Hanging Messages: Using Context-Enhanced Messages For Just-In-Time Communication

by

Emily L. Chang

Submitted to the Department of Electrical Engineering and Computer Science on May 17, 2001, in partial fulfillment of the requirements for the degree of Master of Engineering in Electrical Engineering and Computer Science

Abstract

Recent innovations in wireless and mobile computing have enabled the development of Just-In-Time communication channels using devices such as cell phones and handhelds. However, as with existing systems such as e-mail, there is a real danger that individuals will be overwhelmed by the speed and volume of information flow. The solution presented in this thesis is to locate an agent on each user’s mobile wireless device, which will make use of information about the user to filter incoming communications. Specifically, the system uses contextual information, the current time and location of the recipient, in combination with user expressed preferences, to filter messages. Ideally, a user should never receive any message which is unsolicited or irrelevant to her current context.

The complete implementation includes a central message server and any number of wireless Internet-enabled, location-aware mobile devices, which may send and receive messages. Among the filtering mechanisms used by the agent are a precise hierarchical categorization for messages, user profiling by rules, and multiple location-based operational modes for each user. The end result is a Just-In-Time communication system which allows users to quickly receive relevant, time and location-sensitive messages.

Thesis Supervisor: Pattie Maes
Title: Associate Professor, Media Arts and Sciences
Acknowledgments

I am grateful to my advisor, Pattie Maes, who was constantly available to provide advice, suggestions, ideas, and feedback, and kept my project from running off in strange directions.

I also want to thank Jim Youll, Joanie Morris, Sybil Shearin, and the rest of the Agents, for being generous with their time and experience, and particularly for helping me out early in the year, when I was young and clueless.

To my apartment-mates (both upstairs and downstairs), and especially to Danny Lai, thanks for keeping me sane.

To Mom, Dad, Michelle, and Andrew, thanks for your continued love and support and for always being there for me.
# Contents

1 Introduction .......................................................... 13  
   1.1 Just-In-Time Messaging ........................................ 13  
   1.2 Context-Enhanced Messages .................................... 14  
       1.2.1 Time and Location of Delivery .......................... 14  
   1.3 Client-side Agent Filtering .................................. 15  
       1.3.1 Categories .............................................. 16  
   1.4 Hanging Messages ............................................. 16  
   1.5 Document Overview .......................................... 17  

2 Vision ................................................................. 19  
   2.1 Overview ........................................................ 19  
   2.2 Scenarios of Use .............................................. 20  
       2.2.1 Personal Reminders ...................................... 20  
       2.2.2 Requests from Others .................................... 21  
       2.2.3 Promotional Use ......................................... 21  
       2.2.4 Distributing Information ................................. 23  
   2.3 Research Focus ................................................ 23  
       2.3.1 Importance of Filtering .................................. 23  
       2.3.2 Overview of Mechanisms ................................. 24  

3 Design and Implementation ......................................... 27  
   3.1 System Architecture .......................................... 27  
       3.1.1 Overview ................................................. 28  
       3.1.2 Server Modules .......................................... 29  
       3.1.3 Client Modules .......................................... 30  
       3.1.4 Design Decisions ....................................... 32  
   3.2 Filtering Mechanisms ......................................... 32  
       3.2.1 Message Categories ...................................... 33  
       3.2.2 Modes and Locations .................................... 35  
       3.2.3 User Profiles ........................................... 36  
       3.2.4 Message Management .................................... 37  
       3.2.5 Summary ................................................ 38  
   3.3 Implementation ............................................... 38  
       3.3.1 Server .................................................. 38  
       3.3.2 Client .................................................. 39  
       3.3.3 Communication Protocols ............................... 40
4 Using Hanging Messages 43
4.1 Client Usage ............................................. 43
  4.1.1 Starting a Session .................................... 43
  4.1.2 Composing Messages ................................. 44
  4.1.3 Incoming Messages ................................. 44
  4.1.4 Login Options ....................................... 45
  4.1.5 Message Visibility .................................... 46
  4.1.6 Message Retrieval Options ......................... 47
  4.1.7 Filtering Options .................................... 49
  4.1.8 Help Manuals ....................................... 53
4.2 Installation and Configuration .......................... 54
  4.2.1 Server .................................................. 54
  4.2.2 Client .................................................. 55

5 Evaluation .................................................. 57
5.1 Significant System Characteristics .................... 57
  5.1.1 Connectivity and Bandwidth ......................... 57
  5.1.2 Software Concerns .................................... 58
  5.1.3 Configurability ....................................... 59
  5.1.4 User Privacy .......................................... 60
5.2 Usability .................................................. 61
  5.2.1 User Feedback ........................................ 61
  5.2.2 Implementation Changes ............................ 62

6 Related Work ................................................ 65
6.1 Find-The-Nearest ........................................ 65
6.2 Location and Customization ............................. 65
6.3 Location and Information ................................ 66
6.4 Location and Messaging .................................. 67
  6.4.1 Locust Swarm ........................................ 67
  6.4.2 HaikuHaiku .......................................... 67
  6.4.3 Geocorder ............................................ 67
  6.4.4 comMotion ........................................... 68
  6.4.5 GeoNotes ............................................ 68
6.5 Just-In-Time Filtering .................................... 69
  6.5.1 Pigeon ............................................... 69
  6.5.2 E-mail ................................................. 70

