Continuous Media E-mail on the Internet: Infrastructure Inadequacies and a Sender-Side Solution

David A. Turner and Keith W. Ross
Institut Eurécom
Sophia Antipolis, France
{turner,ross}@eurecom.fr

May 2, 2000

Abstract

This paper examines how the Internet's e-mail infrastructure should evolve to better support continuous media (CM) e-mail, such as audio and video. We assert that the problem is not simply a matter of adding the obvious new functionality to user agents (mail readers), such as audio and video capture, but requires the adoption of a new delivery model. We do this by examining the problems that arise using current methods to deliver video messages, and we show how sender-side storage and media streaming solve these problems. Finally, we describe an implementation strategy that requires changes only to individual sender systems, but enables CM e-mail to be delivered universally to any recipient, thus providing a realistic evolutionary path that can be adopted incrementally by individual mail systems.

1 Introduction

Internet e-mail, introduced in the 1970s, was the Internet's first killer application. It is the original application that provoked user interest in the Internet, and it has now become a common fixture of modern society, both in the workplace and in the home. Up through the early 1990s, almost all Internet e-mail was in simple ASCII. But in recent years, e-mail user agents (mail readers) have become MIME compatible, and are therefore capable to a limited degree of creating and rendering e-mail messages with multimedia content. Paralleling the MIME developments, there has been increased integration of the e-mail user agent (UA) and Web browser. For example, mail readers now parse text messages for URLs, and render them as hyperlinks which, when clicked upon, launch a companion Web browser that retrieves and displays the referenced page.

Thus, e-mail systems have access to the multimedia streaming and rendering technology that has been developed for the Web.

Although the MIME standard has been widely adopted by e-mail user agents, the vast majority of the e-mail messages sent today are void of continuous-media (CM) content, such as audio and video. The lack of CM content in e-mail messages is clearly not due to lack of user interest. Indeed, there are many contexts in which CM e-mail would be highly desirable. For example, a young child who cannot type (or even read) would prefer sending an audio message to sending a text message. Family members and friends may prefer to give their e-mail a personal touch by including video. An office worker may be able to record a CM message more quickly than typing a text message—English is spoken at a rate of approximately 180 words per minute, whereas the average office worker types words at a much slower rate. Possibly the greatest impetus to the widespread adoption of CM e-mail will come from the proliferation of small hand-held wireless devices with Internet connectivity—devices for which voice input is more convenient than text input due to space limitations.

Although current telephony voice mail systems provide much of the benefits of CM e-mail, they do not provide all of the features available to CM e-mail. First, telephony voice mail is voice-only, whereas Internet e-mail can be voice-only, or any combination of voice, text, HTML, video, animation, static images, etc. Second, e-mail is hyperlinkable, providing the message recipient with easy access to referenced Web objects. Third, e-mail can be used to convey information when transporting objects in the form of e-mail attachments. Fourth, e-mail is exclusively asynchronous—the message author and message recipient are guaranteed a temporal separation. Conversely, with telephony voice mail, the "caller" does not know whether he or she will be asked to record a message or will enter into a synchronous conversation with the "called."

Storage capacity should also not be a major hindrance to widespread adoption of CM e-mail. Indeed, the cost of disk storage has been declining at exponential rates, and message storage systems with terabytes of storage are now conceivable. Also, speakers are now becoming a common component of computer systems, thus there is little hindrance from the playback side for the widespread use of continuous media messaging.

There are obvious inadequacies in the existing user interfaces to e-mail systems for the creation of CM e-mail. Current e-mail user agents lack audio and video capture functionality. To send an audio message, the sender must record a message in a separate application and save the recording as a file. Then, inside the UA, the sender incorporates the CM file into the message by including it as a MIME attachment. This whole process needs to be integrated into the interface of the UA, so that users can compose and respond to messages as easily as they do with text e-mail.

Aside from hardware and software inadequacies within the end user's computer system, there also exists a psychological barrier to asynchronous voice and video messaging. Because asynchronous voice messaging is unfamiliar to most users, many are timid when

interacting with voice-based systems, much like the initial timidity displayed by people when first exposed to telephone answering machines. This observation, along with the fact that many users may not have the necessary privacy to be comfortable when composing an asynchronous voice message, was made in [3] and [4]. However, repeated exposure to voice messaging and ergonomic enhancements for increased privacy should gradually reduce these psychological barriers.

The problems mentioned so far are only minor compared to the more fundamental problems related to the underlying infrastructure of Internet e-mail, which are impeding the widespread adoption of CM e-mail. The first contribution of this paper is to identify these inadequacies, which include: (1) a faulty cost model for Internet e-mail, in which the recipient bears much of the cost of e-mail delivery, (2) the duplication of message data within mail systems, which occurs when e-mail is sent to multiple recipients within the same domain, (3) the unnecessary delivery of message data that is not rendered (or not fully rendered) by the recipient, (4) a lack of sensitivity to the end user's access rate, which results in excessive delays when retrieving large messages, and an insensativity to the end user's storage resources, which results in undeliverable messages, and (5) inadequate access protocols, which can not be adapted to CM.

