Active Messenger: Email Filtering and Delivery in a Heterogeneous Network

Chris Schmandt

MIT Media Lab

Stefan Marti

MIT Media Lab

RUNNING HEAD: Active Messenger

Corresponding Author's Contact Information:

Chris Schmandt
MIT Media Lab, E15-368a
20 Ames Street
Cambridge MA, 02139, U.S.A.

Phone: (617) 253-5156
Email: geek@media.mit.edu.

Stefan Marti
MIT Media Lab, E15-384C
20 Ames Street
Cambridge MA, 02139, U.S.A.

Phone: (617) 253-8026
Email: stefanm@media.mit.edu.

Brief Authors' Biographies:

Chris Schmandt is a computer scientist with an interest in mobile computing and mediated communication; he is the head of the Speech Interface Group at the MIT Media Lab. **Stefan Marti** is a research assistant with an interest in telecommunication intermediaries; he is a PhD candidate in the Speech Interface Group at the MIT Media Lab.

ABSTRACT

Active Messenger (AM) is a software agent that dynamically filters and routes email to a variety of wired and wireless delivery channels, monitoring a message's progress through various channels over time. Its goal is to ensure that desired messages always reach the subscriber, while decreasing message volume when the user is less reachable through location awareness. AM acts as a proxy, hiding the identity of the multiple device addresses at which the subscriber may be found and caches channels to guarantee seamless information delivery in a heterogeneous network. Our previous experience with mobile messaging influenced the initial requirements and design of AM. We describe the operation and evolution of AM to meet changing user needs, and how our own communication patterns and expectations have changed as we relied increasingly on mobile delivery.

CONTENTS

1. INTRODUCTION

Active Messenger (AM) is a system that prioritizes email messages and delivers them 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 in a location-sensitive manner. More than just a static message router, AM waits for an active user to read messages on screen, and may attempt to reach a series of devices over time, or to resend messages when a device comes back into range. AM uses sets of explicit and context-sensitive filtering rules, based on a user's recent correspondence, calendar, and location, and transcodes messages to fit different display characteristics of mobile text-based devices, as well as for faxing or speech synthesis for voice delivery.

AM is a two-way system, acting as a proxy for messages or replies from the mobile devices, rewriting them to appear to originate from the user's canonical email address. AM tracks message threads on a per-device basis, and can explain its behavior in handling a particular message. Short structured messages access and modify personal information, such as address book and calendar, and allow access to limited web-based sources.

We built AM because email is an essential communication channel for both of us, but we also need to work outside the office and need almost permanent access to our email. Our experiences are similar to the mobile professional workers studied by Perry, O'Hara, Sellen, Harper, & Brown (2001). This study found that remote awareness and access to colleagues was very important, but available intermittently via email (on laptops) or mobile phones. It was important for these workers to plan ahead so they had needed documents with them, and use dead time effectively. The phone was used in many ways as a "device proxy", to have helpers take dictation, and send email or faxes. AM applies mobile email technology to these problems, allowing closer team awareness through email notification, remote access to many desktop tools and means to obtain forgotten files. Since email is asynchronous, “down time” is excellent for sending from a mobile device.

Accepting mail from colleagues at any time allows the mobile user to be more aware of and active in decision making back at the office. But interruption has great cost. As O'Conaill and Frohlich (1995) found in an observational study, although the interrupted party benefited from the interruption, more than 40% of the time they did not resume the work they were doing prior to it. And as Cutrell, Czerwinski, and Horvitz (2001) found, even when an interruption is ignored, it impairs task performance. But as Hudson, Christensen, Kellogg, and Erickson (2002) point out, the availability problem is a complex tension between wanting to avoid interruption because of distraction from current workflow, and appreciating that at other times the interruption is beneficial.

Mobile email usage is growing fast. The makers of the Blackberry mobile email device have 1.1 million subscribers in the U.S., a number which may double this year (Brady, 2004). Globally, 45.6 billion SMS messages were sent from mobile phones since February 2004[1]. If each message represents a potential interruption in one's

[1] http://www.gsmworld.com/news/statistics/index.shtml

pocket, a system such as AM must have means of filtering messages, possibly being context-aware. For managing messages, AM uses routing rules, a concept going back at least 15 years to the classic *Information Lens* work. As Mackay et al. (1989) found in a field study, Information Lens users had no trouble writing rules, but different users used rather different sets of filtering rules. Other work shows that email users have very different habits, as Whittaker and Sidner found (1996) in a study of mail folder use, a topic related to routing. Terveen and Murray (1996) offered some means to assist users writing rules for their agents; we, too use a rule-based approach with different means of assisting the rule-writing. An alternative to end-user programming via rules is to train an agent by example, as was done by Boone (1998); when routing is conditioned on status of various devices, user location and activity level, message sender, time of day, and media available, such training becomes awkward, although for simple configurations it may be easier on end users.

This paper describes the operation and evolution of Active Messenger as both a research project as well as a tool in daily use by the authors on all of their email for 5 years. We will go on, in section two, to discuss our own nearly 20 years of work in mobile email. From this, we draw in section three a set of design principles for AM, and describe its operation and evolution. Section four emphasizes lessons from actual use of AM for five years, section five presents related work, and in section six we generalize these experiences while discussing alternative architectures.

2. ORIGINS OF ACTIVE MESSENGER

For twenty years, we have been exploring the delivery of mobile email via different media and technologies and to make “the desktop” available to mobile users. During this time we have never attempted to replace desktop (or laptop) computers, because large screens and full-sized keyboards are very well suited for text-based communication. Rather we have sought to enhance email’s utility by providing notification and access from locations or situations in which it would otherwise be unavailable. We built real systems with real, although highly sympathetic, users in a work environment that saw early adoption of intense email usage; if the system did not perform to the satisfaction of the users, they would not hesitated to abandon its usage. In using and improving these systems, we learned a lot which influenced the design of Active Messenger.

2.1. Telephones

In the mid 1980’s we built early voice user interfaces for the telephone, and what has now become known as “unified messaging” in which all message media are available by either screen or telephone; this includes hearing email by text-to-speech synthesis (Schmandt & Arons, 1984). In the early 1990’s we combined listening to email with remote access to ordinary desktop computing tools. This led to *Phoneshell* (Schmandt, 1993), a telephone-based system using text-to-speech to read email and access an address book, calendar, laboratory-wide dial-by-name service, and news, weather, stock quotes, and traffic, in addition to the expected voice mail. Phoneshell was used extensively by a small audience at MIT and Sun Microsystems, where it was made to use Sun’s calendar management tools. Phoneshell has been continually

operational since 1989, with a maximum of 10 users at any time, and three power users for over five years each.

Phoneshell users can initiate messages or compose responses (using the telephone keypad, two key presses per letter) and forward messages as well; replies come from the user's ordinary email address. Initially, sending email from a phone was such a novelty that Phoneshell automatically added a footer about how the message had been composed. Over time, though, we found that we often did not want to reveal that we were not "hard at work in the office" and went to some pains to program Phoneshell to perform capitalization of the reply message so it looked normal. Although speech-based user interfaces were also built (Marx & Schmandt, 1996; Yankelovich, Levow, & Marx, 1995), it is the reliable and less resource intensive touch-tone Phoneshell that remains in use. In addition to being a platform for voice user interface experimentation, Phoneshell evolved in two directions.

2.2. Filtering

Listening to messages is slow, so Phoneshell users constructed static procmail-like rules[2] to sort messages into categories; within each category messages were presented in first-in first-out order. As daily mail volume increased steadily during the 1990s, filtering became even more essential. These rules adequately capture static or slowly varying information, such as family and co-workers, but very mobile users benefit from more dynamic contextual information.

The CLUES filtering agent (Marx & Schmandt, 1996) was built in the mid 1990's to meet this need. On an hourly basis, CLUES creates a set of rules to specify "timely" messages, by 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 a co-author if the user had sent him a message yesterday. If a traveler provides a contact phone number, or just an area code, via calendar, CLUES associates emails with location via the address book, and marks as "timely" messages from senders in that region. Because Phoneshell supports dialing by name, our home-built voice mail system logs caller ID, and the address book maps email addresses to phone numbers, CLUES is also able to detect communication threads that pass between voice and text messaging.

Although simple, CLUES is effective. In a simple example, author A was able to respond to email from a major sponsor in the Bay Area with "are you free for lunch in 15 minutes?"; the sponsor accepted and was please by the prompt response. AM knew its user was in the Bay Area (from his calendar) and used his address book to find the email sender's telephone number, which matched that location. A more sophisticated example came on the same trip when AM delivered a message about "A/V requests for your talk tomorrow" from an otherwise unknown person at PARC. AM did not know that person, but her email address looked similar to several other entries in the address book, all with Bay Area phone numbers.

