The Design and Implementation of the MONTAGE Multimedia Mail System

W. Keith Edwards

Georgia Institute of Technology
Software Engineering Research Center
Multimedia Computing Group
Atlanta, GA 30332-0280
keith@cc.gatech.edu

ABSTRACT

Electronic mail systems capable of transmitting compositions consisting of various unconventional media (such as voice, video, and images) have attracted a substantial amount of interest in recent years. It is important that mailers capable of delivering such documents present them in an organized fashion. We present the Montage multimedia electronic mail system, along with its model for multimedia documents. Montage makes use of a simpler format than more generalized hypermedia systems. It is our belief that the Montage model is more effective than general hypertext for the task of creating user-to-user messages. Furthermore, Montage is designed to be runtime extensible to new media types by its users. Thus the system does not have to know ahead of time all the possible media users may want to send. The system is built on top of existing mail transport protocols for flexibility and portability. We discuss the design of the mailer along with experiences gained from its implementation. The user interface to the system is also presented.

KEYWORDS

Multimedia mail, electronic mail, compound documents.

INTRODUCTION

mon•tage män-'täzh, *n.*

1 *the production of a rapid succession of images in a motion picture to illustrate an association of ideas*

2a *a literary, musical, or artistic composite of juxtaposed more or less heterogeneous elements*

2b *a composite picture made by combining several separate pictures*

3 *a heterogeneous mixture*

Webster's Ninth New Collegiate Dictionary
Copyright © 1989

In the past decade, the proliferation of fast, inexpensive, networked computer workstations has produced an explosion in the use of electronic mail. Electronic mail systems have traditionally been limited to the transmission of pure textual information only.

As computer workstations increase in power, and as the use of windowing interfaces becomes more widespread and standardized, it becomes apparent that capabilities now exist for the transmission and reception of complex multimedia documents consisting of voice, images, video, and other media types in addition to plain text.

The chief advantage of such a system is the increased communication bandwidth between the users of the system. A multimedia mail system should allow people to communicate as freely and without restriction as conventional ''paper'' mail systems do. In current paper mail systems, users can seal most anything within an envelope and expect prompt delivery. Electronic multimedia mail systems should allow similar flexibility.

Furthermore, to be useful for the widest range of people, attention should be paid to current standards for electronic mail transport. It should be possible to build multimedia mail systems on top of existing lower-level mail transport protocols and thereby use existing mail routing software for transport. Furthermore, the system should allow users to send plain ‘‘flat text’’ mail messages to users who do not have multimedia capabilities.

We present an experimental multimedia electronic mail system, called *Montage*, which has been developed at the Georgia Institute of Technology’s Software Engineering Research Center. We believe that this system provides a flexible and convenient means to send and receive complex multimedia documents. This system is built strictly on top of existing lower-level mail routing protocols and thus should work on most systems which support the Unix operating system, X Windows, and Simple Mail Transport Protocol (SMTP) mail transfer systems.

WHY MULTIMEDIA MAIL?

Traditional electronic mail systems have been limited to sending only textual data. While text-based store and forward communication systems are useful, the old adage that ‘‘a picture is worth a thousand words’’ is often true. For example, it is nearly impossible to concisely describe a complex architectural drawing by writing a textual description of it. Many applications involve the use of graphical information and it is extremely inconvenient to transport this information using existing electronic mail systems.

Furthermore, the mode of information delivery most familiar to humans is direct voice communication. The work of Rohr [Rohr 86] indicates that some concepts are inherently graphical while others are inherently verbal. Other research indicates that in many applications, concepts are understood better when presented in a ‘‘mixed media’’ mode [Guastello 89].

While computer-based electronic mail systems enjoy many benefits over paper mail, they continue to lag behind in many respects. With paper mail systems, users can post basically any type of document which can be printed on paper. This includes images and formatted text, all possibly with annotations marked on the document.

With a more flexible electronic mail system it should be possible to combine the benefits of both systems: the convenience and speed of existing computer systems with the flexibility of paper mail systems. Furthermore, it should be possible to use the dynamic aspects of computing technology to open up mail systems to media which have previously gone unused in a store and forward environment, such as audio and video.

DESIGN

Our design for the Montage system was influenced by several desired goals. Foremost, we wanted to create a mail system which implements our model of multimedia messages. Secondly, to be widely used, the system had to be built on top of existing protocols and standards where ever possible. Finally, to meet unforeseen needs, the system had to be extensible by end users.

Requirements

Our design was constrained by several factors. First, we wanted to distinguish between multimedia messages and more complex general multimedia ‘‘documents.’’ Montage is not a hypermedia system. Section 4, *A Model for Multimedia Messages* , ellaborates on this.