To resolve these inadequacies, in this paper we propose a basic scheme of *sender-stored e-mail*, which addresses these problems by taking into account the needs and heterogeneity of CM e-mail users while only requiring incremental changes in the existing e-mail infrastructure. In this scheme, the mail system is responsible for the storage of the CM content of its outgoing messages, which are streamed to the recipient UAs (or media players) in the moment the recipients desire rendering. The entire data stream need not be sent: only those portions of the stream that are requested by the recipient, which he controls through the playback controls of his interface. To accommodate different access rates, the CM server should have the ability to transmit a compressed and layer-encoded version of the CM data that matches the available bandwidth.

We do not restrict our discussion to any particular operating system or application layer software. Our proposals are equally relevant for the growing community LINUX/UNIX users as well as other platforms. Of course, users that rely on older mail readers (such as mh, pine, etc.) will not be provided with the automated procedures available to the more advanced user agents, such as emacs, Netscape, etc. A more serious barrier to users of older systems is the absence of sound cards, whose users will obviously not be able to render audio data. However, these systems represent a quickly diminishing segment of the user community, and so do not present a significant barrier to universal access.

Several authors including ourselves have recognized the importance of sender-stored delivery of e-mail. Gay and Kervella[5] describe a multimedia messaging architecture that supports the creation and playback of synchronized multimedia objects. They describe the MEGHAM system, which is implemented on the X.400 mail standard. Hess

et al.[3] present VistaMail, a multimedia messaging system that emphasizes the unified nature of a video message and relies on extensions to MIME. Along with a detailed explanation of their architecture, they include valuable observations regarding user acceptance. Schürman[6] gives an overview of MIME-based and X.400-based multimedia messaging, and presents the architecture of the BERKOM Multimedia-Mail Teleservice, which is built on the X.400 standard. All three of these approaches rely on some form of distributed storage of message content, where messages include references to separately stored content. MEGHAM and BERKOM both use X.400 approaches, and VistaMail relies on NFS access to CM data files. Our approach is different from these researchers in that we consider CM e-mail schemes that are appropriate for the global Internet. Our schemes require only incremental changes in the existing Internet e-mail infrastructure, and they explicitly address streaming, appropriate cost models for CM e-mail, user heterogeneity, forwarding, annotation, and privacy.

This paper is organized as follows. In Section 2 we explain how CM e-mail differs from traditional text e-mail. In Section 3 we review current delivery procedures for multimedia e-mail. In Section 4 we identify the inadequacies in the existing Internet e-mail architecture to support CM e-mail. In Section 5 we describe our basic scheme of sender-stored e-mail. Section 6 investigates security issues, and describes our proposals to ensure message privacy and integrity. In Section 7 we conclude with a summary, and outline the contribution of a companion paper that addresses issues related to sender-stored e-mail, including QoS, forwarding and annotated replies.

2 How does Continuous Media E-Mail Differ from Text E-Mail?

Given that there is significant user interest in CM e-mail, what changes need to be made in the Internet e-mail infrastructure so that CM e-mail becomes more widespread? To answer this question, we need to look at how CM e-mail differs from traditional text e-mail. The most salient difference is that a CM message, and in particular a video message, is much larger in byte count than a text message. To roughly quantify this difference, consider a message that consists of 180 words. A text version of this message is about 1 KB. For a person who speaks at a rate of 180 words per minute, a 24 Kbps encoding of an audio version of this message is 180 KB. An MPEG-1 1.5 Mbps encoding of a video version of this message is 11,250 KB. Thus, the byte count of an audio message is roughly 100 to 200 times larger than a text message, and the byte count of a good-quality video message is roughly 10,000 times larger than a text message.

The second difference is that CM e-mail has more stringent QoS requirements than traditional text e-mail. When a recipient selects a CM message for rendering, playback should begin within a few seconds. Once playback begins, the audio and video quality

should be good, and the playback should not be plagued with interruptions due to packet loss or client re-buffering. Moreover, the recipient should be able pause playback as well as make temporal jumps forwards and backwards within the message.

The third difference concerns the heterogeneity of user environments. For example, although some users today have high-bandwidth connections to the Internet, 98% of residential users access the Internet at dial-up modem speeds of 56 Kbps or less[7]. The fraction of low-bandwidth modem connections will remain significant for the foreseeable future, as many communities (particularly rural communities) may never get wired for emerging cable and ADSL residential access[7]. In addition to heterogeneous access rates, users are also heterogeneous in terms of local storage. At one extreme are the thin clients with minimal local storage, such as hand-held wireless devices. At the other extreme are systems with tens of gigabytes of storage. The issue of heterogeneity is critical for Internet e-mail because arbitrary collections of user environments need to inter-operate with each other. Although this heterogeneity issue is also present for traditional text e-mail, the issue is of much greater importance for CM e-mail due to its size and QoS requirements.

3 Review of CM E-Mail in the Internet Today

Internet e-mail requires that an arbitrary sender be able to send a message to an arbitrary recipient. This is possible, because of universal agreement regarding the format of messages and their method of transport. In general, an Internet e-mail message is 7-bit ASCII text data that conforms to the specification given in RFC 822, and its transport is accomplished using Simple Mail Transport Protocol (SMTP). These two standards form the basis of Internet e-mail. However, the RFC-822 message format is inadequate for messages that contain anything other than a single body of ASCII text. To provide for multiple-body messages composed of any data type, the Multipart Internet Mail Extension (MIME) was developed. (Reference [1] gives a thorough introduction to Internet e-mail.)

