A Multi-media Message System for Andrew

Nathaniel Borenstein
Craig Everhart
Jonathan Rosenberg
Adam Stoller

Information Technology Center
Carnegie-Mellon University

Abstract

The Andrew Message System (AMS) is a portable, distributed, multi-media, multi-interface system for reading and sending mail and bulletin board (bboard) messages. Mail and bulletin board processing was selected as a ‘‘showcase’’ application to demonstrate how the Andrew file system and user interface toolkit could be applied most usefully to a user’s daily tasks.

The AMS supports multi-media messages, which may include line drawings, hierarchical drawings, spreadsheets, raster images, animations, and equations. It is explicitly designed to support a huge database of messages and an enormous user community. At CMU, it services over 1200 bboards, including netnews, the Dow Jones information service broadtape, and bboards on which newspaper cartoons appear as raster images. The system incorporates a B-tree based ‘‘white pages’’ for doing name lookups, including phonetic matching of misspelled names. In addition, the system supports a number of advanced features such as voting on multiple-choice questions, private bboards, shared mailboxes, and automatic classification of incoming mail messages. The server-based architecture makes it easy for client interfaces to be ported to or built on almost any computer. Currently, interfaces run on IBM RTs, DEC MicroVaxes, Suns, IBM PC’s, Macintoshes, and Vax UNIX and VMS timesharing systems.

Introduction

In the Andrew project [1], the primary focus of the development effort has been on developing two major tools to support an advanced computing environment. Those tools, the Vice distributed file system and the Andrew Toolkit (a.k.a. BE2) for user interface development, are described in detail elsewhere [2,3]. In addition to developing these primary tools, the Andrew project has created the Andrew Message System (AMS), an application suite that uses the Andrew tools to facilitate electronic communication by users of the system. As a large application developed independently of (but in cooperation with) the BE2 and Vice projects, the AMS has been both a useful application in its own right and a driving force in the evolution and improvement of Vice and BE2.

The goals of this paper are twofold: we will describe the key features and architecture of the Andrew Message System, and we will explain how the underlying Andrew facilities simplified the development of such a system and made it more powerful and useful than would otherwise have been possible.

The Goal: A Truly Integrated Message System

The primary goal of the Andrew Message System project was to develop an *integrated* electronic mail and bulletin board system that took advantage of the primary Andrew facilities. The word "integrated" has at least four meanings in this context:

Integration of functionality. The system should integrate many advanced features already present in other systems into a reliable and efficient unified system.

Integration of media. It should be possible to send a collage of anything that can be digitized through the mail, in a seamless and painless manner.

Integration of disparate hardware. The system should be useful not only on advanced workstations, but also on IBM PC's, Macintoshes, and via terminal/dialup access to time-sharing systems based on several operating systems.

Integration of database access. The system should provide an integrated database access mechanism, with as much as possible of the mechanism encapsulated in interface-independent libraries, so that the construction of new user interfaces to the system will be relatively quick and painless.

Design Architecture

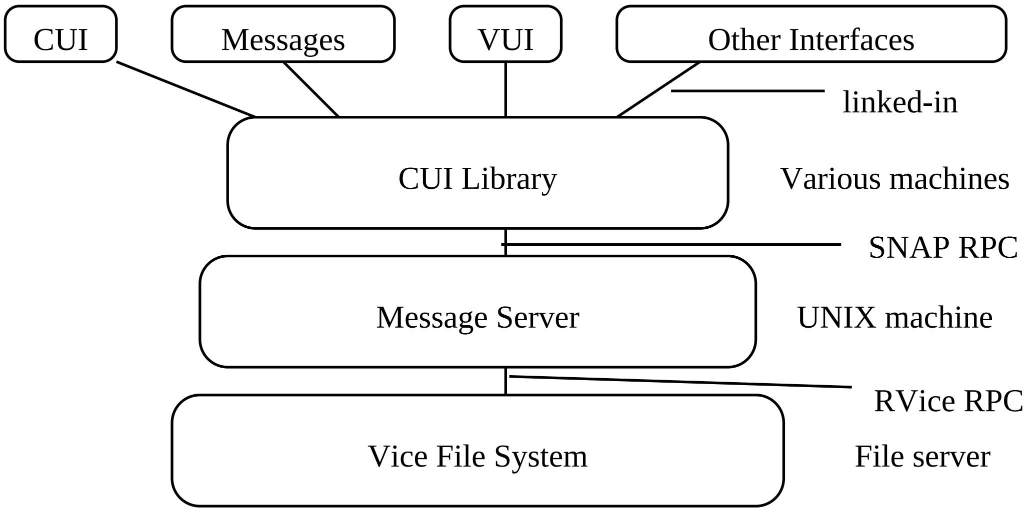
The integration goals described above led to a system architecture in which functionality is divided into three levels:

Actual access to the message database occurs exclusively at the *message server* level. The message server is a process that runs on a machine with full access to the Vice file system. Multiple client programs may simultaneously access data from the same message server, via a remote procedure call mechanism known as SNAP.

Various abstractions relevant to client interfaces, as well as the interfaces that allow communication with the message server, are implemented at the level of the *client library*, generally known as the CUI (common user interface) library. This library is written in portable C, and runs in environments as diverse as UNIX, VMS, MS-DOS, and Macintosh.

Finally, interaction with the user occurs at the *application level*, which is, of course, different for each user interface. Some of the user interfaces are highly portable across environments, while others are not. For example, the most basic user interface, CUI (Common User Interface) will run in every environment where the CUI library runs, and requires no more sophisticated a display than a teletype. A more popular user interface, VUI (Visual User Interface) runs on PC's, and on UNIX and VMS systems with CRT displays. Other more specialized interfaces include Messages (formerly known as AUI, the Andrew User Interface), which runs on a UNIX workstation running X11 or the old Andrew window manger, and MacMessages, which runs only on a Macintosh.

The levels of the system are pictured below:



Of course, this description leaves out many important features. Most notably, it leaves out a complex message delivery system, developed as an alternative to the UCB sendmail program for reliable mail delivery in the Vice environment. Although the delivery system component of the Andrew Message System will be mentioned briefly below, a full description of it is beyond the scope of this paper. [5]

Features

In this section, we outline some of the key features of the Andrew Message System.

Multi-media Messages

The most noticable feature of the Andrew Message System is its support for a rich set of multi-media objects. In particular, the AMS allows users to send, in addition to normal text messages, formatted text, raster images, line drawings, hierarchical drawings, interactive tables, equations, and even animations. The multi-media objects are managed by the Andrew BE2 toolkit, as described elsewhere [2].

Of course, the multiple machine architecture makes multi-media mail particularly challenging. In the AMS, only one user interface, Messages, is currently able to display the full complement of multi-media messages. When other interfaces -- notably those that run on lower-functionality hardware -- display multi-media messages, they ask the message server to give them an "unformatted" version of those messages. Thus, where a Messages user might see something like this:

Date: Wed, 25 Nov 87 10:52:30 -0500 (EST)

From: Nathaniel Borenstein <nsb+@andrew.cmu.edu>
Subject: Animated logo!

Here’s the CMU logo:

Carnegie
Mellon

But the *best* part is that if you click on it and choose the **Animate** menu, it will turn into a cube and tumble around on your screen!

Isn’t that neat?

A user of a lower-functionality interface would instead see something like this:

Date: Wed, 25 Nov 87 10:52:30 -0500 (EST)
From: Nathaniel Borenstein <nsb+@andrew.cmu.edu>
Subject: Animated logo!

Here’s the CMU logo:

[An Andrew/BE2 view (an animated drawing) was include here,
but could not be displayed.]

But the best part is that if you click on it and choose the Animate menu, it will turn into a cube and tumble around on your screen!

Isn’t that neat?

Note that all of the text is still presented correctly, and the full multi-media message persists in the database, so that the frustrated user may seek out an Andrew workstation if he wishes to view the message in its full multi-media splendor.