With this design decision, we were able to greatly simplify implementation. Montage messages proceed along one basic stream of control, unlike hypermedia messages which have a potentially large number of paths through the information.

Another design constraint was that the system should function cleanly over existing low-level transport mechanisms. We do not require that Montage must deliver its messages over TCP/IP connections. Since mail messages may be transferred between several intervening machines which may or may not run Montage, the messages should look for all intents like ordinary Simple Mail Transport Protocol (SMTP) [Crocker 82][Stallings 87] messages. Montage does not rely on any protocols higher than SMTP for transport.

Finally, we deemed that the system must be as extensible as possible. Since we cannot anticipate the needs of all users, we must provide a mechanism to dynamically add support for new media types, without the need to alter and recompile Montage.

Platform

We chose our platform based on availability and portability. While the Montage mail format may be implemented on a variety of platforms, including PCs and Macintoshes, our prototype implementation is for Unix workstations running the X Window

System. This platform ensures a reasonably high degree of portability for the system.

The interprocess communication abilities of Unix and the high level of flexibility in X also greatly simplify the implementation of some parts of Montage, as we shall see.

Specifically, our implementation of Montage was done on a Sun SPARCstation-1/GX machine running SunOS 4.0.3 and X11R4. The interface was built using Hewlett-Packard's widget set for X Windows. Audio support is native on the SPARCstation, video support is provided by a RasterOps video frame buffer board.

The hardware dependent parts of the Montage implementation (audio and video) are localized to simplify the porting process. It should be possible to run Montage on any Unix-based platform with X support. The audio and video functionality can be ported if necessary, but Montage will function perfectly well without audio and video support. In other words, Montage will function as a formatted text and image mailer if these facilities are not present.

Extensibility

It is impossible to know *a priori* all the things users may want to mail with a multimedia mail system. Almost by definition, multimedia mail systems should be able to send and receive a wide array of media.

Therefore, it was a primary goal that Montage be highly extensible by its end users. Montage allows users to modify the behavior of the program at runtime, without the need to recompile the system. Support may be added for new media types and new message transport techniques.

The facilities available for user extensibility will be addressed in following sections.

A MODEL FOR MULTIMEDIA MESSAGES

It is our belief that unlike more general purpose multi- or hypermedia documents, mail messages typically tend to be short and attempt to convey one central idea from the author. This belief has driven our design for the Montage message format.

Hypermedia documents tend to present the user with a large body of information in various media. The user then ''navigates'' through the document along a path that he or she chooses. Hypermedia systems generally tend to be quite interactive--the user is presented with almost all relevant information about a topic and then browses the document to find interesting information [Nielsen 90].

Contrast this model to a typical mail message in which the author is generally trying to convey some small number of central tenets. Unlike more general hypermedia documents, mail tends to be less interactive. The author has an idea to present and in some sense defines the reader's path through the message at composition time.

We believe that by limiting the generality of our message format we can achieve greater levels of usability, simplicity (for both the users and the implementors), and understandability of messages.

Still, however, since various media will be combined within a single message there must be some way to navigate through the packaged information. We have taken a simple approach to this task without having to resort to the less-constrained approach of a full-blown hypermedia system.

In Montage, all media are one of two classes: static or dynamic. The defining characteristic of a dynamic medium is that the information presented changes with time. Static media do not have this temporal component. Examples of dynamic media include video and audio clips. Examples of static media include simple text, formatted (rich) text, and still images.

Obviously it is of great interest to be able to combine various media freely within a single message. The differing characteristics of our two media classes make this packaging difficult. It is natural to think of a message window containing text and image data freely interspersed. Similarly it is natural to think of video and sound data being played at the same time. This mixing of media is common and is something users are used to experiencing on a day-by-day basis.

But does it make sense to combine static and dynamic media? Consider a message containing only text. When the message is opened a window appears that contains the text of the message. Now consider a message containing only voice. When the message is opened the voice clip is replayed. But what about a message containing both voice and text? Does one begin playing the voice as soon as the text window is opened? What about video and text? Does one open two windows, one for the text and one for the video? How does the user know which window to focus attention on? What if the moving video is placed within the text window? The video would become unviewable if the text were scrolled.

In both of these circumstances the mode of interaction would become one where the user's attention is focused on one media while the other is ignored. Some means must be provided to let users control the playback of the dynamic media so that they can decide when to focus attention on the various media. In addition, in our example above where video is presented in one window and text is presented in another, the user has no idea which contains the ‘‘central’’ part of the message. In other words, the user does not know which component of the message to focus on first.

Research supports the idea that users' interactions with multimedia systems tend to be largely ‘‘media-modal.’’ That is, they segregate the information presented to them based on the medium of interaction [Laurel 90].