[2] http://www.procmail.org/

2.3. Facsimile

Almost from the beginning, Phoneshell faced the dilemma of *polling* versus *asynchronous delivery*. Although it worked from any telephone in the world (we carried pocket touch tone generators to Europe), the user was required to place a call to find out if there was mail waiting. The first work around to this problem involved faxing, which gradually became an alternate delivery mechanism. Some users configured automatic faxing of summaries of new emails, the day's calendar and weather forecasts, and news summaries to their homes or, when on the road, hotels; this allowed a morning "catch up" without having to place any call. Faxing provided notification, but without mobility. Faxing became an option to most Phoneshell commands; this was useful for sharing information or reading long messages. For example, when one of us was to be joined by family mid-way through a trip some time zones away, he alerted them to unusually cold weather by having Phoneshell fax home a forecast in the middle of the night.

2.4. Pagers/SMS

Asynchronous textual notification was partly solved in the late 1990's with the appearance of the first alphanumeric pagers with email gateways. Early pagers were receive-only with only regional coverage, but became popular immediately; notification of new messages increased the recipient's responsiveness dramatically. When email arrived at the computer, a copy was forwarded to an agent that executed a procmail rule set (updated every hour by CLUES), to decide whether to forward to the pager. Phoneshell also allowed control over paging, as pagers were very regional at the time. When author A had one pager for the Bay Area and one for Boston, switching was added to Phoneshell, and this was the first motivation for AM.

By shortly thereafter we had a plethora of paging devices, each with different coverage and usage charges, and soon some pagers were two-way. At the same time GSM mobile phones with SMS (Short Message Service) became available in the U.S., providing similar text messaging. While on campus, many messages were sent through *Canard* (Chesnais, 1997), a system using Motorola equipment and a roof-top transmitter; while at work it was desirable to be reachable. Off campus options depended on where one lived. We soon had multiple procmail rule sets for each device and would manually switch them, either from a pager or over Phoneshell, when leaving and returning to work; of course the problem was remembering to do so. We also missed the calendar and address book access provided by Phoneshell, and so provided this using structured messages. For example, the sending the message **rolo Curly Howard** would return Curly's address book to the pager, and **cal tomorrow** would page with tomorrow's calendar entries; these were implemented in a utility called Knothole[3].

The multiplicity of devices, different forwarding rules depending on the device, and change of access modality during the day and the week led to Active Messenger. AM was designed to automate distribution of messages to various devices in an extensible manner, while allowing the widest range of devices possible, including pagers, phone, and fax. AM was required to be always running, monitoring message access and wireless device availability, with caching and retransmission as necessary.

[3] http://web.media.mit.edu/~stefanm/pager/

This is much more complex than Phoneshell, which merely had to parse the mail spool file once per call to determine which messages were new at that time.

3. ACTIVE MESSENGER

By the time we built Active Messenger, we had seven years' of operational experience with mobile messaging via Phoneshell, and more than a year of experimenting with email on pagers, with half a dozen users at a time on each system. Author A and several other users traveled extensively and used these techniques as their sole access to email (this was before the days of the web, cyber-cafes, and WiFi hot spots). Based on this experience, and the needs of and the feedback from these users, we identified a number of design principles which AM had to support:

1. **Filtering of messages to the mobile device is essential.** Interruption in one's pocket with every email is unthinkable. AM uses CLUES and a blacklist.
2. **There is no point reading email if you cannot reply to it,** this is only frustrating.
3. **Device addresses are private; email addresses are public.** In trade for always being connected, users need absolute control over privacy. AM acts as a proxy, hiding device addresses in all correspondence.
4. **Remote access to desktop utilities such as calendar is useful.** This allows these databases to be maintained on a computer with a good user interface, and supports sharing. AM supports access through structured email messages.
5. **It is hard to remember long email addresses and awkward to enter them on handheld devices.** AM uses address books and mail alias files to allow sending messages without recalling the full address of correspondents.
6. **"Real computers" are always preferable for reading email when at hand.** Pagers, PDAs, and phones are simply too small with poor text input. AM waits to deliver mail to a device if a user may read it on screen.
7. **Users forget to activate modes.** We forget to put our phone into vibration mode until it rings; AM determines device modality from ordinary use.
8. **Forwarding, reply to sender, and reply to all recipients are essential.** Though simple commands, these are powerful actions in email communication.
9. **Reliability is crucial.** Mobile users cannot check the status of running processes. This affects mainly the software architecture of AM.
10. **Command syntax and user interfaces must be simple.** Mobile users are distracted and busy. AM tries to "keep it simple."

3.1. AM architecture

Active Messenger was meant to co-exist with Phoneshell as a separate, parallel delivery mechanism; reliability was extremely important in Phoneshell and AM development should not impact that. AM is implemented as two large PERL scripts: the main AM server that runs continuously, and an event driven process that executes when a new message arrives. The event driven process analyzes each message and extracts the user's commands, if any. Such embedded commands trigger subroutines that get information from the web (e.g., weather forecasts), local resources (e.g.,

dictionary lookup), or from the user's personal information (e.g., her calendar, rolodex, etc.), and sends it back. For ordinary incoming email, this process uses CLUES to determine its priority, and unless it is to be ignored, creates a new database entry for the message; it does not immediately forward it. This process also tracks message threads and monitors device activity.

The AM server tracks each message: when it arrived, when it was sent to which device, an estimate if the user read the message, etc. The server periodically parses the user's mail spool file and extracts the status of each message; it also parses several web pages to assess the status of external systems, mapping the information provided to its internal message table. It also tries to determine the user's location by using "finger" information from the most commonly used mail servers, as well as caller ID from Phoneshell. In separate data structures, it also keeps track of when the last message was sent to a device, when the user replied using this device, as well as other parameters that allow it to assess the user's activities in order to send or resend a specific message at the best time to the most appropriate channel.

AM is thus invoked with the arrival of each email (with consideration of race conditions if many message arrive), and knows that a message was received on a device when it receives any other message from that device (device radios are less powerful than cell base stations, so this a safe assumption). It determines whether a user has read a message on screen by examining the user's mailbox and parsing its messages (or via IMAP). Relying on principle 6, AM never deletes or modifies the mailbox; we assume that users do maintenance on a real screen. Once a message is known to have been read, AM is finished with delivery, although it may be called upon to access the message in formatting any reply from the user.

3.2. Filtering and delivery

AM relies on several sources to classify messages. Users specify rules linking particular kinds of messages to user-defined categories, using a modified version of the public domain procmail syntax of Unix regular expressions. 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." AM also uses CLUES to detect "timely" messages. Users indicate the ordering of message priorities, including ordering of the "timely" category. A message is evaluated for timeliness when it arrives, and is assigned to the highest matching priority. If a rule identifies messages from a boss as "very important" and this category is ranked higher than timeliness, a message from a boss about something in tomorrow's calendar is "very important," not "timely."

AM uses the user-defined ordering to determine how hard to attempt to deliver a message. In addition to specifying filter categories and ordering them, the user must indicate which categories should be sent to which devices; this is defined by a text configuration file. Combined with geographic or situational (when a device is active) locality, this mapping allows AM to throttle message delivery when a user is in a less accessible mode. Users also specify how long to wait for a response at each step.

When a user is known to be online and active, there is no immediate reason to deliver any messages as reading on a big screen is preferred, so AM simply observes the progress of the message through whatever mail reader the user employs. If an

important message remains unread for some time, it is then forwarded. Since Canard worked only within a few kilometers of campus, and author A lives out of that range, it was safe to assume that he was “at work” when in range, and a large number of messages were sent to that device. When further away and using a more expensive service (such as Skytel or Iridium) he limited his messages to only the highest priorities. 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.

AM takes a number of timed steps after a message arrives and is sorted. The following example (Figure 1) shows what happens when a new email message arrives at a user’s inbox—let’s call her Clara. Clara has the following lines in her preference file:

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

This describes the channel sequences for important and very important messages. If a message is “important”, it will be sent to the Canard pager. However, if the message is “very important”, it gets sent to a phone, and then to a fax, after Canard. The bracketed numbers specify delay in minutes until a device or channel is used. Let’s assume furthermore that Clara is currently at home and has the following entries in her preference file. She has the channels **canard**, **phone**, and **fax** available. For each channel, a number or address is specified, and the time when it is OK to use the device, which is different for each location:

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

Work
canard  = clara@canard.mit.edu, anytime
phone   = 342-4545, not M-F 19-9, not SU
fax     = 342-5463, anytime
```

A message arrives at 6:57am. The channel sequence specifies the first channel as **canard** in **20 minutes**. However, this initial delay is scaled down inversely proportional to the user's idle time: if the user is idle for more than an hour, the message gets sent immediately. Clara has been sleeping, not logged in, so AM goes to the next step immediately. 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 sends it to the Canard pager. Right after that, the agent tries to schedule the next event, which would be **phone**. The phone call would be in **14 minutes** at 7:11am, but Clara does not allow AM to **phone** at home from Monday through Friday between 10pm and 8am. This channel is currently not available, and the agent skips it. The next entry is **fax**. The delay for sending faxes is specified as **35 minutes**, so Active Messenger schedules a fax for 7:32am, then waits. Clara happens to log in to her computer and read this email message at 7:25am. The “message read” level rises over the threshold, so AM cancels the fax. Note that if Clara read the message on her Canard PDA, and replied, this would have also cancelled the fax.

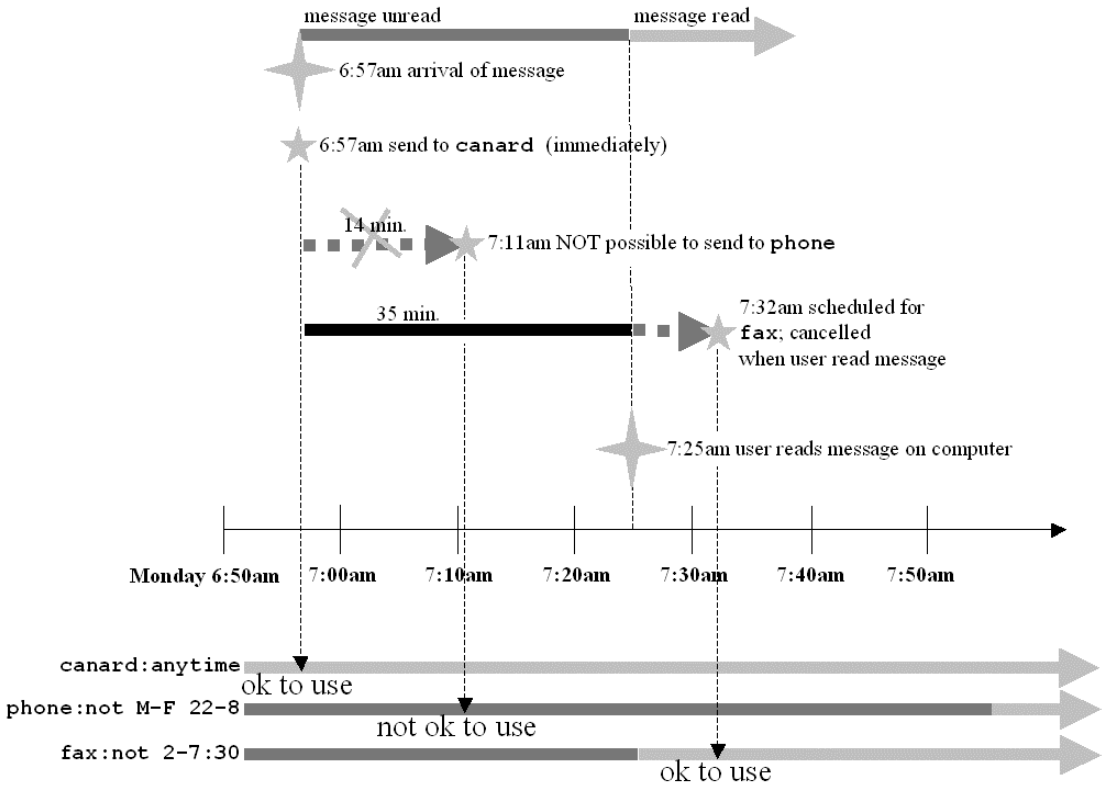


Figure 1: Channel sequence example

The combination of message priority and the user's location causes the message to be routed in different ways. If this message had been classified as just "important" instead of "very important," then AM would have sent it only to Canard, and then terminated. If the "very important" message had been sent much later, e.g., around noon, and if Clara had been reading her email from her computer at work half an hour previous, AM would have scaled down the initial delay for sending it to Canard from 10 to 5 minutes. After 14 minutes waiting—Clara is in her office talking to a visitor—AM would have called her, but this time on her office phone, and delivered the message by reading it to her with text-to-speech.

3.3. Device handoff and intermittent connectivity

AM has operated with a variety of devices or "channels." Over time we have changed devices, requiring new channels for new hardware; some channels have only limited RF coverage and are used occasionally, and some channels support different media. As Figure 2 shows, there are many details of network support and channel characteristics which AM has been required to manage, with programming required for new ones. 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 whether a login session is local or remote; if logged in, the user may read the mail so AM delays longer before sending it to the first in the series of devices. Similarly, if the user phones in to access voice mail or hear email, caller ID may indicate whether she is at home, or a geographic location (area code).

	Is device two-way?	*Is device buffered?*	*Info about device in range?*	*Info about message received?*	*Info about message read?*
Canard pager	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
Wireless Palm[4]	Yes	Yes	No	No	No
Pocketmail	Yes	Yes	No	No	No
Sidekick	Yes	Yes	No	Yes	Yes
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	Yes	Yes	Yes	Yes	Yes
UNIX mail spool file	Yes	Yes	Yes	Yes	Yes

Figure 2: Characteristics of some communication channels

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 many 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. This was not part of the initial AM design but was added later because we found it highly distracting to receive these now unwanted messages. For wireless devices, the device would keep beeping as each message arrived, and then some devices require the user to manually delete each. It can also become expensive. For Pocketmail[5], it means a long download delay to get past stale messages to new ones; this is also expensive when calling from overseas. That this is so annoying helps reinforce our first design principle: minimize the number of messages sent to mobile users.

If AM is not certain of a device's availability, it forwards a few messages as a probe, and then caches the rest. 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. At worst, when a user turns on a device, he can

[4] http://www.palmone.com/us/support/i705/

[5] http://www.pocketmail.com/: essentially a modem in a PDA, but not a terminal emulator. Mail sent to a provided email address is retrieved by dialing a toll-free number.

activate it by sending any message. 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. Although CLUES detects threads, it is not triggered to rewrite its filter rules every time a message is sent. In any case, the device from which the message was sent my not be configured to receive such “timely” messages, as CLUES would classify them. So AM also tracks message “threads,” specifically in response to messages sent from each device, and any replies are sent to this device immediately, then to the chain of devices. Rejected mail is handled in the same way. Again, this was not in the original design, but was added after one user, lost on the Georgia Tech campus, was unable to rendezvous with a colleague because responses to his “help!” message were delayed.

3.4. Role as proxy mailer and PDA

Although most AM devices are email-addressable, users do not wish to reveal their addresses, for a variety of reasons. Different devices are in use at different times, so a message sent to any one may not be delivered for some time. We acquire new devices from time to time and it is a bother to inform all our correspondents. Most important, we wish to keep the addresses of these devices secret and rely on filtering agents to control which messages are delivered to them. Similarly, if our correspondents use filters, they may not recognize the addresses of devices as “ours”; when author B sends pictures from his camera-phone, many are never viewed for fear of viruses because of the phone’s unrecognizable email address.

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. Reply messages from the device are also 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 from our normal email system. Although in practice this is almost always correct, there may be exceptions. Author A’s wife is rather negative about this use of a PDA for email at home (though is happy when it reveals relevant news from members of either family). Yet, she is his highest priority correspondent and he usually replies to her messages very quickly, though often is out of his office. Perhaps she would be more tolerant of AM if it added to those messages a footer saying “He must really love you since he used Graffiti to send you this message!” This comment is made somewhat seriously.

Although some AM devices are wireless PDA’s, many are not. One advantage of a PDA is its calendar, address book, etc, although it is often difficult to migrate these to new devices, and they cannot be shared. We incorporated these features into our mobile messaging architecture; an AM user may access and modify his address book, calendar, and to-do list. 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 data (*stdout* and *stderr*) is sent to the pager as a response. This additional functionality has another desirable side effect: increased network traffic. The more often any device is used, the more accurately AM can infer which device is currently active and hence should receive urgent messages.

3.5. Rules and User Interface

Active Messenger itself has only a minimal user interface; mobile users' direct experience is with whatever mobile device they are using. It would be highly desirable if users could configure these devices to be aware of AM's message classification and, for example, alert in different manners. High priority messages could cause a more audible alert, an interruption less easily ignored, while less important messages may only turn on a "new mail" indicator LED. While we used this approach quite effectively in *Nomadic Radio* (Sawhney & Schmandt, 2000), that project ran only on a local WiFi network. AM relies on proprietary devices on commercial networks and so tolerates whatever user interface (and device communication characteristics or latency) those provide.

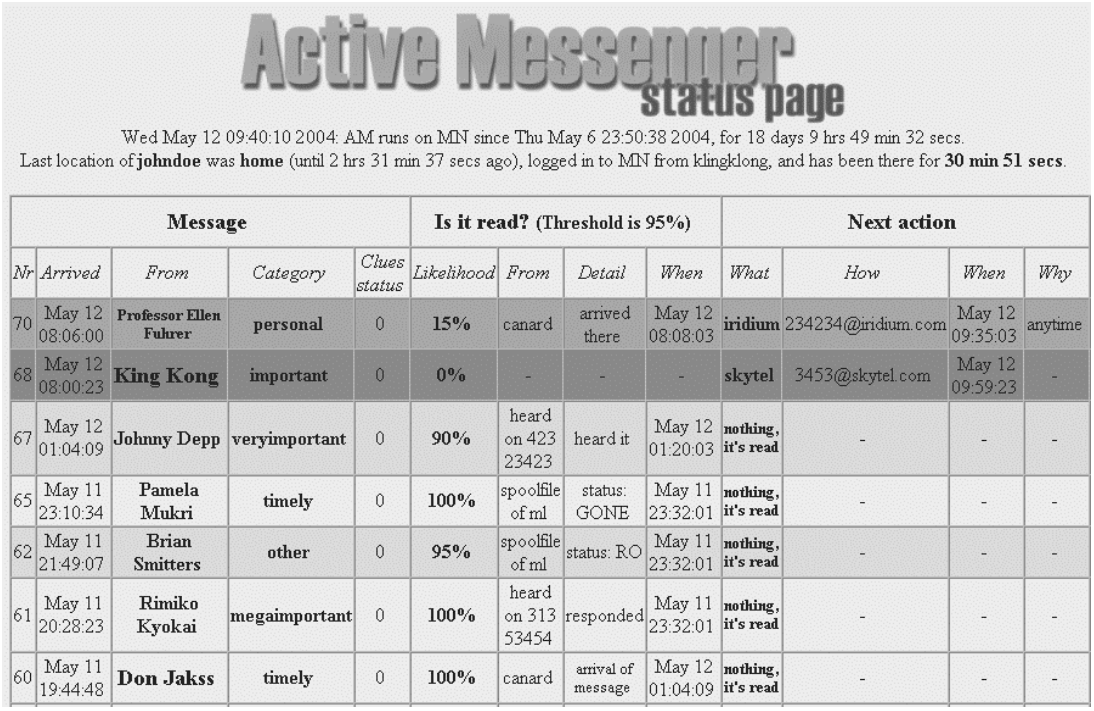
AM users do have significant control over how AM processes their messages. As seen in section 3.2, configuration files specify routing of different message classes, the delay between each device (a zero delay sends a message to multiple devices simultaneously), locations and their associated data, and time-of-day constraints. A user may also specify messages that AM should ignore; author A uses this for three high traffic mailing lists which he wishes to review only when convenient. Text files are also used for CLUES rules, which are regular expressions. For example, the following rules define messages from someone who logs in as "krosenfield" or which have the word "urgent" in the subject line to be of class "veryimportant" while messages from anyone at domain "irs.gov" is of class "important":

