In this chapter:

- *What Is the Internet Mail Model?*
- *Why Follow the Internet Mail Model?*
- *Examples*

1

The Internet Mail Model

IMAP stands for Internet Mail Access Protocol. For much of the Internet email system administrator community, it also stands for flexibility, speed, and power. These attributes come from abilities like being able to store all a user's mail centrally, not demand that she store copies of it on each workstation from which she wants to access her mail. IMAP users also store their mail in an arbitrary number of server-side mailboxes, each of which they can move messages into or out of with any IMAP client. When an IMAP user checks her mail, her client need only download some of the header for each message, not the entire message. When she sees messages in her index she wants to retrieve, she can decide which parts of the 13-part message she wants to download, and which she doesn't. These are capabilities that no other standardized mail access protocol permits.

Before we dive into a more detailed discussion of IMAP, though, let's talk about Internet mail in general. Much of this discussion is a definition of terms. In defining those terms, however, we're discussing the language that is the bedrock of Internet electronic mail.

What Is the Internet Mail Model?

The Internet Mail Model, like the Internet itself, is a collection of standardized components all acting with a common goal. In the case of email, the goal is to provide the framework for carrying electronic messages between one user and another. Each of the end users may be on very different platforms. Their respective sites may have vast geographic, technological, and social differences. Those differences demand that the framework be at once both robust and flexible. The Internet's email framework consists of agents, mailstores, and standards. It may help you to reference Figure 1-1 as you read the chapter. The figure shows how the agents, mailstores, and standards work together.

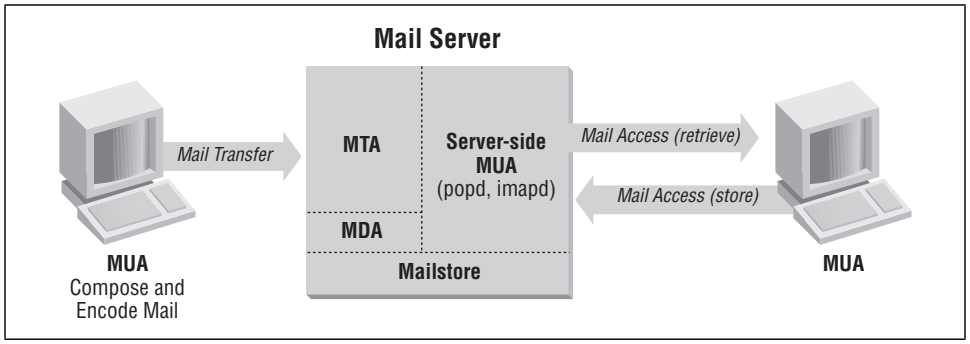

Figure 1-1. Email cycle of life

The Agents (MUA, MTA, MDA)

The software programs that handle Internet messages are called agents. There are three types of Internet messaging agents: the Mail Transport Agent (MTA), Mail Delivery Agent (MDA), and Mail User Agent (MUA).

MTA

An MTA (Figure 1-2) is a program that transmits and receives messages between messaging sites. The sending MTA accepts messages from end user client software and transmits it to a receiving MTA. The receiving MTA receives messages from the sending MTA, determines whether or not the recipient resides locally on the receiving MTA (server) system, and then hands off the message for delivery. If the message is destined for a user on the receiving MTA's system, then the receiving MTA hands the message off to a Message Delivery Agent (MDA) such as */bin/mail*. If the user is not on the local system, then the receiving MTA acts as a sending MTA to pass the message on to the MTA on the remote system.

Figure 1-3 shows typical Internet message headers. Each "Received" header line represents transit through a separate MTA. MTAs do not touch the mailstore. They delegate that work to the MDA.

MDA

The MDA is the trench soldier: the grunt of Internet messaging. All the MDA knows is how to determine which local user the message is destined for and how to put the message in the correct place in the mailstore. Actually, that's not quite *all* the MDA knows. Some super-charged MDAs, such as Procmail, have vast delusions of grandeur, but we'll cover that later in the book. All that's essential to know right now is that the MTA hands the MDA each Internet message destined for a local user and that the MDA is responsible for knowing where to place it in the mailstore.