Primary Media Classes

To overcome these difficulties, Montage introduces the concept of a *primary media class*. Each message in Montage has a primary media class. This class is either static or dynamic. The primary media class is the type of the media presenting the ‘‘main thrust’’ of the communication from author to reader. Various media with the same media class (either static or dynamic) as the primary media class may be combined freely. For example, if a message's primary media class is dynamic, the author may freely intersperse audio and video in a message.

All of the components of the primary media class which compose the principal part of the message are collectively called the *primary component*.

The primary component is presented to the user in a single window. The window contains mechanisms for controlling the presentation of the primary component contained therein (in the case of dynamic media, the controls may be buttons for play, pause, fast forward, and reverse; in the case of static media the controls may be scrollbars to view various parts of the message).

We use the concept of the primary component to project a single ‘‘path’’ through the message, much as in existing mail systems. The primary media and primary component concepts also serves to give the author a means for expressing the central ideas in a message.

If Montage were restricted to using only media with types the same as the primary class, the system would not be very flexible. There is an obvious need for the incorporation of any type of media within a message, regardless of the primary media class. As we shall see, the Montage model allows the use of any arbitrary media in the form of attachments.

Attachments

To augment the power of the system, Montage also makes use of *attachments*. Attachments may be placed on any message of a given primary media class. Attachments themselves are basically submessages and may be of any media type, regardless of primary media class.

Attachments may be considered to be ‘‘margin notes’’ that are not central to the message but still convey useful information. Since they are not presented to the user immediately when the window is opened they are, in some sense, secondary in importance to the information contained in the primary component.

Attachments give us a way to convey additional information while retaining our easy-to-play back, easy-to-understand main message thread. Without attachments, messages would degenerate to a conglomeration of mixed media within a single window which would in many cases be unmanageable.

In Montage, attachments are presented along side the primary component window in the form of icons. The image of the icon represents the type of media in the attachment. Furthermore, attachment icons are ‘‘connected’’ to a particular location in the primary component of the message. A use of attachments may be to provide annotation or supplementary information to the information contained in the primary component.

In the case of static primary media, attachments are connected to a certain physical point in the message. Thus, a voice attachment may be connected to a particular line number in a text message. In the case of dynamic media, attachments are connected to a certain time range in the primary message. Thus, as the dynamic message ‘‘plays back,’’ the various attachments are presented to the user during the time they are relevant. These attachments may be selected as they appear to give additional information provided by the author.

Consider two examples of the use of primary media and annotations to construct a message. A geographically distributed group may be collaborating on a document. The primary component may be the document in question. Attached to this at various points may be audio annotations requesting changes, image data to be considered for review (image data may also appear within the document itself), and video message clips to the various members of the group.

As another example, consider a researcher mailing a video of a presentation to a colleague. The primary media class here is dynamic, and the primary component of the message contains the video and associated audio of the talk. Attached to this at appropriate points are textual and audio annotations (the playback of the presentation may be stopped at any point to review the annotations), image data of the slides used in the presentation, text of papers, program source, and so forth.

Summary of Model

To summarize, the Montage model projects a single path of message traversal. This primary path is reflected in the primary component of the message. The primary component may contain mixed media, but the media it contains must be either all static or all dynamic to focus reader attention.

Additional information may be placed in a message in the form of attachments. An attachment is a piece of annotational information which is secondary to the primary flow through the message. Attachments are not restricted to being of the same class as the primary component.

IMPLEMENTATION NOTES

We have nearly finished a prototype implementation of Montage. This section covers the details of our implementation, including our assumptions and design decisions, the format Montage uses for message interchange, and the user interface to the system.

Overview

It was our desire that Montage be as flexible as possible in the types of media it can use. Therefore, we were determined to provide support even for media with very high bandwidth requirements, although the actual use of such media may be awkward on today's hardware because of pragmatic constraints.

Montage provides full support for interchange of text, still images, audio, and video. Local area networks commonly found today, such as Ethernet, have bandwidth in the 10 Mbps range. Even with Ethernet, applications will almost never see the full theoretical bandwidth available in the system.

The bandwidth provided by such a network is sufficient for text and small image interchange, workable for audio and large image interchange, and unwieldy for video interchange. Nevertheless, network speed and capacity will improve, so it is important to lay a software groundwork for applications which can make use of these faster networks.

Fortunately, the batch-processing style of mail makes full utilization of network resources less important. Whereas a real-time video conferencing system would require a guaranteed portion of the communication bandwidth to function, mail systems can be much less picky. As long as the message arrives in what the user perceives to be a ‘‘reasonable’’ amount of time the system is meeting its goals.

Montage relies on the underlying mail transport agents for actual mail delivery across the network. So as newer mail delivery agents that can efficiently transport large message across a network become available, Montage will be able to make use of these systems.