```
^From.*krosenfield@*
| set veryimportant

^Subject.*urgent
| set veryimportant

^From.*@irs.gov
| set important
```

Those familiar with programming in Unix generally know regular expressions, but this is the most difficult aspect of using Active Messenger, and new users typically take someone else's CLUES rules and modify them. Phoneshell can automatically author a rule on request after speaking a message (AM uses Phoneshell to deliver text messages over the phone); the user chooses a rule based on the author of the email, the Internet domain from which the email arrived, or its subject line. A command-line simulator shows how a message will be classified. AM also presents a continuously updated web page with the status and disposition of recent messages (Figure 3). Although meant as a diagnostic tool, this page reveals message classifications and is an excellent summary of the current state of one's inbox. If there are no highlighted messages (shades of red and pink indicate the priority of messages) then there is nothing urgent, and the authors increasingly refer to this page as a quick mail scan (similar activity was reported by the author of CLUES, who preferred to read his mail



Message					Is it read? (Threshold is 95%)				Next action			
Nr	*Arrived*	*From*	*Category*	*Clues status*	*Likelihood*	*From*	*Detail*	*When*	*What*	*How*	*When*	*Why*
70	May 12 08:06:00	**Professor Ellen Fuhrer**	**personal**	0	**15%**	canard	arrived there	May 12 08:08:03	**iridium**	234234@iridium.com	May 12 09:35:03	anytime
68	May 12 08:00:23	**King Kong**	**important**	0	**0%**	-	-	-	**skytel**	3453@skytel.com	May 12 09:59:23	-
67	May 12 01:04:09	**Johnny Depp**	**veryimportant**	0	**90%**	heard on 423 23423	heard it	May 12 01:20:03	**nothing, it's read**	-	-	-
65	May 11 23:10:34	**Pamela Mukri**	**timely**	0	**100%**	spoolfile of ml	status: GONE	May 11 23:32:01	**nothing, it's read**	-	-	-
62	May 11 21:49:07	**Brian Smitters**	**other**	0	**95%**	spoolfile of ml	status: RO	May 11 23:32:01	**nothing, it's read**	-	-	-
61	May 11 20:28:23	**Rimiko Kyokai**	**megaimportant**	0	**100%**	heard on 313 53454	responded	May 11 23:32:01	**nothing, it's read**	-	-	-
60	May 11 19:44:48	**Don Jakss**	**timely**	0	**100%**	canard	arrival of message	May 12 01:04:09	**nothing, it's read**	-	-	-

Figure 3: AM status web page: each row in the table corresponds to a message; color shows priority

via a CLUES-sensitive basic web mail client for some years after graduating) (Marx & Schmandt, 1996).

Mobile devices, even with browsers, are not appropriate to view complex pages, so an explanation facility using text or text-to-speech was written by Sean Wheeler. When CLUES generates rules, it also generates an explanation template for each rule. Later, a user who receives a questionable message can send **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” with the template filled in. Users tolerate apparently incorrect behavior when (1) there is a reasonable explanation and (2) the explanation can be delivered promptly; a common situation is an unexpected reply message to a Cc from a similarly Cc’d recipient of a previous message. Additionally, AM users can cancel a rule. This was added after author A was flooded with messages about “how to change the toner cartridge in the HP printer on the 3rd floor” because he was in California visiting HP Labs. Cancellation prevents that rule from firing again, at least for a few days, and avoids similar deluges of unwanted messages.

4. EVALUATION AND EVOLUTION

In previous sections we discussed our prior work with mobile email using various technologies, and how years of using mobile email, albeit with a small user community, established the design principles for AM. We then described the architecture and operation of AM, but in doing so we almost immediately encountered further iterations in the design, which we have tried to note explicitly because they

contain valuable lessons for future developers. In this section we focus on further evolution of AM in response to our own changing needs as users, and also on changes in our own communication behavior enabled by AM. This is hard to measure since AM was only one in a series of projects, making it difficult to attribute all user experiences to a specific change in infrastructure. Another difficulty is that during the life of this project we have seen a general increase in the use of email, more and more personal and mobile communication using email-like messaging, and a significant rise of unsolicited and unwanted email ("spam"). All of these have confounding impacts on the use of the medium, but a change is the shift from use of Phoneshell to AM. Although Phoneshell is still used regularly for functions such as address book lookup, dial by name, weather forecasts while on the road, etc., its email reading features are used only under duress when none of the various wireless PDAs (or Pocketmail with its modem) work. Asynchronous notification plus the superiority of a screen over hearing synthetic speech, despite years of familiarity, are the strong points of some technologies used by AM.

Our observations here are based on a small community of highly motivated users: the two authors for AM experience, and up to about five at any time with previous systems. This number is too small, and the systems were too much under evolution for much of the time, to claim more formal evaluation results. We recognize that formal evaluation is highly desirable, but it is difficult. A several week field-trial of Phoneshell with a major U.S. telecommunication company revealed insight on its user interface, but not on its utility to a mobile work force, because company policy strongly discouraged re-routing of internal email to an outside server, and it was too difficult to recreate our operational environment on their premises. Nonetheless, their liaison to MIT, who visited frequently, became an ardent user, making use of weather forecasts and traffic info to chose mode of ground transport when visiting Cambridge. An AM field trial with a major European telecommunication company required porting our software to run in a Windows Exchange environment, with the work group being studied put on a separate network out of fear by their IT department of interference with corporate email. Unfortunately the whole organization vanished in a re-organization halfway through the port.

These cases show some of the difficulties encountered when trying to modify and evaluate email systems, since email is seen as an essential work tool and part of the core networking infrastructure. It is difficult to experiment with email, and mobile email shows little value until users find they can depend on it under stress. But by the same logic, even extremely sympathetic users would not tolerate a mobile email system that interfered with their work to any serious degree, and even conscientious users abandon unwanted devices after a few weeks or months. So in this section, we mainly self-report.

The two authors have used Active Messenger continuously for five years and find it essential to their everyday communication. Several other users have subscribed to the service more casually and one user accesses the Knothole functions only. AM has processed over 300,000 messages for the two main users, who use different filtering settings. Author B, who gets on average 53 messages per day, lets the agent process almost 90 percent of these messages. Author A gets on average 132 messages per day and lets the agent process 38 percent of them. Processing so many messages in a very email-oriented work culture over a long duration puts a high responsibility on the

agent; 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, yet the authors rely on this system and find it hard to imagine life without it.

4.1. Being always connected

We chose to be connected almost permanently, which explains the high value of AM's services to us. But given the choice, we highly enjoy the ease with which we communicate at a distance and the degree to which we maintain contact with both friends and associates. For neither of us is this communication exclusively or even predominantly work oriented. But our contacts have become used to the fact that we are reliably reachable, and are not surprised by a quick response at any time of day or night. In so doing, our patterns of usage may compromise our privacy slightly, but communication is so clearly beneficial that we gladly do so. It is essential, however, that we have the ability to mask our actual situation by use of the proxy mailer, which hides our device identity and thus reveals fewer of our current details.

AM has enabled the authors to be nearly always connected to their email for a number of years, but we should again emphasize that for each of us, a significant portion of our email is social, and not at all work related. Author A received an expected notification of his mother's death and regularly arranges his father's care in a nursing home through email with its staff, and his first hints of his father's medical crises via AM are notification of voice mail from the nurses (although we do not attempt to transcribe the voice mail, we do send notice with the caller ID and name, if found in his address book). On a happier note, AM allowed this user to send, from the hospital room, the announcement of his son's birth to a pre-arranged list of recipients within a few minutes of the event, and maintains many friendships by email.