In order to allow incremental adoption by e-mail systems, the MIME format was designed so that MIME-compliant messages also comply with the syntax rules of RFC 822 messages. This was accomplished by dividing the single body of an RFC-822 message into multiple parts, and encoding non-ASCII data into ASCII. Special headers are used to identify subdivisions of the body, the type and format of the data contained within each subdivision, and the ASCII encoding scheme that was applied to the data. For binary data, such as audio and video, the Base 64 encoding is used. The result of this encoding is to increase the size of the data representation by 33%; thus, continuous-media messages, which are already large, become even larger when transported in ASCII encoded form.

Because a MIME message is also an RFC-822 message, it can pass into the recipient's mailbox through its SMTP server. From a transport and storage perspective, SMTP and the MIME standard together enable multimedia messaging in the Internet. However, other than sending HTML documents and images, multimedia messaging through MIME and SMTP has not become commonplace. In the next section we examine the major barriers to its development.

Let's consider an example in which the MIME message format is used in conjunction with SMTP to deliver a video message. Suppose that Alice recorded herself speaking for one minute using MPEG-1 at 1.5 Mbps. She saves the file with the name "12345.mpg." In her UA she types a short text message to supplement her video message and attaches the video file to her message. She addresses the message to Bob and presses the send button. First, we will examine the structure of the message that her UA creates, and then we will examine how it is transported to Bob's mailbox. (Throughout this paper, we assume that the sender is Alice and the recipient is Bob.)

Fig. 1 contains the message that Alice's UA created. It follows the organization scheme of RFC-822; because there is a header section, followed by a blank line, followed by a body. The header section contains the various pieces of meta-information, such as an identification of the sender, the recipient, message date, etc. In particular, note the presence of the *Content-Type* header, which declares that the body is of type multipart/mixed. This header also defines a *boundary* attribute, which specifies a string of characters used to demarcate the beginning and ending of the various body parts.

In our example, there are two separate parts within the main body. Each part follows the basic structure of a body part; it has a header section, followed by a blank line, followed by a body section. The boundary string is used to begin each part. The final part is also followed by the boundary string, but it is appended with two dashes, "--".

The first part contains a single header indicating that the content-type of the first body part is text/plain. The body contains the text created by Alice. The Content-Type of the second body part is video/mpeg, indicating that the body contains an MPEG data file. The Content-Transfer-Encoding header indicates the data has been transformed into simple ASCII text using the Base 64 algorithm. The Content-Disposition header indicates the recipient UA should treat the data carried by this body part as an attachment, which means the UA should not attempt to interpret the data for immediate display, but should present the user an option to open the attached data. The filename attribute is defined within the Content-Disposition header as a suggested name for the UA to display to the user.

The route Alice's message takes is shown in Fig. 2. First, Alice's UA transfers the message to a mail transfer agent (MTA) provided by her mail service provider. Alice's MTA then relays her message by SMTP to Bob's MTA, that is, the *mail server* that accepts messages into Bob's mailbox. These two steps comprise the push phase of

```
From: "Alice Adams" <alice@aaa.com>
To: "Bob Brown" <bob@bbb.com>
Subject: meeting announcement
Date: Tue, 9 Feb 1999 13:18:45 +0100
MIME-Version: 1.0
Content-Type: multipart/mixed;
     boundary="--myBoundary"

--myBoundary
Content-Type: text/plain

Bob,
Don't forget to bring the Peterson
file to today's Meeting at 2:30.
     Alice
--myBoundary
Content-Type: video/mpeg
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
     filename="12345.mpg"

IsAfdBgA4A6B8kAIA+A2AlwB8ybABmYWNfmA
8k/67+i/6I/n7+ZP62/u3+Kf+p/yUA8gBqAb
  . . .
--myBoundary--
```

Figure 1: Message with Video Attachment

message transport. The final step of transport occurs when Bob runs his UA, which then retrieves the message from his mailbox using a mailbox access protocol.

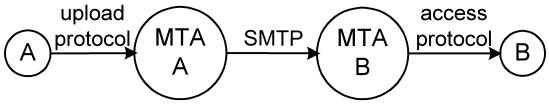

Figure 2: E-Mail Message Transport

Because the message in our example is a one-minute 1.5-Mbps MPEG video clip, its size (after Base 64 encoding) is 15 MB. In today's environment, such a message is frequently too large to pass into the recipient's storage, either because the mailbox lacks available space, or because messages of this size are rejected by policy. If both servers support Extended SMTP, then Alice's MTA announces the size of the message

before it attempts transmission. This allows Bob's MTA to reject the message prior to transmission, rather than mid-stream. However, for the sake of the example, we will assume that Bob's MTA accepts the message and places it in Bob's mailbox storage.

Different UA systems accomplish message transport to and from the user's mail service using different protocols. The first distinction we make is between *standalone* and *Web-based* UAs. In a standalone UA system, a special purpose application runs on the user's computer and communicates intermittently with the remote mail system. It uses SMTP to upload messages to the user's outgoing mail server, and an access protocol, such as POP3 or IMAP4, to retrieve messages from the user's mailbox. Examples of standalone UAs include Netscape's Messenger, Microsoft's Outlook, and Qualcom's Eudora. In Web-based UA systems, the remote mail system provides an interface to the user by sending HTML documents to the Web browser. Messages are thus sent and retrieved over HTTP. Examples of Web-based UA systems include Hotmail and Yahoo! Mail. Fig. 3 graphically depicts the difference between standalone and Web-based UAs.