Interchange Format

While Montage mail headers conform to the SMTP standard, Montage makes use of a custom format for the bodies of electronic mail messages. This format was created because we felt that existing standards were either inflexible in the types of media they allow, or were too unconstrained to present media in an understandable format appropriate for mail messages.

The limitations of existing mail transport agents require that the contents of Montage mail message consist entirely of printable ASCII characters. Although there are some experimental binary mail transport agents, far and away the largest number of mailers available on Unix platforms only support ASCII transfer.

Because of this restriction, we are forced to encode the bodies of Montage messages into ASCII. Our reference implementation uses the standard Unix *uuencode* program to accomplish this. The ASCII-encoded file's size is expanded by 35% after this encoding process. To compensate for this, the message body is first compressed before it is mapped into ASCII.

But what is actually contained in this message body? In Montage, the message body actually consists of several discrete units, called *chapters*. When mail is received by Montage, the body is separated from the header, it is converted to binary from its ASCII format, uncompressed, and then broken down into its components. In our implementation, each of these chapters is stored as a separate file. There is one special chapter, called the *table of contents*, which contains information on the layout of the entire message. The table of contents references and connects all other chapters in the message.

Our first implementation uses existing Unix tools in an effort to rapidly produce an operational version of Montage. We use the *tar* Unix archive program to combine the various chapter files upon message creation, and to break out the chapters upon message receipt.

Montage adds one line to the SMTP mail message header which specifies the version number for the encoding scheme. Since the above method is only one of many possible encodings, future Montage mailers will be able to determine the encoding method by the version number in the header. Note that this is the only addition we need to make to the header to be able to send multimedia messages.

X.400

In 1984, CCITT released a set of standards for electronic mail systems. These standards do not deal with the user interfaces of mail systems, but rather they specify the services available for sending messages across the network.

The X.400 model defines two agents that make up an electronic mail system: the User Agent (UA) and the Message Transfer Agent (MTA). The UA provides the user interface for the system and may interact with other UAs. UAs hand messages off to MTAs for transport. X.400 specifies the interactions between UAs and MTAs, but does not specify the interactions between UAs and the users [Cunningham 84][Cunningham 85].

Montage implements a subset of X.400. Montage provides most of the header fields which are used in communication between UAs and MTAs, but does not provide the inter-UA communication facilities.

It should again be noted that Montage is largely independent of any underlying mail transport agent. The system can be configured to use an administrator-defined mail agent. On our prototype system this is the Unix *sendmail* program. While we have not tried using a full X.400 mail transport agent with Montage we believe that it should function properly.

Chapters

As stated before, a message body consists of one or more message chapters along with a table of contents. The table of contents specifies the relations between the various chapters (relative placement within a message, chapter format, etc.)

Each chapter contains a ‘‘section’’ of the message. Each chapter is represented in one medium. The type of this medium must be specified in the table of contents so that the recipient mailer will know how to ‘‘play back’’ the chapter.

A chapter is basically a single file containing a single message component. Chapters may contain either part of the primary component, an attachment, or the table of contents.

Note that the principle component of the message may be composed of many chapters, as long as the media in those chapters is of the same class as the primary media class (either static or dynamic). Thus, it is possible to have text interspersed with images in the primary component of the message.

Montage treats each chapter as raw data. That is, it associates no real semantic information with each chapter. Instead, the chapter data is handed to a playback module, or handler, which may be internal or external to Montage. The handler which is invoked on a particular chapter depends on the media type of the chapter. These handlers are specified by the users of the system, and Montage provides a mechanism for users to specify external programs which they can use to play back and record message chapters.

Media types are identified to Montage by *tags* which are associated with each chapter via the table of contents. Montage does not predefine any tags and indeed does not even associate any meaning with tags. Tags are defined by the users of the system. When a chapter is encountered the catalog of tags is searched and the user-specified handler (which may be an external program) is invoked on the chapter. The mappings from tags to handlers may be completely specified by users in a per-user database (with ‘‘sensible’’ defaults provided in a system-wide database).

Thus Montage is completely runtime extensible by the user in the media domain.

This ability provides Montage with a great deal of its flexibility. Users can choose their favorite tool for ‘‘recording’’ text (i.e., they can use their choice of editors (or word processors or spreadsheets) to enter text into the mail system). Similarly, they can use their choice of tools for viewing received audio and video chapters. Montage provides some simple means for playing and recording certain simple media. But because the system is not limited to those media which have built-in handlers, users can automatically add support for new media by specifying external programs to be used for media playback and recording. Additionally, this mechanism allows a great deal of user customizability and support for individual user preferences, by allowing users to use their choice of mechanisms for playback and recording of media.