7 Concluding Remarks ........................................ 71
7.1 Future Implementation .................................... 71
  7.1.1 Message Modification ............................... 71
  7.1.2 Message Reactivation ................................ 72
  7.1.3 Multiple Locations .................................. 72
  7.1.4 Complex Filtering Rules ......................... 72
  7.1.5 Recognition Groups .................................. 73
  7.1.6 Agent Learning ...................................... 73
  7.1.7 Encryption ........................................... 73
7.2 Conclusion ............................................... 74
A Server: Class Dependencies 75
B Client: Class Dependencies 77
C Server Communication Protocol 79
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>A personal grocery list</td>
<td>20</td>
</tr>
<tr>
<td>2-2</td>
<td>A request to run an errand</td>
<td>21</td>
</tr>
<tr>
<td>2-3</td>
<td>A special promotion</td>
<td>22</td>
</tr>
<tr>
<td>2-4</td>
<td>Information about a construction site</td>
<td>23</td>
</tr>
<tr>
<td>2-5</td>
<td>Selecting a message attribute</td>
<td>24</td>
</tr>
<tr>
<td>2-6</td>
<td>Inserting a message value</td>
<td>24</td>
</tr>
<tr>
<td>2-7</td>
<td>The new rule is added to the profile</td>
<td>25</td>
</tr>
<tr>
<td>2-8</td>
<td>Known sender list</td>
<td>26</td>
</tr>
<tr>
<td>3-1</td>
<td>Primary interactions between client and server</td>
<td>29</td>
</tr>
<tr>
<td>3-2</td>
<td>Diagram of server architecture</td>
<td>30</td>
</tr>
<tr>
<td>3-3</td>
<td>Diagram of client architecture</td>
<td>31</td>
</tr>
<tr>
<td>3-4</td>
<td>Diagram of category hierarchy</td>
<td>34</td>
</tr>
<tr>
<td>3-5</td>
<td>Editing profile rules for the current mode</td>
<td>36</td>
</tr>
<tr>
<td>3-6</td>
<td>Editing profile rules for all modes</td>
<td>37</td>
</tr>
<tr>
<td>4-1</td>
<td>Logging into the client</td>
<td>43</td>
</tr>
<tr>
<td>4-2</td>
<td>Composing a new message</td>
<td>44</td>
</tr>
<tr>
<td>4-3</td>
<td>An incoming message</td>
<td>45</td>
</tr>
<tr>
<td>4-4</td>
<td>The user can choose a different message server</td>
<td>45</td>
</tr>
<tr>
<td>4-5</td>
<td>Location can be manually set or automatically determined</td>
<td>46</td>
</tr>
<tr>
<td>4-6</td>
<td>Menu options</td>
<td>47</td>
</tr>
<tr>
<td>4-7</td>
<td>Users can assign nicknames to locations</td>
<td>47</td>
</tr>
<tr>
<td>4-8</td>
<td>Users can change their current mode of operation</td>
<td>48</td>
</tr>
<tr>
<td>4-9</td>
<td>Users can set the search radius for messages</td>
<td>49</td>
</tr>
<tr>
<td>4-10</td>
<td>Changing filtering options</td>
<td>49</td>
</tr>
<tr>
<td>4-11</td>
<td>Editing personal profiles</td>
<td>50</td>
</tr>
<tr>
<td>4-12</td>
<td>Selecting a message attribute</td>
<td>50</td>
</tr>
<tr>
<td>4-13</td>
<td>Inserting a message value</td>
<td>51</td>
</tr>
<tr>
<td>4-14</td>
<td>The new profile rule</td>
<td>51</td>
</tr>
<tr>
<td>4-15</td>
<td>Display of existing profile rules</td>
<td>52</td>
</tr>
<tr>
<td>4-16</td>
<td>Editing the known sender list</td>
<td>52</td>
</tr>
<tr>
<td>4-17</td>
<td>Help topics available</td>
<td>53</td>
</tr>
<tr>
<td>4-18</td>
<td>Help manual entry for “Compose”</td>
<td>53</td>
</tr>
<tr>
<td>4-19</td>
<td>Starting the server</td>
<td>55</td>
</tr>
<tr>
<td>4-20</td>
<td>Display of existing profile rules</td>
<td>52</td>
</tr>
<tr>
<td>5-1</td>
<td>The evolution of the <em>Edit My Profiles</em> dialog</td>
<td>63</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The ascendance of a new technology and its subsequent entrenchment into everyday life often depend not on the soundness of the technology itself, but on the development of novel applications that capitalize on its unique advantages. One recent innovation which has yet to gain widespread acceptance is mobile computing, which in the extreme case, allows ubiquitous or “everywhere” computing[1]. Mobile devices certainly exist, in all shapes and sizes, but most consumers limit themselves to simple uses such as cellular (cell) phones and personal digital assistants (PDAs).

The same situation applies to wireless technology; although many industry experts predicted the rapid proliferation of wireless access protocol (WAP)[2] enabled devices, WAP has yet to take root in popular culture. Of course, in both of these cases, the technology itself is only a framework; there are countless opportunities for additional innovation, many of which have not yet been explored. With this in mind, it is quite possible that the most practical solution is to concentrate on developing useful applications that leverage the benefits of these new technologies.

1.1 Just-In-Time Messaging

Since mobile computing and wireless connectivity are particularly complementary, they are often used in conjunction to support a new application; examples include wireless e-mail and wireless web. However, these uses are somewhat unimaginative; a better approach would be to use mobile wireless to enable substantially new functionality, rather than attempting to replace existing technology, such as a web browser on a personal computer (PC).
One possibility is Just-In-Time (JIT) messaging; allowing the sending and receiving of messages in real time, to a mobile device. Although JIT messaging has similarities to both normal cell phone usage and e-mail messaging, it is uniquely placed between the two. JIT messaging is less intrusive than a phone call, but unlike e-mail, it allows extremely time-sensitive communication. Short Message Service (SMS) for cell phones is one example of a simple JIT messaging system; it has become extremely popular in Europe and Japan, and is gaining momentum in the United States.

1.2 Context-Enhanced Messages

The primary reason that JIT messaging is useful is that it allows the sender of a message to exert some control over the context in which a message is delivered. With e-mail, voice mail, and other inbox-style systems, a message sender cannot know when or where the recipient will check her inbox and receive the message. Instant messaging systems such as ICQ, AIM, or Zephyr allow slightly more control; as long as the recipient is at his PC, he will receive the message almost immediately. Cell phones allow instant communication, but often intrude on the current activity of the recipient. JIT messaging differentiates itself from all of these systems, by allowing real time communication, without excessively disturbing message targets.

1.2.1 Time and Location of Delivery

Although SMS and other basic JIT messaging systems allow users to send or receive messages at any time, they do not allow senders to specify a future delivery date or time period. The addition of this simple feature would permit reminder-type messages as well as message queuing; users could queue up multiple messages while they were unoccupied, to be stored for future delivery.

