REIHE INFORMATIK
004/95

Multi-Media Mail in heterogeneous Networks

Gerd Danner und Walter Müller
Universität Mannheim
L15,16
D-68131 Mannheim

Multi-Media Mail
in
heterogeneous Networks

a study by

Gerd Danner and Walter Müller

Lehrstuhl für Praktische Informatik IV
Universität Mannheim

68131 Mannheim

Mannheim, March 1995

1 Motivation

Since its publication in 1982, RFC 822 (Request for Comments) has defined the standard format for almost any textual mail messages in the Internet. Together with the domain name service and the SMTP service, it possibly provides one of the most frequently used means of communication in the Internet and beyond. Though it was state of the art in 1982, during the intervening years it has become more and more clear that in the long run some limitations of this standard imply unacceptable restrictions for the fast-growing user community.

The standard described in RFC 822 is limited to a definition of pure messages. The encoding scheme chosen for the transmission is a 7-bit ASCII character set. However, most European or Asian languages need a more powerful character set. Hence messages in these languages cannot be properly represented with the RFC 822 standard. Furthermore, the standard does not provide any means to include non-textual objects such as images, audio-information or video-streams into a mail body.

Existing coding tools, such as uuencode and uudecode, allow the user to transform binary data to an RFC 822-compliant format and can provide a temporary work-around. But they do not solve the actual problem of a standard for message formats lacking the flexibility to satisfy all future needs.

Not least because of the development of multimedia workstations with high-resolution graphical displays and the ability to process audio data, a demand for a new standard for message passing and encoding that provides "natural" means to handle multimedia information in extension of pure text messages, has become more and more desirable. Possible extensions to the standard could be mail bodies that contain images, audio or video in addition to pure ASCII information.

One example of such an extension to the original standard is the MIME specification published in 1992. MIME is an acronym that stands for Multipurpose Internet Mail Extensions. In addition to pure textual mail bodies, MIME allows

- specification of multiple objects in one message
- messages to be written without any limitations concerning the line or text length
- use of additional character sets
- transmission of binary coded data
- use of images, audio, video within your mail, thus creating multimedia messages.

Within the scope of this study several different MIME-based multimedia mailing systems were installed on workstations from different vendors located in heterogeneous networks and tested for functionality and compatibility.

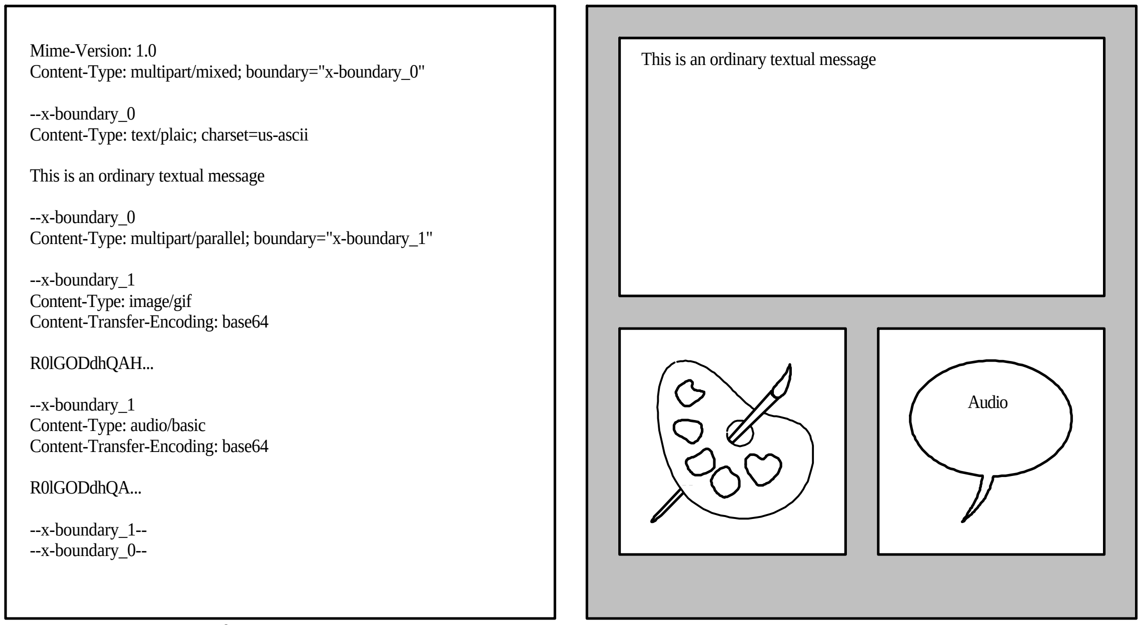
The aim was to use publicly available mailing systems to exchange mails containing high-definition color images, audio and video amongst ordinary textual information in-between heterogeneous systems and to "visualize" the incoming mail on a workstation's graphical display and/or audio device. To test and to assess the functionality and interoperability of the different mail user agents was of special interest. Furthermore, different Local Area (LAN) and Wide Area (WAN) Networking technologies, in particular Ethernet, FDDI and X.25 were used as a transmission medium for multimedia mails.

2 MIME

2.1 The MIME Standard

2.1.1 Basics

The MIME (Multipurpose Internet Mail Extensions) standard is one approach to solve the limitations implied by RFC 822, such as a maximum line length of 1000 characters, a limited overall length, and pure textual information in US-ASCII. It is defined in the Internet Standard document RFC 1521 and describes a new way to specify the format of Internet message bodies. It is capable of dealing with non-textual body parts such as images, postscript, audio and video without introducing any serious incompatibilities to the existing world of RFC 822 mail. It further allows users to have multiple objects in a single message, unlimited line and overall length in text bodies and to use non-ASCII character sets. Figure 1 shows an example of a MIME message format and the corresponding user view.



```
Mime-Version: 1.0
Content-Type: multipart/mixed; boundary="x-boundary_0"

--x-boundary_0
Content-Type: text/plaic; charset=us-ascii

This is an ordinary textual message

--x-boundary_0
Content-Type: multipart/parallel; boundary="x-boundary_1"

--x-boundary_1
Content-Type: image/gif
Content-Transfer-Encoding: base64

R0lGODdhQAH...

--x-boundary_1
Content-Type: audio/basic
Content-Transfer-Encoding: base64

R0lGODdhQA...

--x-boundary_1--
--x-boundary_0--
```

Figure 1: MIME message format

2.1.2 New Header Fields

Mime defines four new types of header fields, extending the RFC 822 standard. These are:

- A MIME-Version header. This field is used by the mailing software to distinguish between messages conforming to the MIME standard and non-conformant message types. Furthermore, a version number allows the future use of different MIME specifications that will extend the current standard.

- A Content-Type header field, which is used to specify the type and subtype of the message body.
- A Content-Transfer-Encoding header field that specifies the coding mechanism used to encode the data to 7-bit-ASCII, allowing the messages to be transmitted using the established Internet email mechanisms.
- Two optional header fields, Content-ID and Content-Description. They can be used to further describe the data in a message body.

The definition of MIME is such that it can be easily extended. It is expected that the set of content type/subtype pairs and their associated parameters will grow with time. Several other MIME-fields, such as character set names, are likely to have new values defined over time. To provide an orderly global development of these definitions, a registration process using the Internet Assigned Numbers Authority (IANA) as a central registry for such values is described in the MIME document.

To provide interoperability between different mailers using the MIME standard to send non-textual information, a minimal subset of the mechanisms mentioned above is defined in RFC 1521, to be supported by a software claiming to conform to MIME.

2.1.3 The MIME-Version Header Field

The MIME standard is capable of handling RFC 822 messages. It also introduces some new content types for handling non-textual mail bodies. If a non-textual mail body is received, the mailing software is expected to process the body and to perform additional tasks to display the MIME message. To provide a way for the mailer to distinguish between messages conforming to the MIME standard and non-conformant message types, a MIME-Version header field is introduced. Furthermore, a version number allows the future use of different MIME specifications that will extend the current standard. Actually, as of now, only one version of the MIME standard is established. Messages that comply with the standard must include a header field with the following verbatim text:

```
MIME-Version: 1.0
```

The MIME-Version header field must appear at least at the top level of a message. In case of a multipart message, the version header is not required to appear in each body part. It is required for the embedded headers of a body of type "message" if the embedded message itself claims to be MIME- compliant.

There is no mechanism that describes the behavior of a MIME-compliant mail-reading software when encountering messages with version numbers other than "1.0". However, conformant software should at least check the version number and warn the user if a message in an unsupported version arrives.

2.1.4 The Content-Type Header Field

The purpose of the content-type field is to provide enough information to the receiving mail user agent to allow the UA to pick an appropriate agent or mechanism to present the data to the user or otherwise deal with the data in an appropriate manner.

The content-type header field is used to specify the nature of the data in the body of an entity, by giving type and subtype identifiers, and by providing auxiliary information that may be required for certain types. After the type and subtype names, the remainder of the header field is simply a set of parameters, specified in an attribute/value notation. The set of meaningful parameters differs for the different types. The ordering of parameters is not significant. Among the defined parameters is a charset parameter by which the character set used in the body may be declared. Comments are allowed in accordance with RFC 822 rules for structured header fields.

In general, the top-level content type is used to declare the general type of data, while the subtype specifies a specific format for that type of data. Thus, a content type of "image/xyz" is enough to tell a user agent that the data is an image, even if the user agent has no knowledge of the specific image format "xyz". Such information can be used, for example, to decide whether or not to show a user the raw data from an unrecognized subtype. Such an action might be reasonable for unrecognized subtypes of text, but not for unrecognized subtypes of image or audio. For this reason, registered subtypes of audio, image, text, and video should not contain embedded information that is really of a different type. Such compound types should be represented using the "multipart" or "application" types.

Parameters are modifiers of the content-subtype and do not fundamentally affect the requirements of the host system. Although most parameters make sense only with certain content types, others are "global" in the sense that they might apply to any subtype. For example, the "boundary" parameter makes sense only for the "multipart" content type, but the "charset" parameter might make sense with several content types.

An initial set of seven content types is defined in RFC 1521. It is expected that additions to the set of supported types can generally be made by creating new subtypes of these initial types. In the future, more top-level types may be defined only by an extension to the standard. If, for any reason, another primary type is to be used, it must be given a name starting with "X-" to indicate its non-standard status and to preclude a potential conflict with a future official name.

In the Extended BNF notation of RFC 822, a content-type header field value is defined as follows:

Content-Type := type "/" subtype *[";" parameter]

The seven initial content types are detailed in the following. They are:

Application

This content type indicates data that do not fit into any of the other categories, e.g. uninterpreted binary data or data to be processed by mail-based uses of application programs. Two subtypes are pre-defined in RFC 1521; however, it is likely that additional subtypes will be defined for applications such as mail-based scheduling systems, spreadsheets and EDI.

Application/Octet-Stream

This subtype indicates uninterpreted binary data. The recommended action for a mail-reading program that receives an application/octet-stream content type is to simply offer the user to write the information into a file. This subtype should also be selected by a mail-composing application, if an unrecognized file type is to be attached to the mail. A mail reader should also interpret an unknown content type as equivalent to Application/Octet-Stream. Further parameters for Application/Octet-Stream are:

Type
The general type or category of binary data. This is intended for human recipients rather than for automated processing.

Padding
This parameter gives the number of padding bits that have to be appended to the bit stream representing the actual contents to produce byte-oriented data. This parameter should be used for enclosing a bit stream in a body if the total number of bits is not a multiple of the byte size.

Application/Postscript

This subtype indicates a body containing a Postscript document. Postscript is a registered trademark of Adobe Systems, Inc. Use of the MIME content type application/postscript implies recognition of that trademark and of all the rights it entails. Be aware that the use of Postscript interpreters can be a potential security risk due to some operations in the Postscript language, such as deletefile or renamefile. Therefore the mailers should be configured to simply send Postscript bodies to the printer.

Text

This content type is used for mail bodies which contain ordinary textual information. An additional charset parameter can be used to specify an ASCII charset different from US-ASCII, thus enabling transmission of messages including, for example, special French, Spanish or German characters. The primary subtype is text/plain, which indicates unformatted text.

Default RFC 822 messages are typed by this protocol as plain text in the US-ASCII character set, which can be explicitly specified as "Content-type: text/plain; charset=us-ascii".

There is a set of predefined charset values including US-ASCII, which is also the default value, and ISO-8859-1 to ISO-8859-9. Additional character sets may be registered with IANA.

Beyond text/plain, there are subtypes defined for using formatted text bodies, which may include text formatting information such as bold, italic or underline. Possible subtypes of the text content type are:

Text/Plain

This subtype indicates plain unformatted text. The character set used is described by the charset parameter. In case non-US-ASCII characters are used in the mail body, it has to be encoded using the quoted-printable algorithm.

Text/Richtext

This subtype indicates a very simple and portable word-processing format that is defined by the MIME standard. It is a very small subset of SGML.

If a mail composing software is given an unrecognized word-processing format, it should first try to translate it to Text/Richtext to retain as much formatting information as possible.

Possible future subtypes may include any readable word processor format, e.g. Text/imake for Framemaker documents, or existing formats for multimedia documents, such as html documents.

Audio

This content type indicates audio data. The initial subtype is Audio/basic. It is expected that richer formats for higher quality or lower bandwidth audio will be defined by a later MIME standard. Of course, audio requires an audio output device (such as a speaker or a telephone) to present the contents to the user.

Audio/Basic

The content of this subtype is audio encoded using simple 8-bit ISDN µ-law [PCM]. A sample rate of 8000 Hz and a single (mono) channel are assumed for this content type.

Further subtype definitions could possibly include support for the SUN audio format or for *.WAV audio files, a format used in Windows and on OS/2 PCs.

Image

This content type indicates that the mail body contains an image. The subtype defines the specific image format. The image content type requires a display device capable of viewing graphical information, such as a graphical display, a printer or a FAX machine. Two initially defined subtypes are Image/Jpeg and Image/Gif.

Image/Jpeg

This subtype indicates an image in JPEG format, using JFIF encoding.

Image/Gif

This subtype indicates an image in GIF format [GIF].

Video

A content type of video indicates that the body contains a time-varying picture image, possibly with color and coordinated sound. The term video is used very generically, rather than in reference to any particular technology or format, and is not meant to preclude subtypes such as animated drawings encoded compactly.

Although in general the MIME standard strongly discourages the mixing of multiple media in a single mail body, it is explicitly permitted for subtypes of video, thus accepting the fact that many so-called "video" formats include a representation for "displaying" synchronized audio information.

Video/Mpeg

This subtype indicates video information according to the MPEG standard [MPEG].

Multipart

The MIME standard allows use of multiple parts, each with its own data type specified within a single message. The multipart content type indicates such a mail body.

To allow a MIME mailer to recognize the distinct parts in the multipart body, each part must be enclosed in a special boundary. Explicitly, each part is preceded by a special string called the encapsulation boundary and the last part of a multipart body is followed by a closing boundary.

To specify the boundary strings a new mandatory parameter called boundary is defined for the multipart content type. The boundary parameter is a string of up to 70 characters that uses a character set known to be very robust through email gateways and not ending with whitespace. The encapsulation boundary is an end of line followed by two hyphens followed by the boundary parameter value. The closing boundary is almost identical to the opening boundary, the difference being that it is followed by two more hyphens at the end of line.

Encapsulation boundaries may not appear within the encapsulations, and may not be longer than 70 characters, not counting the two leading hyphens. Furthermore, the mail-composing software must be able to choose and specify unique boundary values that will separate the body parts.

A typical multipart content type may look like this:

Content-Type: multipart/mixed; boundary=cc123abfd

This indicates a mail body consisting of several body parts. Each part is syntactically identical to an RFC 822 message except for the preceding encapsulation boundary --cc123abfd and the fact that no header area is required. The last body part must be followed by the closing boundary, which is almost identical to the preceding boundary; in the example above it looks like this --cc123abfd--.

The predefined subtypes for the multipart content type are as follows:

Multipart/Mixed

This subtype indicates a mail body with several parts which need not be bundled in any particular way and are completely independent. The different body parts are to be viewed serially by the receiving mailer. Any subtype not recognized by the mailer should be treated as Multipart/Mixed.

Multipart/Alternative

The syntax of a Multipart/Alternative mail body is identical to the Multipart/Mixed body but the semantics differ. A Multipart/Alternative body indicates that it contains the same information in several formats. Thus, a mail-reading agent should be able to choose the "best" format based on the local environment and preferences, e.g. to show the text document that contains the most formatting information that can be displayed with the user's machine and software. Another strategy for a mail-reading software could be to offer the user a list of contained parts if no clear decision can be made, and then let the user decide which one to display.

It is important that mail-composing agents that support Multipart/Alternative bodies place the entities in an increasing order of preference, that is in an order with the most preferred format placed last.

Multipart/Parallel

This subtype is syntactically identical to Multipart/Mixed. However, the distinct parts of the body are intended to be displayed in parallel. Commonly, a receiving mail user agent should present the parts of such a mail body simultaneously. But if using a Multipart/Parallel content type, you should be aware that the ability of the mail reader to do so strongly depends on the hardware and software. Mail readers which do not support Multipart/Parallel - this is indeed the majority - will show the parts serially.

Multipart/Digest

This subtype is syntactically identical to Multipart/Mixed. The difference in semantics is that the default content-type value for a body part is changed from Text/Plain to Message/RFC822. This is to allow a more readable digest format that is largely compatible with RFC 934.