Table of Contents

The table of contents (or *TOC*) is a special chapter that specifies the relations of the other chapters to one another. The format of the TOC is relatively simple, reflecting our simple model of mail usage.

We have chosen a line-oriented ASCII format for the table of contents. Since most TOCs will be relatively small, we felt that the space savings accomplished by encoding the information in a machine readable format would not be as important as the simplicity and ease of debugging provided by a human-readable format.

Each TOC must have at least 3 records. Each record gives some piece of information and each appears on a separate line. The required records are:

1. **`TOCVersion`** The version identifier for the TOC format.
2. **`Class`** The class of the primary message component, either static or dynamic.
3. **`Primary`** The name and type of the message chapter containing a part of the primary component of the message.

In addition to these required records, there are several optional records:

1. **`Attachment`** The name, tag, and position of an attachment. Position is given as a line number in the primary component if the message is static, or as a time offset in the primary component if the message is dynamic.
2. **`Author`** The mail address and (optional) name of the sender.
3. **`CreationDate`** The date the TOC was created.
4. **`Subject`** The subject of the message.
5. **`ID`** A unique identifier for the message on the machine it was created.
6. **`Comment`** Signifies that the rest of the line is to be treated as a comment (ignored). This is primarily used as a debugging tool.

Since it is possible to combine several chapters within the primary section of the message (as long as these media are either all static or all dynamic), there may be several `Primary` fields in the TOC. These chapters will be presented sequentially to the user.

Additionally, some media (for example, a proprietary format for a general purpose multimedia document) may mix static and dynamic media within a single window, even though this mixing is something that ‘‘native’’ Montage does not allow. In such a case, the media in question may reference files which it expects to contain the various media components it needs. These external files may be bundled together with the rest of a Montage message, and are referred to as ‘‘subchapters’’ since they are not used directly by Montage itself, but rather by one of the media that Montage is dealing with. The media which uses these external files must be responsible for managing them. Subchapters are not specified in the TOC since individual applications, rather than Montage itself, deal with them.

Here is a sample table of contents for a Montage message. This message consists of a simple text primary component, with three attachments, a μ-law sound clip (with tag ‘‘CODEC88’’), another simple text segment (with tag ‘‘SimpText’’), and an X Bitmap image (tag ‘‘XBM’’).

```
Author: Keith Edwards <keith@cc>
CreationDate: 18 Oct 90 10:18:52 PDT
Subject: Notes from the meeting
ID: 2848.AA08665
TOCVersion: 1.0
Class: static
Primary: txt.2848.txt SimpText
Attachment: snd.2848.snd CODEC88 27
Attachment: txt.2848.txt SimpText 30
Attachment: xbm.2848.xbm XBM 49
```

Interface

We built the interface for our implementation on top of the X Window System. We perceived two primary reasons for using X. Most importantly, X is accepted as a standard throughout the workstation environment we were targeting as our audience. A primary reason for this acceptance is the portability of X and applications developed for X. As a result of this portability, the interface portions of Montage should recompile cleanly on any workstation which supports X Windows.

Secondly, X provides certain general mechanisms for user customization at runtime. We were able to make use of these mechanisms to create a highly configurable mail system--not only is the interface of Montage configurable, but the actual types of media the system can handle can be changed at runtime. It would have been possible to get this degree of configurability without using X, but we were able to prototype the system much more rapidly by using the facilities already available to X applications.

Our front end is built using a set of user interface objects (*widgets* in the X parlance) provided by the Hewlett-Packard company. We choose the HP widget set based on its perceived completeness and orthogonality. A secondary consideration was that the HP widget set is freely available, unlike some widget libraries (such as Motif and OpenLook) which are available only to sites which pay a licensing fee.

Montage makes use of a multiple-window interface. The first thing a Montage user sees is the *Control Palette*. The Control Palette presents the user with a group of on-screen ‘‘buttons’’ representing the major options available in the system. The primary options available from the Control Palette are

- **New Mail** check in any newly received mail into the Montage system. New mail must be ‘‘checked in’’ before it can be read.
- **Info** presents the user with a ‘‘pop-up’’ window, giving information about the program’s origin.
- **Messages** presents the user with a scrollable list of messages. Users may then view, forward, save, or delete these messages.
- **Compose** is an interface to message creation. Facilities are provided for composition in several media types. Support for new media types may be added dynamically.
- **Iconify** reduces all the Montage windows to a single small window. This is convenient for when the user is not working with the system at the moment.
- **Preferences** brings up a panel which allows the user to customize his or her default preferences.
- **Quit** quits the system.

We will now go into some detail on the mechanics of the Montage interface.