An extension of incorporating this time-sensitivity would be to specify additional metadata, such as the location to which a message should be delivered. The result would be greater control of message delivery context; by specifying when and where the intended recipient will receive a message, the sender incorporates her judgment and ideally, increases the relevancy of the message to the recipient. Time and location are key components of an individual environment, and there are many messages which would only be appropriate
when delivered to a particular location, during a specific time period.

For example, a landlord might want to notify the tenants of a building of a water shutdown. This notice would only be relevant during the water shutdown period, and only to those people that are in the building during that time. Similarly, an individual who is late for a meeting might want to post a message at the meeting location, indicating his estimated arrival time. This message would only be useful until the person arrives in person at the scene. Bulletin board posters and flyers which announce events are extremely time-sensitive; these are only useful until the date and time of the event, and must be manually removed.

Furthermore, by using time and location attributes in conjunction, the actual number of delivered messages may be significantly reduced. A recipient who does not coincide with the location and time specified by a message will never receive that message, and consequently that recipient’s message load is diminished. By requiring the attributes of location and delivery time, a system increases the relevancy of its messages and reduces the information overload that often occurs with high message volume.

1.3 Client-side Agent Filtering

Ideally, a JIT communication system should only deliver those messages which are truly urgent and time-sensitive. Although not as intrusive as cell phones, high volume JIT messaging can also become disturbing, due to its omnipresent nature. One possible solution to this problem is to deploy an “agent” on the receiver’s side to continually examine new messages, filtering out unnecessary, unwanted, or low-priority messages.

An agent is an autonomous program that can oversee a task on behalf of the user. The continuous, adaptive nature of an agent makes it the ideal choice for a task such as real-time message filtering. An agent can reside on the recipient’s client device, monitor the user’s location, query the message server at appropriate intervals for new messages, and process incoming messages, deciding whether or not to present them to the user. Agents typically are personalized by user; they keep track of a user’s unique interests, habits and preferences, and can use these to implement a very diverse set of filtering behaviors[7].
1.3.1 Categories

In order for an agent to be an effective filter, it needs to extract information from the message. In conventional e-mail systems, filters generally classify messages based on sender, recipient, subject or body keywords. In a time and location-enhanced JIT messaging system, the additional attributes of spatial and temporal context can also be used for filtering. However, one of the most important attributes of any message is the subject of its content; i.e. what the message is about. In most messaging systems, the “Subject” field is supposed to summarize the content of the message. Unfortunately, this field is often left empty, or occupied with meaningless phrases such as “RE” or “Fwd” which shed no light on the actual content of the message.

In a JIT messaging system, it is essential that message filtering be robust and aggressive, to prevent user saturation. To allow better client-side filtering, an additional attribute such as “Category” could be added to all messages, which would be selected by the sender from a predefined list, but could be modified by the recipient’s agent based on other attributes.

For example, a sender could specify that a message belonged to the Traffic category, to indicate that the message provided traffic information for a certain location, or the Family category, to let the receiving agent know that the sender is a family member of the recipient. The small amount of extra work that this addition would require from the sender would be far outweighed by the benefit of a more informed user agent on the recipient side.

1.4 Hanging Messages

Hanging Messages is a JIT messaging system that was developed according to the principles described above. It uses the emergent technologies of wireless and mobile computing to enable real time communication; the system requires that users carry their client devices constantly, so that messages will be delivered at appropriate locations. It also requires that they have a continuous Internet connection, in order to retrieve messages in a timely fashion.

In addition to allowing specification of time and location of delivery, Hanging Messages incorporates some additional enhancements such as multiple message recipients, message re-delivery upon recipient request, and message priority types; these features will be described in detail later in this document. The important point is that the Hanging Messages system
has been designed modularly, with the idea of future enhancements in mind. Although it incorporates several interesting features, the Hanging Messages (HM) system concentrates on maximizing the advantages of additional contextual data in messaging, so that the recipient receives only those messages which are highly relevant. It requires the specification of a “Category” attribute, so that in addition to time and location, a client-side agent in this system has a good idea of the message content. By using all of this data effectively in conjunction with the user’s profiles, an intelligent agent can filter incoming messages efficiently and effectively.

Several separate schemes are used by the HM client-side agent to filter messages. The agent assumes that each user has several “modes” of operation, such as “home” or “work”; a different user profile is associated with each mode. User profiles consist of rules that describe user preferences, such as wanting to receive messages from the “Work-Related” category but not from the “Recreational” category while in “work” mode. Additionally, a “Known Sender List” functionality is provided to the user; she can edit a personal list of users and enable this feature to receive messages only from those senders on the list. The combined effect of these policies is a reduction in the number of messages that are actually delivered to the user.

1.5 Document Overview

This chapter discussed how Just-In-Time messaging exploits the new technologies of wireless and mobile computing devices to offer users a new service. It mentioned various extensions and enhancements that would prevent JIT messaging from becoming overwhelming to the user, and introduced Hanging Messages, a JIT messaging system that incorporates several context and other enhancements.

In the next chapter, the vision for the Hanging Messages system is described in more detail; several scenarios are depicted in which the system would be useful, the research focus of agent filtering is clearly delineated, and a general overview of filtering methods available in HM is presented.

Subsequently, in Chapter 3, the overall system architecture is explained and specific filtering mechanisms are described. Important implementation details are also noted.

Specifics about the use of the Hanging Messages system are explained in Chapter 4,
followed by a brief evaluation of the system, its ease of use, utility, and design and implementation choices in Chapter 5.

Chapter 6 discusses related work in the areas of location-aware messaging, intelligent message filtering, and other related areas.

Finally, some concluding remarks are made about issues encountered during development, future implementation paths, and lessons that might be taken from the design and implementation of the Hanging Messages system.
Chapter 2

Vision

2.1 Overview

The Hanging Messages system aims to enable users with a robust and feature-rich Just-In-Time communication system. The primary focus of the system is to ensure that this new service does not contribute to overloading its users with information. Rather, Hanging Messages provides a new channel through which relevant context-sensitive messages can be distributed more effectively.

