Active Messenger: filtering and delivery in a heterogeneous network

Stefan Marti and Chris Schmandt

MIT Media Laboratory
20 Ames Street,
Cambridge, MA 02139, U.S.A.
{stefanm, geek}@media.mit.edu

Abstract. Active Messenger (AM) is a software agent that dynamically filters and routes email to a variety of wired and wireless delivery channels. More than just a router, AM is a dynamic process that monitors a message's progress through various channels over time. It observes which devices have been used to originate or respond to messages, recent log ins, and caller-ID when checking voice or email over the phone. Its goal is to ensure that desired messages always reach the subscriber, while decreasing message volume when the user is less reachable. AM also acts as a proxy, hiding the identity of the multiple device addresses at which the subscriber may be found. It also caches channels to guarantee seamless information delivery for the user in a heterogeneous network.

1 Overview

Active Messenger [8][12] is a system that delivers messages, based on priority, to a variety of devices in a heterogeneous network. Its goal is to ensure delivery of urgent messages across multiple user access methods, while throttling back delivery of less important messages. To support these goals, AM may attempt to reach a series of devices over time, or to resend messages when a device comes back into range, but without sending redundant messages. AM uses sets of explicit and context sensitive filtering rules, based on a user's recent correspondence, calendar, and location. AM also transcodes messages to fit different display characteristics of mobile text-based devices, as well as for faxing or speech synthesis for voice delivery. AM infers device availability from a combination of network supplied information (device in range, device re-entered range, delivery successful) and observation of user originated traffic.

As just implied, AM is a two-way system. It acts as a proxy for messages or replies originated from the mobile devices, rewriting them to appear to originate from the user's canonical, published email address, rather than the address of the particular device. Because the user can originate a message on any device, and expects a reply to appear promptly on that device, AM tracks "threads" of such messages for special delivery. It also provides, through short structured messages, access to and modification of a number of personal information management tools, such as address

book and calendar, and access to limited web-based sources. Additionally, Active Messenger can explain its behavior in handling a particular message, and the user can override that specific contextual rule.

Why a heterogeneous network?

Currently, we employ devices with different network coverage and usage fees. For example, courtesy of sponsor Motorola™ we have campus wide two-way paging from an in house system. In the US, SMS is available on GSM phones, but GSM is not available in every metropolitan area and GSM service providers do not always allow interchange of SMS messages. Other wireless devices exhibit spotty coverage, where a user may switch to a voice channel such as listening to messages over the phone or a portable terminal such as Pocketmail™[1]. Just traveling to a summer home a few hours from Boston crosses three zones of different device access. There are also international issues of which devices work in which countries and at which frequencies.

Additionally, we argue that even if a single mobile device could be employed, a system such as Active Messenger still needs to be aware of multiple access methods. First, many messages will be read on a normal computer or laptop screen and need not be sent to any other device. Second, most users will not want to receive *all* their messages on the mobile device, but only the most important ones. Third, there are situations in which it is very desirable to access messages via fax (perhaps to share with others or deliver to third party) or listen to via synthetic speech over any telephone (perhaps a coin-operated roadside phone in a remote area with no wireless service of any form).

Filtering and delivery

AM relies on several sources to classify messages. Users can specify rules linking particular kinds of messages to user-defined categories, using a modified version of the public domain procmail syntax (e.g., [14]). For example, messages from a daughter or boss may be "very important," messages to a mailing list may be "ignore," and messages from students may be "important."

Additionally, AM employs the CLUES filtering system [9] written by Matt Marx. On an hourly basis, CLUES examines a number of personal information databases to come up with an additional set of regular expressions defining "timely" messages, after consulting the user's log of sent email, dialed phone calls (if using computer telephony tools), calendar, and address book. For example, messages containing the word "Ubicomp" within a few days of a calendar entry such as "Ubicomp paper due" will be tagged as timely, as would a reply message from the conference chair if the user had sent him a message yesterday. If the user provides a contact phone number, or even just an area code, in his calendar, CLUES attempts to associate emails with location via the address book, and marks as "timely" messages from senders in that area.

Users indicate the ordering of message priorities, including ordering of the "timely" category. A message is evaluated for timeliness when it arrives, and is

[1] http://www.pocketmail.com/

assigned to the highest priority for which it matches. If a static rule has been created to identify messages from a boss as "very important" and this category is ranked higher than timeliness, a message from a boss about something in my calendar tomorrow would be "very important" and not "timely."

AM uses the user-defined ordering to determine how hard to attempt to deliver a message. In to specifying filter categories and ordering them, the user must indicate which of these categories should be sent to which devices; currently this is done by editing a text file. Combined with geographic or situational (when a device is carried) locality, this mapping allows AM to throttle message delivery when a user is in a less accessible mode.

When a user is known to be online, there is no reason to hold off any messages so AM simply observes the progress of the message through whatever mail reader the user employs. Since one of our devices—the in house Motorola pager system named *Canard* [2][3]—works only within a few kilometers of campus, and one of the authors lives out of that range, it is safe to assume that she is "at work" when in range, and a large number of messages are sent to that device. When further away and using a more limited or expensive service (such as Skytel™ or Iridium™) he limits his messages to only the highest priorities. But when he can connect in a more direct and non-interrupting manner, e.g. by a wireless Palm VII™[2] PDA, he again prefers to receive a fairly large fraction of his rated messages (which is still only about a third of all his messages). In this way, AM strives to guarantee prompt delivery of very important messages but pace delivery of other messages to limit their degree of annoyance.

In order to deliver a message, AM takes a number of timed steps after a message arrives and is sorted. The following example (Fig. 1) shows what happens when a new email message arrives. Let's assume that the user has the following line in her preference file:

```
Mapping
important = canard(20), vpager(13), phone(14), fax(35)
```

This describes the channel sequence for important messages. It means that if a message is important, it will be sent to the Canard pager (in house paging system), after that to a Voice Pager[3], then to a phone, and then to a fax. The numbers in brackets mean the delay until a device or channel is used.