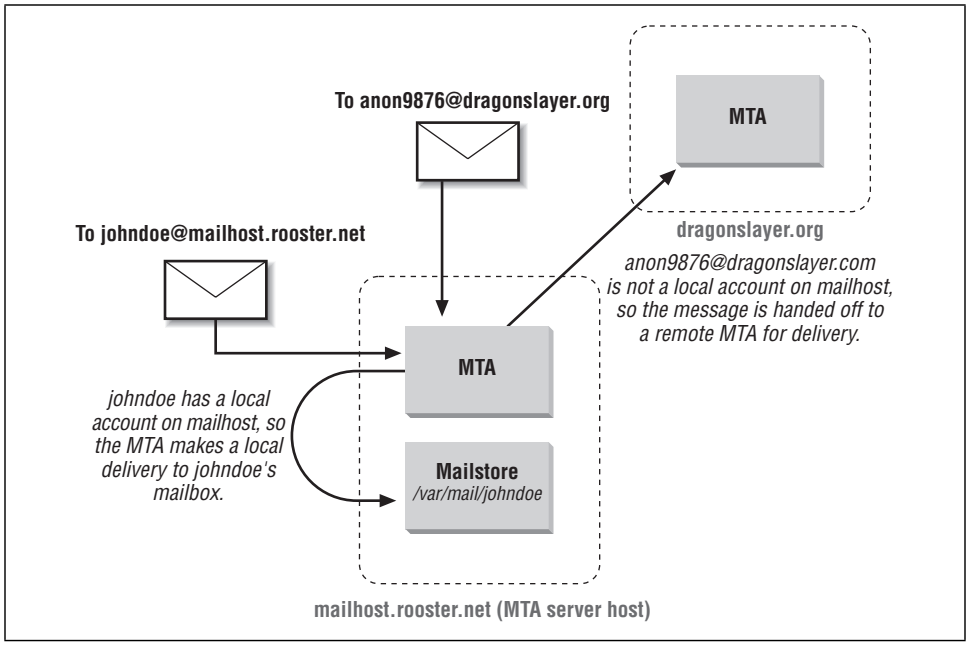

Figure 1-2. The MTA

```
From janedoe@acme.com Tue Jan 27 23:06:24 2000
Return-Path: <janedoe@acme.com>
Received: from mailhost.widget.com (mailhost.widget.com [12.9.120.1.1])
        by workstation1.widget.com (8.8.7/8.8.7) with ESMTP id XAA25079
        for <qpublic@workstation1.widget.com>; Tue, 27 Jan 2000 23:06:23 -0600 (CST)
Received: from acme.com (janedoe@fohnix.acme.com [192.245.137.2])
        by mailhost.widget.com (8.8.8/.8.8.8) with SMTP id XAA09696
        for <qpublic@widget.com>; Tue, 27 Jan 2000 23:10:53 -0600 (CST)
Received: by acme.com id AA27837
  (5.67a/IDA1.5hp for qpublic@widget.com); Tue, 27 Jan 2000 23:10:49 -0600
From: janedoe <janedoe@acme.com>
Message-Id: <200001280510.AA27837@acme.com>
Subject: Re: 1/26/00 Meeting Notes
To: qpublic@widget.com
Date: Tue, 27 Jan 2000 23:10:48 -0600 (CST)
In-Reply-To: <200001271545.JAA24165@workstation1.widget.com> fr0m "J.Q. Public" at Jan 27,
98 09:45:11 am
Reply-To: janedoe@acme.com
Return-Receipt-To: janedoe@acme.com
X-Mailer: ELM [version2.4 PL24]
Mime-Version: 1.0
Content-Type: text/plain; charset=US-ASCII
Content-Transfer-Encoding: 7bit
```

Figure 1-3. Typical Internet email message header

MUA

As we've seen, the MTA, responsible for knowing how to route every conceivable type of legitimate Internet message, is by far the most educated part of the messaging model. The MDA is the hardest-working component. The MUA, on the other hand, is charged with being the most glamorous part of the IMM framework. The MUA is the interface between the MTA and the most unpredictable component of the IMM: the user himself. Strictly speaking, the MUA retrieves mail from the mailstore and sends new messages upstream to the MTA.