Checking In New Mail Clicking the New Mail button causes any mail in the user’s mail to be ‘‘checked in’’ to Montage. Whenever new mail is checked in, the system reads in all new messages from the user’s mail spool file. As the messages are read in, the headers are parsed and a mail cache file is built in the user’s home directory. The cache file contains important header information such as the message sender and subject, and the number and format of any attachments. The system also calculates the byte offset of the start of the actual message body so that the message data can be separated from the header. After the cache file is updated, the individual messages are stored in a subdirectory of the user’s directory, on a one-message-per-file basis. The system spool file is then removed to free up system disk resources.

Users are notified when new mail arrives by the playback of a sound file. Users may also enable an automatic mail check-in function via the preferences panel (see **Preferences**, below).

Basic Message Control The most commonly used Control Panel items are **Messages** and **Compose**. These two items give the interface to message creation, viewing, deletion, and most other common functions.

When the user clicks the **Messages** button to view the list of current messages, the system scans the mail cache file to retrieve the header information. This keeps Montage from having to open, scan, and parse all the message files individually. A new window is opened and the user is presented with a scrollable list of the current messages.

From the *Messages* window (see Figure 1) users have the option to view messages, save messages to a file, delete messages, or reply to or forward messages. All actions on messages are accomplished by highlighting the desired message or messages, and then clicking the button to perform some action on those messages. Since viewing is perhaps the most common operation, users may view messages by double clicking on the message.

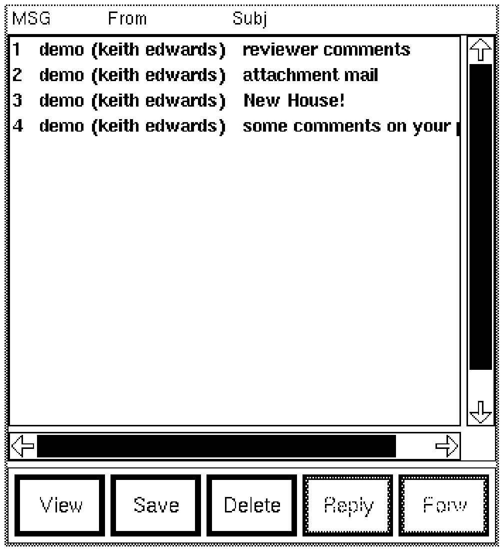

```
MSG   From                   Subj
1     demo (keith edwards)   reviewer comments
2     demo (keith edwards)   attachment mail
3     demo (keith edwards)   New House!
4     demo (keith edwards)   some comments on your
```

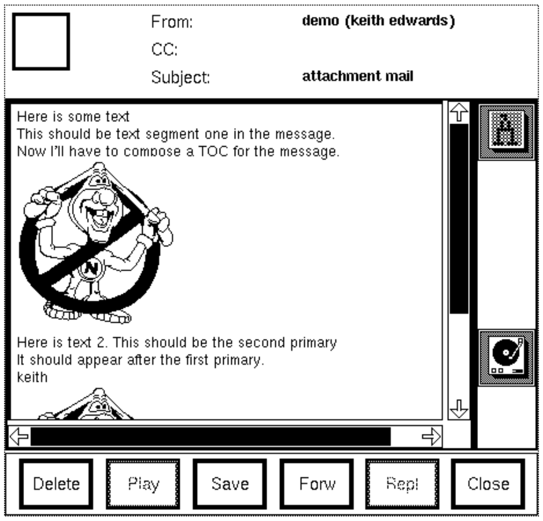
Figure 1: The Message Window

Saving a message brings up a window which prompts for the filename in which to save the message. Currently, messages are saved in their native, bundled format. We will be adding support for saving individual message components.

Replying to a message puts the user into the *Compose* window (to be discussed shortly). When replying, the user is not restricted to using the same primary media class as the original message.

Forwarding simply allows the message to be resent to a new destination.

Message Viewing When a message is viewed, the *View* window (Figure 2) appears on the user's display. The message is automatically unpacked and the primary component of the message is presented in the main portion of the window. Any attachments are displayed along the side of the window and are represented by icons which depict the types of the attachments. There is a single scrollbar attached to the primary window and attachment list.



From: demo (keith edwards)
CC:
Subject: attachment mail

Here is some text
This should be text segment one in the message.
Now I'll have to compose a TOC for the message.