Multiple Machine Types & Interfaces

As previously indicated, one of the major design goals of the AMS was to create an integrated electronic communication environment unifying a wide variety of computer hardware. That this goal has largely been achieved is indicated by a list of the available AMS interfaces available and the hardware/software environments in which they run (current as of winter, 1987/88):

> **Messages** (*IBM RT, Sun, MicroVax*) -- The flagship interface, supporting full multi-media messages.
>
> **CUI** (*IBM RT, Sun, MicroVax, IBM PC, Macintosh, VAX UNIX or VMS timesharing system*) -- the simplest, teletype-oriented interface.
>
> **VUI** (*IBM RT, Sun, MicroVax, IBM PC, VAX UNIX or VMS timesharing system*) -- a screen-oriented interface with a interface style like that of many IBM PC applications
>
> **MacMessages** (*Macintosh*) -- a native Macintosh interface.
>
> **Batmail** (*UNIX/Emacs*) -- a user-supported Emacs interface.

Integration of Message Types

Another goal of the AMS was the integration of various kinds of electronic communication in a common system. The synergistic effects of merging mail and bulletin boards in a common interface are especially beneficial. For example, if a user sees an interesting message on a bulletin board and wants to keep a copy, he may simply put it into one of his mail folders, the same way that he would manipulate his own mail.

In addition, the need to explicitly address multiple protection domains within the message database -- to allow for public bboards and private mail in the same system -- has enabled the creation of various "hybrid" forms of message databases. For example, there are private bulletin boards shared by groups of users and official bulletin boards to which only certain users may post messages but which everyone may read. Users may even give each other access to their mail folders so that, for example, a secretary may process his employer’s mail as if it is a bulletin board he administers.

Rewritten Delivery System

Originally, we had hoped to build the AMS without becoming entangled with issues of message delivery. In particular, we had hoped to be able to simply use the standard Berkeley sendmail program as our delivery engine, and concentrate our efforts on other things. Although the AMS will, in fact, run compatibly with a standard sendmail-based delivery system, that system is simply not reliable in the Vice environment, nor would it be likely to be reliable in any similar environment.

Because *sendmail* was not written with the notion that the file system could temporarily disappear (for example, during network or file server problems), it is very hard to make it behave consistently and reliably when such situations occur. Additionally, sendmail takes as a given the notion that users live on particular machines, whereas, in the Vice environment, it is basically irrelevant what machine the recipient uses, as long that machine is connected to Vice. Sendmail also depends on some features of standard

UNIX not available in the authenticated Vice environment, particularly those related to file protection and special user privileges. Finally, the Vice file system provides a much greater level of authentication than standard UNIX, and sendmail was not well set up to take proper advantage of that authentication information.

For all of the above reasons, an entirely new message delivery system was implemented as part of the AMS. That delivery system will be described in detail in a forthcoming paper; a full description is beyond the scope of this paper.

The delivery system is technically an optional component, in that the AMS can be (and often is) configured to run with the standard delivery system on non-Vice machines. However, several features of the AMS work only in the presence of the AMS delivery system. In particular, the AMS delivery system allows us to make very strong guarantees about message authentication, whereas it is notoriously easy to forge messages on a standard UNIX system. The AMS delivery system also provides a great deal of location-independence; users need not send their mail to a particular machine, but merely to an address that corresponds to an entire Vice installation. Finally, the AMS delivery system provides sophisticated user name lookup features via a mechanism known as the *white pages.*