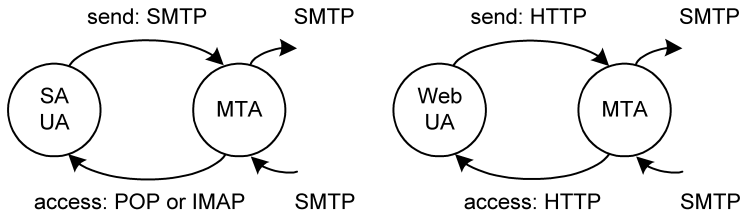

Figure 3: Standalone and Web-Based User Agents

Generally, standalone UAs transfer outgoing messages using SMTP. However, the retrieval of messages is usually accomplished by Post Office Protocol (POP) or by Internet Message Access Protocol (IMAP). POP is designed to be a simple retrieve-and-delete method of mailbox access, where the location of stored message data is in the user's local computer. IMAP provides the ability to leave one's messages in remote storage, organized into a hierarchy of folders. The advantage of IMAP is that one's messages can be accessed from more than a single computer.

Web-based UAs don't run as standalone applications in the user's computer, but operate within a Web browser. The UA can be thought of as running at the remote server site, which generates and sends Web pages that provide the user interface to one's mailbox. These interactive Web pages provide all the UA functionality through HTTP exchanges with the server, including the reading of one's mail and the sending of newly created messages to other users.

A Web interface for text e-mail is possible, because HTML defines input mechanisms to capture keyboard and mouse input and deliver it to an HTTP server. Within the browser window, the user enters text into textboxes, and clicks on various selectable

objects, such as checkboxes, radio buttons, etc. The user submits his input to the server by clicking on a "submit" button, and the browser sends an HTTP POST request to the server that includes the values representing the user's actions. In this way, the remote mail system performs whatever UA function is desired by the user, such as creating a new mail folder, deleting an old message, or sending the user's entered text as an e-mail message to another user.

Web-based systems provide the greatest mobility to the user, because the end user need only have access to a Web browser in order to interact with the mailbox and create and send new messages. Similar to the IMAP approach, messages are kept in remote storage and are organized into a hierarchy of folders.

As an example, suppose that Bob's computer is on a 100 Mbps LAN, and his mail server is also located on this LAN. His UA is configured to check the server every five minutes for the presence of new messages. When a new message is detected, it is immediately retrieved over POP and deleted from the remote store. When Alice's video message (Fig. 1) is detected, Bob' s UA will consume approximately 1 second of LAN bandwidth to copy the data from the mail server's message store to his local storage.

As a second example, suppose Bob connects to his mail server over the Internet with a 28.8 Kbps modem instead of over a LAN. In this environment, his UA will take one hour and ten minutes to retrieve Alice's video message, regardless of the access protocol it uses. Even if his mail service provider is willing to store such large messages in his mailbox, it is not practical for Bob to spend the time and money retrieving them. Another possible barrier is that he doesn't have 15 MB of available disk space to store the video data.

4 Inadequacies of the Existing Internet E-Mail Infrastructure

In this section we identify the key problems with the Internet's e-mail architecture, which are impeding the widespread adoption of CM e-mail:

- Internet e-mail is based on a faulty cost model.
- The storage architecture results in excessive data redundancy.
- Resources are wasted in the transmission and storage of unread, or not fully read, message data. (In the case of CM e-mail, we might say "un-rendered," rather than "unread.")
- Message delivery to the end user is insensitive to the end user's access rate and storage resources.

- The access protocols are inadequate.

Faulty Cost Model

The costs associated with e-mail include (1) consumption of bandwidth when transporting messages into recipient mailboxes, (2) storage of the message until it is deleted from the message store, and (3) consumption of bandwidth when transporting the message from the recipient's message store to the recipient's local machine. These costs are indirectly paid by individuals through subscription payments to an ISP, and possibly telephone usage payments to the telephone company in the case the user's access to the Internet is by modem. Since the sender only pays for a portion of the bandwidth expense, and not for temporary storage of message data, the recipient actually pays more to receive e-mail than the sender pays for sending it. This is contrary to the cost model of ordinary mail, where the sender bears the entire cost of a letter.

Although it is not the only cause of spam[8, 9], this faulty cost model tends to encourage its proliferation. Although it is becoming more feasible to provide users with the required disk storage for CM e-mail, there is a reduced incentive to do so, since this resource would likely contribute to the development of audio and video spam.

Data Redundancy

One impediment to CM e-mail is the current storage architecture, which results in excessive data redundancy within mail systems. In corporations and large-scale ISPs, a single message is frequently distributed to many other recipients within the same mail system. In this case, an exact copy of the message data is duplicated in storage for each recipient, and statically stored until deleted. Forwarded messages and annotated replies also result in further duplication of message data. Thus storage resources, which include active disk space and their protective back up systems, are needlessly consumed with redundant static data.