The MUA typically retrieves messages from the mailstore in one of three ways: by using a mail access protocol like IMAP or POP, by using a remote file access protocol, or by accessing local files. In the case of IMAP and POP, the MUA function is split between two pieces of software: the mail client and a corresponding server process that mediates between the client and the mailstore using POP or IMAP (see Figure 1-4).

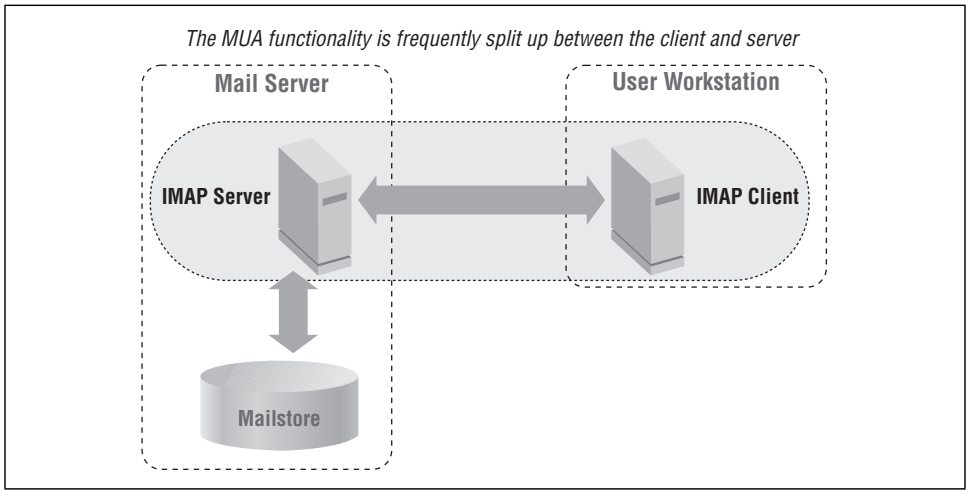

Figure 1-4. MUA function

The Mailstore

The mailstore is the filing cabinet of the mail system. When a user receives a piece of email, it's deposited into his portion of the mailstore. To retrieve his email, he uses an MUA to peer into the mailstore and view his messages. Not too long ago, most mailstores consisted of a single text file per user containing the user's messages concatenated together within that file. Today, mailstores are implemented in a great variety of ways. The volume of email crossing the Internet has grown meteorically, increasing the demand for more efficient and accessible ways of storing mail. Technologies like IMAP that permit hierarchical organization of the user's mail within his part of the server mailstore have resulted in products that abandon

the traditional flat file mailstore. Depending on your messaging product, your mailstore may consist of a single file per message, or all the users at your site may share a single high-performance database in which their messages are stored. The Internet standards don't address implementation details of the mailstore—its concern is primarily with transport and format.

The Standards (RFC 822, MIME, SMTP/ESMTP, POP, IMAP)

Internet standards are defined in documents called Requests for Comments (RFCs). The Internet Engineering Task Force (IETF) is a collection of working groups, each of which is a collection of volunteers collaborating to define new RFCs for the Internet or to revise existing ones. RFCs are the deliverable produced by the IETF working groups. All IETF standards are RFCs, but not all RFCs are standards. Some RFCs are experimental protocols, and some are commentaries on existing practices. Some are somewhat transparent attempts to publish proprietary methods and convince the Internet community to embrace them as standards, and some (usually the ones published on April 1st) are wry bits of geek humor.

In this chapter, though, we skim through the standards that are germane to Internet mail. We group these standards into several categories: formatting and encoding mail, mail transfer, and mail access.

Formatting and encoding mail

RFC 822 (Standard)—Standard for the format of Internet text messages
: This is the big kahuna. This RFC lies at the core of all Internet-based messaging. It defines plaintext messages, which themselves consist of a header in a common format, a single blank line, and a body. If ASCII is the DNA of Internet messaging, RFC 822 messages are the chromosomes…or maybe the cells…oh well.

RFC 2076 (Informational)—Common Internet Message Headers
: RFC 822 defines a standard format Internet message header. RFC 2076 goes into greater detail about the specific header lines, their purpose, and their individual contents.