White Pages

The white pages database is a package that facilitates the mapping of human names to UNIX user names and other mail-related information. It is optimized for efficient operation in a distributed environment by being packaged in a collection of Vice files are organized according to a B-tree discipline. [6] The white pages facility, which features heuristics for phonetic matching of user names, is used primarily for evaluating and rewriting what users type on the "To:" and "CC:" lines of outgoing messages, and for evaluating the addresses on incoming messages from remote sites. The operation of the white pages is most simply demonstrated, however, by a sample dialog with the CUI program, using the "whois" command to probe the white pages database directly. (The user input is in **bold** font.)

```
CUI> whois bond
Verifying name list 'bond'...
The name 'bond' is ambiguous.
What did you mean by 'bond'?
    1 - Eric Bond (eng) <bond+@andrew.cmu.edu>
    2 - Sandra Bond (itc) <sandra+@andrew.cmu.edu>
    3 - None of the Above
Choose one [3 - None of the Above]: 2

Sandra Bond <sandra+@andrew.cmu.edu>
CUI> whois not bronstin
```

```
Verifying name list 'not bronstin'...
Name match was uncertain; please confirm:
    1 - Nathaniel Borenstein <nsb+@andrew.cmu.edu>
    2 - None of the above
Choose one [2 - None of the above]: 1

Nathaniel Borenstein <nsb+@andrew.cmu.edu>
CUI> whois AMS
Verifying name list 'AMS'...
The name 'AMS' is ambiguous.
What did you mean by 'AMS'?
    1 - Anthony Iams (cdec) <ai02+@andrew.cmu.edu>
    2 - Wendi Anne Amos (csw) <wa0a+@andrew.cmu.edu>
    3 - None of the Above
Choose one [3 - None of the Above]:

AMS (Address Validation Error: ambiguous name)
CUI>
```

Because the white pages database is stored in Vice, it is accessed by the message server, which then exports this function to the client interfaces, wherever they may be. Thus the above dialog could take place as easily on a PC as on a Vice-based workstation. The White pages is also used by the Andrew *finger* service.

Miscellaneous Features

In addition to the major areas described above, there are several smaller but still interesting features implemented in the Andrew Message System, some of which will be briefly summarized here.

Magazines. In order to help cope with the flood of information produced by large bulletin board systems, the AMS permits the creation of magazines. Magazines are bulletin boards that volunteers from the user community maintain on topics of their own choosing. For example, the editor of a magazine on "music" might read twenty music-related bboards at a site like Carnegie-Mellon. He can then easily cross-post the messages he considers worth reading on his magazine. Others, lacking the time or inclination to read all twenty of those bboards, can instead read only the magazine, thus seeing only the "cream of the crop".

Automatic Classification of New Messages. When the message server processes new incoming messages from a user's mailbox, by default it simply puts them into a folder called "mail". However, the user may supply a program which the message server will run on each piece of incoming mail to decide where that message should be classified. Currently, the program must be written in a somewhat cumbersome stack-oriented language, although this language is slated for replacement in a future release (see "future plans", below). Although clumsy, the language is powerful enough to permit, for example, hundreds of Internet mailing lists to be sent to the same mail address and then be automatically sorted onto appropriate bulletin boards on the basis of information in the message header.

Voting. AMS users can compose special messages that call for "votes". Anyone reading such messages is automatically prompted (by the CUI library) to vote on the issue in question. Votes are automatically mailed back to an address specified by the vote composer, and there are also a few features to make vote tabulation straightforward. Voting is *not* by secret ballot -- indeed, with the AMS delivery system, the votes are authenticated, so that multiple voting is easily detected. All votes are multiple-choice, although the vote composer may specify whether or not write-in votes are permitted.

Return-receipt requests. On sending a message, an AMS user can designate it as requesting an acknowledgment, or return-receipt. When an AMS user receives such a message, he is automatically asked whether he wishes to send such an acknowledgment. This is particularly useful for users who are separated by unreliable network connections.

Enclosures. Often, users at distantly-connected sites need to share files or other data, and the only means at their disposal is electronic mail. In such cases, the "enclosures" feature of the AMS can be useful. With this feature, a user can specify that all or part of his message is an "enclosure" to be handled specially. When a user receives such a message, he is automatically asked whether he would like to do something special with it, such as write it to a file or (on UNIX) pipe it through a process. Of course, what is written to the file or pipe is the enclosure only, without the message headers, which can save a fair amount of time in processing such messages.