Let's assume furthermore that the user is currently at home and has the following entries in her preference file:

```
Home
canard   = johndoe@canard.mit.edu, anytime
vpager   = 654-4567, not 0-7
phone    = 423-7755, not M-F 22-8, not SU
fax      = 423-7755, not 2-7:30
```

This means that at home, she has the channels **canard**, **vpager**, **phone**, and **fax** available. For each channel, a number or address is specified, and the time when it is OK to use the device.

---

[2] http://www.palm.com/products/palmviix/

[3] http://www.motorola.com/MIMS/MSPG/Press/PR19980109_3072.html

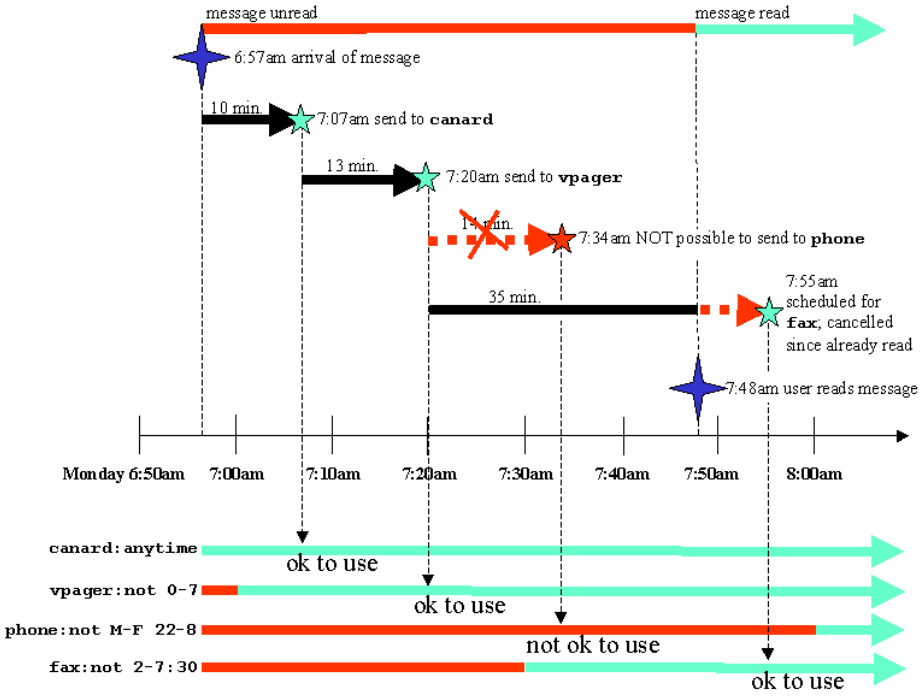

Fig. 1: Channel sequence example

The message arrives at 6:57am. According to the above channel sequence, the first channel would be **canard** in **20 minutes**. However, this initial delay is scaled down indirect proportionally to the user's idle time: if the user is idle for more than an hour, the message gets sent immediately. Because our user has checked email half an hour ago, the delay is scaled down to 10 minutes. Before the agent can schedule this event, it checks if **canard** is allowed at that time at that location. The preference file says **anytime** is ok for **canard** at **home**, so Active Messenger schedules this event, and waits.

After 10 minutes, if the user hasn't read the message otherwise, e.g., by reading it from the mail spool file, Active Messenger sends it to the Canard pager. Right after the sending, Active Messenger looks up the next channel, which is **vpager** in **13 minutes** from then. It's now 7:07am; the time of the sending would be 7:20am, which is a valid time, because the Voice Pager only wants to get messages after 7:00am. Then the agent waits again and checks all available back channels if the message gets read somehow.

After 13 minutes, if the message's is still not read, Active Messenger calls up the Voice Pager number and synthesizes the message with the text-to-speech module. Right after that, the agent tries to schedule the next event, which would be **phone**. The phone call would be in **14 minutes** at 7:34am. Unfortunately, the user does not allow Active Messenger to call her up on the **phone** at home from Monday until Friday after 10pm and before 8am. Therefore, this channel is currently not available,

and the agent skips it. The next entry would be **fax**. It is now 7:20am, the delay for sending faxes is specified by the users as **35 minutes**, and so Active Messenger schedules a fax sending for 7:55am. Then the agent waits again.

The user, however, happens to log in to her computer and read this email message at 7:48am. The "message read" level rises over the threshold, and the message is regarded as read. Therefore, Active Messenger cancels the fax.

Device handoff and intermittent connectivity

In a heterogeneous environment, AM supports graceful device handoff. AM detects "device" presence in a variety of ways. The Unix "finger" command indicates computer activity and can give some hints as to whether a login session is local or remote; if the user is logged in, she is likely to be reading email so AM delays longer before deciding that a message is not going to be read and sending it to the first in a series of possible devices. Similarly, if the user phones in to access voice mail or hear email, caller ID on the call may indicate whether she is at home, or a geographic location (area code).

For mobile devices, different networks provide different indicators of connectivity. Some indicate when a device is within range, some indicate when a device newly arrives in range, some explicitly indicate receipt of a message on the device, and others provide no indication at all. In all cases, messages originated from a device and received through AM (as a proxy) indicate that a user is active on that device. If AM does not know whether a device is in range, it sends a limited number of messages to it before waiting for some indication of successful delivery; this is important because some networks buffer messages internally, and when the user comes back in range, perhaps days later, many stale messages could be waiting, wasting money, bandwidth, and the time spent deleting them on the device.

When AM detects that the user has switched devices or a device is newly within range, it rescans all the messages which might have been sent to the device, to determine whether they are still unread. If so, it then sends them automatically. This handoff could be triggered by a signal from the network, change in status of a previously sent message to "received," or an incoming message suddenly appearing from the device. This caching and resending when appropriate has resulted in very effective "seamless roaming" across devices and networks.