Multi-Purpose Internet Mail Extensions
: The primary motivator for the creation of the working group that created MIME was to support non-ASCII character sets necessary for email in languages other than English. A secondary motivator was a requirement for a standard way to send attachments. Less important, but also a motivating factor, was the need for a standard way to send multimedia content. MIME came about through the realization that a single solution could address all three needs.

Figure 1-5 shows how a graphics file might have been conveyed in days of yore, alongside how it would probably be transported via MIME today. The figure doesn't really do justice to the benefits of MIME. With manual encoding, users were usually stuck to just sending single files in their messages. With MIME, users can send attachments containing any kind of data, of arbitrary length. MIME messages can point to files or other data outside the mail message. The only functional limitation is that the MUA on each end must know how to handle the particular MIME type. If you send an attachment to a colleague of type *application/postscript*, and her mail client doesn't know how to handle that type, you've gained little over manual encoding.

The core of MIME itself is set forth in five RFCs:

- RFC 2045 (Draft standard)—MIME Part 1: Format of Internet Message Bodies
- RFC 2046 (Draft standard)—MIME Part 2: Media Types
- RFC 2047 (Draft standard)—MIME Part 3: Message Header Extensions for Non-ASCII Text
- RFC 2048 (Draft standard)—MIME Part 4: Registration Procedures
- RFC 2049 (Draft standard)—MIME Part 5: Conformance Criteria and Examples

Together, these RFCs define the mail headers, message structure, and data characterization that should permit any computer file or data stream to be conveyed via email without extraordinary demands on the intermediate mail gateways or the receiving client.

MIME appeared on the scene before use of HTTP was widespread. Shortly after MIME began being used, the Web became popular, and suddenly, people needed to send URLs via email. Sending the URL to a file instead of the file itself is popular—instead of sending the file as an attachment, the user sends a pointer to the file instead. Large mail attachments can be problematic. Many ISPs still use only POP service and implement it in such a way that forces users to download all new email without picking and choosing particular messages. Messages with large attachments make downloading POP mail painfully time consuming. SMTP servers often have size limits on messages they'll accept (typically 10–20 MB per message). If a message has a large attachment, it could be rejected by the SMTP server. Sending the URL instead of the file itself gets around those problems. It's no wonder URLs are a popular way of conveying information stored in large files.

A common (much to the consternation of traditionalists) use for MIME nowadays is to send two versions of your message: a *text/plain* version and a *text/html* version with more formatting.