Message

This body type is defined for the common operation of encapsulating another mail message in a mail body. There are three subtypes defined for this content type, including:

Message/RFC822

This subtype indicates a body part consisting of a message according to the RFC 822 syntax. But in contrast to a simple RFC 822 message, a body part of this subtype is not required to include a "From", "Subject" or "Destination Header" field. It should be noted, that despite calling this subtype Message/RFC822, a body part of this type can contain some type of enriched information, e.g. it can be a message conforming to the MIME standard.

Message/Partial

This subtype is defined to allow fragmentation of a large message into smaller pieces, in order to enable the message to be passed through mail transport facilities delimiting the length of a mail body. Thus, a large message can be transmitted separately in smaller pieces which are then reassembled automatically by the receiving user agent. The mechanism is comparable to IP fragmentation/reassembly in the Internet Protocol (IP). Hence, a mail reader interprets a body part with this subtype as a fragment of a larger message that has to be reassembled before delivered to a user.

Three additional parameters are required in the content type field of the subtype Message/Partial. The first one is called Id. A unique identifier that should be as close to world-unique as possible, it is used to merge the parts together. The second one, called Number, is also an integer. It is interpreted as the sequence number of the message part, so that the original message can be reassembled in the correct order. The third parameter is called Total and gives the total number of parts. This parameter is required only on the final part of the message. It may appear optionally in earlier parts.

It should be noted that this process of fragmentation and reassembly is normally handled automatically by the mailing software, and is completely transparent to the user.

Message/External-Body

This subtype indicates that the body part does not contain the data itself. Instead a reference to the data is transmitted. The additional parameters defined for this subtype describe a mechanism for accessing the external data.

A body or body part of this subtype consists of a header, a blank line, and the message header for the encapsulated message. Another blank line indicates the end of the header for the encapsulated information. However, since this content type describes only a link to the actual body, it does not appear in the area that follows. Hence, an example of a complete message body could look like this:

```
Content-type:  message/external-body
               access-type=local-file
               name=-/usr/images/kirk.jpg

Content-type:  image/jpeg
```

The area at the end might be called the "phantom body". Though normally ignored, for the access-type "mail-server" it must contain the complete command that is to be sent to the mail server to retrieve the actual information.

The only mandatory parameter for this content type is the access type field. Whether the other parameters are considered to be mandatory or optional depends on this access type field. The following list gives an overview on all previously defined access types:

FTP and TFTP
This access type indicates that the information can be retrieved using the FTP [RFC 959] or TFTP [RFC 783] mechanisms. Two more parameters, name and site, are required to localize the file on a remote machine. To gain access to the data, the user normally needs a login id and a password for the remote workstation.

Anon-FTP
This access type is identical to FTP except that the user does not need a login id or a password to gain access to the data (anonymous FTP).

Local-File and AFS
The access type local-file indicates that the actual body of the message is located on the local machine's file system. AFS indicates a file that is accessible via the global AFS file system. In both cases the name of the file containing the actual data is the only additional mandatory parameter.

Mail-Server
This access type indicates that the actual data is available from a mail server. One more mandatory parameter called server, giving the email address of the mail server, is required. Because some mail servers need multiline-line commands to retrieve the data, this command is not an additional parameter but provided in the "phantom body".

In addition to the parameters specific to the different access types, the standard defines two optional parameters for all access types: an expiration parameter specifying a date after which the existence of the external data is not guaranteed and the size parameter used to specify the size of the actual data.

X-TypeName

A content type value beginning with the characters "X-" indicates a private value. This value can only be used by cooperating mail user agents, if their content and interpretation are bilaterally defined. Any type used without a public definition must start with an "X-" prefix and any publicly specified values shall never begin with "X-".

The MIME standard strongly discourages the use of top-level types starting with an "X-" prefix. Instead subtypes of existing content type definitions should be extended by this mechanism whenever possible.

Private subtypes starting with an "X-" may be defined between cooperating mail composing and mail reading programs without outside registration. However, the name used for a privately defined content type should be chosen carefully to avoid misinterpretation of the mail body, which otherwise could possibly lead to conflicts between different user communities of MIME.

The MIME standard does not allow the use of a content type without a subtype specified.

If no content type is specified, either by error or by an older user agent, Text/plain with charset US-ASCII is assumed as the default value.

It should be noted that the list of content-type values given here may be augmented in time, via the mechanisms described above, and that the set of subtypes is expected to grow substantially.

When a mail reader encounters mail with an unknown content-type value, it should generally treat it as equivalent to "application/octet-stream".

2.1.5 The Content-Transfer-Encoding Header Field

Two of the restrictions implied by the SMTP protocol are the limitation of a mail message to 7-bit US-ASCII data and a total line length of less than 1000 characters. Thus, many content types which could usefully be transported via email but have a "natural" representation as 8-bit character or binary data cannot be transmitted due to these limitations.

MIME provides several mechanisms for encoding and decoding such data to short-line 7-bit US-ASCII format. MIME uses the Content-Transfer-Encoding header field to specify which of the provided mechanisms to use to encode/decode the data. Thus, the Content-Transfer-Encoding field is used to indicate the type of transformation that has been used in order to represent the body in a format that can be transported using RFC 822-compliant mail bodies.

It should be clear that at least two different coding mechanisms are desirable. One that is to be used to en/decode an 8-bit X-ASCII (this means any kind of 8-bit ASCII encoding) to an almost readable 7-bit US-ASCII representation and another one that is to provide a more compact coding algorithm for binary data.

The predefined values for the Content-Transfer-Encoding field are:

Base64
Quoted-Printable
7Bit
8Bit
Binary

The values 7Bit, 8Bit and Binary all mean that no encoding has been done. Its task is to introduce a clear labeling that will allow gateways between future mail transport systems with extended capabilities in transporting data to handle data according to their specific needs.

7Bit
This label means that the data is represented as short lines of US-ASCII characters.

8Bit
This means that the lines are short, but there may be non-US-ASCII characters in them.

Binary
This label indicates that not only may non-US-ASCII characters be part of the message, but also that the lines may exceed the limitation implied by the SMTP service.

The two other labels, Base64 and Quoted-Printable, both imply an encoding that consists of lines no longer than 76 ASCII characters.

Quoted-Printable
This encoding scheme seems to be most appropriate for data that consists primarily of printable ASCII characters. This encoding method will represent as themselves ASCII characters compliant to 7-bit US-ASCII. To encode any non-7-bit character, an escape mechanism has been established that uses the equals sign (=) as the escape character. Thus, any non-printable character is shown as an equals sign followed by a two-digit hexadecimal number. An equals sign in the message is represented in the same way. Lines longer than 76 characters are wrapped after the 75th character, using the escape character as a delimiter.

Base64
The Base64 encoding scheme provides a mechanism suitable for dense coding of binary data. A sequence of three bytes is represented as four printable characters. The character set chosen for this encoding is a 65-character subset of US-ASCII, enabling 6 bits to be represented per printable character.

2.1.6 The Optional Content-ID Header Field

This optional header field allows the labeling of message bodies. Thus it is possible that one body contains references to another. This could be useful for constructing a higher-level user agent. The Content-ID header is syntactically identical to the Message-Id field and like it, should also be generated to be world-unique.

2.1.7 The Optional Content-Description Header Field

The content-description field gives the user the ability to associate descriptive information with the body part. For example, it may be useful to mark an audio body as "The famous words spoken by N. Armstrong while taking the first steps on the moon". Such text may be placed in the content-description header field.

This concludes our introduction into the MIME standard with its header fields and content types. In the next section we will present a number of widely-used MIME mailers.

2.2 MIME-compliant Mailing Software

2.2.1 Andrew Message System

The Andrew Message System (AMS) provides a multimedia interface to mail and bulletin boards. AMS contains many advanced features including authentication, return receipts, automatic sorting of mail, vote collection and tabulation, enclosures, audit trails of related messages, and subscription management. It also provides a variety of interfaces that support alphanumeric terminals (ttys) and low-function personal computers in addition to the high-function workstations.

The Andrew Message System is part of the Andrew Toolkit (ATK), which is a portable user-interface toolkit that runs under X11. The toolkit provides a dynamically-loadable object-oriented environment that allows a programmer to easily create multimedia applications based on pre-defined ATK objects and user-defined objects. Such an object can include information of many different media types, e.g. formatted text using different fonts, raster images, spreadsheets, bitmap-oriented drawings, equations, simple animations, or even other multimedia objects implemented earlier. With the toolkit, programmers can create new objects that can be embedded as easily as those that come with the system. Many objects, e.g. a help system, a system-monitoring tool, and editor-based shell interface, and support for printing multimedia documents, are included in the release.

Additional features not mentioned above are:

- Extensive folder management. AMS provides an elaborated folder management system using binary-coded folder directories. Nested folders are supported as well as a graphical folder tree.

- Inline displaying capabilities. Many media types, e.g. raster images, bitmaps, spreadsheets, or simple animations can be displayed inline. Thus, the user does not have to worry about a confusing number of additional applications to view a multimedia mail.

- Aliases. AMS provides an alias system for frequently used mail addresses.

- MIME support. Mails written with the AMS can either be encoded in the Andrew format or in the MIME format. Received MIME-compliant mails can be decoded using the metamail package. The body parts are displayed either inline or by use of an external viewer specified in the mailcap file.

Currently, ATK runs on several variants of UNIX, including AIX, SunOS, Solaris, Ultrix, BSD, HP UX, Linux, IRIX, and SCO. A future release to additionally support the OS/2 operating system is planned.

The Andrew Toolkit is a joint development of Carnegie Mellon University and the IBM Corporation at the Information Technology Center. It is freely available and the current release, 6.2, can be obtained via anonymous ftp from the Internet host ftp.andrew.cmu.edu.

2.2.2 Elm with Metamail

Elm is a user agent system for reading, sending and managing electronic messages. It is designed to run with sendmail or any other UNIX mail transport agent. Originally, its task was to replace the existing line-oriented mailing software providing an interactive, screen-oriented and user-friendly interface. Thus, it is intended to be a full replacement of programs such as Berkeley Mail or mailx. Actually, elm is more than a single program, including programs like "frm" to list a table of contents of mail folders and "printmail" to quickly paginate mail files to allow clean printouts.

As described in [NB94], the metamail package can be used to extend elm to have MIME support. This enables the user to write mails that contain not only purely textual information, but also binary data, graphic, audio, video, Postscript and other files. The mechanism to attach a non-textual body part is quite simple. The part has to be specified within the editor using the following syntax:

```
[include path/filename content-type encoding]
```

For example, [include /opt/doc/mime.ps application/postscript base64] would instruct elm to include an application/postscript body part in base64 encoding, containing the file /opt/doc/mime.ps. Before sending the mail, elm parses it in order to replace these statements with the corresponding MIME syntax and to include the encoded data in the mail body.

Other features provided by elm are listed below:

- Elm comes with an extensive documentation set that includes The Elm Users Guide, The Elm Reference Guide, The Elm Alias Users Guide, The Elm Filter System Users Guide, The Elm Forms Mode Guide, and The Elm Configuration Guide.

- The ability to receive and transmit "forms" as defined by the AT&T Mail System.

- A mail filter program allowing the user to specify folders other than inbox to contain newly delivered mails.

- Special outgoing mail processing. Among other things, elm provides the feature to send and receive encrypted mail.

- Customized header lines. The content of the .elm/elmheaders file will be included immediately after the regular headers of all outbound mail.

- An answering-machine transcription program to allow automated processing and answering of incoming mail.

- An alias mechanism to easily handle frequently used mail addresses.

- The newmail program, which monitors one or more folders and notifies the user of new mail.

- The fastmail program, providing a batch mailing system that works as quickly as possible.

Elm uses Berkeley-style mail folders. It can handle multiple folder directories containing multiple folders at one time. A folder within such a folder directory is represented by a file containing the messages which are simply concatenated one after another. Thus, nested folders are not provided. Additionally, for Elm 3.0, the alternative use of MH-like folders has been suggested.

Elm is currently available for practically all UNIX-based systems. It has also been ported to DOS/Windows and OS/2.

The first version of elm was written by Dave Taylor. Later, the Elm Development Group, a cooperative venture of volunteers, was founded to develop the system further. The current version is 2.4. A new version, 3.0, has already been announced. Elm can be obtained via anonymous ftp from many sites. In Germany, for example, from ftp.th-darmstadt.de in the /pub/networking/mail/elm directory.

2.2.3 Exmh

Exmh is a TCL/TK-based interface to the MH mail system. Internally it uses the well-known MH programs to manipulate the mail folders and messages. Thus, it is compatible with the command line use of MH programs. Alternatively, for MH novices the details of running MH programs are hidden behind the graphical interface.

To run exmh a version greater than 3.3 of wish (the preferred version is 3.6) and version 6.7 or 6.8 of MH is needed. The current version 1.5.3 of exmh is not ready to be run with TK 4.0. Version 1.6 is announced as compatible to TK 4.0.

Exmh comes with a built-in editor called sedit, providing a set of text editing and formatting commands, e.g. automatic line breaks, changing fonts (bold, italic, smaller, larger, etc.), cut and paste, and some emacs-like editing key-bindings. Sedit supports the insertion of MIME-conformant body parts, as well as the use of 8-bit character sets.

As well as providing the usual graphical interface layer on top of MH commands, exmh has a number of other features:

- MIME support. exmh displays richtext and text/enriched directly. Other body parts are shown as headers. A pop-up menu under the right mouse button can be used to invoke external viewers, either by using the entries in the local mailcap file or by passing the mail body to metamail. The built-in editor allows simple composition of text/enriched format and the insertion of multimedia mail bodies, such as images, video, Postscript, etc.

- Nested folder display. The currently used folders appear as a set of labels, one for each folder. This is similar to xmh, except that the folder labels are highlighted to indicate the current folder, the target folder for moves, folders with unread mail in them (this is very useful if using mail filtering), and folders with nested folders under them. Unlike xmh, arbitrarily nested folder structures are supported by exmh.

- Scan listing highlights. The scan listing is also highlighted to indicate the current message, unread messages, and messages marked for move or delete. Either color or monochrome highlighting is used, depending on the display.

- Facesaver bitmap display. If there is a facesaver database on the system, exmh displays the bitmap face of the person that sent the current message.

- Background processing. exmh can be set up to run inc periodically, check for new messages, run the MH msgchk program, or tally the messages in the mail spool file.

- Mail filtering. Exmh is designed to work with external agents that filter arriving mail into different folders.

- Pick and Fast Search facilities. These interfaces allow selection of messages by patterns in the header fields, by date, or by sequence name.

- Preferences user interface. Exmh can be configured to use another editor, or key binding. Additionally, there are many knobs and dials to adjust exmh's behavior.

- Pretty Good Privacy (PGP). If PGP is installed on the system, it can be used from exmh to digitally sign, encrypt, and decrypt messages.

Exmh uses MH-style folders, that is, folders are represented by subdirectories, in the ~/Mail directory. Mails within a folder are sequentially numbered distinct files within the corresponding subdirectory. Exmh provides an import tool to convert Berkeley-style mailbox folders into MH folders.

Exmh was developed by Brent Welch at Xerox-PARC. The latest version can be obtained via anonymous ftp from parcftp.xerox.com. It can be found in the pub/exmh directory.

2.2.4 Mercurius

Mercurius is a MUA which facilitates the composing and reading of multimedia electronic messages compliant to the MIME standard as defined in RFC 1521. This allows exchange of multimedia messages over existing standard Internet mail handling systems as defined in RFC 822, using sendmail, smail, or mmdf as mail transfer agents, and TCP/IP or uucp as transport protocols.

Mercurius provides an easy-to-use visual interface, written in the TCL/TK script language, for manipulating MIME messages, supporting many features one might expect from modern GUI mailers:

- multiple mailboxes management
- inline display facilities for plain text, MIME richtext and bitmaps.
- support of many MIME content types by use of the mailcap mechanism.

The Mercurius MUA is freely available and can be obtained via anonymous ftp. The current version is located on the Internet host ftp.lii.unitn.it in the pub/mercurius filesystem. It can be run on several different UNIX systems, including Solaris, NextStep, Ultrix, and OSF/1.

2.2.5 Metamail

Metamail is not a multimedia mailing system itself. Instead it is a toolkit consisting of several parts that provide facilities to de- or encode MIME-compliant mails. Its typical use is as an add-on to existing mail systems, e.g. elm. [NB91] gives a description of how to write patches for the most commonly used mailers so that they can support the metamail package. Usually the latest releases of these mail systems should already include these patches, which can then be built in via compile flags. Figure 2 shows the cooperation of a mail user agent with the metamail package.

The metamail package consists of the following tools: metamail, audiocompose, audiosend, getfilename, mailto-hebrew, mailto, metasend, mmencode, richtext, showaudio, showexternal, shownonascii, showpartial and showpicture.