Another aspect of modifying device priority is "threaded" messages. A user can send a message to any Internet address from any device. This could be someone who appears in some predefined filter rule. If not, on the next hour when CLUES runs, that person will become "timely"; until then a reply from that person would most likely be unrated.

In any case, the device from which the message was sent my not be configured to receive a message of whatever category, if any, the reply would have, so it would be missed. In order to avoid this unwanted behavior, AM tracks message "threads," i.e. new messages originated from each device. An incoming message is matched against these threads, and if a match is found, the message is immediately sent to the original originating device. It is then sent through the normal chain of devices, in case that device is no longer in range.

Similar behavior is required to notify the user of rejected messages, perhaps because of ill-formed or mis-typed destination fields. It is very frustrating to send a

message and be wondering why the recipient has not replied when, in fact, the message was never delivered and the rejection notification was missed!

Role as proxy mailer

Although most AM devices are email-addressable, an AM user may not wish to reveal their addresses, for a variety of reasons. Different devices may be in use at different times, so a message sent to any one may not be delivered for some time. We may change devices from time to time and don't wish to bother having to so inform all our correspondents. Most important, we may wish to keep the addresses of these devices secret and rely on filtering agents to make sure that only the desired messages are delivered to them. Similarly, if our correspondents use filters, these filters may not recognize the addresses of our devices as "ours."

Our correspondents need not to know our "physical" device addresses, but only the unique "logical" one: our canonical email address. AM transforms it to the appropriate physical address, ensuring that device addresses will be used purely as physical addresses.

To this end, AM users do not mail directly from their devices, but rather send specially formatted messages to themselves (messages of the form "m <recipient list> (optional subject) message body"). The recipient list is interpreted at the AM server, allowing the use of nicknames (Unix mail aliases) or last names from one's address book, in addition to canonical Internet addresses. The AM server repackages these messages so that they appear to come from our normal home email addresses. It also updates files to track outgoing message threads.

Similarly, incoming messages sent to the devices actually appear to the device to come from our home mail accounts. This means that any reply will be sent back to that account; AM puts enough information in the subject line of the message that a reply can be matched against the message for which the reply is intended. So the reply message, too, is repackaged, with the original subject, our canonical mail address, and a copy of the original message appended (this "original copy" can be suppressed by terminating the reply text with the '-' character).

These methods hide the identity of our devices and make all messages appear to originate with our normal email system.

Connectivity vs. PDA

We have tried to avoid allegiance to any particular device, and instead have emphasized AM's ability to function in a heterogeneous network, including multiple media. We may equally well read a message on a small screen as listen to it spoken over a mobile phone; the choice will depend on which device has connectivity, the content of the message, and the user's current activity (e.g. driving vs. riding on the train). In fact we have been pleased at how easily AM can be extended to support any text-based device with an email address.

However, one advantage of a "primary" device is its role as a PDA, keeping a user's calendar, address book, etc. So we incorporated these features into our mobile messaging architecture. In addition to sending mail by proxy, as just described, over the phone or by text pager a user may access and modify his address book, calendar, and to-do list. Over both media users may also access a variety of local and web-

based databases, including weather forecasts, dictionary lookups, news headlines, and traffic reports. Additionally, from a text device one may execute an arbitrary Unix command line on one's office computer; the output (*stdout* and *stderr*) are sent to the pager as a response.

For the telephone-based interface, these features are activated by touch-tone or speech recognition user interfaces. From text devices, the features are activated by sending structured messages. For example, initial keywords such as "wx," "traf," and "def" take the rest of the message and send it to the weather, traffic, or definition (dictionary) service. A sub-system called *Knothole* [6] parses these incoming messages, contacts the appropriate databases or services, and filters down the text responses as appropriate for small screen devices.

Besides the convenience of "one device does it all," this additional functionality has another desirable side effect: increased network traffic. The more useful any device is, the more it will be used. The more any of a set of devices is used, the more accurately AM can track device usage and infer which device is currently active and hence should receive urgent messages.

Confidence in the system

This leads to a sensitive topic from the user's perspective: confidence in the system. Almost by definition, when the user is really depending on a mobile system such as AM, he is away from normal desktop computers and will have difficulty accessing system logs, glancing at his mailbox, etc. If a long period has elapsed since the last message, does the system know where (logically) the user is? Is the system running? Is the system filtering correctly? Why hasn't a reply to an urgent message been received?

AM users can employ several methods to verify system integrity. The first is to simply send a request for any personal information, such as an address book lookup; absence of a prompt reply indicates severe problems, although these could be in the device or its network as well as in the AM server. For more detail about message forwarding, a user may send a "sum" request, which summarizes recent email received and forwarded; lost messages may then be retransmitted on request. (Note: this feature is mostly obsolete due to the success of the "seamless roaming" methods described above.)

Since a system such as AM is never going to do what the user wants all the time, it resorts to several methods to at least explain what it did. The first is a web page that shows how it rated each message and what steps were taken when (Fig. 2).

Message					**Is it read?** (Threshold is 95%)				**Skytel**				**Fax**		**SMS**
Nr	*Arrived*	*From*	*Category*	*Clues status*	*Likelihood*	*From*	*Detail*	*When*	*Sent*	*Message ID*	*Arrived*	*When*	*Sent*	*Fax ID*	*Sent*
70	May 12 08:06:00	**Professor Ellen Fuhrer**	**personal**	0	**15%**	canard	arrived there	May 12 08:08:03	-	-	-	-	-	-	-
68	May 12 08:00:23	**King Kong**	**important**	0	**0%**	-	-	-	May 11 20:30:02	4232	yes	May 11 20:35:06	May 11 20:31:51	83	-
67	May 12 01:04:09	**Johnny Depp**	**veryimportant**	0	**90%**	heard on 423 23423	heard it	May 12 01:20:03	May 11 20:25:02	1213	no	-	-	-	-

Fig. 2. Part of AM status web page: its main table lists 35 parameters for each email message

This provides a first level of help facility, though it is useful only when the user has access to a browser.