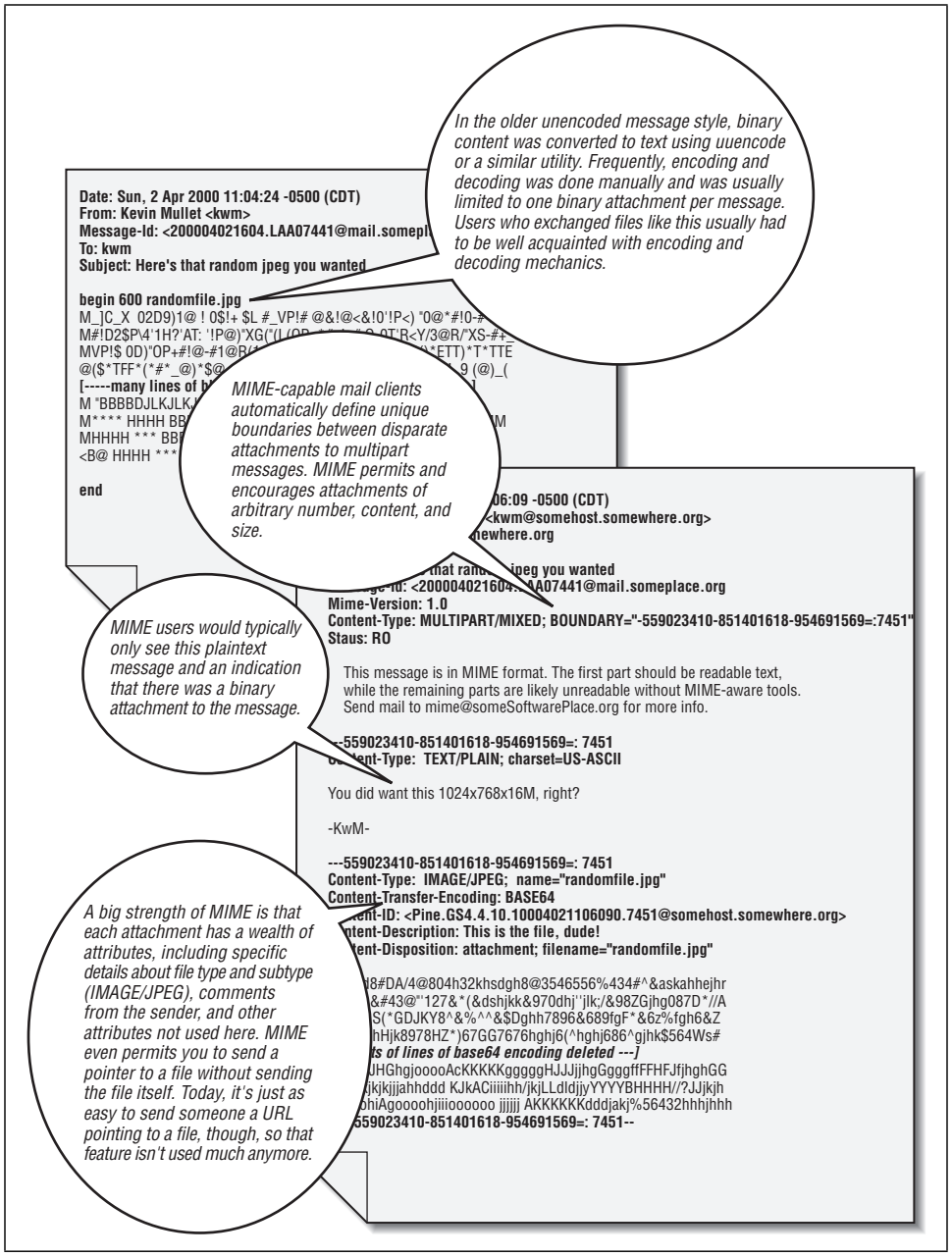

Figure 1-5. Transport using MIME

Mail transfer

The RFCs mentioned in the previous section provide the framework for the construction of an email message. Now we need some standards for describing how that message is conveyed upstream to other hosts on the Internet. The most important of these is RFC 821: Simple Mail Transfer Protocol (SMTP). SMTP defines how mail is transported from one place to another, whether that transport is from your MUA to the MTA or between two MTAs. Figure 1-6 shows an example of a typical SMTP conversation between an MUA and an adjacent MTA. There are additional RFCs that augment SMTP into what is frequently called Extended SMTP, but RFC 821 remains the core standard defining Internet mail transport. A list of additional RFCs may be found at the Internet Mail Consortium site (*http://www.imc.org*).

Mail access

We've looked at standards for conveying mail to a server. We've seen yet other standards specifying how the email should be formatted. It only stands to reason that we now need standards for accessing and managing the mail in a mailstore. POP and IMAP are two of those standards. Later, we'll go into vivid detail about the differences between IMAP and POP. A mail access protocol is a means by which the mail client software may perform operations on messages that have already made it to the mailstore. Note that we don't just say "read" messages that are in the mailstore. Although POP is a "read-heavy" protocol, IMAP permits users to add messages to the mailstore, move them around, and change their attributes or the degree of access other users have to them.

Why Follow the Internet Mail Model?

Unlike closed commercial mail systems, Internet messaging is defined by a series of specifications that are free and open for all. Consequently, an Internet messaging system can be built using a variety of products from several vendors, with assuring that each product will interoperate with all the other products. This is especially important because the open standards defining the Internet itself make for a highly complex environment in which each component of a messaging model must know what to expect of the others. Communication standardization is the soul of the Internet.

The Internet repeatedly proves the fact that there is no problem so large that it cannot be solved by the principle of "divide and conquer." A browse through some of the messaging-related RFCs from the early 1970s shows how early ARPAnet* engineers struggled to send email back and forth. Early on, they relied on the

* Advanced Research Projects Agency network (R.I.P. 1970–1990).