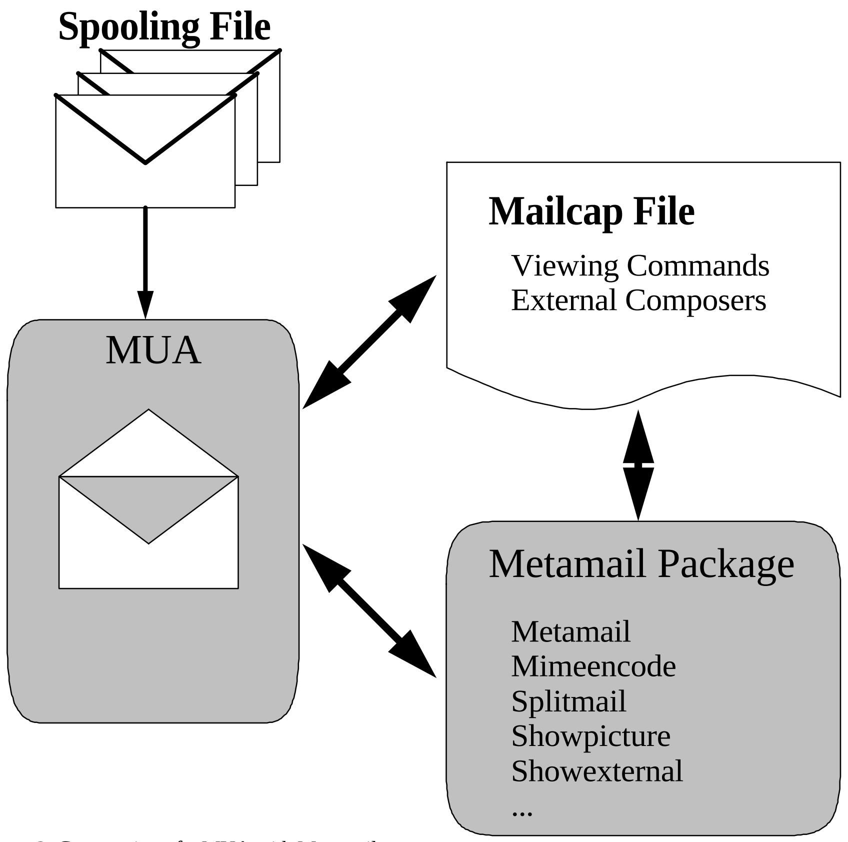


Figure 2: Cooperation of a MUA with Metamail

The metamail package is copyrighted by Bell Communications Research, Inc. (Bellcore). However, it is permitted to use, copy, modify, and distribute the toolkit for any purpose and without fee. It consists of the following tools:

Audiocompose

The audiocompose program can be used to compose an audio fragment for inclusion in a mail message. It will allow a user on a multimedia workstation to enter an audio fragment which will be written to a file. The user is given the chance to listen to his clip after recording it, and to re-record it if he so desires. The audiocompose program uses simple PCM to digitize the analogous data. Audiocompose can be configured to use an external audio composing program by specifying the RECORD_AUDIO environment variable.

Audiosend

The audiosend program will allow a user on a properly-equipped workstation to send an audio fragment as electronic mail. If an argument is provided, it will be used as the To address. Otherwise, the user will be asked to provide the To address. In any event, the user will be prompted for the Subject and CC fields. Audiosend will then let the user record his message. Afterwards, the user will be given the options of delivering the mail, listening to the message, re-recording the message, or aborting. Like audiocompose, this program can also use an external audio composing program.

Extcompose

The extcompose program provides the ability to include a reference to "external" data, i.e. data which is not included in the mail message itself but is otherwise available on the network via a suitable mechanism.

The user will be prompted for all necessary information. External data may be referenced as local or AFS files, as files available for FTP (anonymous or authenticated), or as data available via a mail server mechanism. The user will be asked to provide a MIME content type value to describe the type of data being referenced, and to specify whether or not the data has been encoded with one of the algorithms for encoding data for email. Extcompose uses this information to create MIME-compliant header fields allowing the receiving mailing software to retrieve the data referenced in the mail body.

Getfilename

The getfilename program will ask the user for the name of a file in a specified format, and will copy that file to a temporary file name. This is useful for mail composing software using the mailcap mechanism to add functionality.

For example, the following line in a mailcap file allows a mailer to help the user include Postscript in outgoing mail:

```
application/postscript ; lpr -Pipa %s ; description="Include
a Postscript File";compose="getfilename Postscript %s"
```

MailTo-Hebrew

The mailto-hebrew program will start an xterm using the appropriate font in order to send mail in mixed Hebrew and English.

MailTo

The mailto program is a very simple user interface for sending multimedia mail in MIME format. Its use is very similar to that of the Berkeley "mail" program, however it provides additional features for composing MIME-compliant messages.

As its name implies, mailto is for sending mail, not for reading it. None of the mail-reading features of the Berkeley mail program have been implemented in mailto.

The main difference between mail and mailto is that the latter can be used to generate enhanced mail in MIME format. However, mailto is intended to be a very simple multimedia mail generator. There are, accordingly, lots of things it cannot do. However, it has the virtues of being extremely simple, extremely similar to a well-known program (mail), and highly configurable, using the "mailcap" file mechanism.

Basically, mailto can include the following things in mail:

Simple formatted text, using the MIME type "text/richtext". This allows you to add emphasis to your message using underlining, bold text, italic, centering, etc.

Non-text data. Metamail can include pictures, sounds, and other non-textual data in the middle of any mail message. The mailcap configuration mechanism can even make this process reasonably user-friendly, but a knowledgeable user can include non-textual data even in the absence of a proper mailcap entry.

Text including non-ASCII characters, such as Hebrew or Russian. Currently, mailto directly supports only the ISO-8859-* family of character sets, which means that it does not meet the needs of Asian users, in particular. However, languages that cannot be expressed in the ISO-8859 family can still be included in the same way non-text data can be included.

Metamail

The metamail program provides the infrastructure for multimedia mail handling based on mailcap [NB94] configuration files. The purpose of the mailcap files is to tell metamail what program to run in order to show the user mail in a given format. At a site where all mail reading interfaces have been modified to call metamail for non-textual body parts, extending the local email system to handle a new media type in the mail becomes a simple matter of adding a line to a mailcap file. The mailcap mechanism is described in the appendix of this document.

In general, users will never run metamail directly. Instead, metamail will be invoked for the user by the user's mail-reading program, whenever a non-text message is to be viewed, since every mail-reading interface should call metamail whenever non-textual mail is to be displayed, unless the mail is of a type that is already understood by the mail-reading program itself. Metamail consults the mailcap file(s) to determine what program to use to show the message to the user.

The metamail program has built-in support for a few key content types. In particular, it supports the text type, the multipart and multipart/alternative type, and the message/rfc822 type. This support is incomplete for many subtypes - for example, it only supports US-ASCII text in general. The built-in support can be overridden by an entry in the mailcap file on the user's search path. Metamail also has rudimentary built-in support for types that are totally unrecognized. According to the MIME standard, metamail strips off all mail headers, decodes the body and saves it in a file.

It should be noted that metamail determines the type of a message using the "Content-type" header, as defined in RFC 1049 and RFC 1521 (MIME). However, using command line options, metamail can be made to work with mail that is not in Internet format, including X.400 messages.

Metasend

The metasend program will allow a user to send one or more pre-existing data files as non-text multimedia mail. With no arguments, the program will ask the user for the To, Subject, and CC fields. It will then ask for the name of a MIME content type. Next, it will ask the user for the name of an existing file containing that type of data. After this, it will ask what encoding type, if any, should be applied to this data. Finally, it will ask if the user wants to include information from an additional file, in which case the last three questions will be repeated for the next file. Alternatively, all of this information can be provided on the command line.

If more than one file is given, the parts will be combined into a single multipart MIME object.

Metasend is intended largely for mail hackers. A much friendlier interface to non-text mail is provided, for example, by mailto.

Mimeencode

The mimeencode program simply converts a byte stream into (or out of) one of the standard mail-encoding formats defined by MIME. Such an encoding is necessary because binary data cannot be sent through the mail. The encodings understood by mimeencode are preferable in several respects to the use of the uuencode/uudecode programs for use in mail. uuencode is known not to work very well in a number of circumstances. In particular, uuencode uses characters that do not translate well across all mail gateways (particularly ASCII <-> EBCDIC gateways). Also, uuencode is not defined as a standard. Thus, there are several variants floating around, encoding and decoding things in different and incompatible ways.

Mimeencode implements the encodings defined for MIME, e.g. base64 and quoted-printable encoding, as uuencode replacements, and should be more robust for email use.

Patch-Metamail

The patch-metamail program will attempt to retrieve a patch file for the metamail distribution, using anonymous ftp to a trusted server (e.g. thumper.bellcore.com), and then install that patch in the local copy of the distribution.

In general, patch-metamail is intended to be called via metamail, with a mailcap entry like the following:

```
application/x-metamail-patch; patch-metamail source-tree-root
%{patchnumber}; needsterminal
```

Richtext

The richtext program allows users to view "richtext" files on an ASCII terminal. Richtext is a very simple mark-up language for sending text bodies with additional formatting information through the mail. The richtext program is able to display text in bold or italic, and to underline text. It also implements most of the richtext commands that have to do with indentation and justification, as well as the "excerpt" and "signature" commands.

Showaudio

The showaudio program will allow a user on a multimedia workstation to listen to an audio message. Thus, it should be typically invoked by the metamail program to "display" audio parts of an email message.

Showexternal

The showexternal program will attempt to obtain the real body of a mail message that is included only by reference, using the MIME type "message/external-body". It is intended to be called by metamail via a mailcap entry like the following:

```
message/external-body; showexternal %s %{access-type} \
%{name} %{site} %{directory} %{mode} %{server} ; \
needsterminal
```

If it succeeds in fetching the message body, showexternal will then call metamail to actually display it.

Shownonascii

The shownonascii program allows parts of a mail message in a non-ASCII character set to be viewed under X11. By default, it will open up an xterm using the font given to shownonascii as an argument, running the "more" command to view all of the files named on the command line.

Showpicture

The showpicture program will allow a user on workstation with a graphical display device to view a picture using a specified external viewing program. The showpicture program is intended to be called via the mailcap mechanism, typically invoked by the metamail program.

Splitmail

The splitmail program will take an email message and break it up into smaller pieces using the "message/partial" type defined by the MIME standard. Thus allowing long messages to be transmitted through RFC-822-compliant gateways. By default it will take the message either from standard input or the named file, and will produce a set of partial message files with names like "/tmp/split.1" for the first part, and so on.

The default chunk size for splitting messages is 250000 bytes at most sites, though this is also a compile-time option. This can be overridden with the -s switch, or with the environment variable SPLITSIZE.

Messages smaller than the chunk size will not be turned into partial messages, but will be written to a single file or delivered as a single message.

Showpartial

The showpartial program will save the body of a MIME-format message that is of type "message/partial" for later use. When it is called on the last piece of a splitted message, it will put the pieces together and call metamail to display the full message. It is intended to be called by metamail via a mailcap entry like the following:

```
message/partial; showpartial %s %{id} %{number} %{total}
```

When the message is finally displayed, via metamail, the parts are all discarded, and the user is asked whether or not to delete the file containing the entire message.

2.2.6 Meuf

Meuf (Mail Extended Using Faces) is a mail user agent running on X11 on several UNIX platforms. The current version, 3.0, has been designed to fully support the MIME standard.

The user agent provides two internal editors for composing MIME-compliant messages. The first simply allows sequential attachment of non-textual files to a mail body, the second is more sophisticated: It allows one to edit the full range of MIME content types, especially multipart/mixed, multipart/alternative and multipart/parallel structures, using a graphical interface. Thus an overview of the structure of the whole message is always provided.

The following list gives some of the features of meuf:

- GUI with faces support. Meuf uses XBM bitmaps to visualize received mails and folders.
- Drag-and-drop facilities. The user agent provides a drag-and-drop interface for folder management, message handling and message composition. Additionally, the built-in wastebasket can be used the same way.
- Alias mechanism to substitute a nickname for long complex addresses and personal distribution lists.
- "Trusted Users" features.
- Online help system.
- Internal MIME support. The system does not rely on the metamail package.
- Support for the composition and display of all MIME content types.
- Parsing of Sun-Attachments created using the Sun Mail Tool.
- MIME-clipboard allowing cut-and-paste operations of MIME parts between different messages.
- Extraction of forwarded MIME messages for MIME restitution.

- Documentation. The package includes a User's Guide and an Administrator's Guide (both written in French).

Meuf was developed at the École Nationale Supérieure des Télécommunications de Paris by Daniel Glazman. It is freely available and can be obtained via anonymous ftp. The binaries for SunOS, Solaris and HP-UX of the current version, 3.0, can be found at the Internet host lara0.exp.edf.fr in the /pub/mime/meuf-3.0 directory.

2.2.7 PINE

PINE™ - a Program for Internet News & Email - is a tool for reading, sending, and managing electronic messages. It was designed specifically with novice computer users in mind, but can be tailored to accommodate the needs of "power users" as well. PINE uses Internet message protocols (e.g. RFC-822, SMTP, MIME, IMAP, NNTP) and runs on Unix workstations and PCs.

PINE supports the MIME (Multipurpose Internet Mail Extensions) standard by allowing the user to save MIME objects to files, and in some cases, PINE can also initiate the correct program for viewing the object. It can be configured to use the system's mailcap configuration file to determine the program to be invoked to process a particular MIME object type.

PINE's message composition editor, Pico, is a very simple and easy-to-use text editor offering paragraph justification, cut/paste, and a spelling checker (spelling checking is only available on Unix systems). Pico does not have integral multimedia capability, but any type of data file - including multimedia - can be attached to a text message and sent using MIME's encoding rules. According to the MIME standard, PINE uses the content type Application/Octet-Stream for attachments of an unrecognized format. PINE's support of the MIME standard, providing the ability to attach any file to a mail body, allows exchange of formatted documents, spread-sheets, image, files, etc., via Internet email.

The following list gives some of PINE's features:

- Online help specific to each screen and context.

- Message index showing a message summary which includes the status, sender, size, date and subject of messages.

- Commands to view and process messages: Forward, Reply, Save, Export, Print, Delete, Capture Address, and Search.

- Message composer with easy-to-use editor and spelling checker. The message composer also assists in entering and formatting addresses and provides direct access to the address book.

- Address book for saving long complex addresses and personal distribution lists under a nickname.

- Message attachments via the Multipurpose Internet Mail Extensions (MIME) specification.

- Folder management commands for creating, deleting, listing, or renaming message folders. Folders may be local or on remote hosts.

- Access to remote message folders and archives via the Internet Message Access Protocol (IMAP) as defined in RFC-1176 and revisions.

- Internet news support via either NNTP or IMAP.

PINE provides extensive folder management. It can handle multiple folder directories containing multiple folders at one time. The folder directories must be specified in the setup. A folder within such a folder directory is represented by a file containing the messages which are simply concatenated one after another. Thus, nested folders are not provided.

PINE and Pico are trademarks of the University of Washington. PINE, Pico, and UW's IMAP server are copyrighted, but freely available. The latest versions, including source code, can be found on the Internet host "ftp.cac.washington.edu" in the file "pine/pine.tar.Z" (accessible via anonymous FTP).

Unix PINE runs on a wide variety of systems including Ultrix, AIX, SunOS, SVR4, and Linux. PC-PINE for DOS is available for Packet Driver, Novell LWP, FTP PC/TCP, and Sun PC/NFS. PC-PINE for Windows/WinSock is a version with the same user interface, but compatible with the WinSock interface.

2.2.8 Tkmailto

Tkmailto is a Tk-based mail composer for mail, which supports the creation of MIME-conformant messages. Its basic functionality is based on the program mailto. Actually, tkmailto has been designed to provide the same functionality as mailto, but with a better graphical user interface. However, tkmailto is a completely new implementation and does not share any code with mailto. It is designed to run with sendmail or any other UNIX mail transport agent.

Like mailto, tkmailto cannot be used to read or process messages. But it provides a fancy interface to create MIME-conformant mails. So it can possibly be used as an external editor for another mailing software which does not provide a MIME-capable editor, or which has only an editor with a rudimentary MIME interface, such as elm or PINE.

The following list gives some of the features of tkmailto:

- Highly configurable. Tkmailto uses the file tkmailtorc to read-in a system and a user configuration, e.g. frequently used ftpdomains, ftp-directories, mailservers, content-types, transfermodes and aliases can be put there and will then show up as menu items.

- Graphical user interface. Tkmailto provides a fancy GUI, which makes it easy to use, even for novice computer users.

- Support of additional header fields, such as, reply-to or bcc.

- Cut-and-paste of whole message parts. Tkmailto can display an overview of all inserted parts. The individual parts can be cut and pasted via menu options.

Tkmailto was developed by Johan Lindbladh at Siemens KWU in Erlangen as the practical part of his master's thesis. The mail composer can be obtained via anonymous ftp from harbor.ecn.purdue.edu. The author can be reached at tet90jl@tintin.hik.se. Any comments, suggestions or bug reports can be mailed to him.

3 Other MM-Mail Standards

3.1 X.400/ Berkom MMM

3.1.1 The X.400 Recommendations

In 1984 the CCITT (International Telephone and Telegraph Consultative Committee) published the X.400 standard, a suite of recommendations describing the OSI's Message Handling System (MHS). X.400 provides an international electronic messaging service for reliable transfer of electronic mails. Two revisions of the standard appeared in 1988 and in 1992, however, most available implementations are still based on the standard of 1984. The functional model of the MHS is taken from a model for message handling developed in 1979 by IFIP (International Federation for Information Processing). The various entities (and their associated protocols) that make up the set of X.400(88) recommendations are identified in Figure 3.