A more portable confidence builder is an explanation facility for the CLUES dynamically filtered messages, written by Sean Wheeler. At the time that CLUES generates its rules, it generates an explanation template for each rule. Later, a user who receives a questionable message can send a request "explain <message number>" and receive back a response such as "You sent mail to this person yesterday at 5:15pm" or "This mail comes from a domain which matches an entry in your calendar tomorrow." Users are in fact much more tolerant of apparently incorrect system behavior when (1) there is at least a reasonable explanation and (2) the explanation can be delivered promptly, when and where the behavior is first noticed.

Additionally, AM users can cancel a rule. For example, one of the authors was bombarded with messages about "how to change the toner cartridge in the HP printer on the 3rd floor" because he had a visit with HP later in the week, according to his calendar. Cancellation prevents that rule from firing again, at least for a few days, and avoids similar deluges of unwanted messages.

2 Related work

IPulse™

IPulse™[4] is a commercial product that mediates between two subscribers by finding a way to get a text message or audio stream through, according to the preferences of the receiver. It can connect users to each other by computer, phone, pager or mobile phone through a simple point-and-click contact. It also allows users to customize their communications by setting up individual profiles that indicate when, by whom and how they want to be reached. It alleviates the contacting person from the burden of finding the right channel. The iPulse™ framework consists of a client application and a back-end server system. The main function of the framework is to provide users with a simple and secure way of establishing communication sessions with other users or services, running either on IP or other networks like PSTN.

Active Messenger addresses the same problems as iPulse™. However, iPulse™ is a proprietary system that is directed towards service providers. Although the manufacturer writes that new services can be implemented easily, legacy services such as stand-alone paging systems or fax may not be integrated.

OnTheMove

OnTheMove[5] is a three-year project that ended 1997. It focused on how to deliver multimedia content to mobile devices. It is based on the a middleware prototype called *Mobile Application Support Environment* (MASE) that is located between the wireless networks, e.g., GSM, DECT, UMTS, and the applications, e.g., video

[4] http://www.ericsson.com/ipulse/
[5] http://www.sics.se/~onthemove

conferencing, personal newspaper, etc. MASE stores user preferences, detects the location of the user, and adapts to the status of the wireless networks and the available bandwidth. "It hides the complexity of networks from applications, making different wireless networks appear as a seamless and homogeneous communication medium. Multimedia conversion allows content to be delivered to a mobile device in a format that is appropriate to its capabilities and also the characteristics of the network in use. The location manager provides a means of determining geographical position (…) A session manager provides resilience to unplanned disconnection and a replica manager shows how file synchronization can be achieved." [4]

Although this project addresses the same problems as Active Messenger, no information is available about if or how MASE filters and prioritizes communication. It is also not clear how the user specifies her preferences for certain channels depending on the importance of an event. Active Messenger fits well in the general framework set by the *OnTheMove* project.

The Mobile People Architecture

The *Mobile People Architecture* (MPA) [1][13][10] is a framework for connecting people instead of their devices. It focuses on finding people and communicating with them personally, as opposed to communicating only with their possibly inaccessible machines like cellular phones and pagers that are turned off. The *personal proxy* has a dual role. As a *tracking agent*, the proxy maintains the list of devices or applications through which a person is currently accessible. As a *dispatcher*, the proxy directs communications and uses *application drivers* to convert the message into a format that the recipient can see immediately.

The framework of the MPA is more general than the one of Active Messenger, because it includes also stream-to-message conversion. E.g., if the user receives a phone call and is currently reachable through email only, the *personal proxy* converts the voice mail to an email and sends it to the user's computer. The system looks at the pitch of an incoming voice message to decide where the message has to be forwarded.

A significant difference between Active Messenger and the Mobile People Architecture is that the MPA does not take several steps over time to guarantee the delivery of a message, trying multiple channels and awaiting possible user reactions.

Priorities and Mobile Manager

Microsoft Research's *Notification Platform*[6] contains several projects that are about context-sensitive services and incoming messages. *Priorities* [5] is trained by the user to assign certain priority levels to incoming email. Using an adaptation of the *Support Vector Machine* method to determine the urgency of each message, the program can announce important email with special audio cues. Priorities senses when the user is busy by monitoring her activity. Given a stream of sensed keystrokes and mouse movements, the agent waits an appropriate amount of time after she has stopped inputting text to interrupt her with a message. Priorities will forward the high priority email messages and scheduled alerts to the user's cell phone or pager if it senses that she is away from her desk.

[6] http://www.futureofsoftware.net/il40010/il40010.asp

Mobile Manager[7] delivers email, calendar, and reminder information from the user's Outlook email manager to her mobile device, taking in account information from *Priorities*. The user can set up to four different profiles, each with different notification rules. It can also deliver notifications at customized time intervals, after a specific number of messages have been accumulated, or after the user's desktop PC has been idle for a specified time.

Although these projects include some features of CLUES filtering and AM, they do not allow sending messages to several devices in turn, awaiting user reactions. AM goes further by, e.g., supporting graceful device handoff, taking in account if a message was read on a mobile device, and which communication channels are active.

Others

There are commercial products that implement a partial functionality of AM, mainly the information request functionality for two-way wireless devices. Most of them do not have any forwarding or routing capabilities.

PocketGenie™[8] by the *WolfeTech Corporation*™ is an add-on service for two-way pagers. It provides limited browsing and query-and-response access to Internet content. A content menu includes directories, reference sections, package tracking, financial updates, news, sports, horoscopes, traffic and road conditions.

The cross-modal messaging platform by *MessageMachines*™[9] is a service for delivering messages to instant messaging, mobile phones, pagers, Palm™, and other devices. Through automatic and remote controls, the subscriber determines where and how messages should be delivered, absolving the sender from knowing the best way to reach them.