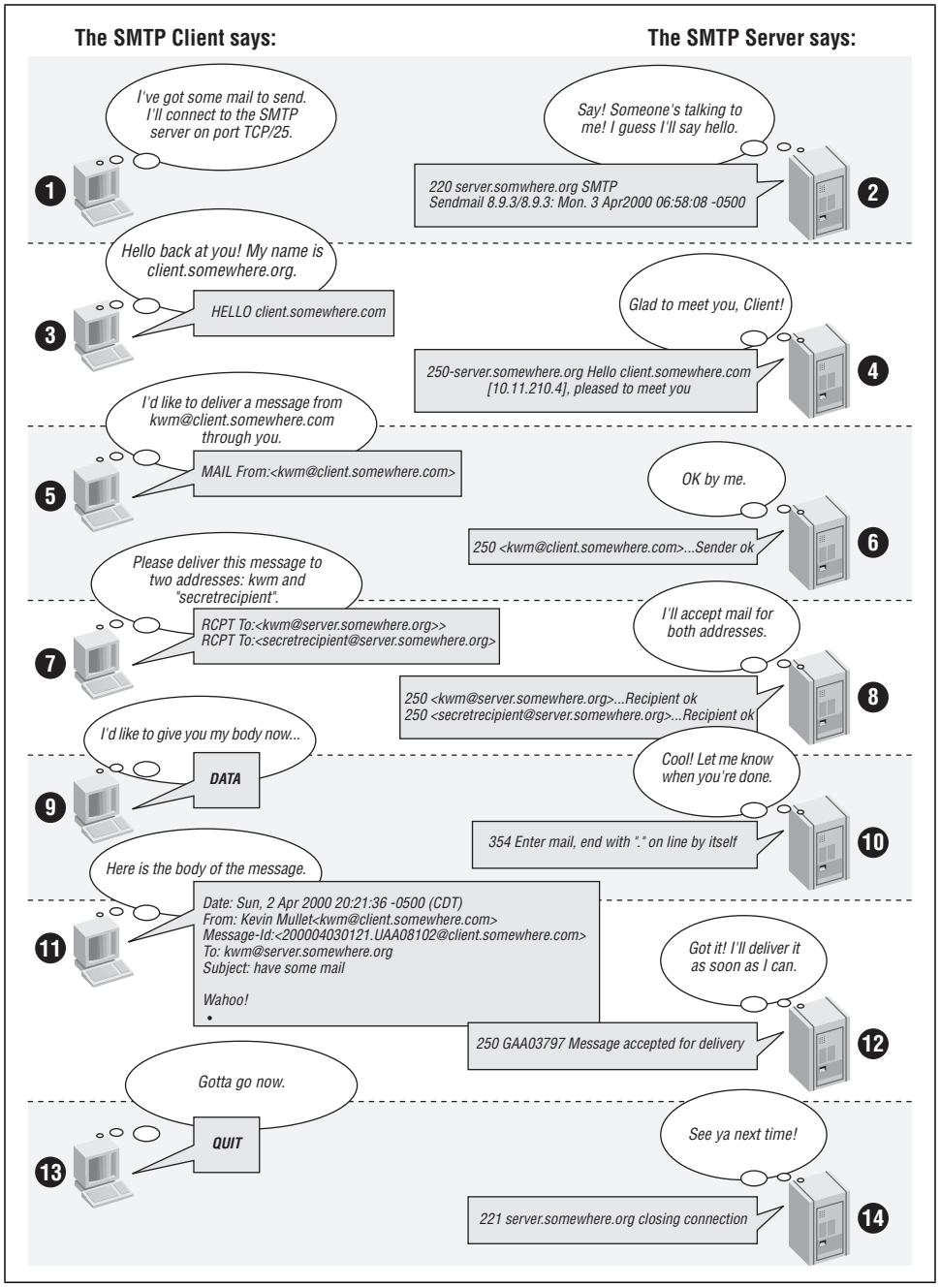

Figure 1-6. Under the hood with SMTP

weak model embraced by many modern-day closed-source solutions: email as a file-copying program. RFC 469 (circa 1973) kicks around the idea of an email infrastructure based on passing files around using FTP. Even in those early discussions, innovative ideas were alluded to, such as active links to other documents, redirection to central document repositories,* permanent email archives, and content from arbitrary non-textual sources. Those ideas suggested the need for a hierarchy of standards and protocols.