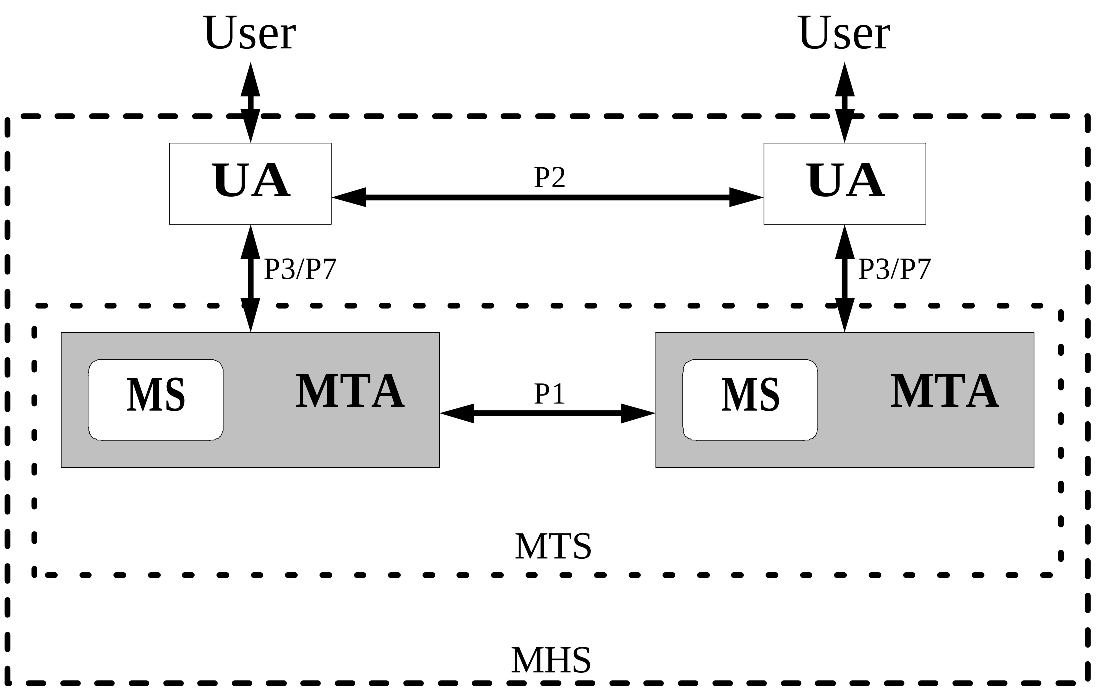


Figure 3: X.400 Functional Model

Berkom stands for "Berliner Kommunikationssystem". It is a very large testbed for a new generation of communication systems, including protocols for high-speed networks and many innovative applications. As one project within Berkom, the X.400-based Multimedia Mail System MMM was developed.

The user interacts with the system using a mail user agent (MUA), which is a software package designed for reading received messages, and for composing and sending electronic mails. Since the MHS is intended to transfer a variety of message types, there must be a pre-defined protocol between the UAs. This is the role of the P2 protocol, for example, the standard relating to simple person-to-person messaging is known as the interpersonal messaging (IPM) protocol.

Once a message has been composed, the UA passes it to a message transfer agent (MTA) which is part of the message transfer system (MTS). The UA interacts with the MTA using the P3 protocol, which provides the submission and delivery procedures as well as additional functions such as charging.

The message transfer agent is then responsible for delivering the electronic mail. Thus, it interacts with the other MTAs using the P1 protocol. A routing is done and the message is passed from one MTA to another following a store-and-forward principle. Once the mail is received by the MTA connected to the addressed UA, the transfer agent will try to deliver the mail to the user agent by means of the P3 protocol. Since it is possible that the user agent is switched off or out of service, the MTA needs to store the mail until it can be delivered. To act as such an intermediary between the MTA and the UA is the task of the message store (MS). If the UA cannot be reached directly, the message transfer agent places the mail in the message store. Later, the user agent can retrieve the electronic mail via the P7 protocol.

3.1.2 The X.400 Message Format

The electronic message being sent using the MHS can be separated into two parts, the envelope and the content. As for "normal" hand-delivered mail, the envelope provides all the information, e.g. the addresses of the sender and the receiver, needed by the MTAs to route the electronic mail through the MTS to the addressed UA. The envelope content, however, contains the information that is to be transported between two user agents. Since there are many different message types, each associated with its own P2 protocol, additional control information is needed by the user agents to handle the message. Thus, the content can also be separated into a header part, containing the control information needed by the user agents, e.g. to specify the message type, and the contents body, which is actually the user information that is to be transported. As an example, figure 4 shows the structure of a message conformant to the IPM protocol (interpersonal messaging protocol).

The interpersonal message system is an example of a P2 protocol, and is defined in X.420. It describes the format of a private or business message, which is sent from the originator's user agent and received by the addressee's user agent, defining the meaning of the header fields and the content of the body parts.

The body of an IPM-conformant message contains a sequence of body-parts, such as text or audio, which are described by the standard. X.420 comprises the following body parts:

ia5-text
A simple textual body part, using an 8-bit character set (international alphabet no. 5).

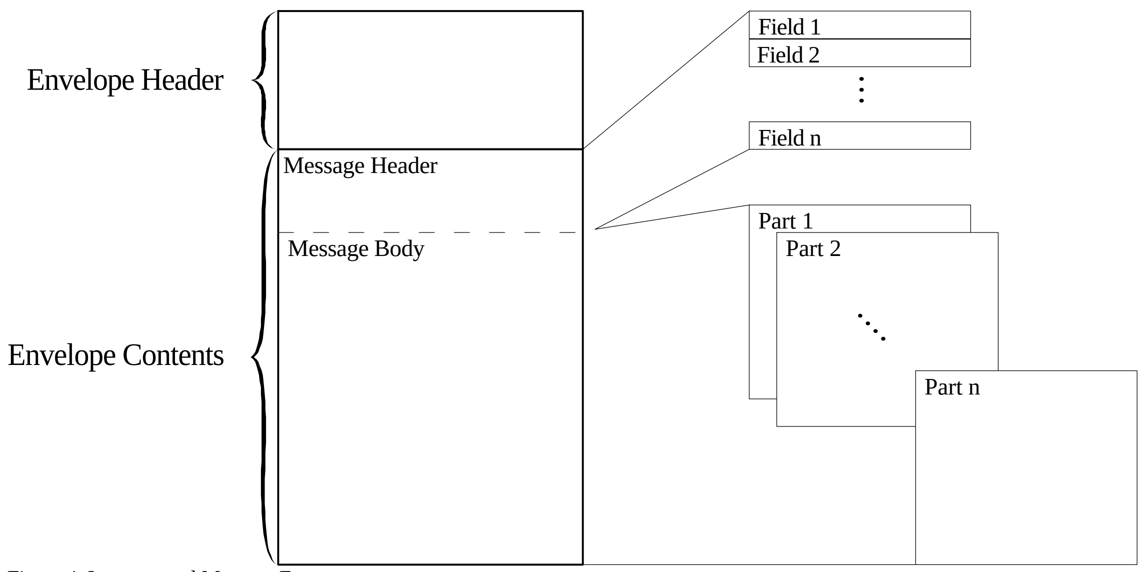


Figure 4: Interpersonal Message Format

voice
A body part containing speech. It must be mentioned that the parameters of such a body part and the digitized speech-encoding technique that those parameters might identify and parameterize, are for further study.

g3-facsimile
A g3-facsimile body part represents Group 3 facsimile images.

g4-class1
A group 4, class 1 body part represents a final-form document of the sort that is processable by Group 4 Class 1 facsimile terminals.

teletex
A teletex body part represents a teletex document.

videotex
A videotex body part represents videotex data.

encrypted
An encrypted body part represents the result of encrypting a body part of a type defined by the X.420 recommendation. The parameters of such a body part, and the encryption technique that those parameters might identify and parameterize, are for further study.

message
A message body part represents an IPM and, optionally, its delivery envelope.

mixed-mode
A mixed-mode body part represents a final-form document of the sort that is processable by mixed-mode teletex terminals and Group 4 Classes 2 and 3 facsimile terminals.

bilaterally-defined
A bilaterally defined body part represents an information object whose semantics and abstract syntax are agreed upon bilaterally by the IPM's originator and all of its potential recipients.

nationally-defined
A nationally defined body part represents an information object whose semantics and abstract syntax are nationally defined by a country whose identity is agreed upon bilaterally by the IPM's originator and all of its potential recipients.

externally-defined
An externally defined body part represents an information object whose semantics and abstract syntax are denoted by an object identifier carried by the body part.

3.1.3 Berkom MMM Extensions

The Berkom Multimedia Mail Service is based on CCITT-Recommendation X.400(88) using the IPM format as its standard message format. To additionally send media types not explicitly specified in the X.400 standard, Berkom MMM makes use of the externally defined body part. New types introduced to the message format are image, audio, video, link and external references.

Since the introduction of new externally defined body parts concerns only the P2 protocol, no changes to the X.400 MTS are required. Instead, new viewing and editing facilities must be added to the user agents to support the new types.

The following table summarizes the body parts allowed in the Berkom MMM Service, and their encoding.

Component	**Representation**
text	IA5 (8 bit)
document	ODA/FOD26 + Corrigenda
audio	Industry Implementation of G.711, G.721
image	Image Interchange Format
audio/video	Phase 1: SMP Phase 2: MPEG, or M-JPEG
external reference	Distinguished Object Reference
link	textual

External References
Since the attachment of multimedia data objects can lead to very large messages that consume a lot of bandwidth during transport, and a lot of storage space on a receiver's file system after receipt, the Berkom MMM teleservice provides an external reference mechanism. This allows the originator of a multimedia mail message to move the data object to a server (Global Store), create an external reference to it, and simply mail the external reference encoded in an X.400 body part to the recipient.

The recipient has two means of viewing the referenced object. He can either copy the data from the server to a local file or use a real-time viewing service to view the data as it arrives from the server.

To support external references, Berkom MMM makes an extension to the functional model of X.400 (Fig. 5).

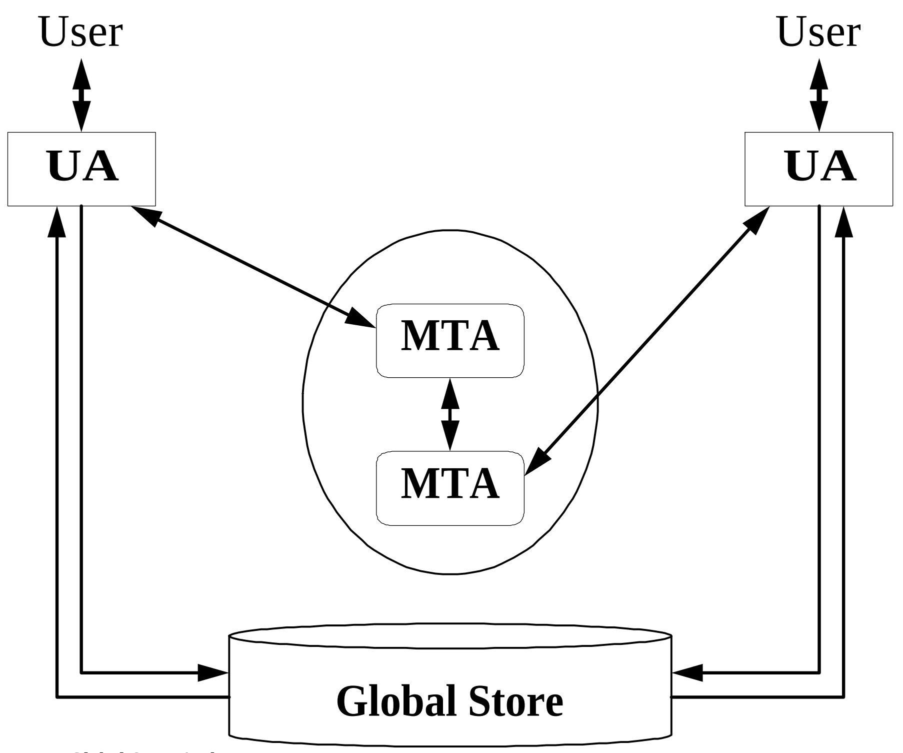


Figure 5: Global Store Architecture

Links
Berkom MMM additionally provides a body part, which can be used for creating links between related parts of the mail body, thereby structuring the message. This hyper-media structure allows a recipient of an electronic mail to jump between associated pieces of information. An example would be to annotate certain text passages with audio or video comments by defining links between the text and other body parts (e.g. audio or video).

3.2 Sun Mail Tool

3.2.1 The Mailtool User Agent

Mailtool - Sun's OpenWindows interface for the mail program - provides facilities for reading, storing, composing, and sending mail messages using standard Internet mail-handling systems such as SMTP, uucp, etc. In extending the support of ordinary RFC 822-compliant messages, mailtool allows to send and receive mails including non-textual attachments, such as images, audio, postscript documents, or raw binary data. Sun's approach to overcoming the limitations implied by RFC 822 is to use the uuencode/uudecode mechanism for en-/decoding message parts which cannot be represented using a 7-bit US-ASCII character set. Additionally, some new header fields have been introduced that provide the information needed by the receiving mail user agent to decode the attachment and to invoke an external viewer to display the non-textual information.

The MUA is part of the Solaris distribution. Although it is a proprietary standard, there are several freely available mail user agents, which at least provide facilities to read messages created with mailtool. Additionally, some tools exist to convert MIME messages to mailtool-format and vice versa.

3.2.2 uuencode/uudecode

uuencode can be used to convert a binary file into a 7-bit ASCII-encoded representation that can be sent using SMTP or other common Internet mail handling systems. The syntax of uuencode is as follows:

```
        uuencode [source-file] file-label
```

The content of source-file, or the standard input, if no source-file argument is given, is converted into an ASCII representation using four ASCII characters to encode three octets of the original file. Additionally, some control information is added, so that the size is expanded by 35%. The file-label argument is required. It is included in the encoded file's header as the name of the file into which uudecode is to place the binary (decoded) data. uuencode also includes the permission modes of source-file, (except setuid, setgid, and sticky-bits), so that file-label is recreated with those same permission modes. The access permissions are included as a three-digit octal number. A uuencoded file called kirk.gif with the permissions -rw-rw---- will look like the following:

```
begin 660 kirk.gif
M24DJ $X'...

end
```

The encoded data is enclosed within a begin/end statement. 660 is the octal equivalent for the access permissions -rw-rw----, and kirk.gif is the name of the file used by uudecode to write the decoded data to.

uudecode reads an encoded file, strips off any leading and trailing lines added by mailer programs, and recreates the original binary data with the filename and the mode specified in the header. The syntax is as follows:

```
        uudecode [-p] [encoded-file]
```

The optional parameter -p causes uudecode to write the decoded information not to the file given in the header, but simply to standard output, allowing uudecode to be used in a pipeline. If no name of an encoded-file is given, uudecode tries to read from standard input.

3.2.3 Additional Header Fields

Sun defines seven additional header fields to provide to the receiving mail user agent the information needed to decode the attached data, and to display it using an external viewer. All header fields have an X-Sun- prefix. They will be described in the following.

X-Sun-Data-Type
This header field can be seen as the equivalent to MIME's Content-Type header field. The purpose of this header field is to describe the data contained in the body fully enough that the receiving mail user agent is able to invoke an appropriate mechanism to present the data to the user, e.g. by passing the decoded information to another application, such as audiotool, etc, which then displays it. Possible values for this header field are text, jpeg-file, gif-file, tiff-file, ae-file (used to transmit an appointment), postscript-file, audio-file, etc. If the attachment is in an unrecognized format, mailtool sets the X-Sun-Data-Type header field to the value default.

X-Sun-Data-Description
The X-Sun-Data-Description header field is comparable to the Content-Description field of RFC 1521. It is used to associate descriptive information with the body part.

X-Sun-Data-Name
The task of the X-Sun-Data-Name header field is to hold the name of the body part being attached to the mail. Normally, this is a file name on the local file system. If using uuencode, the X-Sun-Data-Name corresponds to the file-label in the header created by uuencode.

X-Sun-Encoding-Info
If existent, this header field provides the information needed by the receiving mailing agent to properly decode the attached data, e.g. if the mail body contains a uuencoded file, the X-Sun-Encoding-Info header field will have the value uuencode.