Thinmail Inc.[10] is a forwarding system that filters email attachments and creates private links. Users sending email from wireless devices can use Thinmail to reformat messages, stripping and storing attachments, changing HTML mail to plain text, and previewing documents while a filter selectively blocks senders. A scanning mechanism interprets commands sent as email messages: document forwarding, converting it to text, or printing it to any fax machine. The service also acts as a proxy, so that all email appears as if it came from the user's standard address.

3 Implementation

The Active Messenger was built for the specific communication needs of the members of the MIT Media Laboratory Speech Interface Group. They have a multitude of communication channels and devices available. First, there are networked PC's and workstations in the offices, and all users have PC's with dial-up or even permanent network connections at home. Email can be delivered to alphanumeric pagers with different ranges (*Canard* [2][3] is MIT Campus wide,

[7] http://www.microsoft.com/office/outlook/mobile/default.htm
[8] http://www.wolfetech.com/
[9] http://www.messagemachines.com/
[10] http://www.thinmail.com/

SkyTel™ USA nation wide in major metropolitan areas), as well as to text-capable cellular telephones (Short Messaging Service based on the GSM standard). Email can also be transformed to fax messages and sent to the most likely location of the user, be it at home or in the office. Furthermore, AM can call up wired or cellular phones and read the email message using a text-to-speech module, or leave messages on answering machines and voice mail systems. Additionally, a user can read email and listen to voice mail from Phoneshell [11], the Speech Group's phone interface to email and voice mail.

3.1 Channels

One of the challenges of Active Messenger is its need to support a variety of channels with different characteristics (Table 1). The different characteristics of the devices and the networks that communicate with them require varying message handling strategies on the part of the forwarding agent.

One characteristic is whether the channel is full duplex or not. Some paging systems are do not support any reply or acknowledgement mechanism; old style text pagers are one example, the Iridium™ worldwide system was another, and the voice pagers a third. It is very difficult for AM to know whether a one-way channel has successfully delivered a message.

A second characteristic is buffering. Many channels store undelivered messages and send them when the target device is back in range or powered up. AM would prefer that a channel support minimal buffering. If it sends many messages to a buffered channel that is not sending to the target device, these messages will eventually be delivered, but at that date, they may no longer be relevant, as they user may have read them using some different access method. Unwanted messages are annoying to the recipient, as they may trigger a per-message charge and must be deleted by hand on the target device. So AM in fact does its own buffering, and relies on channel buffering only for short periods after a device has been seen to be active.

A third channel characteristic is clearly the supported medium. Text messages can be sent as text to a pager, converted to image for faxing, or synthesized to speech for a telephone or voice pager. A voice message cannot currently be converted to text; instead we send a text message displaying the caller's number (and name, if we can find it in a personal address book or campus-wide phonebook) and message length.

Fourth, a channel may be asynchronous—messages arrive when sent and generate an alert, such as a pager—or it may be synchronous—messages are delivered when the user polls the network, such as the Palm™ VII or voice mail. AM doesn't need to know this characteristic directly, but it does influence the manner in which it relies on the channel for buffering. Additionally, the synchronous channels will also buffer outgoing messages from the device, which means they tend to arrive all at once when the user “sends” them. This has resulted in some software race conditions that needed to be worked out, but had not been present when only asynchronous devices were used.

Finally, and perhaps most important, the channels reveal varying amounts of information as to the status of a sent message and the state of the device. We were able to modify the in-house paging system to send back notification from the channel

when the device was detected "back in range" of the transmitter, and also ascertain when a message had arrived at the pager. (Since the pager radio is much lower strength than the base station, it is actually possible for the message to arrive at the pager, but for the acknowledgement message to be too weak to be received, however). Many channels (Palm™, Pocketmail™, and SMS messaging) provide zero feedback. We were able to obtain partial feedback from the Skytel™ network by using their web interface, which allows the sender to check message receipt; the cost to AM is that this is a more complicated protocol than simply using SMTP to send a text email.

Active Messenger tries different strategies to compensate for this missing information. The agent watches the user and tries to infer from her behavior if a device may be able to receive messages, if a single message arrived and was read by the user.

Table 1: Characteristics of some communication channels

	Is device two-way?	*Is device buffered?*	*Info about device in range?*	*Info about message received?*	*Info about message read?*
Canard pager [2][3]	Yes	Only for 10 minutes	Only *back in range*	Yes	Possible
SkyTel™ pager	Yes	Yes	No	Yes	No
Short Messaging Service (GSM)	Yes	Yes	No	No	No
Iridium™ pager	No	Yes	No	No	No
Palm7	Yes	Yes	No	No	No
Pocketmail	Yes	Yes	No	No	No
Fax machine	Yes	Yes	Yes	Yes	No
Playing message to Voice Pager	No	Yes	No	No	No
Playing message to phone	Yes	Yes (answering machine)	Yes	Yes	No
Playing message to cellular phone	Yes	Yes (voice mail)	Yes	Yes	Yes
Phoneshell [11]	Yes	Yes	Yes	Yes	Yes
UNIX mail spool file	Yes	Yes	Yes	Yes	Yes

New channels can be added easily to AM by adding subroutines that are specific to the new channel. In many cases, existing device driver subroutines can be recycled and adapted easily by an average programmer.

3.2 Software Architecture

AM consists of two main programs: the event driven code, and the server process.

Event Driven Code

Whenever a message arrives to the user's mail spool file, this code gets executed, once per message, and then stops. It has the following structure (Fig. 3):

First, the email gets read into memory and parsed. MIME elements like attachments are stripped. Then the user's preference file is loaded to determine if the message should be ignored. Next, CLUES generates the message category. Finally, AM tries to determine if the new message is part of a thread by comparing it with earlier message subject lines and addresses. All this information is stored in files.