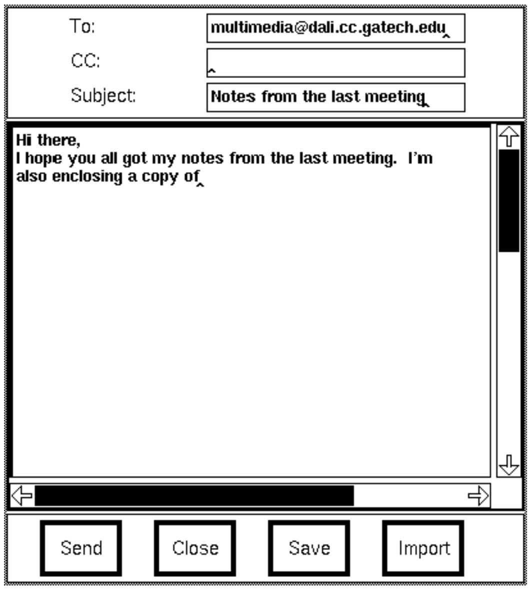
Here is text 2. This should be the second primary
It should appear after the first primary.
keith

Figure 2: The View Window

What is actually displayed in the primary component window depends on the media themselves. Typically, static media will be presented in a scrolling window. The handling of dynamic media is more complex, however. Users have control over sound via a control panel much resembling a tapedeck. Video messages are controlled by via a window with a set of controls much like a video cassette recorder. As the dynamic media are played, any attachments scroll by the user to the side of the control panel. If desired, users may stop playback and view attachments relevant to the portion of the primary component they are viewing.

The *View* window also gives users options to delete, save, forward, and reply to messages. These functions work the same as those presented in the *Messages* window.

Composing New Messages Clicking the **Compose** button brings up the *Compose* window, which is an interface to message creation (see Figure 3). Users fill in the necessary components of the message header (recipient, subject line, and any carbon copies) by typing in the appropriate fields.



To: multimedia@dali.cc.gatech.edu
CC:
Subject: Notes from the last meeting

Hi there,
I hope you all got my notes from the last meeting. I'm also enclosing a copy of

Figure 3: The Compose Window

The main portion of the *Compose* window is taken up by a region where the primary component is created. Users may type into this window directly, or they may import files of various types via the **Import** button.

The **Import** button brings up a window which allows users to insert external files as either attachments or as parts of the primary component. The user must specify the type of the inclusion so that Montage may build the table of contents for the message correctly.

Because of the ability to type directly into the *Compose* window's primary component area, the input of text is simple. However, it is desirable to be able to record other media as easily as text. It is often cumbersome to have to use tools external to Montage to record some message component into a file and then import the file into a message. Thus the *View* window provides a menu for **Compose Tools**. These tools allow users to incorporate media into messages almost as easily as text.

For example, Montage supports a sound recording tool from the **Compose Tools** menu. This tool (resembling a tapedeck, see Figure 4) allows users to record and edit sound clips. A similar video tool allows the recording and editing of video messages. Users may configure Montage to add new items and actions to the tools menu.

Users click the **Send** button to pass the message on to the mail transport agent. The *View* window contains buttons to close the window, and save the message to a file.

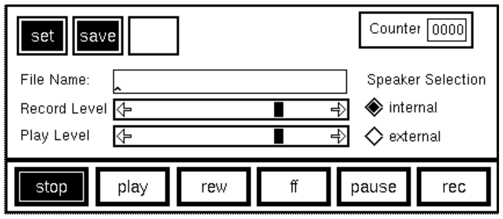


Figure 4: A Sound Tool

Hiding Montage Montage uses a multiple-window interface. Each window in the system may be shrunk or ‘‘iconified’’ individually.

It may be desirable at times to iconify all the windows of the application at once to effectively hide the application. This may be accomplished by the **Iconify** button on the Control Palette. This button reduces all the Montage windows to a single small window. Any windows which have been iconified individually are ‘‘joined’’ into the Montage icon. All windows will be returned to their original states when the icon is clicked.

User Preferences Montage supports user-configurable preferences via the **Preferences** button. This button brings up a control panel which allows users to select and modify a variety of system parameters, including changing the speakers to which audio output will be routed, setting audio play and record levels, setting whether or not message headers will be displayed, and so on.

PERFORMANCE

Overall mail system performance depends on many variables: the underlying network used for mail transport, the size of the messages, and even the habits of the users (some users may want to send only sound clips while others may be content to send simple text most of the time).

Designers of mail systems must be concerned with two aspects of performance: mail delivery latency and message size. Mail delivery latency is the actual time required for transmission of a typical message across the network. Message size is the amount of physical storage required to actually store the mail message.

The delivery latency is dependent on the speed of the underlying network which is in turn dependent on the mail delivery agent chosen for message transport. Since Montage is usable with any RFC-822-compliant mail transport agent, we are not directly concerned with network latency and performance may be characterized by one metric: message size. In most cases message size will be proportional to delivery latency for a given network and transport agent, so message size is an effective measure of performance.