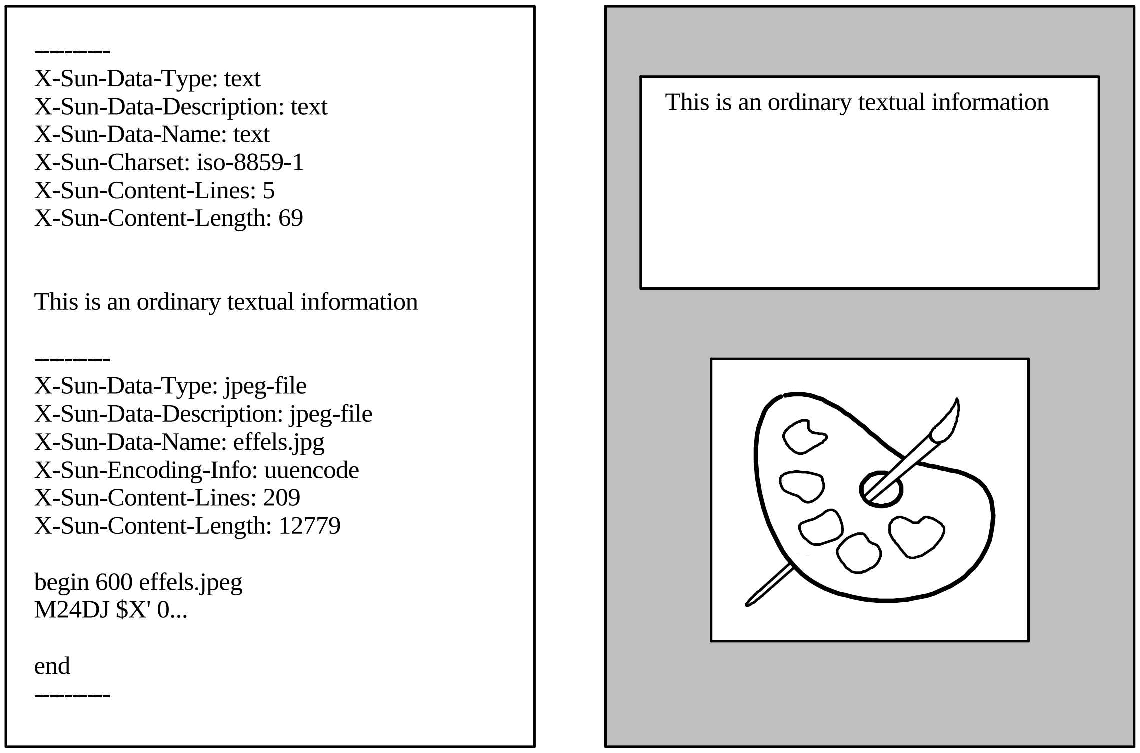
X-Sun-Charset
Mailtool allows the use of different character sets within a mail message. The X-Sun-Charset header field is used to specify the character set being used in the following mail body. Possible values are US-ASCII, ISO-8859-*, etc.

X-Sun-Content-Lines
This header field provides information on the total number of lines in the attached message body.

X-Sun-Content-Length
This header field provides information on the total number of characters in the attached message body.

3.2.4 Overall format of a Mail Message

Since mailtool uses simple Internet mail handling systems, the messages created by mailtool are in 7-bit US-ASCII representation. If using attachments, mailtool concatenates the individual message parts one after another. To separate the attachments, a single line holding "----------" is used as a boundary. Below this boundary, the X-Sun-* header fields appear to fully describe the content or the attachment. To separate the header from the body, a blank line is used. Thus, a complete mailtool-compliant mail has the same structure as a MIME-compliant mail consisting of an enclosing multipart/mixed content type, containing several "simple" attachments. A sample mail and its corresponding user view is given in figure 6.



```
----------
X-Sun-Data-Type: text
X-Sun-Data-Description: text
X-Sun-Data-Name: text
X-Sun-Charset: iso-8859-1
X-Sun-Content-Lines: 5
X-Sun-Content-Length: 69


This is an ordinary textual information

----------
X-Sun-Data-Type: jpeg-file
X-Sun-Data-Description: jpeg-file
X-Sun-Data-Name: effels.jpg
X-Sun-Encoding-Info: uuencode
X-Sun-Content-Lines: 209
X-Sun-Content-Length: 12779

begin 600 effels.jpeg
M24DJ $X' 0...

end
----------
```

This is an ordinary textual information

Figure 6: Mailtool Message Format

4 Testing

4.1 The Testing Environment

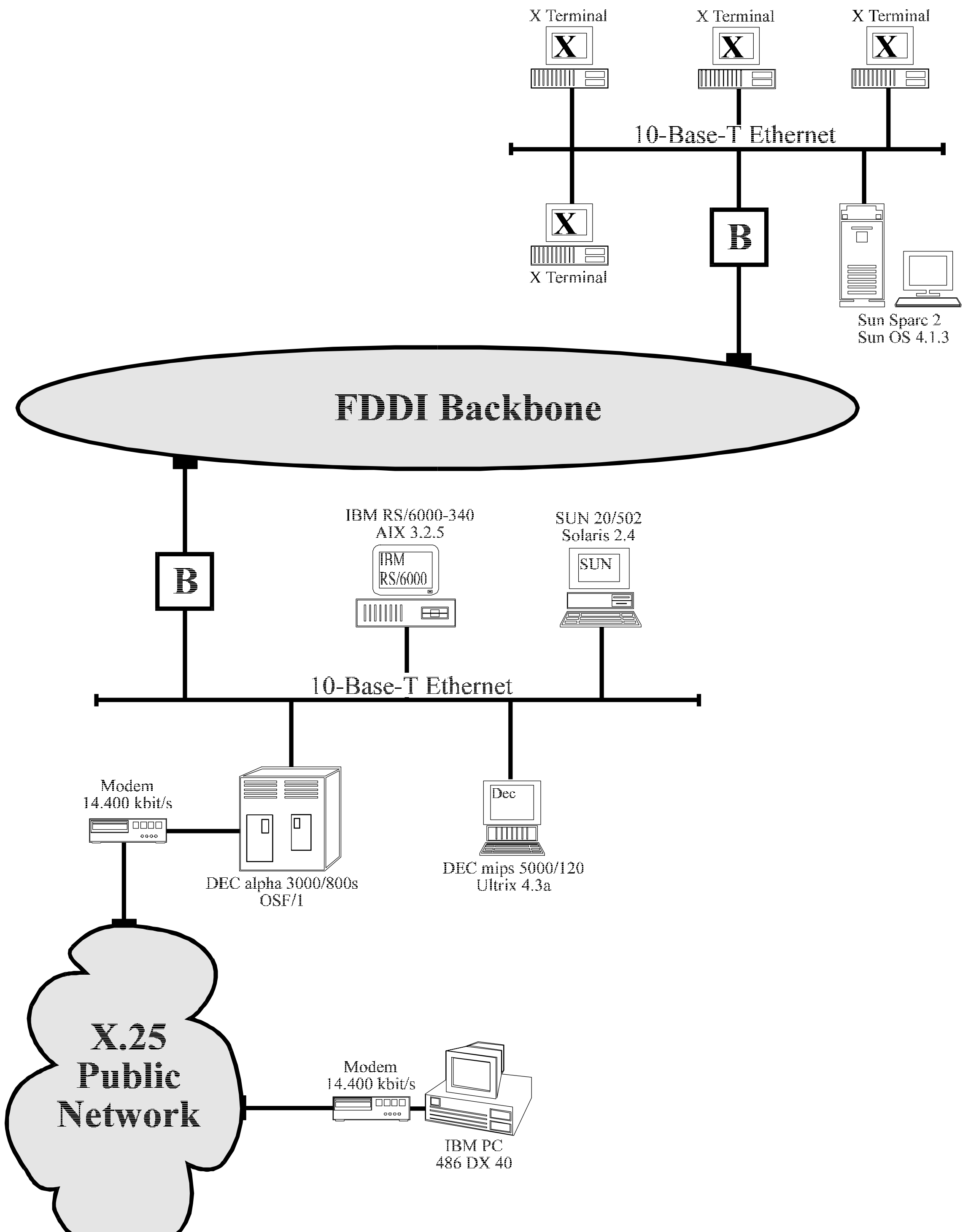


Figure 7: The Testing Environment

Figure 7 illustrates the environment we used to conduct the tests. It consisted of two LANs interconnected with a FDDI backbone. Additionally, we used a 14.400 kbit/s modem link over a X.25 network to connect an IBM PC to the university network.

The following table shows which mail user agent was installed on which system.

	Sun 20/502 Solaris 2.4	Sun Sparc 2 Sun Os 4.1.3	IBM RS/6000 AIX 3.2.5	DEC alpha OSF/1 2.1	DEC mips Ultrix 4.3a
AMS	X				
Elm	X	X	X	X	X
Exmh	X[(1)]	X	X	X	X
Meuf	X				
Mercurius			[(2)]	X	
Metamail	X	X	X	X	X
PINE	X[(3)]	X	X[(3)]	X	X
Tkmailto	X	X			

[(1)] locking problem with mh. The system lock does not work properly on NFS mounted file systems being exported by a Ultrix server.
[(2)] AIX does not support the lockf system call needed by Mercurius.
[(3)] The system lock does not work properly on NFS mounted file systems being exported by a Ultrix server. Only dot-file-locking possible.

4.2 MIME Conformance Tests

4.2.1 Encoding Test

Each mailing agent tested during the study provides an internal editor. The aim of the encoding test is to assess the capability and user-friendliness of these internal editors to create a MIME-conformant mail. All editors have been tested extensively and have been matched against the following criteria:

E.1 generate a "MIME-Version: 1.0" header field, if necessary.

E.2 encode any non-7-bit information and add the appropriate MIME header fields.

E.3 allow the editing of the following content types

E.3.1 *text*

E.3.1.1 allow the editing of a text/plain mail body with charset=US-ASCII

E.3.1.2 allow the editing of a text/plain mail body with charset=ISO-8859-1 using a standard ASCII keyboard (most workstations are equipped with such a keyboard), do the encoding as quoted printable and add the appropriate header fields.

E.3.1.3 provide text/richtext formatting facilities and add the appropriate header fields.

E.3.2 *multipart*

E.3.2.1 generate the multipart/mixed boundary, if necessary.

E.3.2.2 allow the creation of a multipart/mixed body part, e.g. nested within another body part.

E.3.2.3 allow the creation of a multipart/alternative body part, e.g. nested within another body part.

E.3.2.4 allow the creation of a multipart/parallel body part, e.g. nested within another body part.

E.4 allow the insertion of one of the following content types. Recognize the format and add the appropriate MIME header fields.

E.4.1 *text*

E.4.1.1 allow importation of a file containing a text/plain mail body with charset=US-ASCII.

E.4.1.2 allow importation of a file containing a text/plain mail body with charset=ISO-8859-1.

E.4.1.3 allow importation of a file containing a text/richtext mail body.

E.4.2 *application*

E.4.2.1 allow the insertion of an application/postscript body part.

E.4.2.2 generate the content type application/octet-stream upon encountering an unrecognized body-part.

E.4.3 *audio*

E.4.3.1 allow the insertion of an audio/basic body part.

E.4.4 *image*

E.4.4.1 allow the insertion of an image/gif body part.

E.4.4.2 allow the insertion of an image/jpeg body part.

E.4.4.3 allow the insertion of an image/tiff body part.

E.4.5 *video*

E.4.5.1 allow the insertion of a video/mpeg body part.

E.4.6 *message*

E.4.6.1 allow the insertion of a message/external-body body-part.

4.2.2 Decoding Test

The aim of the decoding test is to assess the MIME conformance of the mailing software tested during the study. The mailers have been tested for their capability of handling at least the MIME-specific header fields and content types which can be found in the following list. Two other points considered were the provided display facilities (inline or external viewer) and the user friendliness.

D.1 recognize a "MIME-Version: 1.0" header field.

D.2 recognize the Content-Transfer-Encoding header field and do either a Base64 or a Quoted-Printable decoding.

D.3 recognize and interpret the content-type header field. Avoid showing data to the user with a content type other than text/*. Provide at least the facility to save the body part.

D.4 recognize and handle the following content types:

D.4.1 *text*

D.4.1.1 recognize and display textual mail with the character set US-ASCII.

D.4.1.2 recognize textual mail with the character set ISO-8859-1. At least inform the user, that the mail might contain characters that might be displayed incorrectly.

D.4.1.3 recognize and display textual mail of the text/richtext content type.

D.4.2 *application*

D.4.2.1 recognize the application/postscript content type. Decode the body part and save it to a file, display it or send it to the printer.

D.4.2.2 recognize the application/octet-stream content type. Decode the body part and offer the user to save it to a file.

D.4.3 *audio*

D.4.3.1 recognize an audio/basic body part. Decode the information and send it to the audio device, or offer the user to save it to a file.

D.4.4 *image*

D.4.4.1 recognize an image/gif content type. Decode the body part and display it.

D.4.4.2 recognize an image/jpeg content type. Decode the body part and display it.

D.4.4.3 recognize an image/tiff content type. Decode the body part and display it.

D.4.5 *video*

D.4.5.1 recognize a video/mpeg content type. Decode the body part and display it, or offer the user to save it to a file.

D.4.6 *multipart*

D.4.6.1 recognize a multipart/mixed structure. Display all relevant information on the message level and the body part header level and then display or offer to display each of the body parts individually.

D.4.6.2 recognize a multipart/alternative subtype. Choose the "best" alternative for the local site and display it. Do not show the user any redundant parts.

D.4.6.3 recognize a multipart/parallel subtype. Display all relevant information on the message level and the body part header level and then display or offer to display each of the body parts in parallel.

D.4.7 *message*

D.4.7.1 recognize a message/external body subtype. Display all relevant information on the message level and the body part header level. Then offer the user to fetch the data using the method described by the access-type parameter. Display the fetched data, or offer to save it to a file.

4.3 MIME Testing Results

4.3.1 Andrew Message System

The Andrew Message System (AMS) provides a multimedia interface to mail and bulletin-boards called messages. Besides many advanced features including authentication, return receipts, automatic sorting of mail, vote collection and tabulation, enclosures, audit trails of related messages, and subscription management, AMS allows to receive and write multimedia mails using the MIME standard to encode non-textual parts.

To edit a multimedia mail, AMS provides a composer called sendmessage. Like any other ATK application, sendmessage is able to make use of pre-defined insets. Thus, it is possible to "internally" edit a huge number of media types, including raster images, bitmaps, spreadsheets, equations, postscript, or simple animations. Even user-defined insets developed using the ATK can be included.

The following tables give the results of the tests towards the specified criteria:

E.1	Yes. Upon encountering any format which cannot be sent using the RFC 822 mechanisms, sendmessage asks whether the mail should be encoded using either Andrew or MIME format.
E.2	Yes. (see E.1)
E.3.1.1	Yes.
E.3.1.2	Yes. Inline formatting, such as bold , italic, bigger, smaller, subscript, superscript, underlined, different fonts, or different colors is supported.
E.3.1.3	No. Sendmessage provides no means to insert non-US ASCII characters using a standard ASCII keyboard.
E.3.2.1	Yes. (see E.1)
E.3.2.2	No. Sendmessage does not recognize content type definitions within a text body and encloses it like an ordinary text body within a multipart/mixed or multipart/alternative boundary.
E.3.2.3	No. (see E.3.2.2)
E.3.2.4	No. (see E.3.2.2)
E.4.1.1	Yes. Even the charset parameter is set properly.
E.4.1.2	Yes. Sendmessage recognizes the format and sets the adequate header fields. It even sets the charset parameter properly.
E.4.1.3	No. Sendmessage does not recognize the format, and it is not possible to set the content type by hand.
E.4.2.1	No. ATK supports a postscript inset, but upon sending a mail, the inset will be encoded using the application/andrew-inset content type. Further, the postscript inset only allows to write a postscript part, not to insert an already existing one.
E.4.2.2	Not tested. Sendmessage provides no means to insert an unknown part into the mail.
E.4.3.1	No.
E.4.4.1	Yes. Sendmessage provides the internal editing and importing of GIF files using either the inset raster or image.
E.4.4.2	Yes. Sendmessage provides the internal editing and importing of JPEG files using either the inset raster or image. Upon sending, the image is converted to image/gif.
E.4.4.3	Yes. Sendmessage provides the internal editing and importing of TIFF files using either the inset raster or image. Upon sending, the image is converted to image/gif.
E.4.5.1	No.
E.4.6.1	No.

If using MIME encoding to send the mail, sendmessage creates a multipart/alternative content type presenting two different views of the mail. The first one only includes the parts which can be shown on a TTY display, the second one includes the whole message.

Insets provided by AMS, which cannot be found in the MIME specification, are encoded using the application/andrew-inset content type and base64 encoding. They can be displayed using the ezview command, which can be obtained via anonymous ftp

Concerning the non-supported MIME content types, it must be pointed out that using the ATK, it is possible to write insets which support, for example, video/mpeg. These insets can then be used like any pre-defined insets, thus allowing steady extension of the system. The language provided by ATK is an object-oriented C. This language has nothing to do with C++, rather it is based on K&R-C, but with an additional pre-processor to cope with the extensions in terms of object orientation [NB90].

D.1	Yes.
D.2	Yes.
D.3	Yes.
D.4.1.1	Yes.
D.4.1.2	Yes. ISO-8859-1 characters can be displayed inline.
D.4.1.3	Yes. Messages supports inline displaying of bold, italic, underline, superscript, subscript, underline, bigger, smaller, center, left, right, etc. formatting as well as using different fonts and colors.
D.4.2.1	Yes. Messages recognizes the content type and passes it to metamail to show it using an external viewer.
D.4.2.2	Yes. Messages recognizes the content type and offers to write the content to a file.
D.4.3.1	Yes. Messages recognizes the content type and passes it to metamail to show it using an external viewer.
D.4.4.1	Yes. The image/gif content type is displayed inline.
D.4.4.2	Yes. The image/jpeg content type is displayed inline.
D.4.4.3	Yes. Messages recognizes the image/tiff content type and passes it to metamail to show it using an external viewer.
D.4.5.1	Messages recognizes the content type and passes it to metamail to show it using an external viewer.
D.4.6.1	Yes.
D.4.6.2	Yes. Messages recognizes a multipart/alternative structure. It selects the best alternative and offers one or more buttons to show the other alternative(s).
D.4.6.3	Yes. Messages recognizes a multipart/parallel structure, however, the parts can only be shown serially.
D.4.7.1	Messages recognizes the content type and passes it to metamail to show it using an external viewer.