If the message originates from the user's mobile devices, AM scans it for information requests from the user. E.g., by typing "wx bos," the user can request the weather forecasts for Boston. The most generic message though is a "reply," generated using the pager's "reply" mechanism. Back on the subscriber's computer (or rather, that computer which received the message), the sequence number is detected in the pager's reply, and the subscriber's message is re-packaged with a return address that is the subscriber's normal email address, an appropriate subject line, and the original message attached.

Server Process

The server process—one instance per user—has the following structure (Fig. 4): After initialization, it goes into a loop where a sequence of modules is executed sequentially.

First, AM checks if new messages have arrived. If there are new messages, they are loaded into the main data structure that keeps track of all messages and events per message. Immediately after loading a new message, a first event is scheduled for this message: Where and when has this message to be sent to, if at all? The events are stored in the main data table.

Then AM tries to determine the current location of the user by using the Unix "finger" command, caller ID information from Phoneshell [11], as well as information from Activity Server [7], a proprietary location server developed by the Speech Interface Group. In addition to the Unix "finger" information, this system collects location information by monitoring the user's telephone lines, as well as from a network of active IR badges.

In the next step, AM checks the message read status of all messages, going through the list of messages: What is the likelihood that each message is read? It does that by checking the user's mail spool file, web pages of external channels (Canard, Skytel™), and other resources. After that, AM checks the status of all communication channels: Did a mobile device register and come back in range? If so, is it necessary to resend messages?

Then AM goes through the message list and checks if events are due. If so, the messages get sent to the specified channels, using channel specific device drivers.