Before too long, however, the problem of how to best implement email was divided and conquered. As we've seen, a special-purpose protocol (SMTP) was developed exclusively for transport of the messages from one place to another. Other protocols were developed for accessing the mail once it arrived at the destination mailstore (IMAP and POP). Standards were developed for the format of messages themselves and for encapsulating the payload of those messages (MIME). With the standards in hand, it was an entirely manageable task for the various parts that make up Internet email to interoperate, because the means of doing so was a widely discussed and published set of industry standards. System administrators no longer needed to worry about whether or not the sendmail, QMail, and Postfix MTAs would talk to each other. Nor did they need to concern themselves with whether any IMAP-compliant client would be able to retrieve mail from their UW IMAP, Cyrus, or proprietary IMAP server.

Now that we have a fair idea of what are the major components of Internet email and why we have that model in the first place, let's look a bit closer at some real-world email transactions.

Examples

The stage is set, and now we're ready to introduce the players.

Mail Routing

Think about what happens when mail is sent by a user on a PC using Netscape to someone who uses the text-mode, Unix-based MUA, PINE. As seen in Figure 1-7, the sender (Netscape user) sends the message by SMTP to the MTA running on the sender's ISP's mail server. Once the mail arrives at the mail server, the MTA asks the MDA to store the message in the local message store. When the recipient reads his mail, he runs PINE on the mail server itself, which views his INBOX as a local file. Although in the example, no *network* mail access protocol is involved while reading the mail, you might think of the mail access protocol in this case as a way

* Sixteen years before the Web and 18 years before Gopher!

of converting the mail into something the client can understand. In those terms, the mail access protocol can be thought of as being hardcoded in the client.

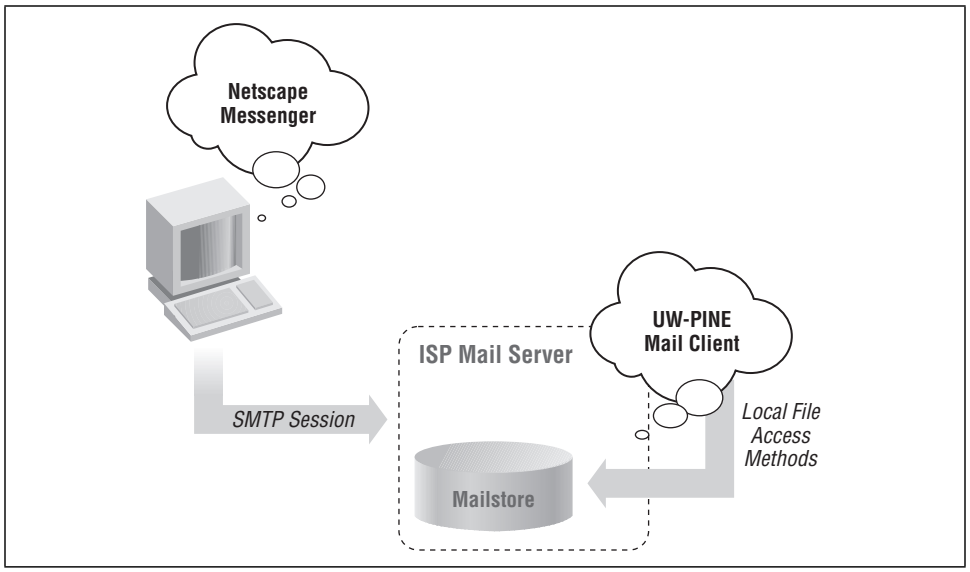

Figure 1-7. Mail routing example

Examples of Agents

An agent is a program that performs a task on behalf of a human user, either directly at the human's behest or indirectly under instruction from another agent. Usually, a chain of agents work in concert to get the information from its point of origin to its intended destination. Unlike the previous examples, Internet email agents perform their duties in the open and usually with the full consent and support of all the participants. Here are some examples of some of the more popular Internet email agents. We'll go into more detail about some of these later, but it's always good to solidify theory with some real-world examples as soon as possible.