Users of HM send and receive all messages on their client devices. Agents reside on these clients, monitoring the user’s location and the current time, querying with the message server at appropriate intervals for new messages, and processing incoming messages, deciding whether or not to present them to the user.

Messages can be filtered by the agent into three types: active, passive, and ignored. Active messages will alert the user when they are delivered, while passive messages are not read unless the user explicitly queries a location for existing messages. Ignored messages are never seen by the user, but are kept so that they can be re-filtered if a user profile changes.

The hypothesis is that messages that are actually delivered will be more useful to the recipient, since the agent has performed some processing. However, the user contributes to the behavior of the agent, and has ultimate control over her profile. In addition, the sender has submitted his input by choosing the time and place of delivery, so that the recipient only receives the message if she comes within the designated location during the specified time period.
2.2 Scenarios of Use

One question that is commonly posed involves scenarios of use. Why should a sender turn to a new system rather than sending her messages through established channels such as e-mail or SMS text messaging? The following sections describe some example scenarios in which the added features of the HM system might be particularly useful.

2.2.1 Personal Reminders

By incorporating delivery time and location, Hanging Messages allows users to effectively send themselves reminders that will be triggered at appropriate times and places. For example, a user could send himself a shopping list as shown in Figure 2-1, specifying the grocery store as the place and a particular day as the time period. This message would not only remind the user to go grocery shopping when he passed by the store, but would present him with his previously constructed shopping list.

![Figure 2-1: A personal grocery list](image)

Similarly, if the user is a student who often forgets to turn in paperwork on campus, she can send herself a message the evening before the form is due, while she actually remembers the errand. When she walks by the student services office the next day, she will be alerted by her mobile device, and be reminded of her errand while she is in the area and it is convenient to carry it out.
2.2.2 Requests from Others

Other users of HM can send messages with location-specific requests. For example, if a user is working on a group project, and needs a specific book, she can send a message to all members of her group in the vicinity of the bookstore, such as in Figure 2-2. A group member who passes by and does purchase the book can then reply with a message to the same location stating that he has fulfilled this request, to prevent repeat actions.

![Figure 2-2: A request to run an errand]

In this scenario, it would be useful if the user who purchases the book, were able to modify the original message, to indicate that the task has been completed, or delete it altogether, to reduce message volume. However, this ability would have many implications regarding access control and would require substantial consideration of authentication and authorization issues. For these reasons, this functionality was not implemented in this version of Hanging Messages.

2.2.3 Promotional Use

One of the more interesting features of Hanging Messages is that recipients need not be specified. This allows messages to be sent that may be applicable to any user in a given area. To prevent misuse, messages that are un-targeted are treated differently; specifically they require the sender to specify a category. The message typing scheme is explained in further detail in the section that discusses system design.

One advantage of allowing untargeted messages is that companies can send promotions
to any user who comes to a certain location, without specifying individual usernames. As a precaution, all untargeted messages are designated as passive, by default. Currently the user cannot change this policy; perhaps in the future this will be an option. Furthermore, the category system prevents HM users from being deluged with passive commercial messages. In fact, users can choose to ignore all untargeted messages. However, because the categories are hierarchical and adaptable, as described in subsequent sections, and most people do appreciate some types of promotions, it is anticipated that most users will not decline all categories of untargeted messages.

Figure 2-3 shows an example of a promotional message; in this scenario, KFC has decided to promote its special of the day by placing a message near its store, so that passersby will be notified of the promotion, and perhaps decide to enter the restaurant to investigate further.

One would expect this type of policy would discourage organizations from the practice of soliciting or collecting user addresses, since any users who would appreciate the information being distributed would have already chosen to receive that category of untargeted message. As a further deterrent, users have the option to construct a “Known Sender” list, so that only messages from users on the list may be designated as active.
2.2.4 Distributing Information

Another benefit of allowing unspecified recipients is that location based news or other information can be posted for general interest purposes. Examples of messages include traffic advisories, perhaps posted a few miles from the congestion spots, crime bulletins, which might be posted near dangerous locations, or construction details such as developer or architect contact information. Figure 2-4 illustrates a message that might be posted near the construction site for MIT's new Stata Center.

![Figure 2-4: Information about a construction site](image)

A more time-sensitive example might be a message about a meeting change; on many occasions, meeting locations change at the last minute. A Hanging Message could be posted to that effect, informing any late attendees of the changes, without requiring the effort on the organizers' part of determining all possible participants.

2.3 Research Focus

2.3.1 Importance of Filtering

For a system like Hanging Messages to be useful, it must provide a function that cannot be well satisfied through other channels, and it should not cause the user "more trouble than it's worth" which is a common complaint regarding pagers and cell phones. Hanging Messages aims to reduce the amount of trouble taken by the user by placing an agent on the receiving front lines, between the sender and the recipient. In practice, the system can only
be successful if the client side agent is an effective guard against unsolicited or otherwise unwanted messages.

2.3.2 Overview of Mechanisms

Several mechanisms were developed to address the issue of effective message filtering, beginning with user profiles. Simple rules can capture user preferences with regard to the timing, subject, or sender of incoming messages. Any attribute, including location, time of delivery, or message category, can be used to define a rule.

![Figure 2-5: Selecting a message attribute](image)

![Figure 2-6: Inserting a message value](image)

Figure 2-5 and Figure 2-6 illustrate the steps of creating a new rule. A message attribute
is selected from a dropdown list, and a message value is associated with the attribute.

A consequence is defined for those messages which match the conditions specified in the first steps. Consequences are generally changes in message state; making messages passive or active. Users can also add filtering rules which change message categories based on other attributes such as sender.

In this case, the user wants to specify that messages which are in the Task-Oriented category should be active.

After constructing the rule, the user taps the Add button to add the new rule to the user profile, as shown in Figure 2-7.

![Figure 2-7: The new rule is added to the profile](image)

To handle sets of preferences which divide along activity-related boundaries, users may have different modes of operation, such as “work” and “home” modes, which correspond to user profiles. Each user has one user profile for each mode, and the rules for each profile are separate from all other profiles. Additionally, each user has a profile that applies to all modes, to capture universal preferences.