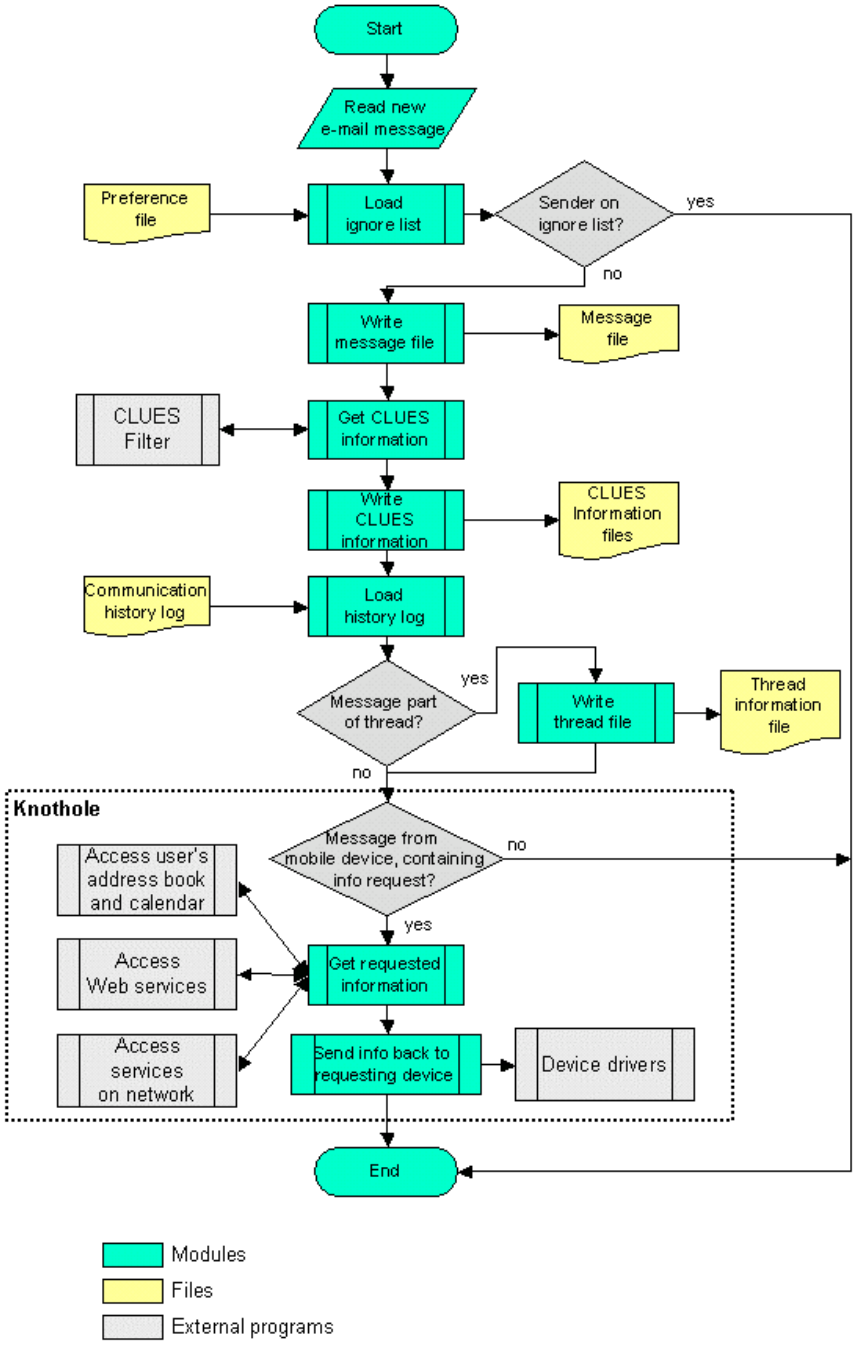

Fig. 3. Event driven AM code

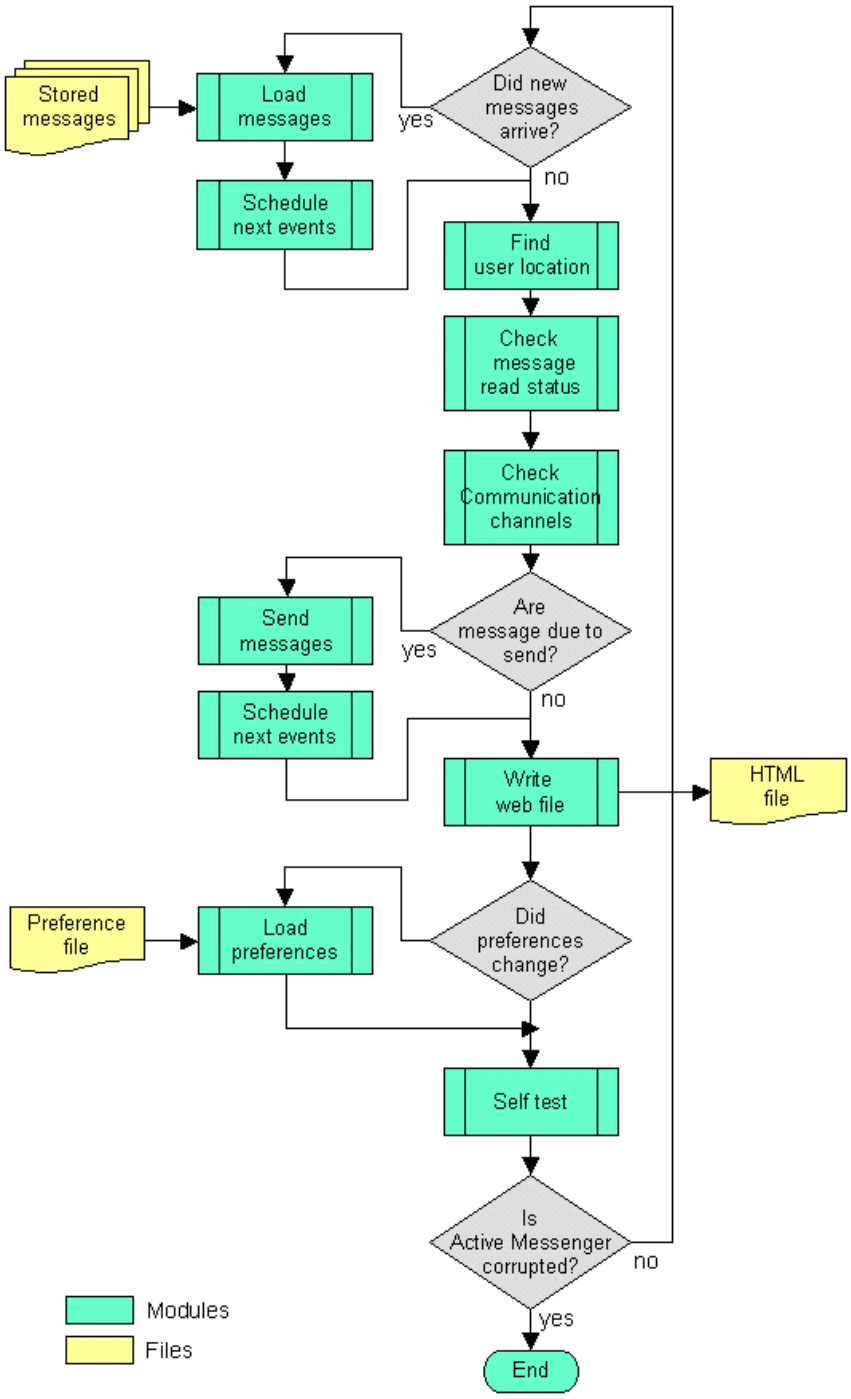

Fig. 4: AM server process

After a message is sent, AM immediately schedules the next event for this message, taking in account the user's preferences, her current location, the current time, the current status of the channels, etc. After that, AM stores all content of its data structures to a web page so that the activities of the agent can be monitored. If the user preference file has changed, it also reloads all users preferences. Like that, changes can be made to the behavior of the agent without restarting the program.

By editing the user preference file, the user can change most of the internal variables. Eventually, AM conducts a self-check and determines if it has to be restarted.

4 Why is this a difficult problem?

From a user's perspective, the hard part of this problem is ensuring that desired messages are received in a timely manner minimizing annoying device behaviors. From a system's perspective, the challenge is monitoring a number of asynchronous processes and channels and inferring message delivery status from channels that may reveal minimal information. From a user interface perspective, the challenge is to exhibit the right behavior at the right time, presenting messages in a manner appropriate to the characteristics of the channel.

As explained earlier, AM actually runs as two components, one triggered by an incoming message and the other running continuously to monitor message traffic channel events. The first is simpler, but is prone to many race conditions, since multiple messages can arrive in quick succession, and some databases must be shared among all processes, and locked accordingly. We also need to avoid using temporary files without assigning each a name unique to the file system, lest multiple processes write to the same file out of order.

For the server process, first and foremost is that this process needs to be highly reliable. In fact, a shell process spawns and monitors the Active Messenger; AM performs periodic internal consistency checks, e.g. to guard against memory leaks, and self-terminates if its operation is questionable. Another aspect to this problem is that AM connects to a variety of services using protocols such as http and telnet; these connections may time out, or the remote server may be very slow, and AM must handle these service failures gracefully.

Finally, it is difficult to keep even dedicated users satisfied when they are on the road. Users struggle with marginally reliable communication channels and are often connecting while busy doing other tasks, or late at night when they would rather be sleeping. In these times of distraction, lapses in performance are very frustrating. For example, from time to time AM buffering has had bugs and multiple copies of a single or, worse, batch of messages gets sent. This results in many irrelevant interrupts, or in large message upload time over synchronous channels. It is also difficult while on the road for a user to check the status of AM or restart it, so we quickly get frustrated. This has made reliability a higher priority than new features.

5 Summary/Evaluation

Two people have used Active Messenger continuously for approximately two years and find it an essential part of their everyday communication infrastructure. The two users have different filtering settings. User A, who gets on average 53 messages per day, lets the agent process almost 90 percent of these messages. User B gets on average 132 messages per day and lets the agent process 38 percent of them. Processing nearly 40000 messages per user over a duration of two years puts a high responsibility on the agent. These two users have depended on AM for over a year and have found that it significantly impacts their communication habits; they grow very frustrated when AM is not operational. While this endorsement is of limited value as these users are the system designers, AM must be providing value if they rely on it 24/7 for handling their mail in work and social environments. As with any emerging technology, it is hard to appreciate the value of a new service; the authors have relied on this system since 1999 and find it hard to imagine life without it.