Wasted Resources

Recipients of CM content will often only want to render small fractions of messages [10]. Recipients will question paying for the delivery and storage of entire CM messages when they only render small fractions of messages they receive. Furthermore, in the case of distribution lists, a recipient may not render any of the CM content for certain messages, particularly for messages containing CM spam. Thus, recipients (or recipients' ISPs) may continue limiting the size of messages they are willing to accept into their message stores, even as bandwidth and storage resources continue to expand.

A closely related problem is the waste of Internet backbone bandwidth in the current mail storage paradigm. Because CM e-mail is delivered in its entirety to the recipient's mailbox, backbone bandwidth is wasted when the recipient only partially renders the message. This problem is exacerbated with distribution lists, which would send a copy of the CM data to each recipient. This waste will have an impact on the cost of the Internet—a cost shared by all Internet users.

Heterogeneous Users

Users are highly heterogeneous in terms of their access rates. The recipient of a short video clip who has T1 access to the Internet might have no problem retrieving the message data, but for a person with modem access, the message retrieval delay would be excessive. It is not always possible to know in advance the transmission rate over which the recipient will be retrieving his mail, because users may access their mailboxes from different computers. Unlike the Web's CM servers[11], there is currently no mechanism within e-mail that allows the recipient to tradeoff quality degradation in exchange for better response time.

Many existing message stores do not have the capacity for storing CM messages that include video or good quality audio. Thus, recipient mail transfer agents typically reject such messages, which makes sending CM messages to such recipients impossible. Although this restriction should eventually ease as disk storage continues to increase, ISPs may choose to continue rejecting large messages to conserve both storage and bandwidth costs in order to keep their subscription prices as low as possible, and thus remain competitive. For example, users of Yahoo's free e-mail service have a storage limitation of 3 MB. All incoming messages that exceed 3 MB are rejected, and Yahoo users can not send e-mail with attachments that exceed 1.5 MB.

Inadequate Access Protocols

The message store access protocols, such as POP and IMAP, treat message data as discrete objects, which are transfered in their entirety before any rendering of them is started at the user's local system. Thus recipients must suffer start up delays when rendering messages with CM, which is particularly problematic for users behind low-bandwidth connections, such as modems and many of the emerging wireless devices.

Recipients of CM content will want to pause/resume playback and make temporal jumps within the message. Although the IMAP standard allows a UA to fetch a byte range of a message, greater functionality, such as that provided by Real Time Streaming Protocol (RTSP), may be necessary to support satisfactory user interactivity.

5 Sender-Stored E-Mail

We use the term *streaming* to describe a client-server system of CM delivery that provides: (1) playback delays not exceeding a few seconds, (2) good audio or video quality without interruptions during playback, and (3) user interaction that includes pausing, temporal jumps and playback rate adjustment. In order to accomplish these objectives, the CM should be layer-encoded, so that the delivery system can send only those data layers that result in an encoding rate that is less than the available bandwidth. For example, suppose the stored media is comprised of 9 layers, with each layer containing data that is rendered at 25 Kbps. If the available bandwidth is 55 Kbps, then the CM delivery system should send the first 2 layers, resulting in a data rate of 50 kbps. Sending any more data would result in buffer starvation at the client if the CM were to be rendered without a significant startup delay.

In this section we propose sender-stored delivery of CM e-mail, where the CM portion of the message is placed in a CM server in the sender's mail system, and a *base message* containing a reference to this data is sent to the recipient in the form of a small text message. This is different from the traditional delivery mechanism, which we refer to as *bulk delivery*, in which the data comprising the entire message is sent in a single transaction. The recipient's UA uses the base message to instantiate an appropriate media player, which will stream the CM from the sender's CM media server. Fig. 4 illustrates the sender-stored delivery process.

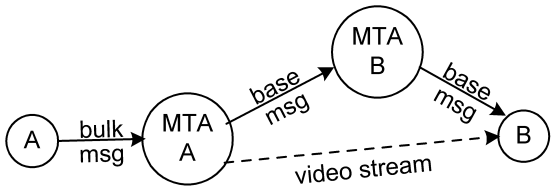

Figure 4: Sender-Stored CM E-mail

Sender-side storage combined with streaming solve all five of the problems described in Section 4. Under sender-stored e-mail delivery, the sender now bears the cost of message storage, and the recipient only pays for the bandwidth used in transmitting that portion of the message data that he chooses to render. Disk storage is not consumed with redundant message data. Bandwidth is not consumed in transmitting non-rendered CM, nor is disk storage wasted in holding such data. Recipients with limited mailbox storage will now be able to receive CM messages. Recipients behind slow network connections are not encumbered with excessive retrieval delays. Additionally, streaming message content from a remote store enables thin clients with relatively small local memories to render CM messages.

With sender-storage mechanisms in place, recipient systems may opt to accept only small-sized messages into their message stores, thereby forcing senders to resort to sender-stored delivery. The primary motivation for mail service providers to do so will be to reduce their storage and bandwidth costs.

5.1 UA Design for Sender-Stored E-Mail

To make sender-side delivery possible to all possible recipients, we limit propose a design that makes changes only to sender systems and leaves recipient systems unchanged (except for possibly a one-time automatic installation of a media player in the form of a plug-in, ActiveX control, etc.).