Generally, a content type that cannot be displayed inline is displayed as a button. Upon pressing the button, messages passes the content type to metamail, which then shows it using the information in the mailcap file to invoke an appropriate external viewer.

AMS makes a good overall impression. The pre-defined editing facilities provided by sendmessage will satisfy most of the user's needs. Additionally, ATK provides a mechanism to extend the range of media types supported. Further, AMS provides a variety of interfaces that support TTYs and low-function personal computers as well as high-end workstations running X11. Indeed, it was the only system considered in this study tthat supported both TTY interfaces and GUIs. In respect to this, MIME encoding is done using an enclosing multipart/alternative structure with two embedded parts, one for TTYs and the other to be displayed on graphical devices.

On the other hand, sendmessage does not support the editing of complex MIME messages, e.g. nested multipart/* structures. It only provides means to sequentially attach MIME objects to a mail, which are then combined within a multipart/mixed structure. Sendmessage itself makes no use of the information stored in the mailcap file, e.g. no external composing agents are invoked, even if they are specified in the mailcap file. The restriction not to provide a mechanism to insert non-US ASCII characters delimits the use in non-English-speaking countries. A last point of criticism is the folder management, which is completely incompatible to folders used by any other mailing software considered in the study.

Concerning the ATK, the definition of another object-oriented C-language instead of simply using C++ will possibly prevent programmers from using it, although the underlying ideas are really fascinating and the "class-library" provides a huge framework, allowing creation of multimedia applications with little effort in a short space of time.

4.3.2 Elm with Metamail

For the testing we used Elm Version 2.4, which can be configured to use metamail for MIME support at compile time. Hence, the metamail package also has to be installed, and the PATH-variable must contain the path of metamail's executables.

The mechanism to attach a non-textual body part is quite simple. The part has to be specified using the following syntax within the editor:

```
[include path/filename content-type encoding]
```

Before sending the mail, elm parses it in order to replace these statements with the corresponding MIME syntax and to include the encoded data in the mail body.

When receiving new mails, elm scans them for the MIME-Version: 1.0 header field. If elm encounters this header, it directly passes the entire message to metamail, which is then responsible for displaying it.

The following tables give the results of the tests towards the specified criteria:

E.1	Yes.
E.2	Yes.
E.3.1.1	Yes.

E.3.1.2	No. Elm's internal editor does not provide a mechanism to insert non-US-ASCII characters.
E.3.1.3	No. The internal editor does not provide any formatting facilities.
E.3.2.1	Yes. Having written the mail, elm scans the mail body for attachments and encloses the text bodies and attachments within a multipart/mixed body part.
E.3.2.2	No. Although the user can insert MIME headers within the text body, elm does not recognize this and encloses them like an ordinary text body within a multipart/mixed boundary.
E.3.2.3	No. (see E.3.2.2)
E.3.2.4	No. (see E.3.2.2)
E.4.1.1	Yes. After inserting the text, the header field charset=US-ASCII is automatically created.
E.4.1.2	Yes. The insertion of a file containing text/plain with charset=ISO-8859-1 is possible, but the charset header field is not set properly.
E.4.1.3	Yes. The same restrictions apply as for E.4.1.2.
E.4.2.1	Yes.
E.4.2.2	Not tested. Elm's mechanism for including non-textual body parts does not allow to insert a body part without specifying the content type.
E.4.3.1	Yes.
E.4.4.1	Yes.
E.4.4.2	Yes.
E.4.4.3	Yes.
E.4.5.1	Yes.
E.4.6.1	No.

Upon encountering any MIME-compliant mail, elm passes it directly to the metamail program. Thus, the decoding done and the results of the decoding test are exactly the same as for the metamail package.

D.1	Yes.
D.2	Yes.
D.3	Yes.
D.4.1.1	Yes.
D.4.1.2	Yes. If properly configured in the mailcap file, metamail can display text with charset ISO-8859-1 e.g. using an xterm with an 8-bit character set.
D.4.1.3	Yes. Metamail displays richtext in the current terminal window using a simplified displaying mechanism, e.g. it displays italic characters inverted.
D.4.2.1	Yes. The part can be displayed using an external viewer.
D.4.2.2	Yes. The part can be written to a file or seen as text.
D.4.3.1	Yes. The part can be displayed using an external viewer.
D.4.4.1	Yes. The part can be displayed using an external viewer.
D.4.4.2	Yes. The part can be displayed using an external viewer.

D.4.4.3	Yes. The part can be displayed using an external viewer.
D.4.5.1	Yes. The part can be displayed using an external viewer.
D.4.6.1	Yes.
D.4.6.2	Yes.
D.4.6.3	Yes.
D.4.7.1	Yes. Metamail uses the showexternal program to fetch data located on a ftp or mailserver site.

Overall, elm makes a good impression. Although, its screen-oriented user interface does not seem to be state-of-the art any more, e.g. it does not show all currently available commands to the user, has no elaborated file-selection and folder-selection mechanisms, does not provide a file-name completion, etc. Thus, an inexperienced elm user is forced to consult the manuals or the on-line help frequently if he wants to use commands that are not currently displayed in the status line. Compared with PINE, PINE's approach to solving this problem is exemplary.

The incorporated MIME support is sufficient for most needs. Due to passage of all MIME-compliant mails directly to metamail, the display capabilities include all content types and structures. The mechanisms for attaching non-textual body parts to a mail allow the simple insertion of any content type except of message/external. However, it is not possible to generate mails with a more complicated structure, e.g. containing a multipart/alternative part with even more multipart/mixed or multipart/parallel nested within it. Further, elm makes no use of the mailcap file to invoke external composers. Additionally, the forwarding of a MIME-compliant message leads to corruption of the MIME format.

4.3.3 Exmh

Exmh's MIME support mostly satisfies the user's needs. Exmh is able to recognize and decode all MIME-subtype definitions of RFC 1521. It shows all relevant header information on the body parts to the user, always supports saving the parts, and either uses the mailcap rules to externally display the parts, or passes them directly to metamail.

Exmh's built-in editor called sedit provides a set of text-editing and -formatting commands, e.g. automatic line breaks, changing fonts (bold, italic, smaller, larger, etc), cut and paste, and some emacs-like editing key bindings. Further, it supports the use of 8-bit character sets. To include any non-textual information, sedit provides a file dialog to include a file containing the body part. Upon selecting a file, another dialog appears, proposing a content type for the selected file and allowing the user to choose a content-transfer encoding algorithm.

The following tables give the results of the tests towards the specified criteria:

E.1	Exmh generates RFC 822 messages by default. By inserting any non-7-bit information the MIME header is generated immediately.
E.2	Yes.
E.3.1.1	Yes.

E.3.1.2	Exmh provides special key bindings allowing the user to insert non-7-bit ASCII characters.
E.3.1.3	Exmh provides richtext formatting, such as bigger, smaller, bold, italic, underline. The formatting is displayed correctly (WYSIWYG).
E.3.2.1	Yes.
E.3.2.2	Exmh allows to edit the MIME headers by hand. Thus, it is possible to create any message/* body part at will.
E.3.2.3	see E.3.2.2
E.3.2.4	see E.3.2.2
E.4.1.1	Yes. The charset parameter is not set.
E.4.1.2	Yes. The charset parameter is not set.
E.4.1.3	Yes. Though, the content type specification has to be corrected by hand.
E.4.2.1	Yes.
E.4.2.2	Yes.
E.4.3.1	Yes.
E.4.4.1	Yes.
E.4.4.2	Yes.
E.4.4.3	Yes.
E.4.5.1	Yes.
E.4.6.1	Yes.

D.1	Yes.
D.2	Yes.
D.3	Yes.
D.4.1.1	Yes.
D.4.1.2	Exmh is able to display messages with charset ISO-8859-1 inline.
D.4.1.3	Exmh is able to show richtext formatting inline.
D.4.2.1	Yes. Exmh shows the relevant header information of the body part. By clicking on it with the right mouse button, a menu appears, allowing the user to show the body part using the mailcap rule, saving it to a file, or passing it to metamail to decode it. Thus, the body part is displayed externally.
D.4.2.2	Yes. (see also D.4.2.1)
D.4.3.1	Yes. (see also D.4.2.1)
D.4.4.1	Yes. (see also D.4.2.1)
D.4.4.2	Yes. (see also D.4.2.1)
D.4.4.3	Yes. (see also D.4.2.1)
D.4.5.1	Yes. (see also D.4.2.1)
D.4.6.1	Yes. Exmh decodes the parts and shows the header information on each of them. The user can invoke a context-sensitive menu on each of the parts by using the right mouse button. These menus provide the facilities to externally view the parts.
D.4.6.2	Yes. Exmh chooses the best alternative. Further, it informs the user that there are different views of the same information. The user can switch to another alternative using the context-sensitive menu.

D.4.6.3	Exmh recognizes a multipart/parallel structure. It allows the user to save the parts to files, show them "inline" (all external viewers are called immediately), or to pass this subtype directly to metamail.
D.4.7.1	Yes. Using the context sensitive menu, the user can fetch the data or save a reference to it to a file.

Overall, exmh makes a very good impression. We have enjoyed using it during the test. Its graphical user interface makes it easy to use, even for computer novices. Writing mails with simple attachments is also very easy. Though only experienced users can use the full MIME functionality, such as multipart/alternative subtypes, or nested multipart/* structures.

The version used in the study has a problem with forwarding MIME-conformant mails. Actually, the forwarding of MIME messages is a topic that has also been discussed in the comp.mail.mime newsgroup. Additionally, it would be nice, if exmh used the "compose=" fields of the mailcap file to invoke external composers for the body parts.

4.3.4 Mercurius

Mercurius provides two editors for creating messages. One is a simple text editor for writing Rfc 822-compliant mails. The other one provides rudimentary MIME support. It allows attachment of any file to a mail-body, however, the choice of content type and transfer encoding is automatic. This precludes use of user-defined content types, e.g. text/x-html, or content types not currently supported, such as video/mpeg, without loss of the content type information, which is necessary for the receiving UA to properly display the body part. Textual parts are always enclosed in a text/richtext content type, even if they are simple Rfc 822-compliant mails.

The following tables give the results of the tests towards the specified criteria:

E.1	Yes.
E.2	Yes.
E.3.1.1	Yes.
E.3.1.2	No. Mercurius provides no means to insert non-US-ASCII characters with an ordinary ASCII-keyboard.
E.3.1.3	Yes. Mercurius supports richtext formatting, however, text justification does not work although it is offered by a pull-down menu.
E.3.2.1	Yes.
E.3.2.2	No. Mercurius allows only the editing of messages with sequential attachments.
E.3.2.3	No. (see E.3.2.2)
E.3.2.4	No. (see E.3.2.2)
E.4.1.1	Yes.
E.4.1.2	Yes, however, Mercurius does not recognize the character set of the imported file and uses charset=US-ASCII instead.

E.4.1.3	Yes, since Mercurius always uses the content type text/richtext for textual message parts. The imported file is not displayed properly, but the message has the right format.
E.4.2.1	Yes, however, Mercurius chooses image/x-ps as content type instead of using application/postscript. This is not compliant to the MIME standard.
E.4.2.2	Yes.
E.4.3.1	Yes. An audio part can be attached.
E.4.4.1	Yes. An image/gif part can be attached.
E.4.4.2	Yes. An image/jpeg part can be attached, but it is necessary, that the jpeg-file has the suffix .jpeg. Otherwise it is taken as application/octet-stream.
E.4.4.3	Yes. An image/tiff part can be attached, but it is necessary, that the tiff-file has the suffix .tiff. Otherwise it is taken as application/octet-stream.
E.4.5.1	No. Mercurius does not recognize the mpeg-format and chooses application/octet-stream instead.
E.4.6.1	No. Mercurius provides no means for editing a message/external-body content type.

D.1	Yes.
D.2	Yes.
D.3	Yes.
D.4.1.1	Yes.
D.4.1.2	Yes.
D.4.1.3	Yes. Richtext formatting can be displayed inline (even text justification and indentation is displayed properly)
D.4.2.1	Yes. Postscript documents are displayed using an external viewer.
D.4.2.2	Yes. The data can be stored to a file.
D.4.3.1	Yes. Audio/basic content types are recognized and displayed.
D.4.4.1	Yes. Gif images can be displayed with an external viewer.
D.4.4.2	Yes. Jpeg images can be displayed with an external viewer.
D.4.4.3	Yes.Tiff images can be displayed with an external viewer.
D.4.5.1	No. Mercurius does not recognize the video/mpeg content type. Instead, the UA shows the contents as application/octet-stream and offers to save it to a file.
D.4.6.1	Yes.
D.4.6.2	Yes. Mercurius recognizes the message/alternative content type and shows the best version to the user. It does not inform the user that alternative views of the data exist.
D.4.6.3	Yes. Mercurius recognizes the message/parallel content type. However, the parts are shown serially.
D.4.7.1	No. Mercurius shows a message/external body simply in textual form.

The UA provides a nice graphical user interface written with the script language Tk/Tcl. Its editor for MIME-compliant messages seems to be a bit unbalanced. It provides a fantastic support of richtext formatting. Attached parts are displayed as small icons, and an external viewer is invoked by clicking the icon with the mouse. On the other hand, it does not support such elementary content types as video/mpeg or message/external-body. Further, it provides no means to insert non-US-ASCII characters. A last point is that the UA does not support the creation of complex multipart/* messages, e.g. nested multipart/parallel content types within a multipart/mixed enclosure.

Mercurius does not use the configuration mechanism as described in [NB94]. Instead it makes use of a mailcap file of its own, which differs from that defined in the Internet draft. For example, it is not possible to specify an external viewer for the content types that are not supported, such as message/external body or video/mpeg. Additionally, it is not possible to specify external composers for the creation of the message part. However, it has been announced that a future release of Mercurius will support the standard mailcap as a resource for finding displaying facilities for body types not internally supported by Mercurius.

4.3.5 Metamail

Although, metamail itself is not a multimedia mailing system, but a toolkit consisting of several parts providing facilities to de- or encode MIME-compliant mails to existing mailing systems, it can also be used on a stand-alone basis for testing purposes.

To write mails, the metamail package provides the simple mailto program, a mail-composing agent similar to Berkeley Mail, but with some extensions allowing the creation of MIME-compliant mails.

To decode MIME-compliant mails received, the mails are passed to the metamail program, which is then responsible for all the decoding operations and the presentation of the received body parts.

Since metamail was developed by Nathaniel S. Borenstein, who has also been involved in the MIME definition process, the package can be regarded as a reference implementation. Thus, all mailers claiming to be MIME-compliant should allow at least the functionality provided by the metamail package.

For the encoding test, the mailto program, which is part of the metamail package, was used.

E.1	Yes.
E.2	Yes.
E.3.1.1	Yes. The charset parameter is not created automatically.
E.3.1.2	Yes. Mailto provides a way to insert 8-bit characters, but it seems to be much too complicated for daily use.
E.3.1.3	Yes. (see 3.1.1)
E.3.2.1	Yes. Having written the mail, mailto scans the mail body for attachments and encloses the text bodies and attachments within a multipart/mixed body part.

E.3.2.2	No. Although the user can insert MIME headers within the text body, mailto does not recognize this and encloses them like an ordinary text body within a multipart/mixed boundary.
E.3.2.3	No. (see E.3.2.2)
E.3.2.4	No. (see E.3.2.2)
E.4.1.1	Yes, if the corresponding item in the mailcap file is properly configured. At least, the field "compose=getfilename text %s" is needed.
E.4.1.2	No, unless no external composer for this body part is available which creates all the header information, including the charset-parameter, etc. Thus, the field "composetyped=external-composer text %s" is needed in the corresponding mailcap entry. Otherwise, the text will be encoded in base64.
E.4.1.3	Yes, if the corresponding item in the mailcap file is properly configured. At least, the field "compose=getfilename text %s" is needed.
E.4.2.1	Yes.
E.4.2.2	Yes.
E.4.3.1	Yes.
E.4.4.1	Yes.
E.4.4.2	Yes.
E.4.4.3	Yes.
E.4.5.1	Yes.
E.4.6.1	Yes.

For the decoding test, the received mails have been passed directly to the metamail program, which is part of the metamail package.

D.1	Yes.
D.2	Yes.
D.3	Yes.
D.4.1.1	Yes.
D.4.1.2	Yes. If properly configured in the mailcap file, metamail can display text with charset ISO-8859-1 e.g. using an xterm with an 8-bit character set.
D.4.1.3	Yes. Metamail displays richtext in the current terminal window using a simplified displaying mechanism, e.g. it displays italic characters inverted.
D.4.2.1	Yes. The part can be displayed using an external viewer.
D.4.2.2	Yes. The part can be written to a file or seen as text.
D.4.3.1	Yes. The part can be displayed using an external viewer.
D.4.4.1	Yes. The part can be displayed using an external viewer.
D.4.4.2	Yes. The part can be displayed using an external viewer.
D.4.4.3	Yes. The part can be displayed using an external viewer.
D.4.5.1	Yes. The part can be displayed using an external viewer.
D.4.6.1	Yes.