Is it obsessive to carry work contact around in one's pocket? It certainly helps group work if a response is necessary to continued progress on a project, or where a quick reply can avoid wasted effort on a problem with a known solution. Getting notifications while mobile can save time; author B was returning from a trip to Europe, prepared to rush into work to a meeting, but when, while still at the airport, he got the email of a schedule change, had time to stop at home to drop off his luggage and change clothes first. This can be useful to discretely catch up on correspondence during meetings in which it would be too disruptive to use a laptop, and AM sometimes makes commute time more productive for one of us who uses public transport from some distance. This advantage depends on AM's proxy software, which properly formats the outgoing email, as mail directly from PDA's is poorly formatted.

Being connected can make one appear to be at work while actually doing something else, and this can be pleasant. Author A, while on a hiking trip, noticed that on a high hill he had network connectivity to the Lab. An email from his superior to all faculty arrived, and, knowing the requested information, he sent it back while eating lunch with a view. Later he was commended for such prompt service, and no one ever needed to know that he hadn't even come into work that day. Being connected in this way can make it easier to take time off work for family matters, and in general makes telecommuting more viable (as might be predicted by Perry et al.) The two authors, in a senior student-advisor relationship, communicate with each

other frequently, at many odd hours. The advisor is not bothered by the fact that his advisee rarely appears in the Lab before lunch, because he often works from home and this is confirmed by his high degree of responsiveness via email.

But the other side of the equation is balancing mobile work time with time off work to be with family, and both authors are able to exploit their almost permanent availability for extended free time. In fact, both authors continue to use AM to monitor messages from work, but by replying to less of them, they gradually decreases their quantity, due to the "timeliness" component of the filters. Since vacation activity for author A's family is largely outdoors, he values AM's ability to deliver detailed weather forecast information. In summer he tracks precipitation back home and controls his irrigation system and thermostat remotely depending on local weather. Even while on vacation it is necessary to respond to family eldercare emergencies. Finally, some vacations have seen email used to arrange rendezvous with local friends or get dining recommendations from them.

There are some advantages to using mobile devices for the work emails instead of laptops while on vacation, in addition to their size. Because it is more difficult to reply, replies are kept terse, and only important messages viewed, leaving more attention for family.

Sometime we better understand our habitual use of technology when we change how we use it. Author A's family has access to a vacation house at a beach resort, where they visit off-season when it is less crowded. Wireless coverage is not available, so author A habitually relied on his relatively slow Pocketmail dialup device to get email as well as weather forecasts while there. While driving from home, wireless coverage would cease en route, but AM would switch to Pocketmail, including resending missed messages, when he first used it.

Recently, it has been necessary for him to carry a laptop to do some writing over the weekend. With the laptop handy, he dialed in and could read all his email, which meant there was less waiting on his return. The laptop could also access weather radar images, telling much more about nearby storm activity than just the text forecast. But now he spent an hour every evening reading email, and both his wife and young child noted this several times. From his point of view, this was precious family time. He resolved this by bringing the laptop when needed, but using the Pocketmail device and relying on AM's filters to allow him to respond to only the most pressing mail.

This extended example reveals the kinds of trade-offs that users of mobile communication must make. Details and decision points will vary widely, but the essence of each decision is the individual's choice between being accessible to people whom he cannot see versus focusing on whatever he is doing at any particular moment.

It was important to learn the social image created by pulling a PDA out of one's pocket and switching attention to it in the middle of a conversation. In anticipation of an important message, it is easy to forget that this is rude behavior and insulting to the other; a word of explanation, however, or waiting a short while for a break in conversation, makes for less intrusive interaction. Another personal habit fostered by AM is less need to remember things; when we think of something, we can

immediately send off a relevant message, or remotely add it to our to-do list, and then forget about it.

4.2. Reliability and redundancy

As with any software agent, the user must have confidence that the system is running, and operating correctly. For Active Messenger this is doubly important because when it matters is precisely when the user has difficulty monitoring it. While in the office, we generally read our messages using whichever screen-based mail reader we each prefer; only when away from the office do we really need AM.

AM puts a sequence number on each message. Thus whenever a message arrives, the user can see if it was out of sequence, and request retransmission. Although such system failures have been minimized due to extensive debugging, it helped boost confidence in earlier days. When switching between equivalent channels (see below) occurs, AM sends a message to the newly activated device, as there may not be a pending message for delivery so the user may not know which device is "active". Similarly when AM restarts from scratch (generally less than once per month) and hence has to rebuild its map of current channel activity, a warning is sent to some devices. A user can help the channel detector by responding with any command, though this is not required.

AM is now highly reliable and running almost continuously, with an uptime of more than 99 percent. Nonetheless, over the years of using it we have only come to more appreciate the need and attractiveness of redundant access methods. If a message is really important, a single channel is not adequate. While traveling, we have used AM's ability to select alternate delivery channels and media many times. We have experienced wireless networks going down, batteries accidentally discharging, wireless services being suspended because a fraud-conscious credit card company rejecting the monthly service fee payment, and losing or forgetting devices. At various times, all the available combinations of access methods through Phoneshell and AM have been used and many of these channels are easily enabled remotely simply by using one to send a message.

4.3. Message size and quantity

The need for user-defined per-device message sizing and limiting the number of messages sent has been apparent from the start. Although each new generation of wireless devices in the U.S. initially includes an "unlimited data" plan, these all revert to a price per kilobyte eventually, and some devices like Pocketmail are slow. AM removes attachments from messages (looking for plain text only), strips most of the header, abbreviates subject lines on reply messages, and also attempts to filter out "included" text and portions of messages, as well as signatures. Sometimes this is incorrect, so a **full** command was added to send the entire plain-text portion; as mobile devices improve it may be useful to send some types of attachments, but with a few exceptions, this is not possible yet. AM also allows a per-device maximum message size. If the message is longer, "[XXX chars]" is appended to the message, so the recipient knows what is left. The **rest** command will deliver the entire remaining unsent portion of the message. This is a rather recent addition that has been most welcome, due to increasing wireless data costs.

Filtering is desirable: in a world of too much email and too many interruptions, we need more control. CLUES plus static filtering are surprisingly good at catching on the order of 95% of the messages that matter. So far we have not used AM to automatically delete mail. However, author A routinely travels a week without tending to the messages not delivered by AM, and employs the same activity on a daily basis with the day's incoming mail. Every few months a message slips through the attentional cracks, and sometimes this causes problems. Although unfortunate, this does not begin to offset the perceived benefits of using AM, and it is also likely that some messages would be missed without AM, due to email overload.