In the case of a standalone UA system using POP or IMAP, we propose two basic design alternatives. Our first proposal (illustrated in Fig. 5) is to leave the outgoing MTA unchanged, and use it simply to forward the base message as it would a usual text message. In this design, a CM server would need to be running, and the sender's UA would require permission to write into its storage. The UA would be responsible for selecting the name of the media file, encoding the CM data into the format required by the media server, and constructing the base message that references the CM it has placed within the storage of the CM server.

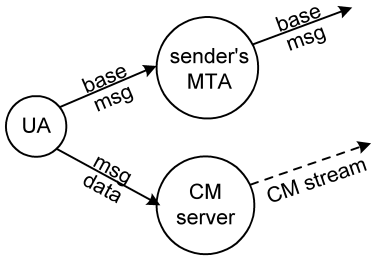

Figure 5: UA Formats Sender-Stored Message

Our second proposal (illustrated in Fig. 6) for the standalone UA system is to have the UA format the message as if it were going to be delivered in bulk to the recipient, using the MIME format as illustrated in Fig. 1. When the outgoing MTA receives the message, it extracts the CM data from the body of the message, and saves it as a file in the storage of the media server. It then constructs a base message, which it delivers to the recipient. By using attachments, a simple version of sender-stored delivery could be implemented without changing existing standalone mail clients.

In the case of a Web-base UA system, there is no real separation between the outgoing mail server and UA, and thus the two design alternatives mentioned above do not apply. The difficulty with Web-based UAs is performing audio and video capture. An

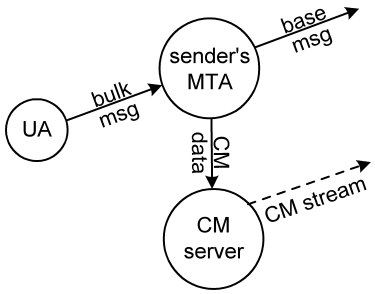

Figure 6: MTA Formats Sender-Stored Message

HTML interface for text e-mail is enabled by the FORM and INPUT tags, which allow the user to enter text data, select checkboxes, etc. The user's input is conveyed back to the server with an HTTP POST request on execution of the FORM's SUBMIT function. An analogous process is not available for user audio or video input, because HTML tags have not been defined and implemented to provide for audio/video capture and transmission to a remote server. A method of extending the HTML user input mechanism to include input from generic devices has been proposed in an Internet Draft[12]. If this mechanism were implemented in browsers, Web-based systems could be extended in a straightforward manner to enable audio and video message input. In the meantime, Web-based systems need to rely on the installation of capture software in the form of a plug-in, ActiveX control or Java applet. (Version 2 of the Java Media Framework (JMF) enables audio and video capture from within Java applets. At the time of this writing, a beta-release of Version 2 is available[13].)

5.2 Implementation of Sender-Stored CM E-Mail

We now describe in greater detail our proposal for the implementation of sender-stored CM e-mail. We do this for the case of a UA designed to handle CM storage and base message creation, as shown in Fig. 4. Thus, in this design, no changes are needed at the recipient or the sender's outgoing MTA. After Alice creates a message and issues the command to send, the UA stores a CM file in her mailbox storage. The UA then formats a base message with a reference to this file, which it sends along the normal route taken by a traditional e-mail message during the push phase of e-mail transport. When the recipient checks for new messages in his mailbox, the pull phase of transport for this message begins. His UA will build a list of messages that are in his mailbox, comprised of senders' names, subject headings, message dates, etc. When he selects the new message, he will have the option to render the CM.

In order to deliver sender-stored CM e-mail with adaptive streaming to an arbitrary

recipient, the base message needs to be processed by a media player outside the recipient's UA. With current UAs, the only way to accomplish this is by formatting the base message as an HTML document that links to a secondary object in the sender's mail system. When the recipient clicks on the link, his UA will instantiate its companion Web browser and have it send the HTTP request to process the hyperlink. When the sender's Web server receives the HTTP request from the recipient's browser, it will be able to ascertain the browser type, its version number, and the operating system on which it is running. With this information, it can tailor a response that is suitable for the recipient's environment. Unfortunately, knowledge of the available bandwidth between CM server and the recipient can not be known in advance, and so the CM server must begin transmitting data with an initial *conservative* bandwidth assumption or via pre-configuration of the recipient's CM agent, as is done with the RealAudio agent. While transmission proceeds, the application can adjust the start up delay and rate of data compression as it refines its bandwidth estimate.

We now identify two choices for the secondary object to which the base message links, depending on whether the implementation uses the plug-in approach or Java applet approach. If it uses the plug-in approach, the secondary object is a file, such as a Synchronized Multimedia Integration Language (SMIL) document[14], which provides the media player plug-in with the information it needs to stream the CM data from the sender's CM server and render it to the recipient. If the implementation uses the Java applet approach, the secondary object is an HTML document, which references a Java applet that performs the same function as the plug-in.

Assuming the plug-in approach is used with a SMIL document, the SMIL document will be requested by the recipient's browser with an HTTP GET request. When the Web server in the sender's mail system receives this request, it will send an HTTP response that includes the SMIL document within the message body and set the Content-type header to a value that causes the recipient's Web browser to instantiate the media player plug-in. As an example, suppose Alice sends Bob a video message, and her system uses sender-stored delivery with the plug-in approach. Fig. 7 shows the base message that Alice's mail system delivers to Bob's mailbox.