Also, a single message is only delivered once to any given recipient, regardless of whether the message has multiple or no recipients, and even if the user revisits the specified location at a later time. In future implementations, users may be able to modify this default behavior.

A few additional policies and features were implemented into the system with the intent of restricting message delivery. As mentioned previously, untargeted messages become pas-
sive by default, since they are less likely to be urgent by the recipient's standards. A user can also create a "known sender" list, such as shown in Figure 2-8, so that only messages whose senders are on that list may be marked active.

![Known sender list](image)

**Figure 2-8: Known sender list**

In addition, the interface was designed to allow recipients to easily ban senders, making future messages from the banned sender passive or ignored. All of these features are intended to selectively reduce message volume according to reasonable criteria, so that the user does not waste time reading irrelevant messages.
Chapter 3

Design and Implementation

Most messaging systems consist of two primary components. Message servers store and relay the messages, and user clients present an interface for sending and receiving messages. Beyond these basic tasks, the division of labor is less clearly defined. In a complex system such as Hanging Messages, the interesting question is the relationship between client and server: how functionality is divided between the two sides, how they communicate, which entity takes precedence, and other such issues.

This chapter addresses this question by presenting the system architecture, outlining the responsibilities of each major module, and by describing and discussing the rationale behind key design decisions.

3.1 System Architecture

When designing the HM system, the first question was whether it was necessary to have central servers at all. If it could be built as a peer-to-peer[8] system, scalability and connectivity concerns would be reduced. However, some features of the system clearly required functionality that could be best provided by a central server, so the decision was made to make Hanging Messages a client-server system.

For example, in order to deliver messages at the correct times and places, it is necessary for messages to be queued, either on the client device, or on a server. If the messages were queued on the recipient side, clients would have to receive all messages addressed to them, regardless of time and location specifications. This policy would greatly increase the number of messages delivered to the average client. Furthermore, it might invade the privacy of the
sender, who might have assumed that message contents would only be divulged if location and time constraints were met. Since the message is stored on disk on the recipient’s device, there is a slight possibility that the recipient might be able to retrieve the message regardless of time location restrictions. With server side queuing, this risk is not a concern.

The server is also useful as a canonical source of updated information. For example, both the category hierarchy and user authentication information are stored on the server side, so they can be easily modified by an administrator; whenever a client session begins, it accesses this information and updates its own data structures accordingly.

Lastly, because of existing central server functionality, features such as message management were easily added to the Hanging Messages system. Message management refers to the idea that the server can store some message state, and vary its treatment of the message based on this state. For example, in the HM system, a message is only delivered once to a given recipient. Although this functionality could have been achieved by keeping track of previous deliveries on the client side, again this would have required a greater number of messages being delivered to clients, with a corresponding increase in network traffic.

For all of these reasons, it was decided that a peer-to-peer design would not efficiently meet the needs of the system. A client-server model was determined to be the best choice for accomplishing the desired functionality of the Hanging Messages system.

3.1.1 Overview

The primary interactions between client and server are illustrated in Figure 3-1. The HM system assumes that there is a constant connection between the server and the client devices, and that the server is continuously listening for commands from the clients. Each client operates as both a sender and a receiver, relaying new messages to the server, and regularly querying the server for incoming messages.

When the server receives a new message, it stores the message in a database, indexing by location, recipient, and delivery time. Queries from clients specify these three attributes, and the server returns all matching messages to the client. The server then discards the query, so that it does not retain any knowledge of a client’s location.

Although there might be some advantages, it is a potential privacy hazard to accumulate user location data on a central server. Should the server be compromised, the secrecy of messages might not be able to be preserved, but at the least, client locations should not
be disclosed. To achieve this result, clients track their own locations, and periodically query the server for messages existing at a given location. The server, after checking for a message match, discards the location information, which subsequently only exists on the user’s device. By shifting the responsibility of location tracking to the client side, a user’s positional information becomes as secure as her client device. Additionally, there is no single point of failure that would allow all user locations to be compromised.

Messages that reach client devices are first processed by user agents. The agents reside on the client, and have access to user profiles which encapsulate message retrieval preferences. By physically locating user information on the client, the user’s privacy is preserved, while the filtering agent can benefit from extensive user profiling. Incoming messages which match user preferences are delivered to the user’s screen.

### 3.1.2 Server Modules

The server itself has several distinct functions, as illustrated in Figure 3-2. In order to ensure user privacy and to allow updates of information such as categories, clients of the HM system undergo a login process before they are allowed to send and receive messages from the server. At the beginning of each client session, when the user starts the HM client software, he is presented with a login screen. Before any messages can be viewed, composed, or received, the user must be verified by the server, and category information is transmitted by the server back to the client.
The primary functionality of the server is implemented by the message manager, which receives and stores incoming messages, delivers outgoing messages, and tracks the delivered state of each message with respect to all recipients. Messages are persisted by the Wherehoo server, which also handles expiration issues; relevant time periods are specified when inserting a record or querying for a record, and Wherehoo returns only those records which correspond with the specified time period.

Wherehoo[9] is a search engine which was built by Jim Youll of the Software Agents Group at the MIT Media Lab. It is optimized for location-specific searches, allows specification of relevant time periods, and has a socket interface. An infrastructure element of the Software Agents Group’s Impulse[10] project, Wherehoo was designed to bind information or data to precise geographic locations. It identifies locations by Global Positioning System (GPS)[11] coordinates, and allows specification of height, bearing, and search radius. Hanging Messages uses Wherehoo by storing a message as raw data, and binding the data to a location; an ideal solution for persisting messages indexed by location.

3.1.3 Client Modules

The basic functions of the client are sending, retrieving, and filtering messages, as well as performing location translation. Figure 3-3 details the client architecture.

Location translation allows users to nickname locations and use these names when com-
posing messages, rather than having to ascertain the GPS coordinates of the destination. The locations are converted before the message is transmitted to the server. Location mappings are stored exclusively by the client device, so that any user’s frequented locations cannot be compromised by a server failure.