Since filtering is imperfect, and wireless networking is improving, why not just send all email to the mobile device? Using a protocol such as IMAP, AM would send only header information, and let the user select messages to receive in full. This approach is sometimes useful, and we each use this delivery mode on occasion, either while searching for a particular expected message or with some idle time while traveling check messages that fell through the filter. But we certainly do not want this mode at all times, because the excessive interruption very quickly becomes quite annoying. Interruption enables quick response, sometimes leading to semi-synchronous email conversations; when some event sends many messages, we turn off device alerting for our sanity, and check messages in batch mode. But in doing so we lose responsiveness, and if not for responsiveness, much of the motivation for receiving mobile messages is gone; sending messages would still be attractive.

4.4. Parallel sending to equivalent channels

AM was originally written to utilize channels in a sequential manner; as the user became harder to contact, with more expensive channels, fewer messages would arrive. As the constellation of mobile devices we use has changed more recently, some of them are considered interchangeable, or "try in parallel", rather than "try serially." Author A currently has a single preferred wireless device, but regularly visits locations beyond the wireless service area. In this case he switches to the Pocketmail "modem in PDA" device and calls up several times a day to get messages. It is almost never the case that he uses these two devices at the same time (although AM supports it).

So AM was changed such that these are equivalent devices. When a message is received from one, showing that it is active, messages are no longer sent to the other. Switching between them is thus seamless; this has been highly satisfactory. In addition each of these devices "times out" and goes inactive if it has not been heard from for six hours (a user configured time period); this is to prevent messages being sent to the device when it is not in use, which will be delivered later in any case (see section 3.3). But sometimes the user knows that this device is going to be used on a daily basis for some time; the **enable** command (and corresponding **disable**) was added to override system defaults. These two strategies provide a blend of agent and user management of these equivalent devices.

4.5. Impromptu uses and flexibility

As with mobile phones, some of the surprises of always-connected email are the unexpected uses. For example, CLUES forwards messages to mobile users if it can determine that they are co-located (at the metropolitan level, based on telephone area

codes); this allowed author A to be a surprise guest at a friend's house warming party on the way home from a vacation near Seattle. An escape mechanism allows an AM user to execute a Unix command line, with *stdout* and *stderr* sent to the requesting device; this is used in combination with the "grep" command to look up a variety of information. By providing a filter rule that allows messages to be forwarded if they contain a secret key word, an AM user can always tell someone how to reach him. Finally, both authors have a filtering rule allowing execution of Knothole commands in response to messages not from them if the subject line of the email contains a code word. Author A's daughter has started using it to get weather forecasts while at school.

Because AM supports delivery to any email address, it has been used at times to send messages via another user's account; this is especially useful when visiting overseas, where mail access may be difficult (a situation which is gradually improving). Using the code word invokes the AM response-via-proxy mechanism. This scheme was also used to temporarily forward messages to commercial email service providers (such as Yahoo) to allow remote web access, before the Media Lab supported a web mail client, by treating Yahoo mail as just another device.

One of the key features of AM's configurability has been the resulting flexibility to accommodate changing communication technology and user behavior. Author A previously left his preferred device always "enabled"; when he went out of range to his summerhouse or on vacation, he would just send a message from Pocketmail to re-route messages to it instead (and catch any messages lost while in transit). Leaving the device enabled meant that in the morning it contained messages that had arrived overnight, and offered a means of quick scanning them and replying to urgent ones. At this time his PC connectivity from home consisted of a dialup modem, and he turned the PC off overnight; later he switched to a broadband connection, and now just leaves a laptop in "sleep" mode on a convenient table. The laptop is so much faster to connect than the old "boot and dial" PC that he now uses it, and prefers not to clutter the PDA with redundant info. However, when he opens up the laptop in the morning, the first page he visits is the AM message status discussed previously, as this provides a quick overview of important messages, just as the messages on the PDA did before. We have numerous similar anecdotes, and cannot over emphasize the importance of flexibility and end user configuration in our rapidly changing networking environments.

4.6. Why a heterogeneous network?

Currently, we employ devices with different network coverage and usage fees. In the U.S., SMS is available on GSM phones, but GSM has not been available in every metropolitan area until recently 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. Just traveling to a summer home a few hours from Boston crosses three zones of different device access. Since work on AM began in 1998, both authors have gone through three generations of mobile devices; twice we have been able to say that none of our devices had been handled by the previous version of AM. Had we been dependent on a single network for wireless

infrastructure, we would not have had the flexibility to adopt new device technologies so easily.

Even if a single mobile device sufficed, a system such as AM 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 hear via synthetic speech over any telephone (we have used a coin-operated roadside phone in a remote area with no wireless service at all, or another country with different wireless spectrum assignments).

The "right" device is always the one I am using currently. Whatever the motivation for carrying one over another, the charged and operational device in my pocket (or the hotel fax machine if that is my only option) is the best for delivery. In this AM excels, and has allowed us to make device decisions based on factors such as power/battery management, need for a voice phone, service cost, even the size of the pocket in a particular shirt. We have little wireless device and service provider loyalty; if rates increase, or a more useful device appears, we will switch. The high churn rate of mobile phone subscriptions suggests that we are typical users in this way.

5. RELATED WORK

IPulse[6] was an early attempt to provide multi-media Instant Messaging style connectivity between PCs, phones, and other mobile devices. It was a client-server system designed to establish connections in a secure and appropriate manner. AM similarly hides device addresses and chooses appropriate media for them. *OnTheMove* (Harmer, 1998) also focused on delivery of multimedia content to mobile devices. It was based on prototype middleware called Mobile Application Support Environment (MASE) located between the wireless networks, e.g., GSM, DECT, UMTS, and the applications, e.g., video conferencing, personal newspaper, etc. MASE stored user preferences, detected the location of the user, and adapted to the status of the wireless networks and the available bandwidth, but it is not clear how this project addresses prioritization of messages or device preferences by users. The *Mobile People Architecture* (MPA) (Maniatis et al., 1999) is a framework for finding people and communicating with them personally, as opposed to their devices. Its proxy has a dual role, as both a tracking agent, maintaining the list of devices or applications through which a person is accessible, as well as a dispatcher, using application drivers to convert the message into a deliverable format. As opposed to AM, MPA makes a single routing decision, at which point it has finished with the message.

Microsoft Research's *Notification Platform* also provides context-sensitive services and message management. *Priorities* (Horvitz, Jacobs, & Hovel, 1999) is trained by the user to assign certain priority levels to incoming email. Using a support vector machine method to determine the urgency of each message, the program can

[6] http://www.onforum.com/exhibits/ericsson_01/

announce important email with special audio cues. Priorities senses when the user is busy by monitoring her activity, and waits an appropriate amount of time after she has stopped inputting text to interrupt her with a message. Priorities forwards the high priority messages and scheduled alerts to the user's cell phone or pager if she is away from her desk. *Mobile Manager*[7] delivers email, calendar, and reminder information from Outlook to her mobile device, using information from *Priorities*. The user can set up to four different profiles 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 PC has been idle for a specified time. More recently, Horvitz and Apacible (2003) have approached the interrupt problem by letting the user train the system by assigning cost to various interruptions and value to their contents, so the agent delivers only information worth being interrupted for. 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. Additionally, AM uses end-user programming of rules, while these projects use machine learning from supervised training.