Bob's UA will display the HTML document in its window. When Bob clicks on the video message link, his UA will instantiate a companion Web browser to send the HTTP request for the file "12345.smil." The Web server running on host 'mail' will return the SMIL document within an HTTP response as shown in Fig. 8. Notice that the Content-type header is set to the (unofficial) MIME type *application/cmail*. When the browser receives this response, it will instantiate the helper application associated with this MIME type, and pass to it the SMIL document in the body of the response message. Then, the helper application will stream the data from Alice's CM server and render the video to Bob. If this were to be the first time Bob plays video mail generated from Alice's system, his browser would alert him of the need to install a new plug-in to

```
From: "Alice Adams" <alice@aaa.com>
To: "Bob Brown" <bob@bbb.com>
Subject: meeting announcement
Date: Tue, 9 Feb 1999 13:18:45 +0100
MIME-Version: 1.0
Content-Type: text/html;
     charset="iso-8859-1"
Content-Transfer-Encoding: 7bit

<HTML><BODY>
Bob,<P>
Don't forget to bring the Peterson file
to today's Meeting at 2:30.<P>
Alice<P>
<A href="http://mail.aaa.com/12345.smil">
video message</A>
</BODY></HTML>
```

Figure 7: Base Message

process the contents of the HTTP response. The most popular browsers in use today guide the user through a simple installation procedure. During this installation, the new MIME type is associated with the plug-in, and thus the processing of future media objects with this MIME type will be delegated to the plug-in.

```
HTTP/1.1 200 OK
Date: 12 Feb 2000 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Content-type: application/cmail

<smil><body>
<video src="rtsp://mail.aaa.com/12345.mpg">
</body></smil>
```

Figure 8: HTTP Response with SMIL Document

If a Java applet approach is used, the video message link in the base message of Fig. 7 will reference an HTML document, rather than a SMIL document. This document would then contain an APPLET tag so that the browser would retrieve and run the applet, which would then be able to stream the data from the CM server and render it to Bob using classes from the Java Media Framework. Fig. 9 shows the HTML document that is sent to Bob's browser. With this approach, we pass the SMIL document to the

applet through a PARAM sub-element of the APPLET element. Similar to the plug-in approach, if Bob's browser doesn't support the Java Media Framework or the version of the Java virtual machine needed to work with these classes, he may be prompted to allow the automatic installation of the enabling software.

```
<HTML>
<BODY>
<APPLET code="cmail.class">
<PARAM name="SMIL">
<smil><body>
<video src="rtsp://mail.aaa.com/12345.mpg">
</body></smil>
</PARAM>
</APPLET>
</BODY>
</HTML>
```

Figure 9: HTML Document with Applet

Whether a plug-in or applet player is used in all of these examples, an RTSP session is established with the CM server that provides access to the CM in the sender's outbox. The player then issues RTSP commands to set up and control a UDP-based transport channel to deliver the video data with real-time properties[15]. Within seconds of clicking on the play button, the video is rendered to Bob. Because RTSP is being used to control the data stream, Bob can pause, rewind, fast-forward, and jump to arbitrary points within the stream using the media controls available to him.

6 Security and Privacy

Because of the ease by which data communications on the Internet can be intercepted, users are increasingly encrypting their e-mail messages to ensure privacy. Additionally, users are becoming more wary of the potential for computer break-ins, which provide intruders with access to the contents of e-mail messages. In this section, we describe a procedure of protecting sender-stored CM e-mail from security threats such as break-ins, communication monitoring and man-in-the-middle attacks.

In the following discussion, we assume the reader is familiar with public key cryptography and methods of obtaining reliable public key certificates [17]. Developers of sender-stored e-mail systems can rely on the extensive support for security that is now provided by Web servers, browsers, UAs, and encryption libraries.

One way to secure sender-stored CM e-mail is to encrypt the CM data with the public key of the recipient, and store the CM in encrypted form with the CM server. Then,

only the recipient can decrypt the CM data with his private key. Such an approach would secure the message data from being rendered by an intruder that has broken into the CM server, and by attackers who have access to the communication channel between the CM server and message recipient. However, there are two drawbacks to this approach. First, decryption using an asymmetric key is time consuming, and would most likely introduce a significant delay when rendering the CM. Second, if there were multiple recipients of the message, then a separate copy of the CM data would need to be stored with the server for each recipient.

To avoid the problems that result from encrypting the CM with the recipient's public key, a symmetric key can be used instead. In this case, the sender's system locally generates a random secret symmetric key using a pseudo-random number generator seeded with a secret known only to the sender. This symmetric key is used to encrypt the data to be stored with the CM server. The recipient's public key is then used to encrypt the symmetric key for secure delivery to the recipient in the base message.

In the case that there are several recipients of the message, the symmetric key can be securely delivered to all of them using their individual public keys. Note that all forward recipients of the base message will also have access to the CM data, because they will have the required symmetric key.