In order to receive messages in timely fashion, the message retrieval module continuously queries the server for messages. Queries specify current location, as determined by the GPS module, and intended recipient, which is the current user. Once received, messages are processed by the filtering unit, which uses user profiles to determine which messages should be brought to the user’s attention. The user is unaware of any activity until messages are approved by the filtering agent, and delivered to the screen.

Outgoing messages are composed through the user interface, and require only location translation before being relayed to the server interface. The interface converts outgoing message and queries into server commands, just as it transforms incoming server data into incoming message objects. All client communications are done wirelessly; the client device is as mobile as a cellular phone.
3.1.4 Design Decisions

Without addressing the details of the filtering mechanisms, there were several key decisions made in the design of the basic messaging system that are worthy of mention. Deciding to use a client-server model permitted central user verification and other extra features. Similarly, locating user-specific functionality on the client, including the filtering agent, reduced privacy concerns, allowing a much richer profiling agent.

Other decisions were more policy-based. For example, the Hanging Message system depends heavily on communication between client and server. This approach assumes widespread and robust wireless connectivity, which may not currently be the case. However, for true Just-In-Time functionality, it is essential that the client-server connection be solid. As previously discussed, the alternative, storing all of a recipient’s messages on his client, generates excessive network traffic and compromises sender privacy. Furthermore, it is expected that wireless connectivity will improve as wireless becomes prevalent.

Throughout the design of the system, emphasis was placed on disturbing the user as little as possible, with the understanding that this might divert some messages which might be considered useful. A conscious decision was made to err on the side of restraint. With an application such as Hanging Messages which can draw the user’s attention away from her current context, it is important that this ability not be misused. Other channels can be used for vital communications which must always be delivered.

The Hanging Messages system is not meant as a substitute for e-mail, phone, and other channels; it is meant to augment them, adding a functionality that is useful in a new and different way. In fact, existing e-mail systems could be used as a front-end application for Hanging Messages, an alternate client interface from which to send or receive HM messages.

3.2 Filtering Mechanisms

By adding location, time, and occasionally category designation as required attributes for messages, messages in the HM system are already automatically “filtered.” Currently, an employee who goes on vacation for a few weeks returns to an inbox full of messages, many of which may be outdated. If e-mail messages were tagged with an expiration date, as Hanging Messages are, obsolete messages would never be received at all, reducing information overload.
Similarly, location can be used to eliminate unnecessary messages. Many e-mail messages are sent to mailing lists, sometimes entire companies. If specifying location were an option, notifications about facilities, for example, would only be received by those employees who were on-site. Telecommuters, or those on business trips, would never see the messages at all, since these messages are completely irrelevant to them.

However, this type of filtering only incorporates the sender’s judgment and does not adjust to the recipient’s preferences. In order to incorporate the latter, the HM system employs an agent on the client’s side to process incoming messages. The added message properties also make it possible to better process incoming messages in this filtering process.

The Hanging Messages system filters messages into one of three types: active, passive, and ignored. Active messages automatically appear in the user’s inbox, and may be accompanied by an alert noise, depending on user preferences. Passive messages may quietly appear in the user’s inbox, or be hidden from view until the user explicitly asks to see his passive messages. The user should never see ignored messages at all. Messages are not intrinsically active or passive; these assignments are made by the agent during the filtering process.

Message filters examine the attributes of a message, compare them with user profiles, and change the type if appropriate. Most of the filtering mechanisms described in this section focus on making profile rules more effective at determining the relevancy and priority level of an incoming message. A few additional policies which help reduce overall message volume on the recipient’s side are explained as well.

3.2.1 Message Categories

The Hanging Messages system allows un-targeted messages: messages sent without a specific recipient. This feature allows, among other possibilities, dissemination of widely applicable information such as traffic reports, posting of general interest items, and local broadcasting of promotions and other services.

However, to allow client devices to effectively filter unwanted messages, users are required to specify a category when sending a new message. Categories are stored on the server, and retrieved by the client at the beginning of a session, allowing rapid transmission of category modifications or additions. They have different levels of specificity, with the levels arranged in a hierarchy.
For example, a user’s agent might accept messages from the general category “Promotions” or alternatively, it could accept messages only from the sub-category “Food” but not “Entertainment” depending on her preferences. This flexibility is significant, since it allows much more effective filtering. Figure 3-4 illustrates the default list of categories. New categories, which may be added to the server only by a system administrator, cannot duplicate existing category names.

In order to ensure that users will not be bothered by unsolicited and unwanted communication, messages with no specified recipient must be categorized as Informational or Promotional.

In addition, while senders of messages can specify a category when composing a message, this category can be overridden by the recipient’s preferences. For example, a recipient can state that all messages from a certain sender are Work Related → Administrative, even if the sender marks them as Work Related → Urgent. This management policy gives the recipient ultimate control over message filtering and delivery.

By adding the category attribute, the filtering agent can better determine matches between messages and user preferences. The result is that only relevant messages get
through. This benefits both the sender and the recipient; the sender receives fewer messages, and can focus more attention on each individual message.

3.2.2 Modes and Locations

User preferences and tendencies often surface along activity-related divisions; users may never want to receive messages while places such as classrooms, churches, courts, or other location where it is inappropriate to break one’s attention. In order to handle these preferences, each user has multiple modes of operation, each with an associated user profile. A user also has a set of default filtering rules that applies to all operational mode; these are the rules that can override category designations.

Users can edit the number of modes, as well as the filtering rules that apply to them. The default locations modes and profiles are:

- **Home** - Personal and Informational messages are active, all other messages are passive, Promotional messages are ignored.

- **In Transit** - Urgent Personal and Urgent Work Related Messages are active, all other messages are passive.

- **Meeting** - Personal and Work Related messages are passive, all other messages are ignored.

- **Social** - Urgent Personal messages are active, all other messages are passive.

- **Work** - Work Related and Urgent Personal messages are active, all other messages are passive, Promotional messages are ignored.