This project has evolved over several years, and the evolution shows the development of features that were desired by its users; even a small user base can reveal significant differences between message handling approaches.

The predecessors of Active Messenger relied on the telephone for message delivery and spawned CLUES because a non-visual user interface does not support message browsing well. Although highly filtered phone access is quite powerful, the asynchronous nature of even one-way text pagers vastly enhanced the communication potential of a rapid exchange of messages.

Since this was before GSM service (with SMS messaging) was available in the US, we had immediate need for multiple devices; asynchronous delivery was very difficult to give up once users had "tasted" it.

As wireless networks proliferated additional paging options became available, with various costs and coverage areas. The immediate precursor to AM was a scheme whereby a user could specify, by sending a text message or choosing from a telephone-based voice menu, which device should receive messages. A per-device set of filters was also invoked. Additionally, as Knothole was developed, one of the first features it supported was "message summary" and delivery of relevant messages that had been missed while using a more restrictive device. This led to a flurry of intentional activity when entering or leave transmission range to make sure forwarding was set up correctly and to retransmit missed messages. The switch to automatic network sensing, and prompt delivery of previously unsent messages when back in range, were very powerful and almost immediately became "how did we live without this?" features.

Managing message threads has been a relatively recent feature, occasioned by user frustration on the road. Once we began to accept AM as a reliable system, it was increasingly assumed that if an expected message did not arrive, it had not been sent. But without thread management, a message sent from a "high priority" device would not receive the response; we quickly forgot this while busy and on the road and would simply assume that we would get the reply. So threading was added.

The explanation facility has been valuable mostly for debugging and understanding system behavior, but also, combined with the web page, provides reassurance that AM is operating as expected, or at least has a good excuse for a sometimes surprising

message delivery. It has also influenced user behavior in some ways. For example, if I send a message with "hi" in the subject, CLUES will flag all incoming messages with the same or similar subject as a timely possible reply. This has fostered a move away from such generic subject lines.

6 Acknowledgements

We would like to thank the members of the Media Lab Speech Interface group who helped during the design and evaluation phase of this project: Sean Wheeler, Keith Emnett, Natalia Marmasse, Nitin, Sawhney, Kwan Lee; Allen Milewski and Walter Bender for reading early drafts of this paper; Pascal Chesnais and Joshua Randall for helping with Canard issues.

References

1. Appenzeller, G., Lai, K., Maniatis, P., Roussopoulos, M., Swierk, E., Zhao, X., Baker, M.: *The Mobile People Architecture*. Technical Report CSL-TR-99-777, Computer Systems Laboratory, Stanford University (1999). http://gunpowder.stanford.edu/~laik/projects/mpa/publications/TechReport/html/TechReport.html (last visited 2002, February 18)
2. Chesnais, P. R.: Canard: A framework for community messaging. In *First International Symposium on Wearable Computers*, Cambridge, Massachusetts (1997) 108-115. http://www.cothink.com/Papers/canardwearables.pdf (last visited 2002, February 18)
3. Chesnais, P. R.: *A Framework for Designing Constructionist Approaches to Community-Centered Messaging*. Ph.D. thesis, Massachusetts Institute of Technology (1999)
4. Harmer, J.: *The OnTheMove project*. BT Laboratories, Martlesham Heath, Ipswich, England (1998).
5. Horvitz, E., Jacobs, A., Hovel, D.: *Attention-Sensitive Alerting*. Proceedings of UAI '99, Conference on Uncertainty and Artificial Intelligence (1999) 305-313. ftp://ftp.research.microsoft.com/pub/ejh/priorities.pdf (last visited 2002, February 18)
6. *Knothole homepage* [WWW Document]. http://www.media.mit.edu/~stefanm/pager/ (last visited 2002, February 18)
7. Manandhar, S.: *Activity Server: You Can Run But You Can't Hide*. Proceedings of the 1991 USENIX Conference, Nashville, TN (1991) 299-312.
8. Marti, S.J.W.: *Active Messenger: Email Filtering and Mobile Delivery*. Master's thesis, Massachusetts Institute of Technology (1999). http://www.media.mit.edu/~stefanm/thesis/am.html (last visited 2002, February 18)
9. Marx, M., Schmandt, C.: *CLUES: Dynamic Personalized Message Filtering*. Proceedings of CSCW '96 (1996) 113-121. http://www.media.mit.edu/speech/papers/1996/marx_CSCW96_clues.pdf (last visited 2002, February 18)
10. Roussopoulos, M., Maniatis, P., Swierk, E., Lai, K., Appenzeller, G., Baker, M.: *Person-Level Routing in the Mobile People Architecture*. To appear in Proceedings of the USENIX Symposium on Internet Technologies and Systems (1999). http://mosquitonet.Stanford.edu/publications/USITS1999/USITS1999.html (last visited 2002, February 18)

11. Schmandt, C.: Phoneshell: The Telephone as a Computer Terminal. *Proceedings of ACM Multimedia '93* (1993) 373-382. http://www.media.mit.edu/speech/papers/1993/schmandt_ACM93_phoneshell.pdf (last visited 2002, February 18)
12. Schmandt, C., Marmasse, N., Marti, S., Sawhney, N., Wheeler, S.: Everywhere Messaging. *IBM Systems Journal*, Vol. 39, Nos. 3&4 (2000) 660-677. http://www.research.ibm.com/journal/sj/393/part1/schmandt.pdf (last visited 2002, February 18)
13. *The Mobile People Architecture homepage* [WWW Document]. http://mpa.stanford.edu/ (last visited 2001, April 19)
14. *Procmail homepage* [WWW Document]. http://www.procmail.org/ (last visited 2002, February 18)

February 17, 2002