Sending a message via Montage will add somewhat to the ‘‘native’’ size of the information being transmitted. Essentially the native size of a Montage message is the sum of the sizes of the attachments, along with the primary message component and the message header. When the message is sent, a table of contents is added. Furthermore, all of the message body undergoes a packaging process which adds some to the size of the message.

Our goal is to minimize the increase in size by (1) optimizing the packaging process for size, and (2) making the table of contents as small as possible. We will address each of these issues in turn.

The packaging process basically consists of ‘‘bundling’’ the various message components into one file. To minimize the size of the resulting file, it is compressed using adaptive Lempel-Ziv coding [Welch 84] as implemented by the Unix *compress* program). The amount of compression obtained by using this algorithm depends on the size of the input and the distribution of common substrings. Typically, for English text, the compression will be in the range of 50% to 60%. Some files, such as μ-law sound files, are already encoded and cannot be compressed further.

Since the result of this compression is a binary file, it must be mapped into ASCII characters for transmission via RFC-822 mailers. This mapping process expands the file’s size by 35% (3 bytes become 4, plus some control information is added). Thus, for fairly long text messages, Montage can achieve message sizes smaller than those for messages containing the same, but unencoded text (original message size $\times .55 \times 1.35 = .74$ of the original message size).

Of course added to any encoded message is a table of contents. This table of contents specifies the ‘‘layout’’ of the message. Our format for the table of contents is ASCII-based. We deliberately chose this format for simplicity in implementation and to ease debugging. Usually the table of contents for a given message will be very small.

The minimum size for a table of contents is variable. Since the table of contents contains entries for things such as sender name and subject there is no fixed minimum size. But in general, a simple table of contents will be on the order of 300 bytes. Each attachment will add something on the order of 60 bytes for information regarding attachment type and placement. So we see that the contribution of the table of contents to the size of the total message is negligible.

STATUS

The design and a first-pass implementation of Montage have been completed. Currently the system allows static primary media, and attachments of either text, audio, or several image formats. Facilities have been completed to allow easy composition of messages by typing or importing text, importing images, and recording sound via a simple sound editor mechanism.

Work is progressing on the addition of full dynamic primary media capabilities. We hope to soon have store-and-forward video transmission along with synchronized sound output.

Montage is currently running as a complete functional mail system which can serve as a replacement for existing Unix mail systems. That is, Montage may already be used to replace users’ text mail composition and reading programs; the remaining work lies in the area of support for additional media.

CONCLUSIONS, CAVEATS, FUTURE DIRECTIONS

Our work on Montage was based on a set of assumptions which we believe greatly enhanced the usability of the system, as well as simplified the implementation. The initial response within the local research community at the Software Engineering Research Center and outside sponsors has been favorable.

In the near future, we plan to polish the implementation of the system to the point where we have a robust, workable mail system which may then be integrated into an actual work environment. Further work will be done on enhancing the video capabilities of the system and integrating the mailer with existing tools and media.

Much work still needs to be done on increasing the usability of the user interface. An online, context-sensitive help system is also needed.

We feel that Montage represents a flexible, workable system for the exchange of multimedia documents while presenting a friendly, intuitive interface to users.

ACKNOWLEDGEMENTS

The research and development of this system were supported by a grant from BellSouth Corporation.

REFERENCES

Crocker, David H. [1982] Standard for the Format of ARPA Internet Text Messages, Internet Request For Comment (RFC) 822, August 13, 1982.

Cunningham, I. [1984] Electronic Mail Standards to Get Rubber-Stamped and Go Worldwide. *Data Communications*, May 1984.

Cunningham, I., and I. Kerr. [1985] New Electronic Mail Standards. *Telecommunications*, July 1985.

Guastello, S., M. Traut, and G. Korienek. [1989] Verbal Versus Pictorial Representations of Objects in a Human-Computer Interface. In *International Journal of Man-Machine Studies*, July 1989, Vol. 31, No. 1, pp. 99-120.

Laurel, Brenda, Tim Oren, and Abbe Don. [1990] Issues in Multimedia Interface Design: Media Integration and Interface Agents. In *ACM SIGCHI Proceedings, 1990*. (Seattle, Washington April 1-5). pp. 133-139.

Nielsen, Jakob [1990] *Hypertext and Hypermedia*. Academic Press Inc., San Diego, CA, 1990.

Rohr, G. [1986] Using Visual Concepts. *Visual Languages*, S. Chang, T. Ichikawa, and P. Ligomenides, eds., Plenum Press, New York, 1986, pp. 325-348.

Stallings, W. [1987] *Handbook of Computer-Communications Standards, Volume 3: Department of Defense (DoD) Protocol Standards*. New York: Macmillan, 1987.

Welch, Terry A. [1984] A Technique for High Performance Data Compression. *IEEE Computer*, vol. 17, no. 6 (June 1984), pp. 8-19.