The system currently requires the existence of the “Home” mode, because when a user starts a session, the default mode is “Home.” All other modes are optional and can be edited or deleted. In this implementation, modes must be manually changed by the user. A better method might be to associate modes with known locations. For example, the “Work” mode would be associated with the location of a school, lab, or workplace. These default associations could be manually overridden by the user.

This mechanism assumes that user profiles are generally consistent for a given mode. A user who might wish to receive personal messages as soon as possible while at home,
will probably not want personal messages while in an “Uninterruptible” meeting or class. However, it also allows the user the flexibility of changing modes; for example, if a user works from home, he can switch to “work mode” while still physically at home.

3.2.3 User Profiles

Profiles on the HM system consist of rules which encapsulate user preferences. Rules are comprised of attribute-value pairs and a consequent message type or category. Rules may address any number of message attributes in conjunction, but consequent actions are limited to assigning a different message type or changing the category assignment of the message.

For example, a rule might state that a message should be active if it is of category “Humor” and it arrives at location “media lab” as illustrated in Figure 3-5. Similarly, the same figure illustrates a rule that makes messages active which are from sender “mom” and are of category “Urgent-Personal.”

![Figure 3-5: Editing profile rules for the current mode](image)

Rules which apply to all modes can use either the category attribute or the status attribute in defining a consequence. For example, in Figure 3-6 one rule states that a message should be passive if it arrives after 2am and before 9am, and the other changes the category assignment to “Food” if the sender is “mcdonalds.”

Depending on type, messages are presented to the user in different ways. Active messages can pop up in a separate window with an accompanying alert noise, or simply appear in the user’s inbox, depending on user-specified preferences. A toggle button controls whether
passive messages appear in the user’s inbox, or are not shown at all. Ignored messages never appear.

Another aspect of the user profiles is the “known sender” list. Users can optionally create a list of senders from which messages are allowed to be active. All messages from unknown senders are automatically categorized as passive. This allows a user to reduce her use of the HM system to the most essential communications from a select group of people.

### 3.2.4 Message Management

One other interesting question involves the further management of messages once they have been delivered to a recipient. Some messages have multiple recipients, and need to be delivered to each one which enters the specified location. Others have no designated recipient at all, but must not be delivered to a single user multiple times.

The Hanging Messages system solves this problem with the following protocol. Messages are stored until they expire, regardless of delivery status. A recipient list is associated with each message on the server, even if the message does not specify recipients. This list keeps track of which users have already received the message, so that it will not be delivered to the same user again.
3.2.5 Summary

All of these filtering mechanisms are designed in order to maximize the relevancy of messages that are actually seen by users of the HM system. It is anticipated that the combination of category switching rules, profiles associated with operational modes, the option of known sender lists, and policies such as restriction of un-targeted messages will allow effective agent filtering, making the system a useful alternative communication channel for its users.

3.3 Implementation

In order to enable effective Just-In-Time messaging, the Hanging Messages system uses several recent advances for the client implementation, in conjunction with established technologies for the server.

3.3.1 Server

Server development concerns included reliability, and portability, as well as quick implementation time. Java was eventually chosen as the development language for several reasons. The Java specification[12] includes a comprehensive networking package, and although performance issues were considered, ease of implementation was a more significant factor. Java servers are widely deployed, and tend to be reliable, at least for moderate use, because it is easier to write bug-free code in Java.

The Hanging Messages server is a Java application that can be run on any platform with a Java Virtual Machine. It is implemented according to JDK 1.3.0 specifications; the latest Java Development Kit (JDK) release from Sun. The server currently runs on RedHat Linux 7.0[13], but has also been tested on Windows 2000[14]. Clients of the HM system use the TCP/IP protocol to communicate with the server.

Server code was written with the intent of making the code clear and robust, so that it would be easily extensible if additional features were desired.
3.3.2 Client

Determining Location

In order for the Hanging Messages system to function properly, clients must be able to quickly and accurately determine their locations. Some possible methods include triangulation using Radio Frequency (RF) transmissions or cellular phone towers, use of Global Positioning System (GPS) technology, and various beacon-type systems. Differential GPS (dGPS) is even more accurate than GPS, to less than a meter; however, it is not well supported or widely used.

For this project, GPS was selected as the most appropriate, because it is accurate to 10 or 15 meters, and does not require the use of a cell phone or installation of hardware at relevant locations. In addition, there are widely available commercial GPS modules that can interface with most types of mobile computing devices. Although GPS signals deteriorate indoors and near tall buildings, they are adequate for the purposes of the Hanging Messages system.

Hardware Selection

Hanging Messages clients must constantly communicate in real-time with the server. One method of achieving this is to equip the client with a wireless modem, so that client-server interaction can be done over the Internet. The choice of client hardware is thus limited to those mobile devices which have at least two input/output ports; one to connect with the GPS module, and the other for wireless Internet functionality. The TRG Pro[15], developed by HandEra, is one device that fits these requirements. The TRG Pro is essentially a Palm VII[16] that is equipped with an extra CompactFlash[17] slot and enhanced audio capability. However, the TRG shares the limited memory capability and restrictive development environment of other Palm OS devices.

The Compaq iPaq H3600 Handheld PC[18] contains a StrongARM 206 MHz processor and is built with 32 MB of RAM. Windows CE is the primary operating system, which offers several development options for client software. An extension cradle adds a CompactFlash interface, and the form factor of the handheld is among the best in its class. The primary problem with the iPaq is availability; at this point iPaqs are back-ordered for three months or more, and the extension cradles are even harder to acquire. Additionally, there is no
existing Java Virtual Machine for this platform. For these reasons the iPaq was excluded from consideration as a hardware platform for the Hanging Messages client.

Like the iPaq, the Casio Cassiopeia e125[19] comes loaded with Windows CE. With a 150 MHz processor, 32 MB of RAM, and a built-in CompactFlash slot, the Casio is comparable in functionality to the iPaq H3600. Although the device is slightly larger, the availability of the Cassiopeia e125 Pocket PC made it the best choice for this implementation.