MTAs

Any discussion of Internet MTAs could also easily be entitled "Sendmail and some alternatives." The old slogan that "nobody ever got fired for buying IBM" could be adapted equally well to sendmail. Leave it to the Internet community, though, not to leave well enough alone. Recently, viable contenders have come onto the scene, such as Qmail and Postfix.

sendmail is to email what the Internet is to networking. The genesis of sendmail was Eric Allman's Delivermail, which he wrote to connect ARPAnet email to numerous other networks' email. As demands for routing more types of email

were made of Delivermail, Allman rewrote it to be much more flexible and user-configurable, and sendmail was born. The ever-increasing flexibility acquired in the twenty-plus-year history of sendmail has come at the price of a complex interface.

Both Qmail and Postfix represent the strategy of performing the MTA mission with many smaller utilities, each of which carries out a narrow part of the MTA function rather than using one larger piece of software. Each has arguably resulted in a system with more straightforward configuration at the expense of a greater degree of process complexity. Both lay claim to being able to transit a bodacious number of messages using a ridiculously small amount of hardware. Which one you're likely to prefer is probably one of those coffee or tea propositions where you'll just have to try each and see which you like better.

We'll go into more detail about MDAs and MUAs later, but let's briefly touch on some of them.

MDAs

As the agent that actually places the email in the user's mailbox, the MDA is probably one of the most often used programs on your mail server. While a single MTA process could handle a large amount of mail bound for one site or a large number of messages from one process, each MDA process usually lives and dies for the processing of a single message.

Examples of MDAs include */bin/mail*, */usr/lib/mail.local*, *procmail*, and Cyrus *deliver*. Each of these programs takes an RFC 822–formatted message on standard input and delivers it to a mailbox. In the case of *procmail*, however, a mailbox could be local or remote, so *procmail* is one of those examples that could be either an MDA, an MUA, or both, depending on the nuances of how it's used.

MUAs

As the frontend to the mail system, the MUA is the highest-profile element in the chain of elements between each Internet email sender and recipient. Ironically, the failure of the MTA or MDA has much greater impact than the failure of a single MUA.

PINE, Eudora, Microsoft Outlook, Netscape Messenger, and Mulberry are all MUAs. Additionally, even sendmail and *imapd* can be considered MUAs. sendmail is frequently used as an MUA to generate messages programmatically. If you were to fire off a sendmail process as shown below, you would be running sendmail as an MTA. The following command starts sendmail running as a daemon and tells it to process the queue every 15 minutes:

```
% /usr/lib/sendmail -bd -q15m
```

If, however, you fired off a sendmail process as follows, you would be using sendsmail as an MUA:

```
% echo "Subject: Hey you!"|/usr/lib/sendmail -v kwmullet@yahoo.com
kwmullet@yahoo.com... Connecting to mx1.mail.yahoo.com. via esmtp...
220 mta220.mail.yahoo.com ESMTP
>>> EHLO security.unt.edu
250-mta220.mail.yahoo.com
250-PIPELINING
250 8BITMIME
>>> MAIL From:<kwm@security.unt.edu>
250 ok
>>> RCPT To:<kwmullet@yahoo.com>
250 ok
>>> DATA
354 go ahead dd
>>> .
250 ok dirdel
kwmullet@yahoo.com... Sent (ok dirdel)
Closing connection to mx1.mail.yahoo.com.
>>> QUIT
221 mta220.mail.yahoo.com
%
```

In addition, each time you use your MUA to connect to an IMAP server, you create an IMAP process on the server exclusively to service the IMAP requests between your MUA and the mailstore. That process is also considered part of the MUA—a server-side MUA. To further distinguish between the server and the client side of the MUA, let's refer to the client side as the MUA and the server side as the Mail Access Agent.

We'll have plenty of details later about MUAs and MDAs. We won't have so much information about MTAs, because they're "SMTP plumbing" in the scope of this book.