D.4.6.2	Yes.
D.4.6.3	Yes.
D.4.7.1	Yes. Metamail uses the showexternal program to fetch data located on a ftp or mailserver site.

Mailto only provides a very simple interface to write MIME-compliant mails. Besides which the use of some features is so complicated that the program cannot at all be considered to be user-friendly.

On the other hand, mailto demonstrates perfectly how a mail-composing agent should incorporate the mailcap file to invoke external composing agents for the editing of non-textual message parts. Actually, mailto was the only mail composer considered in the test that used the information provided by the mailcap file. Furthermore, it was also the only program that by means of MIME's message/partial mechanism split into several parts mails that were possibly too long for transport via SMTP gateways.

The decoding done by the metamail program is also exemplary. It also makes extensive use of the mailcap file to get information on how to display message parts which cannot be shown inline, using external viewers. Upon encountering an unrecognized content type value, mailcap offers to view the part as text, to write it to a file or to skip it. Thus, in conjunction with a properly defined mailcap file, metamail is able to display any MIME-compliant mail.

4.3.6 Meuf

Meuf is the only MUA considered in the study that comes accompanied by two inline editors for MIME messages. These provide not only different capabilities of editing MIME-compliant mails, but also have completely different user interfaces.

The first simply allows sequential attachment of non-textual files to a mail body. The window is splitted into four parts. The first contains the header information, the second is for editing a plain text using the US-ASCII character set. The third shows the currently attached non-textual body parts, and the last subwindow displays different icons, one for each content type supported. Attaching a non-textual body part is a matter of dragging such an icon into the attachment part of the window. This mechanism is comparable to the templates used by the OS/2 operating system. By clicking on an icon within the attachment subwindow, it is possible to specify the properties, e.g. the filename, of the content type to be attached.

The second editor is designed especially to create MIME messages which make use of multipart/mixed or multipart/alternative content types. The window is splitted into three parts. The first is for editing the header of the message. In the second, the user can create the tree of the structure of the mail. This can be done by simply dragging the template icons from the third subwindow and dropping them into the second one. The message is displayed as a tree - comparable to that of a hierarchical file system displayed by a graphical file manager - showing the content types as nodes and using links to visualize the structure of embedded body parts. This editor was used to test the criteria E.3.2.2 to E.3.2.4, whereas we used the simple editor to do the rest of the testing.

The following tables give the results of the tests towards the specified criteria:

E.1	Yes.
E.2	Yes.
E.3.1.1	Yes.
E.3.1.2	Yes. It is possible to insert a text/plain body part using the extended characters of ISO-8859-1, however, the charset- and content-transfer-encoding parameters must be set by hand or pre-configured as the default values for the text/plain content type. The latter is essential if using an extended character set in the main edit field of the simple editor.
E.3.1.3	No. The next release, 3.1, is intended to support richtext formatting facilities.
E.3.2.1	Yes.
E.3.2.2	Yes.
E.3.2.3	Yes.
E.3.2.4	Yes.
E.4.1.1	Not supported. Meuf provides no means to import a text file into a text/plain mail-body.
E.4.1.2	Not supported. (see E.4.1.1)
E.4.1.3	Not supported. (see E.4.1.1)
E.4.2.1	Yes.
E.4.2.2	Not tested. Body parts can only be inserted selecting the actual content type to be inserted. Meuf provides for insertion of a raw file, which is then given the application/octet-stream content type.
E.4.3.1	Yes.
E.4.4.1	Yes.
E.4.4.2	Yes.
E.4.4.3	Yes.
E.4.5.1	Yes.
E.4.6.1	No. Meuf cannot create a message/external body subtype internally, but an external composer is invoked if specified in the meufcap file. However, the extcompose program of the metamail package cannot be used as an external composer since meuf's configuration mechanism does not provide a "composetyped" parameter. Thus, the created header field is not in the correct format. No other external composers are available for this content type.

D.1	Yes.
D.2	Yes.
D.3	Yes.
D.4.1.1	Yes.
D.4.1.2	No. If displayed inline, the data is shown unencoded. Passing the body part to metamail does not work either since meuf strips off all relevant header information before invoking metamail.

D.4.1.3	No. If displayed inline, the data is shown without formatting. Passing the body part to metamail does not work, as well, since meuf strips off all relevant header information before invoking metamail.
D.4.2.1	Yes. Meuf offers to display the body using an external viewer, or to save the information to a file.
D.4.2.2	Yes. The body can be written to a file.
D.4.3.1	Yes. External viewing and saving is offered.
D.4.4.1	Yes. Gif images can be displayed inline or written to a file.
D.4.4.2	Yes. External viewing and saving is offered.
D.4.4.3	Yes. External viewing and saving is offered.
D.4.5.1	Yes. External viewing and saving is offered.
D.4.6.1	Yes. External viewing and saving is offered.
D.4.6.2	Yes. The multipart/alternative structure is recognized. The user is asked which alternative he wants to see. An overview on the structure of the whole message can be displayed.
D.4.6.3	Yes. Meuf recognizes a multipart/parallel subtype, however, the parts are displayed serially.
D.4.7.1	Yes. A message/external body can be shown using an external viewer, such as the showexternal program of the metamail package.

Meuf provides a very powerful and easy-to-use GUI. The approach to editing a complex MIME-compliant message as a hierarchical tree seems to be well suited to create mails containing nested multipart/mixed, multipart/alternative, or multipart/parallel content types. It is also well suited to allow graphic display of the structure of a received message and to allow direct access to each attached content type by simply clicking it with the mouse is a nice feature.

Meuf's displaying capabilities do not rely on the metamail package. This is achieved by providing inline decoding facilities and displaying capabilities for plain text and gif-images. External viewers for all other content types may be specified in a text-oriented configuration file called meufcap. The software package provides a GUI-oriented editor for modifying this file.

The syntax of the meufcap file is similar to that of the mailcap mechanism. Some extensions have been made, e.g. it is possible to define a set of encoding algorithms for a given content type. This information is then used by the graphical editors to display a selection of transfer encoding which can be applied to the content type. However, some fields defined by the mailcap mechanism are not supported by the meufcap configuration file (see E.4.6.1).

However, the editing facilities provided by the current version should be improved. The UA should be able to automatically recognize the use of a non-US-ASCII character set. Further the inline formatting of richtext should be supported. A last point of criticism is that a message/external body content type cannot be created using the currently available external composing agents.

Finally it should be mentioned that the version used in the study, a 3.0 alpha release for Solaris 2.3, was very unstable and often crashed. Even such simple actions as changing to another folder sometimes ended up in a core dump, which makes this version of the software unacceptable for daily use. Although a beta release was announced for the 1st of March, it has not become available yet.

4.3.7 PINE

Generally, PINE supports the MIME standard by allowing the user to save MIME objects to files. For the content types image/*, PINE can also initiate an external viewer to display the object. The viewer can be specified within the setup of PINE. For the study, PINE was configured to use the system's mailcap configuration file. Thus, any content types that cannot be handled internally are passed to the viewers specified in the mailcap file.

PINE's message composition editor, Pico, is a very simple and easy-to-use text editor. It does not provide richtext formatting facilities. Nor does Pico have integral multimedia capability, but any type of data file - including multimedia - can be attached to a text message and sent using MIME's encoding rules. According to the MIME standard, PINE uses the Content-Type Application/Octet-Stream for attachments of an unrecognized format.

The following tables give the results of the tests towards the specified criteria:

E.1	Yes. The "MIME-Version: 1.0" header field is always generated.
E.2	Yes.
E.3.1.1	Yes.
E.3.1.2	No. Pico does not support the insertion of non-US ASCII characters
E.3.1.3	No. Pico does not support any text/richtext formatting facilities.
E.3.2.1	Yes. Pico always generates a multipart/mixed subtype as the outermost boundary.
E.3.2.2	No. Pico does not support user-defined multipart/* structures. Since Pico does not display the MIME headers of the body parts, it is not possible to insert these structures by hand.
E.3.2.3	see E.3.2.2
E.3.2.4	see E.3.2.2
E.4.1.1	Yes. Pine selects the proper content type value, but encodes the body part using base64 encoding. The charset value is set properly.
E.4.1.2	No. Pine does not recognize a file containing text with charset ISO-8859-1. It selects application/octet-stream as content type and does a base64 encoding.
E.4.1.3	Yes. Pine selects the proper content type value, but encodes the body part using base64 encoding. The charset value is set properly.
E.4.2.1	Yes.
E.4.2.2	Yes
E.4.3.1	Yes.
E.4.4.1	Yes.

E.4.4.2	Yes.
E.4.4.3	Yes.
E.4.5.1	Yes, but Pico does not recognize the video/mpeg format. Thus it chooses the application/octet-stream subtype and does a base64 encoding.
E.4.6.1	No.

D.1	Yes.
D.2	Yes. (see also D.4.4.1)
D.3	Yes. (see also D.4.6.1)
D.4.1.1	Yes.
D.4.1.2	Yes. For the test PINE was running in an xterm using an 8-bit font, and thus was able to display these mails inline.
D.4.1.3	Yes. PINE is able to display the text/richtext subtype inline. Though, it uses a simplified display, e.g. underlining instead of a bigger font.
D.4.2.1	PINE uses the mailcap rule to handle this subtype. Thus, the body part is shown external.
D.4.2.2	Yes.
D.4.3.1	Yes. PINE uses the mailcap rule to handle this subtype. Thus, the body part is shown external.
D.4.4.1	Yes. PINE uses the mailcap rule to handle this subtype. Thus, the body part is shown external. A restriction is, that PINE cannot properly decode an image in quoted printable encoding.
D.4.4.2	see D.4.4.1
D.4.4.3	see D.4.4.1
D.4.5.1	Yes. PINE uses the mailcap rule to handle this subtype. Thus, the body part is shown external.
D.4.6.1	PINE properly handles a multipart/mixed subtype, if it is not layered within other subtypes, e.g. within a multipart/alternative part.
D.4.6.2	The same restrictions apply as for D.4.6.1.
D.4.6.3	The same restrictions apply as for D.4.6.1.
D.4.7.1	Yes. PINE passes a message/external body subtype to metamail. Thus, the body part is shown external.

PINE makes a good first impression. It is user-friendly, providing an on-line help system and always showing the currently available commands in the status line. Thus, the program is easy to learn, even for novice computer users. Furthermore, in addition to simple text-oriented mail, it supports the attachment of non-textual body parts.

On the other hand, it was not the primary intention of PINE's developers to write a MIME mailer. Thus, the support of MIME is limited to sequential attachments of multimedia objects using a MIME encoding. No external composing agents are invoked, even if they are specified in the mailcap file. Further, the MIME headers cannot be edited by hand, making it impossible to use any user-defined subtypes or to create any multipart subtype other than multipart/mixed. The restriction not to provide a means of inserting non US-ASCII characters delimits the use in non-English-speaking countries, and PINE does not provide any text formatting facilities. Finally, upon receiving a message/partial body part, PINE cannot display the message, although it invokes metamail's showpartial program.

4.3.8 Tkmailto

Tkmailto is a Tk-based mail composer that supports only the creation of MIME-conformant messages. Like mailto, tkmailto cannot be used to read or process messages. Thus, the testing was reduced to the editing test.

The editor provides a means to easily insert, modify or cut almost all MIME-specific content types. It is highly configurable, so that most of the information needed to attach the body part, e.g. content types, ftp domains, ftp directories, mailservers and transfer modes can be selected with a menu and must not be typed in by hand.

The following table gives the results of the tests towards the specified criteria:

E.1	Yes, the "MIME-Version: 1.0" header field is always generated.
E.2	Yes.
E.3.1.1	Yes.
E.3.1.2	No. Tkmailto does not provide to insert any non-US-ASCII characters.
E.3.1.3	No. Tkmailto does not provide any text formatting facilities.
E.3.2.1	Yes. Tkmailto automatically encloses all text bodies and attachments into a multipart/mixed body part.
E.3.2.2	No. Although the user can insert MIME headers within the text body, tkmailto does not recognize this and encloses it like an ordinary text body within a multipart/mixed boundary.
E.3.2.3	No. (see E.3.2.2)
E.3.2.4	No. (see E.3.2.2)
E.4.1.1	Yes, but the body is encoded using the base64 algorithm and no charset parameter is set. The latter can be done by hand.
E.4.1.2	Yes. The same restrictions as for E.4.1.1 apply.
E.4.1.3	Yes. The same restrictions as for E.4.1.1 apply.
E.4.2.1	Yes.
E.4.2.2	Not tested. Body parts can only be inserted, selecting the actual content type to be inserted from a menu. This menu provides for insertion of a raw file, which is then given the application/octet-stream content type.
E.4.3.1	Yes, via the "insert raw file" menu option, the user can insert an audio/basic body part. The content type must be selected from a configurable list of defined content types or be typed in by hand.
E.4.4.1	Yes.
E.4.4.2	Yes. (see also E.4.3.1)
E.4.4.3	Yes. (see also E.4.3.1)
E.4.5.1	Yes. (see also E.4.3.1)
E.4.6.1	Yes.

Tkmailto makes a good overall impression. Its editing facilities will satisfy most of the user's needs. The graphical user interface makes it easy to learn and use. If properly configured, the attachment of MIME objects is simply a matter of selecting several menu options.

On the other hand, tkmailto does not support the editing of complex MIME messages, e.g. nested multipart/* structures. It only provides means to sequentially attach MIME objects to a mail, which are then combined within a multipart/mixed structure. Then it must be mentioned that tkmailto makes no use of the information stored in the mailcap file. Instead, the user must twice define the content types he wants to use: first, in the mailcap file and second, in tkmailtorc. Furthermore, no external composing agents are invoked, even if they are specified in the mailcap file. The restriction not to provide a means to insert non-US-ASCII characters delimits the use in non-English-speaking countries. The last point of criticism is the lack of text formatting facilities.

4.4 Interoperability Tests

4.4.1 Scenarios

The aim of the interoperability test is to prove the practical suitability of the MUAs mentioned above to exchange multimedia mails between different users working on workstations from different vendors and using different mail user agents. We created five scenarios showing some possible real-life situations in terms of some different kind of information one user would like to send to another one using a specific UA. To reduce the extent of this test, we concentrated on five mail user agents including four MIME-compliant systems and Sun's mailtool. The MIME software packages considered are Elm, PINE, exmh, and tkmailto[1]. The mailers should prove their abilities to send and receive the messages without any loss of information, e.g. a video sent by one mailer should be recognized by the receiving agent and displayed using either internal or external displaying facilities.

The created scenarios should reflect five different situations a user of multimedia mail could run into.

S.1
A mail consisting only of a formatted richtext part using the ISO-8859-1 character set is created on one UA and received by another one.

S.2
A mail including a textual body part and a GIF-image attachment is created on one UA and received by another one.

S.3

[1] Since tkmailto is a mail composer, only its editing facilities are considered.

A mail including a textual body part and a MPEG-video attachment is created on one UA and received by another one.

S.4
A mail including a textual body part and an audio attachment is created on one UA and received by another one.

S.5
A mail including a slide-show is created on one UA and received by another one. The term slide-show is meant to be a sequence of images, each accompanied by an audio part describing the image. Using the MIME vocabulary, the message consists of an enclosing multipart/mixed structure with several nested multipart/parallel bodies, each containing an image and an audio content type.

4.5 Interoperability Test Results

4.5.1 Test Results

The results of the interoperability tests are summarized in the following tables. The first line of the table shows the names of the UAs used for creating and sending the message, the first column shows the receiving mailers.

S.1 richtext

	Elm	PINE	exmh[3)]	tkmailto	Sun mailtool
Elm	N[1)]	N[2)]	Y	N[6)]	N[7)]
PINE	N[1)]	N[2)]	Y[4)]	N[6)]	N[7)]
exmh	N[1)]	N[2)]	Y[4)]	N[6)]	N[7)]
Sun mailtool	N[1)]	N[2)]	N[5)]	N[6)]	N[7)]

[1)] Elm provides no facilities for editing text/richtext content types.
[2)] PINE provides no facilities for editing text/richtext content types.
[3)] Exmh supports only a subset of richtext formatting, e.g. provides no text justification. Therefore, content types including formatted text are specified as text/enriched instead of text/richtext.
[4)] Since no graphical terminal is running, a simplified mechanism is used to display richtext format.
[5)] Sun mailtool recognizes a MIME content type. It is possible to save it to a file or view it as plain text including the formatting commands, but without any formatting.
[6)] tkmailto provides no facilities for editing text/richtext content types.
[7)] Sun mailtool provides no facilities for editing text/richtext content types.

S.2 text & image

	Elm	PINE	exmh	tkmailto	Sun mailtool
Elm	Y[8)]	Y[8)]	N[9)]	Y[8)]	Y[10)]
PINE	Y[8)]	Y[8)]	N[9)]	Y[8)]	N[11)]

exmh	Y	Y	Y	Y	Y[10]
Sun mailtool	Y	Y	N[9]	Y	Y

[8] If running in a terminal window under X11, an external viewer is invoked. Otherwise, the data can be saved to a file.
[9] Since exmh does not automatically insert a blank line between the text/plain header and the following text/plain body part, the receiving user agents are not able to distinguish between the header and the body and thus misinterpret the textual part of the message. We reported this to exmhbugs@parc.xerox.com.
[10] The Sun format is recognized and decoded using the sun2mime tool of the metamail package as specified in the mailcap file.
[11] PINE does not recognize the Sun format and shows the whole message as plain text.