Assuming the secret symmetric key is long enough, possession of the CM data would be useless to a person without the decryption key. Thus, there is no need to authenticate access to the CM server. Nevertheless, requiring authentication may be prudent in that it adds one more barrier to a potential attacker. Also, it can control whether forward recipients are allowed access to the data through the sender's CM server. If the sender doesn't want to service requests from forward recipients of the base message, then the server can require that the client prove it is one of the originally intended recipients of the message. Alternatively, if the sender is willing to service requests from forward recipients of the base message, then the server can require that the client prove it has possession of the decryption key.

Other measures are required to fully secure the system described above. For instance, a virus could be running in the sender or recipient computer that intercepts the CM data in its non-encrypted form as it passes through the sender's media capture system or the recipient's media rendering system. Other potential threats include inadequate erasure of the decrypted symmetric key or the non-encrypted CM data from the memory or persistent storage of either the sender or recipient systems, and exposure of the sender's secret used to seed the pseudo-random number generator. These and other potential threats would also need to be guarded against by careful implementation of an overall security solution for the involved systems.

7 Conclusion

In this paper, we have identified the major weakness of Internet e-mail that obstruct the development of CM messaging. These include a faulty cost model, in which the recipient bears much of the cost of e-mail delivery; the duplication of message data within mail systems, which occurs when e-mail is sent to multiple recipients; the wasteful delivery of non-rendered message data; and a lack of sensitivity to the end user's access rate, which results in excessive delays in retrieving large messages. We showed how all of these problems are solved with a sender-stored message delivery architecture, where a small text message (base message) is sent that allows the recipient system to stream the CM message content from the sender's storage.

In order that our sender-stored e-mail delivery architecture be backward compatible with existing systems, we described an implementation in which changes are only required to sender systems. We described the three major Internet UA systems (POP, IMAP and Web-based), and we discussed how each of these systems could accommodate sender-stored e-mail. We also described how existing security mechanisms based on public key cryptography can be use to ensure the privacy and integrity of sender-stored CM e-mail.

Because sender-stored e-mail is a major paradigm shift for existing e-mail, it engenders several new problems. First, there is a QoS problem, which results from the streaming delivery of CM across a bandwidth-limited network path. Second, there is the problem of deciding when to delete message data from the sender's outbox, which the recipient may wish to access at an unknown point in time. Third, sender-stored e-mail introduces complexities into the process of forwarding and replying with annotation (embedding pieces of the original message in the reply). We address all of these issues in a companion paper[20].

References

[1] L. Huges. *Internet E-mail: Protocols, Standards, and Implementation* Artech House, Norwood, MA, 1998.

[2] J. Reynolds, J. Postel, A. Katz, G. Finn, and A. DeSchon. The DARPA experimental multimedia mail system. *IEEE Computer*, Oct 1985.

[3] C. Hess, D. Lin, and K. Nahrstedt. Vistamail: An integrated multimedia mailing system. *IEEE Multimedia*, Oct-Dec 1998.

[4] D. Turner and K. Ross. Asynchronous audio conferencing on the Web. In *Advances in Intelligent Computing and Multimedia Systems*. International Symposium on Intelligent Media and Distance Education, Baden-Baden, Germany, Aug 1999.

[5] V. Gay and B. Dervella. MHEGAM: A multimedia messaging system. *IEEE MultiMedia*, Oct-Dec 1997.

[6] G. Schürmann. Multimedia mail. *Multimedia Systems*, October 1996.

[7] E. McPhillips. The structure and trends of the ISP market. Hewlett-Packard Laboratories Technical Report HPL-IRI-1999-002, Jun 1998. http://www-iri.hpl.hp.com/.

[8] S. Hambridge and A. Lunde. Don't spew: A set of guidelines for mass unsolicited mailings and postings (spam). RFC 2635, Jun 1999.

[9] D. Dern. Postage due on junk e-mail: Spam costs Internet millions every month. *InternetWeek*, May 1998.

[10] J. Padhye and J. Kurose. An empirical study of client interactions with a continuous-media courseware server. In *Proceedings of NOSSDAV '98*, Cambridge, UK, Jul 1998.

[11] RealNetworks, Inc. SureStream: Delivering superior quality and reliability. http://www.real.com/devzone/library/whitepapers/surestrm.html.

[12] J. Salesman. Form-based device input and upload in HTML. Submitted to the W3C, Jul 1999.

[13] Sun Microsystems, Inc. Java Media Framework API, Jul 1999. http://java.sun.com/products/java-media/jmf/index.html.

[14] P. Hoschka, ed. Synchronized multimedia integration language (SMIL) 1.0 specification. W3C recommendation, Synchronized Multimedia Working Group, Jun 1998.

[15] H. Schulzrinne, A. Rao, and R. Lanphier. Real time streaming protocol (RTSP). RFC 2326, Apr 1998.

[16] Bruce Schneier. *Applied Cryptography, second edition*. John Wiley & Sons, 1996.

[17] C. Kaufman, R. Perlman, and M. Speciner. *Network Security: Private Communication in a Public World*. Prentice Hall, 1995.

[18] S. Dusse, P. Hoffman, B. Ramsdell, L. Lundblade, and L. Repka. S/MIME version 2 message specification. RFC 2311, Mar 1998.

[19] T. Dierks and C. Allen. The TLS protocol version 1.0. RFC 2246, Jan 1999.

[20] D. Turner and K. Ross. A comprehensive architecture for continuous media e-mail on the Internet. In submission.