show the connection termination phase from aaa@xxx.com to bbb@yyy.com.
16. User aaa@xxx.com sends a message to user bbb@yyy.com, which is forwarded to user ccc@zzz.com. Show all SMTP commands and responses.
17. User aaa@xxx.com sends a message to user bbb@yyy.com. The latter replies. Show all SMTP commands and responses.
18. In SMTP, if we send a one-line message between two users, how many lines of commands and responses are exchanged?

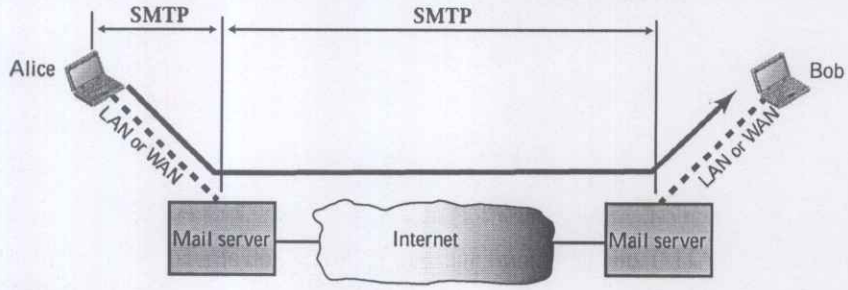
Research Activities

19. A new version of SMTP, called ESMTP, is in use today. Find the differences between the two.
20. Find information about the *smileys* used to express a user's emotions.

23.3 MESSAGE TRANSFER AGENT: SMTP

The actual mail transfer is done through message transfer agents (MTAs). To send mail, a system must have the client MTA, and to receive mail, a system must have a server MTA. The formal protocol that defines the MTA client and server in the Internet is called **Simple Mail Transfer Protocol (SMTP).** As we said before, two pairs of MTA client-server programs are used in the most common situation (fourth scenario). Figure 23.8 shows the range of the SMTP protocol in this scenario.

Figure 23.8 *SMTP range*



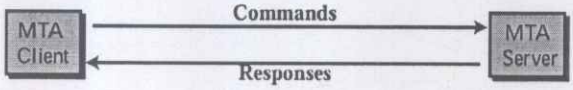
SMTP is used two times, between the sender and the sender's mail server and between the two mail servers. As we will see shortly, another protocol is needed between the mail server and the receiver.

SMTP simply defines how commands and responses must be sent back and forth. Each network is free to choose a software package for implementation. We will discuss the mechanism of mail transfer by SMTP in the remainder of the section.

Commands and Responses

SMTP uses commands and responses to transfer messages between an MTA client and an MTA server (see Figure 23.9).

Figure 23.9 *Commands and responses*



Each command or reply is terminated by a two-character (carriage return and line feed) end-of-line token.