AM uses simple thread detection by looking at the sender of a message and its subject line. Lam (2002) developed a system that summarizes messages by looking at thread reply chains as well as commonly found features. It uses a heuristic-based approach to filtering e-mail to remove signature, header fields, and quoted parent messages. Employing a similar set of rules to ensure thread consistency would be perfectly appropriate for future mail agents.

PocketGenie[8] by the *WolfeTech Corporation* is a service for two-way pagers, providing 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. *Thinmail Inc.*[9] 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. The service also acts as a proxy, so that all email appears as if it came from the user's standard address.

6. CONCLUSIONS AND FUTURE DIRECTIONS

We have described Active Messenger, a mail forwarding agent designed to prioritize incoming email and direct it to one or more mobile devices. We have discussed its origins, initial requirements analysis, operation, and effectiveness. The usage data is gleaned from two users, who are also the creators of the system and the authors of this paper. Many of the particular devices and our own mail management habits are very idiosyncratic, but, as related research supports, the general importance of email in the workplace and mobile email for the mobile worker, as well as

[7] http://www.microsoft.com/miserver/techinfo/omm/default.asp
[8] http://www.wolfetech.com/
[9] http://www.thinmail.com/

associated problems of filtering and interruption, are key motivators for any mobile messaging system. AM's overall architecture, i.e. a server tracking delivery over time and processes to handle each incoming message, generalizes well.

This paper makes two specific contributions over the related work we have cited. First, we have attempted to describe many of the operational issues we have encountered using various devices for mobile email access, and the software solutions we incorporated into AM to manage them successfully. Although those changes to AM's delivery procedures were not radical, they have significantly enhanced the utility of AM and made its use more satisfying; many of these techniques are appropriate to other mobile message delivery systems. Second, we have reported qualitative results and a number of user experiences in a multi-year longitudinal interaction. Although certainly biased, these results offer some general insights into both the potential as well as the down side of "everywhere messaging."

More important, we verified our design principles that can guide future work. Our core issues are relevant to any mobile system, including the effects and expectations of connectivity, tradeoffs in device power consumption and bandwidth utilization versus degree to which messages are filtered, and caching and managing intermittent network connectivity. By far the most important lesson is the value of flexibility and configuration; each user is unique, devices and networks have different characteristics, and user's needs vary over time. It is the spontaneous improvisation based on system characteristics, in real life, out-of-office situations, which will win user loyalty.

Active Messenger, as well as much of the related work, reveals the tension between the desire to be always connected and the personal and social costs of such connectivity. Notification of incoming mail, typically audible, is a distraction from our current tasks and intrudes on our social settings; as author A's wife once said in a restaurant, "Are we having dinner together or are we reading our email?" Even inaudible notification, such as a blinking red LED on a device in one's pocket, has distracted some of these with whom we regularly interact enough for them to complain. Although we certainly can, and sometimes do, turn off notification, we often forget to turn it back on. With notification disabled, we lose the opportunity for timely responses and semi-synchronous email "conversations." Many such conversations are with precisely the same family members who are annoyed by the notifications that occur in their presence; they cannot appreciate the role of AM technology since they never directly witness it in operation on their own messages.

While AM cannot fully solve this tension, as it cannot be aware of our current social context or our relationship with the sender of a message, it does help minimize the number of interruptions, through filtering, and allows a user to control, on a per-location basis, the classes of messages which will be sent to a particular mobile device as well as automatic time of day constraints on delivery. AM also assists by adapting to device usage; simply sending a message from an inactive device enables it, while ignoring an active device will eventually cause it to go idle. Ideally, AM could interact with a device and control as well as monitor whether a user responds to the alert by reading the message, but this has not yet been possible with currently available devices.

Limiting delivery to the "most important" messages—however importance is determined—also frees up personal time while still offering limited responsiveness. Author A recently stopped using wireless delivery when the service provider ceased support for his wireless device. So he was pleased to have wireless network access for his laptop computer at a conference he attended. But, with full email access, he succumbed to the temptation to read *all* his mail, and as a result paid only partial attention to the talks. He found it disturbing to have traveled half a day to the conference and then spend his time reading email, bud did not want to simply refrain as it was his means of staying in touch with his family. So although AM could simply provide notification for every email, this may be highly undesirable, even with notification disabled.

Of course, relying on an agent to deliver important messages raises the possibility of ignoring wanted message that erroneously do not get tagged by the agent as important. For this reason, AM does not change existing email storage or email reading software at all; all messages are stored and eventually each AM user must read and/or delete these. Of course messages which AM does not deliver will not be read as quickly, but our strong belief is that generating an audible alert with every incoming email far exceeds tolerable levels of interruption. And the same author's recent "disconnected" experience also demonstrates that use of conventional mail tools does not prevent temporarily missing important messages. With more than 100 incoming emails daily, it is impossible to keep up, and the number of valuable email message he misses under each condition is qualitatively similar. This is a problem that is endemic with email.

For some, strong separation of work and family time may suffice limiting interruption. But neither of the authors works with a regular daytime schedule, and even if they did, many of our colleagues live in widely disperse time zones. Both of the authors work highly unconventional schedules and the freedom this provides is important to their jobs satisfaction (and may be one of the best advantages to working at a university). Because we both use AM, we communicate frequently and freely and hence coordinate our activities effectively. The separation of location from work/personal mode requires communication tools that assist both aspects of life.

Still, AM is not quite as malleable as we would like. Because, as just mentioned, it cannot monitor the actions performed by the user at a particular device, it cannot know the user's actual behavior. Sometimes this results in a wasteful redundant delivery of messages, which can be frustrating if the messages must ultimately be deleted manually. And although it is possible (and has been done many times in the development of AM) to add support for a new device, experience shows that modifying AM code is often more time consuming than anticipated. Improving AM's extensibility would require a significant redesign of the AM infrastructure.

We have now shifted our research focus from email to voice call management, with emphasis on giving that agent a physical (and auditory) embodiment. Active Messenger itself is finished as a research system, though we see absolutely no reason to stop using it. It has matured to a level to be close to a product. Certainly it would have to support commercially available office software in a product, and would probably supply a GUI and simulator for rule authoring. A stronger device abstraction model might allow end users to more easily configure AM to support new devices, or at least simplify the task for a system maintainer.

One question to address in closing is, why not a homogeneous network? Certainly there will one day be the perfect device to handle all media and operate on a single network; does this obviate Active Messenger? We think not, but first make two observations. First, we stick to our design principle that screens and keyboards make for easier mail reading, and as AM effectively considers the user's local computer as its first device, there can be no single device solution for users who are sometimes mobile but sometimes use a desktop or laptop computer. Second, although Europe may be close to a homogenous network model, with GSM and emerging UTMS standards in a densely populated area, the U.S. is not. Author A spends at least four weeks per year in places without wireless coverage, including a summer home on an island. Yet, it is just these places where mobile connectivity is sometimes most appreciated, even if it requires a trip to a coin-operated landline telephone down the street or at a trailhead. And the regulatory environment in the U.S. seems to consider network diversity to be an indicator of competition, one of the goals of governmental control.