S.3 text & video

	Elm	PINE	exmh	tkmailto	Sun mailtool
Elm	Y[8]	N[13]	N[9]	Y[8]	N[14]
PINE	Y[8]	N[13]	N[9]	Y[8]	N[14]
exmh	Y	N[13]	Y	Y	N[14]
Sun mailtool	N[12]	N[13]	N[9]	N[12]	N[14]

[12] Sun mailtool does not recognize the mpeg-video format. The attachment can only be saved to a file.
[13] PINE does not recognize the mpeg-video format. The content type is set to the value application/octet-stream, and thus the content information is lost.
[14] Sun mailtool does not recognize the mpeg-video format. The x-sun-content type is set to the value default, and thus the content information is lost.

S.4 text & audio

	Elm	PINE	exmh	tkmailto	Sun mailtool
Elm	Y[15]	Y[15]	N[9]	Y[15]	Y[10]
PINE	Y[15]	Y[15]	N[9]	Y[15]	N[11]
exmh	Y[15]	Y[15]	Y	Y[15]	Y[10]
Sun mailtool	Y	Y	N[9]	Y	Y

[15] The audioplayer defined in the mailcap file is used to display the information.

S.5 dia-show

	Elm	PINE	exmh[17]	tkmailto	Sun mailtool
Elm	N[16]	N[16]	Y[18]	N[16]	N[16]
PINE	N[16]	N[16]	Y[18]	N[16]	N[16]
exmh	N[16]	N[16]	Y[19]	N[16]	N[16]
Sun mailtool	N[16]	N[16]	Y	N[16]	N[16]

[16] The creation of complex MIME structures, such as nested multipart/* content types, is not supported.
[17] Exmh's editor can be used to create complex MIME structures, however, the boundaries must be edited by hand.
[18] The body parts are shown serially and the user gets no information on the actual structure of the message.
[19] In certain situations exmh behaved a bit strangely while displaying the body parts, e.g. an enclosed multipart/parallel vanished and the whole message had to be redisplayed to bring it up again. We reported this to exmhbugs@parc.xerox.com.

5 Summary

The MIME approach seems to be the most reasonable effort for allowing the sending and receiving of multimedia messages using standard Internet mail transport facilities. Providing new header fields, such as MIME-Version, Content-Type, and Content-Transfer-Encoding, it is now possible to include various kinds of information types, e.g. audio, images, richtext, or video, into a RFC 822-conformant mail. Making use of these headers, it is possible to fully describe an attached body part, so that a receiving mail user agent is able to display it without any loss of information. Additionally, the definition of the "multipart" and "message" content types allows the creation of hierarchical structured mails, e.g. a message containing two alternative parts of information, one that can be shown using a simple ASCII-terminal, the other to be displayed on a multimedia workstation. Allowing the definition of bilaterally defined content types and providing a standardized means of establishing new content types prevent MIME from being a one-way road and supply mechanisms to extend MIME for future use.

MIME has been defined in such a way that no restrictions imposed by either RFC 821 or RFC 822 are violated, thus allowing MIME messages to be sent using already established and frequently used transport agents such as sendmail. Therefore, MIME possibly leads to a smooth transition from already established text-oriented RFC 822 mail to electronic messages containing many kinds of media since they can be used in parallel. This transition will be additionally reinforced by the facts that MIME is not a proprietary standard, is freely available, and that many MIME-compliant MUAs have already been developed that can be obtained via the Internet and used for free.

The MIME-conformant mail user agents considered in this study are all of a high quality. They all provide at least the minimum MIME conformance as defined in RFC 1521. On the other hand, no user agent tested fulfilled all the demands we defined in 4.2, though some of them came pretty close.

Deciding which mailer to use strongly depends on the environment. Consider the four following scenarios:

Someone working on a Unix system with an X11 Window-System will probably use a mail user agent providing a graphical user interface. Thus, meuf, Mercurius, AMS, and exmh are the possible candidates. Since the current version of meuf is not stable enough, and Mercurius' MIME-editing and -displaying facilities are restricted to certain content types, AMS or exmh will win out. Both rely on the metamail package to provide MIME support. Exmh additionally needs the MH package, since it is only a graphical user interface for MH commands. The displaying facilities of these two user agents are almost equivalent. However, exmh supports the composition of more MIME content types than does AMS, e.g. nested multipart/* structures or message/external body content types can be inserted into a mail message. The editor of exmh, sedit, provides very flexible editing facilities since the message is shown in a pure textual format including all MIME headers and even encoded body parts in base64 representation. On the other hand, showing attached information as a sequence of characters will probably confuse a novice computer user. Thus, we would recommend using exmh as the mail user agent, if working on an X11 window system, but providing an alternative editor, such as tkmailto, for less-experienced users.

If working on a system equipped with an ASCII-terminal, one has to use one of the character-oriented mail user agents. In this paper, we present three mail user agents, which can be run on a terminal: elm, PINE and AMS. However, installing the whole Andrew package to simply read mails on a terminal screen seems akin to using a sledgehammer to crack a nut. PINE and elm are more reasonable solutions. Elm supersedes PINE as well as far as the composing and the displaying facilities are concerned. On the other hand, PINE offers easier attachment of multimedia objects to a mail since the user does not have to know anything about content types or transfer encodings. Additionally, PINE's user interface seems to be more user-friendly than elm's. It is a matter of taste, but we would still prefer PINE, although it's MIME support is not as good as elm's.

Another case might be a user who works on an X11 UNIX workstation in his office and also wants to read his mail at the week-end or during the holidays, or who has to travel around a lot, connected to the office via a modem using radio traffic. Since a common 14.400 kbit/s modem connection will have to cope with a very limited bandwidth, and electronic mails containing multimedia objects tend to be very large (a simple audio file encoded using PCM at a sampling rate of 8000 Hz takes about 8 kB per second), he should use a mail user agent running on a terminal, such as elm, AMS or PINE. He could then read the textual parts of the mail, save the attachments to the workstation's file system and selectively download them, if needed immediately. However, this might lead to conflicts in terms of folder management and aliases if he wants to use a mailer with a GUI back in the office, since there are two major styles for folder management (Berkeley and MH) and each mailer uses its own aliases system. A possible solution is to use AMS, which provides both an ASCII-terminal interface and a graphical user interface. Another possibility, following the recommendations given above, is to use exmh as a GUI-based mailer and PINE on the terminal. This minimizes the conflicts mentioned above since PINE can work with MH folders as long as they are not nested.

A final scenario is someone working on a PC using a 64-kbit ISDN connection to a mail server over a TCP/IP protocol stack. The exclusive use of a 64-kbit link by one user will surely allow the transmission even of large multimedia mails with a reasonable consumption of time[2]. Thus, the choice of a mail user agent depends on the system running on the PC. If using the Linux operating system with XFree, one could use the exmh mail user agent. Elm and PINE are two alternatives for a PC running a DOS/Windows system. Elm has also been ported to OS/2. OS/2 Warp also includes a mail user agent within the Bonus Pack. This MUA, Ultimedia/2 Lite, also provides minimal MIME conformance.

[2] It is a fact, that until autumn 1994 two single 64-kbit links connected the University of Mannheim to the Internet, dealing with all the email, WWW, telnet, and ftp traffic. It was not too fast, but it worked.

6 Acknowledgment

We express here our appreciation both of the time and development effort spent by many people all over the world on designing software packages such as the MIME-conformant user agents considered in this study and of their subsequent free distribution to the Internet community.

We would like to thank our colleagues at IBM ENC Heidelberg for the excellent cooperation during this project.

7 References

[GIF] Graphics Interchange Format (Version 89a), Compuserve Inc., Columbus, Ohio, 1990

[PCM] CCITT, Fascicle III.4 - Recommendation G.711, Geneva, 1972, "Pulse Code Modulation (PCM) of Voice Frequencies".

[MPEG] Video Coding Draft Standard ISO 11172 CD, ISO IEC/TJC1/SC2/WG11 (Motion Picture Experts Group), May, 1991.

[NB90] Borenstein, Nathaniel S., Multimedia Applications Development with the Andrew Toolkit. Englewood Cliffs, New Jersey. 1990

[NB91] Borenstein, Nathaniel S., Adding Diverse Multimedia Format Support to Established RFC 822 Mail and Bulletin Board Readers, Bellcore, 1991.

[NB94] Borenstein, Nathaniel S., A User Agent Configuration Mechanism for Multimedia Mail Format Information, Internet Draft, IETF, May 1994.

[PS] Adobe Systems, Inc., PostScript Language Reference Manual, Addison-Wesley, Second Edition, 1990

[X400] CCITT, X.400 Message Handling System and Service Overview, Recommendation X.400(92), ITU, 1993.

[X420] CCITT, X.420 Message Handling Systems - Interpersonal Messaging System, Recommendation X.420(92), ITU, 1993.

[MR93] Rose, Marshall T., The Internet Message - Closing the Book with Electronic Mail, Englewood Cliffs, New Jersey, 1993

[FH92] Halsall, Fred, Data Communications, Computer Networks and Open Systems, Addison-Wesley, Third Edition, 1992

[MA94] M. Altenhofen, et al., The BERKOM Multimedia Teleservices, Proc. IWACA '94, Heidelberg, 1994

[RFC 783] Sollins, K., TFTP Protocol (revision 2), RFC 783, MIT, June 1981.

[RFC 822] Crocker, D., Standard for the Format of ARPA Internet Text Messages, STD 11, RFC 822, UDEL, August 1982.

[RFC 934] Rose, M. and E. Stefferud, Proposed Standard for Message Encapsulation, RFC 934, Delaware and NMA, January 1985.

[RFC 959] Postel, J. and J. Reynolds, File Transfer Protocol, STD 9, RFC 959, USC/Information Sciences Institute, October 1985.

[RFC 1049] Sirbu, M., Content-Type Header Field for Internet Messages, STD 11, RFC 1049, CMU, March 1988.

[RFC 1521] Borenstein, N. and N. Freed, MIME Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies, RFC 1521, Bellcore, Innosoft, September 1993

[RFC 1522] Moore, K., Representation of Non-ASCII Text in Internet Message headers, RFC 1522, University of Tennessee, September 1993.

8 Appendix

A A User Agent Configuration Mechanism

A.1 The Format of the Mailcap File

The mailcap mechanism provides a way to inform multiple mail-reading user agent programs about the locally installed facilities for handling mail in various formats. Thus, it is a user agent configuration mechanism for multimedia mail format information.

Moreover, this mechanism makes it easy to individually configure the mail-reading software, e.g. extending the set of media types supported at a site becomes a simple matter of installing a binary and adding a single line to the mailcap file, or if a user wants to use the program XY to display gif pictures instead of the pre-configured program AB, of simply putting a corresponding line into his own mailcap file located in his home directory.

Each time a mail-reading user agent encounters a body part of a content type which the software itself cannot handle, it looks up the mailcap file for information on how to display this specific content type. The configuration information will be obtained from the first matching entry in a mailcap file, where "matching" depends on both a matching content type specification, an entry containing sufficient information for the purposes of the application doing the searching, and on the success of any test in the "test=" field, if present.

The syntax of a mailcap file is quite simple. Any line that starts with "#" is a comment. Blank lines are ignored. Each line not starting with a "#" character defines a single mailcap entry for a single content type. Long lines may be continued by ending them with a backslash character, "\".

Each individual mailcap entry consists of a content type specification, a command to execute, and (possibly) a set of optional "flag" values. For example, a very simple mailcap entry would look like this:

```
        text/plain; cat %s
```

A general mailcap entry may consist of the following mandatory fields:

The "type" field
The "type" field is simply any legal content type name, as defined by RFC 1521. This field will be matched against the "Content-type" header to choose the right mailcap entry to handle the current message.

The "command" field
The "command" field is any UNIX command, and is used to specify the interpreter for the given type of message. It will be passed to the shell via the system facility. A sequence of "%s" within the command string will be expanded to the name of a file that contains the body of the message. "%t" will be replaced by the content type field, including the subtype, if any. If the command field contains "%{" followed by a parameter name and a closing "}", then all those characters will be replaced by the value of the named parameter, from the Content type header, e.g. %{charset} will be expanded to the value of the character set parameter of the text/plain header.

The optional fields, that can appear in any order, are as follows:

The "compose" field
The "compose" field may be used to specify a program that can be used to compose a new body or body part in the given format. Its intended use is to provide the necessary information to mail-composing agents which allows to call external composing agents for the editing of the body parts.

The "composetyped" field
The "composetyped" field is similar to the "compose" field. It is to be used when the composing program additionally needs to specify the Content type header field to be applied to the composed data.

The "description" field
This field simply provides a textual description of the type of data enclosed in the message body part. This field is to be used optionally by mail readers that wish to describe the data before offering to display it.

The "edit" field
This field is to specify a program which can be used to edit the body part. It may be identical to the compose field.

The "nametemplate" field
This field gives a file name format, such as %s.gif, leading to the suffix ".gif" being appended to the name of the temporary file which is to be passed to the viewing command.

The "print" field
The "print=xxx" field is a command that is executed to print the data instead of display it interactively.

The "test" field
The "test=xxx" field is a command that is executed to determine whether or not the mailcap line actually applies. That is, if the content type field matches the content type on the message, but a "test=" field is present, then the test must succeed before the mailcap line is considered to "match" the message being viewed.

The "textualnewlines" field
If this field is set to any non-zero value, it indicates that this type of data is line-oriented and that if encoded in base64, all new lines should be converted to canonical form (CRLF) before encoding, and will be in that form after decoding. In general, this field is not needed for text/* content types.

The "x11-bitmap" field
This field names a X11-bitmap file in xbm-format that is to be used to visually indicate the presence of a body part of the specified type/subtype.

The optional flags can be used to specify additional information on the mail-handling command.

needsterminal
If this flag is given the named interpreter needs to interact with the user on a terminal.

copiousoutput
This flag should be given whenever the interpreter is capable of producing more than a few lines of output on stdout, and the content is intended to be piped through a pagination program, such as more.

needsx11
This flag indicates, that this mailcap entry should only be used if the X11 window system is running.

A.2 A sample mailcap file

```
# Mailcap File modified by GD to meet specific needs of the local site
#
# last modified: 03/15/95

#==============================================================
# text/* content-types
#==============================================================
text/plain;     cat %s;\
                test=test "`echo %{charset} | tr '[A-Z]' '[a-z]'`" = iso-8859-1;\
                description="text/plain in ISO-8859-1"; \
                compose="getfilename text/plain %s; \
                copiousoutput

text/plain;     cat %s;\
                description="text/plain"; \
                compose="getfilename text/plain %s";\
                copiousoutput
```

```
text/x-html;     Mosaic %s; \
                 description="A html Page";\
                 compose="tkwww %s";\
                 needsx11

text/richtext;   shownonascii -*-courier-medium-r-*-*-12-* -e richtext -p %s;\
                 description="text/richtext in ISO-8859-1"; \
                 test=test "`echo %{charset} | tr '[A-Z]' '[a -z]'`"  = iso-8859-1; \
                 compose="getfilename text/richtext %s";\
                 copiousoutput

text/richtext;   shownonascii -*-courier-medium-r-*-*-12-* -e richtext -p %s;\
                 description="text/richtext"; \
                 compose="getfilename text/richtext %s";\
                 copiousoutput

text/enriched;   shownonascii -*-courier-medium-r-*-*-12-* -e richtext -p %s;\
                 description="text/enriched in ISO-8859-1"; \
                 test=test "`echo %{charset} | tr '[A-Z]' '[a-z]'`"  = iso-8859-1;\
                 compose="getfilename text/richtext %s";\
                 copiousoutput

text/enriched;   shownonascii -*-courier-medium-r-*-*-12-* -e richtext -p %s;\
                 description="text/enriched"; \
                 compose="getfilename text/richtext %s";\
                 copiousoutput

#==================================================================
# application/* content-types
#==================================================================
application/postscript; \
                 lpr %s ;\
                 description="A Postscript File";\
                 compose="getfilename application/postscript %s"

#==================================================================
# image/* content-types
#==================================================================
image/jpeg;      showpicture -viewer display %s; \
                 description="An image in JPEG format"; \
                 compose="getfilename image/jpeg %s"

image/gif;       showpicture -viewer display %s; \
                 description="An image in GIF format"

image/gif;       showpicture -viewer display %s; \
                 compose="xwd -frame | xwdtopnm | ppmtogif > %s\; exit 0"; \
                 description="An X11 window image dump in GIF format"
```

```
image/*;        showpicture -viewer display %s; \
                description="An image"; \
                compose="getfilename image/* %s"

#==================================================================
# video/* content-types
#==================================================================
video/mpeg;     mpeg_play %s; \
                description="A video in MPEG encoding"; \
                compose="getfilename video/mpeg %s"; \
                needsx11

#==================================================================
# audio/* content-types
#==================================================================
audio/basic;    showaudio %s; \
                compose=audiocompose %s; \
                edit=audiocompose %s; \
                description="An audio fragment"

#==================================================================
# message/* content-types
#==================================================================
message/partial; \
                showpartial %s %{id} %{number} %{total}

message/external-body; \
                showexternal %s %{access-type} %{name} %{site} \
                                %{directory} %{mode} %{server}; \
                composetyped = extcompose %s; \
                description="A reference to data stored in an external location"
```

I. CONTENT

II. ILLUSTRATIONS