Folder creation announcements. Because the AMS makes it very easy for users to create new bulletin boards, the process of subscribing to new bboards must be very simple. One mechanism that supports this is folder creation announcements. When a user reads a message that is a folder creation announcement, he is automatically asked whether or not he wishes to subscribe to the newly created folder, or bulletin board. Such announcements are typically created automatically when the system creates a new bboard, but may also be created by users themselves.

Assessment of the Andrew Environment

The AMS represents a great deal of development effort by the AMS developers themselves, independent of the work of the Vice and BE2 development efforts. However, the AMS would not be nearly the system it is without the work of those other groups. In this section, we will describe how the AMS benefits from the more general Andrew facilities, and what price had to be paid for those benefits.

The Vice File System

The AMS as we know it would not be possible without the Vice file system. The AMS was designed specifically to handle enormous message databases (i.e. gigabytes of message text), which would simply be impractical to store on individual workstations. In addition to the support for the scale of the system, Vice also gave us much stronger authentication and richer file protection than does the normal UNIX file system. As a result, in the Vice environment we can make a relatively strong guarantee to our users that mail from other local users is not forged, and we can easily permit the creation of private bulletin boards and the sharing of mailboxes.

The most important benefit of Vice for our purposes, however, was the simplification of the database. Rather than having to replicate the database on multiple machines, we can simply store it in one place -- Vice -- and let all the interested message server processes read the data from that place. This was a tremendous simplification over any scheme for sharing bulletin boards among multiple machines that do *not* share a common file system.

On the other hand, Vice poses a few notable problems for developers, when contrasted with normal UNIX. Most notably, Vice does not support setuid programs, hard links between files not in the same directory, or per-file protection information (Vice protection is on a per-directory basis). The absence of these features placed additional constraints on the design of some of our algorithms.

Perhaps most important, the use of Vice *or any other distributed file system* is likely to seriously diminish the reliability of applications that were written for a non-distributed file system. In particular, most pre-existing UNIX utilities have no idea that they should check the return value of the close() system call, nor of what they should do if such calls fail. (The failure of a close() call is almost inconceivable on the standard UNIX file system, but is a common point of failure on Vice. Such failures may correspond to file server failures, or to other problems, notably related to file protection and storage quotas.) This kind of incompatibility can necessitate massive rewrites of pre-existing software; in our case, it was a major factor in our decision to rewrite virtually the entire mail delivery system.

The Andrew BE2 Toolkit

The BE2 Toolkit is the heart of the multi-media features of the AMS. Without it, the AMS simply would not have multi-media mail, or would have had to implement the interpretation of a multi-media datastream itself. Thus, clearly, the benefit to the AMS of using BE2 is enormous.

Currently, we know of no user-level drawbacks to the use of BE2 in the AMS. Although the development process was made more painful by the fast-paced evolution of BE2 and the frequent necessity to devote significant time to converting to the latest version of BE2, those problems are basically irrelevant to the AMS as the users see it.

As a programmer's interface, BE2 was in general extremely useful. BE2 is well enough designed to permit the programmer access at whatever level of abstraction he finds necessary. That is, it rarely suffers the deficiency of "protecting" the programmer from the feature he needs. Additionally, it only rarely forces the programmer to use a lower level of abstraction than he desires.

The only major negative factor in BE2 programming is the initial complexity faced by the new programmer. BE2 is a very clever and complex system, with a relatively high "entry cost". It is hard to imagine an uninitiated programmer understanding the most trivial "hello world" BE2 program in less than half an hour. However, this situation can be expected to improve somewhat as better documentation and tutorials become available.

Status Report

Statistics

The Andrew Message System has been in use by the Carnegie-Mellon campus for about a year and a half, and for a lesser time at other sites. In order to convey the size and scope of the CMU installation, we will mention a few statistics about the AMS at CMU as of the fall of 1987.