But, these concerns aside, AM could thrive in a homogeneous network. We presume such a network would be cellular and would know the approximate location of the user via GPS or other means. All that is required is to make this data available to AM and it can apply its location-specific filtering rules. Since users may also log in via Internet cafes, etc. and their devices may be turned off or out of power, caching and just-in-time message delivery is still needed. This new AM could be built inside the network and offered as a service to subscribers, who would benefit from sharing their calendars and address books with the service. Or, with communication from either the network or directly from a location-aware device, the AM server could be housed on customer premises and even encrypt traffic to a client on the device, which would make it more attractive to the corporate IT world.

We admit our bias, but Active Messenger has changed our email communication for the better, for both work and play. Any future system for mobile email access would benefit from careful consideration of our design features and the lessons gained from our years of usage.

NOTES

Background. This article is based on the Master's thesis of the second author.

Acknowledgments. 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, Vidya Lakshmipathy, Gerardo Vallejo, Ivan Chardin; Allen Milewski and Walter Bender for reading early drafts of this paper; Pascal Chesnais and Joshua Randall for helping with Canard issues. We would also like to thank the reviewers for numerous helpful comments.

Authors' Present Addresses.

Chris Schmandt
MIT Media Lab, E15-368a
20 Ames Street
Cambridge MA, 02139, U.S.A.
Email: geek@media.mit.edu.

Stefan Marti
MIT Media Lab, E15-384C
20 Ames Street
Cambridge MA, 02139, U.S.A.
Email: stefanm@media.mit.edu.

HCI Editorial Record. (supplied by Editor)

REFERENCES

Boone, G. (1998). Concept Features in Re:Agent, an Intelligent Email Agent. *Proceedings of the AA-98 Conference on Autonomous Agents,* 141-148. New York: ACM.

Brady, D. (2004). The Brains Behind BlackBerry: Research In Motion's co-CEOs keep taking wireless e-mail to the next level. *BusinessWeek* April 19 2004, 55. New York: McGraw-Hill Companies Inc.

Chesnais, P. R. (1997). Canard: A framework for community messaging. *Proceedings of the ISWC '97 First International Symposium on Wearable Computers*, 108-115. Los Alamitos, CA: IEEE Computer Society Press.

Cutrell, E. B., Czerwinski, M., & Horvitz, E. (2001). Notification, disruption, and memory: Effects of messaging interruptions on memory and performance. *Proceedings of INTERACT 2001 Conference on Human-Computer Interaction*, 263-269. Amsterdam, Netherlands: IOS Press.

Harmer, J. (1998). *The OnTheMove project*. BT Laboratories, Martlesham Heath, Ipswich, England.

Horvitz, E., and Apacible, J. (2003). Learning and Reasoning about Interruption. *Proceedings of the ICMI 2003 International Conference on Multimodal Interfaces*, 20-27. New York: ACM.

Horvitz, E., Jacobs, A., & Hovel, D. (1999). Attention-Sensitive Alerting. *Proceedings of UAI '99 Conference on Uncertainty and Artificial Intelligence,* 305-313, San Francisco: Morgan Kaufmann Publishers.

Hudson, J., Christensen, J., Kellogg, W., and Erickson, T. (2002). "I'd Be Overwhelmed, But It's Just One more Thing To Do": Availability and Interruption in Research Management. *Proceedings of the CHI '02 Conference on Human Factors in Computing Systems*, 97-104. New York: ACM.

Lam, D.S. (2002). *Exploiting E-mail Structure to Improve Summarization.* Unpublished Master's thesis, Massachusetts Institute of Technology.

Mackay, W. E., Malone, T. W., Crowston, K., Rao, R., Rosenblitt, D., & Card, S. K. (1989). How do experienced information lens users use rules? *Proceedings of the ACM Conference on Human Factors in Computer Systems (CHI '89)*, 211-216. New York: ACM.

Maniatis, P., Roussopoulos, M., Swierk, E., Lai, K., Appenzeller, G., Zhao, X., & Baker, M. (1999). The Mobile People Architecture. *SIGMOBILE Mobile Computing and Communications Review*, 3(3), 36-42. New York: ACM.

Marx, M., & Schmandt, C. (1996). CLUES: Dynamic Personalized Message Filtering. *Proceedings of the CSCW '96 Conference on Computer Supported Cooperative Work*, 113-121. New York: ACM.

Marx, M., & Schmandt, C. (1996). MailCall: message presentation and navigation in a nonvisual environment. *Proceedings of the CHI '96 Conference on Human Factors in Computer Systems,* 165-172. New York: ACM.

O'Conaill, B., & Frohlich, D. (1995). Timespace in the Workplace: Dealing with Interruptions. *Proceedings of the CHI '95 Conference on Human Factors in Computer Systems,* 262-263. New York: ACM.

Perry, M., O'Hara, K., Sellen, A. Harper, R., & Brown, B.A.T. (2001). Dealing with mobility: understanding access anytime, anywhere. *ACM Transactions on Computer-Human Interaction (ToCHI 2001),* 4(8), 1-25. New York: ACM.

Sawhney, N., & Schmandt, C. (2000). Nomadic Radio: Speech & Audio Interaction for Contextual Messaging in Nomadic Environments. *ACM Transactions on Computer-Human Interaction (ToCHI 2000),* 7(3), 353-383. New York: ACM.

Schmandt, C. (1993). Phoneshell: The Telephone as a Computer Terminal. *Proceedings of the first ACM international conference on Multimedia,* 373-382. New York: ACM.

Schmandt, C., & Arons, B. (1984). Phone Slave: A Conversational Telephone Messaging System. *IEEE Transactions on Consumer Electronics CE-30(3),* 21-24. New York: Institute of Electrical and Electronics Engineers, Inc.

Terveen, L.G., & Murray, L. (1996). Helping users program their personal agents. *Proceedings of the CHI '96 Conference on Human Factors in Computing Systems,* 355-361. New York: ACM.

Whittaker, S., & Sidner, C. (1996). Email overload: exploring personal information management of email. *Proceedings of the CHI '96 Conference on Human Factors in Computing Systems*, 276–283. New York: ACM.

Yankelovich, N., Levow, G.A., & Marx, M. (1995). Designing SpeechActs: Issues in Speech User Interfaces. *Proceedings of the CHI '95 Conference on Human Factors in Computing Systems*, 369-376. New York: ACM.

FOOTNOTES

1. http://www.gsmworld.com/news/statistics/index.shtml
2. http://www.procmail.org/
3. http://web.media.mit.edu/~stefanm/pager/
4. http://www.pocketmail.com/: essentially a modem in a PDA, but not a terminal emulator. Mail sent to a provided email address is retrieved by dialing a toll-free number.
5. http://www.palmone.com/us/support/i705/
6. http://www.onforum.com/exhibits/ericsson_01/
7. http://www.microsoft.com/miserver/techinfo/omm/default.asp
8. http://www.wolfetech.com/
9. http://www.thinmail.com/

FIGURE CAPTIONS

Figure 1: Channel sequence example

Figure 2: Characteristics of some communication channels

Figure 3: AM status web page: each row in the table corresponds to a message; color shows priority

FIGURES

Figure 1: Channel sequence example

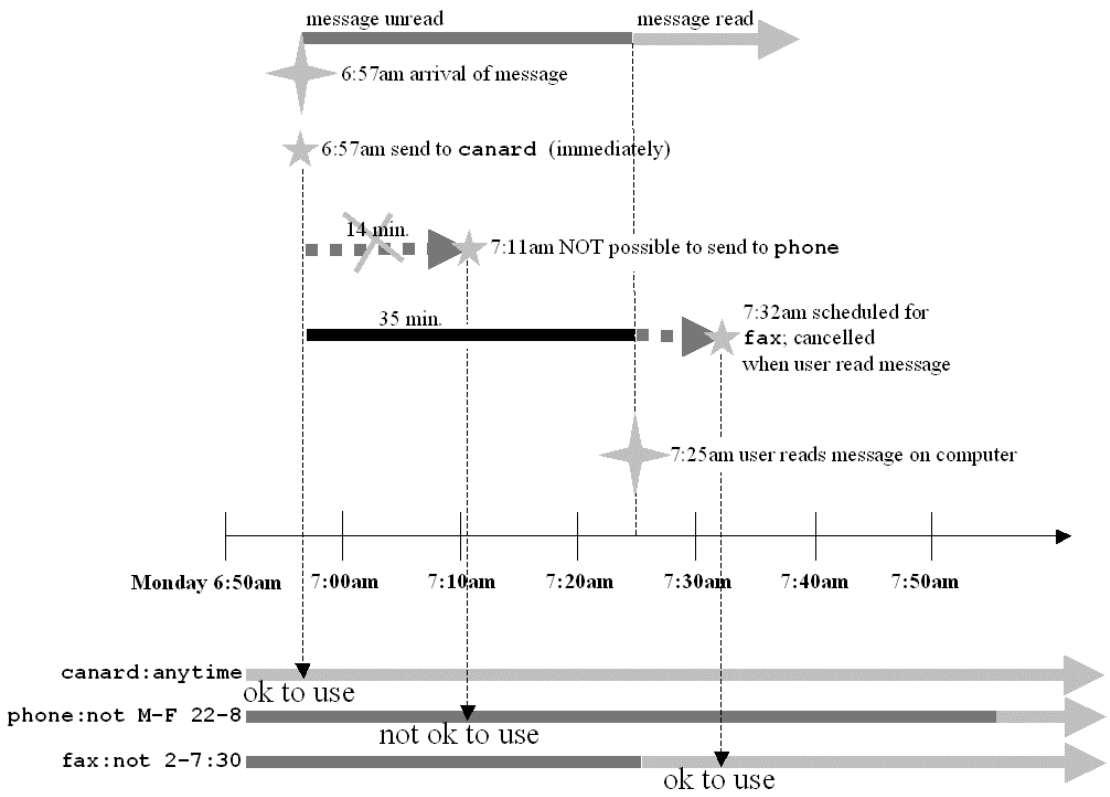
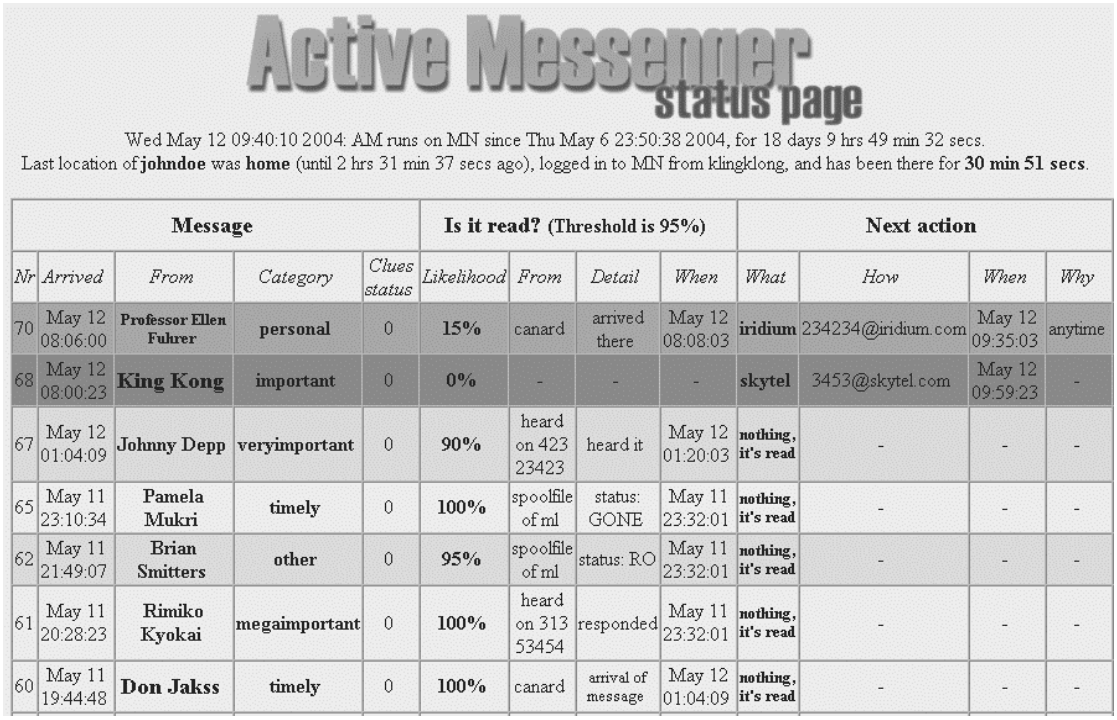


Figure 2: Characteristics of some communication channels

	Is device two-way?	*Is device buffered?*	*Info about device in range?*	*Info about message received?*	*Info about message read?*
Canard pager	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
Wireless Palm[10]	Yes	Yes	No	No	No
Pocketmail	Yes	Yes	No	No	No
Sidekick	Yes	Yes	No	Yes	Yes
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	Yes	Yes	Yes	Yes	Yes
UNIX mail spool file	Yes	Yes	Yes	Yes	Yes

[10] http://www.palmone.com/us/support/i705/

Figure 3. AM status web page: each row in the table corresponds to a message; color shows priority



Active Messenger status page

Wed May 12 09:40:10 2004: AM runs on MN since Thu May 6 23:50:38 2004, for 18 days 9 hrs 49 min 32 secs.
Last location of **johndoe** was **home** (until 2 hrs 31 min 37 secs ago), logged in to MN from klingklong, and has been there for **30 min 51 secs**.

Message					Is it read? (Threshold is 95%)				Next action			
Nr	*Arrived*	*From*	*Category*	*Clues status*	*Likelihood*	*From*	*Detail*	*When*	*What*	*How*	*When*	*Why*
70	May 12 08:06:00	**Professor Ellen Fuhrer**	**personal**	0	**15%**	canard	arrived there	May 12 08:08:03	**iridium**	234234@iridium.com	May 12 09:35:03	anytime
68	May 12 08:00:23	**King Kong**	**important**	0	**0%**	-	-	-	**skytel**	3453@skytel.com	May 12 09:59:23	-
67	May 12 01:04:09	**Johnny Depp**	**veryimportant**	0	**90%**	heard on 423 23423	heard it	May 12 01:20:03	**nothing, it's read**	-	-	-
65	May 11 23:10:34	**Pamela Mukri**	**timely**	0	**100%**	spoolfile of ml	status: GONE	May 11 23:32:01	**nothing, it's read**	-	-	-
62	May 11 21:49:07	**Brian Smitters**	**other**	0	**95%**	spoolfile of ml	status: RO	May 11 23:32:01	**nothing, it's read**	-	-	-
61	May 11 20:28:23	**Rimiko Kyokai**	**megaimportant**	0	**100%**	heard on 313 53454	responded	May 11 23:32:01	**nothing, it's read**	-	-	-
60	May 11 19:44:48	**Don Jakss**	**timely**	0	**100%**	canard	arrival of message	May 12 01:04:09	**nothing, it's read**	-	-	-