Currently at CMU there are approximately 1350 bulletin boards on the system; most of these are publicly readable, but about a hundred or so are semi-public. (Semi-public bboards are bboards which are publicly *visible*, but which are only readable by designated users. No statistics are available on the number of completely private bboards, as users may create these completely independently and secretly.)

The 1350 bboards include all of UNIX netnews, all of the Dow Jones information service, and hundreds of Internet mailing lists, in addition to hundreds of CMU-only bboards. Among the CMU-only bboards are a suite of bboards used by the consulting staff to answer users' questions and problems; these bboards are visible to the developers as private bboards, so that the developers can easily intervene with assistance for the trickier questions.

Taken together, annual postings on all of these bboards amounts to at least ten gigabytes, with a new message showing up, on average, about every 40 seconds. Helping users cope with this flood of information are sixteen user-edited "magazine" bboards.

Currently there are about 1800 regular users of the system at CMU. The "typical" user is hard to describe. A large number read only a few "official" bboards, but another large group reads dozens or even hundreds of bboards. One productive staff member manages to get work done despite reading 377 bboards regularly.

Future Plans

The future of the Andrew Message System holds many new and exciting developments in store. Work is progressing or planned in the following areas:

> **FLAMES**: Work is progressing on FLAMES, the Filtering Language for the Andrew Message System. This will be a powerful and general LISP-like language for operating on the message database. The two primary initial applications will be database queries and automatic classification of incoming messages. In the future we will also be concentrating on improved customization facilities, intelligent selection of messages (as in the Information Lens system [4]), and automatic network-based information services.
>
> **Name resolution**: Our White Pages technology has proven so successful that we are considering ways to expand it to include names from non-Andrew and non-local sources, leading eventually toward the goal of a "national white pages".
>
> **Expansion of Client Base**: Efforts are underway to increase the functionality available to AMS users on systems such as the Macintosh and VMS.
>
> **Event calendars**: Ordinary bboards are inadequate for announcing specific events for future dates; special features are planned to help make event planning simpler and more natural.
>
> **Two-dimensional bulletin boards**: Eventually, we hope to make our bulletin boards optionally visible in a two-dimensional manner, where responses are clearly associated with the messages to which they respond.
>
> **Conferencing systems**: It is our hope to eventually bridge the gap between bboard and conferencing systems by providing a way to conduct an interactive conference on a bulletin board.
>
> **Improved Customization facilities**: Although the system is already highly customizable, there is still much room for improvement, particularly in the marking and annotation of messages in the database, and in the user interface to the customization facilities.

Acknowledgments

In addition to the authors of this paper, dozens of other people have contributed to the development of the Andrew Message System. Although space does not permit us to mention them all here, special credit should be given to Brian Arnold, Mark Chance, Bob

Glickstein, David Kovar, Sue Pawlowski, Larry Raper, Mike Sclafani, and Aaron Wohl for their work on the AMS. We also wish to acknowledge our indebtedness to the 'advisor' staff (our most demanding and helpful users), the Andrew documentation group, the Vice group, the BE2 group, the ITC management, the networking group, and the support staff, all of whom helped make the AMS a reality.

References

[1] Morris, et. al, Andrew: A Distributed Personal Computing Environment, Communications of the ACM, Volume 29, Number 3, March, 1986, pp 184 - 201.

[2] BE2 reference **** NEED TO FILL IN COMPANION PAPER INFO ****

[3] Vice reference **** NEED TO FILL IN COMPANION PAPER INFO ****

[4] Malone, et. al, *Intelligent Information-Sharing Systems*, Communications of the ACM, Volume 30, Number 5, May, 1987, pp 390 - 402.

[5] Rosenberg, et. al, An Overview of the Andrew Message System, Proceedings of SIGCOMM ´87 Workshop, Frontiers in Computer Communications Technology, Stowe, Vermont, August, 1987.,

[6] Lehman, Philip L., and S. Bing Yao, Efficient Locking for Concurrent Operations on B-Trees, ACM Transactions on Database Systems, Volume 6, Number 4, December, 1981, pp 650-670.