Software Development

There are several options for Windows CE software development, including C, C++, Visual Basic, and Java. A basic desktop version of the HM client was first developed in Java in order to test server and communication implementations. Originally the plan was to develop the Pocket PC version in C++ using eMbedded Visual C++. However, the development environment assumed the use of Microsoft Foundation Classes (MFC), which would have been restrictive and unnecessarily complex for this application.

Finally it was decided to revise the desktop implementation for use with PersonalJava[20], Sun's Java Virtual Machine (VM) for Pocket PC. This process involved revising code so that it was compatible with JDK 1.1 specifications, eliminating use of unsupported classes, and optimizing code for use on a smaller, slower platform.

The Java implementation of the client software, running on the PersonalJava VM, currently functions at a slightly less than ideal speed, but it is adequate for prototype purposes. It is expected that Java VMs for mobile devices will continue to improve, as will the processing speeds of the devices themselves, so that a future implementation of the Hanging Messages client will be suitably responsive for constant use.

3.3.3 Communication Protocols

The server expects all communication from HM clients to use TCP/IP on a designated port. In order to achieve this, clients use the PocketSpider CF+ Wireless Modem[21], which enables a Cellular Digital Packet Data (CDPD)[22] link, to connect to the Internet.

The client server communication protocol was designed to allow message stores and queries by any client with a valid username. The server responds to valid inputs with appropriate responses, and all other inputs with descriptive error messages. Clients can terminate communication at any time, and can abort in the midst of a logical series of
commands. The complete Hanging Messages server communication protocol is documented in Appendix C.
Chapter 4

Using Hanging Messages

4.1 Client Usage

4.1.1 Starting a Session

When the HM client software is started, the user is presented with a login screen, as shown in Figure 4-1. A user must authenticate himself at the beginning of each session, before being allowed to send or receive messages. To ensure user privacy, messages are not displayed in the inbox until the user is verified.

![Login Screen](image)

Figure 4-1: Logging into the client

During the login process, the user's name and password are checked against server data, and the most updated category hierarchy is retrieved from the server. For this reason, the process of logging in takes a few seconds to complete.
4.1.2 Composing Messages

To compose a new message, a sender can either tap the Compose button at the bottom of the graphical interface, or select the Compose menu item, under the Messages menu.

Figure 4-2: Composing a new message

Figure 4-2 shows the compose dialog as well as an example message. The Location and Delivery Time Period fields are required; all others are optional. If no recipient is specified, the message is marked as un-targeted, and a category specification is required. For convenience, when the user composes another new message, the location and delivery time fields are pre-populated with values from the last message.

4.1.3 Incoming Messages

Incoming messages which are not filtered away by the agent arrive in the user’s inbox, which is displayed as a list. In this implementation of Hanging Messages, there is no other place to store messages besides the inbox, and thus no concept of a directory structure. This feature may be added in the future, if usage patterns tend towards storing large numbers of messages.

The messages are ordered by delivery time, but message senders and subjects are also shown in the inbox header list, as illustrated by Figure 4-3.

To read a message, the user can tap twice on the message header, and a dialog will appear which displays the entire message.
4.1.4 Login Options

There are two options that are accessible from the login screen: Change Server, and Location Mode.

The default message server for all Hanging Messages clients is soundwave.media.mit.edu.

To change the server used by the client, the user can tap the Change Server button on the login screen, and enter the name or IP address of the new server. Figure 4-4 shows the interface for changing the default server.

Location can be determined automatically by the device, by reading GPS coordinates from an attached module, or manually, through user settings.
To change the mode through which location is determined, the user can tap the *Location Mode* button on the login screen, or the *Set Location* item, under the *Options* menu. The *Location Mode* dialog, depicted in Figure 4-5, allows the user to select the desired mode from a dropdown list. If *Manual* mode is selected, the user should also enter the latitude and longitude coordinates of the desired location at this time.

### 4.1.5 Message Visibility

The user can choose the method of message delivery for both passive and active messages. Active messages can either pop up of their own accord, with an accompanying alert, or be quietly delivered to the user’s inbox. The modes can be toggled by tapping either the *Popup/ Quiet* button at the bottom of the user interface, or the *Popup/ Quiet Active* menu item, under the *Messages* menu. Both the *Messages* menu and the *Options* menu are illustrated in Figure 4-6.

Passive messages can also appear in the inbox alongside active messages, or they can be hidden from view entirely. Either the *See/ Hide Passive* button or the corresponding menu item can be used to toggle passive message visibility settings.
4.1.6 Message Retrieval Options

Location Names

Location names can be selected from a dropdown list, or typed in manually and verified against the user's existing location names. To create a new name-location association, the user can use the Save Location feature, or add the mapping explicitly to a plain text file.

The Save Location option allows the user to save the location that she is currently at, and assign it a nickname that can be used in future messages. To use this option, the user taps the Save Location item, under the Options menu, and enters the desired nickname in the text
field, as shown in Figure 4-7. Subsequently, the nickname can be used while composing new messages, and will be substituted for location coordinates in matching incoming messages.

Operational Modes

![Image: Change Mode interface]

Figure 4-8: Users can change their current mode of operation

The application assumes that most users have different operational modes such as "Home" or "Work" in which they have different message reading behaviors. Accordingly, each user has multiple modes, each with separate user profiles.

In the current version of Hanging Messages, users can change their operational mode by tapping the Change Mode item, under the Options menu. Figure 4-8 shows the user interface for changing modes. In future versions, modes may automatically change based on current user location.

Search Radius

By default, the HM client searches for messages within a certain radius: 100 meters. To change the search radius of a particular client, the user can tap the Set Radius item, under the Options menu, and enter the desired search radius, as illustrated in Figure 4-9.

A possible improvement to this method would be the addition of automatic search radius determination, based on speed of travel. For example, a user on a bike or in a car would move more quickly than a user on foot, and would probably want to use a larger search radius, since she could quickly reach a greater range of locations.
Set Location

The Set Location feature was previously described while discussing modes of determining location.

4.1.7 Filtering Options

Several filtering options are available to the user, to help reduce message volume and increase message relevancy. The filtering options are accessible from the Filtering menu, illustrated in Figure 4-10.

As part of its filtering scheme, the Hanging Messages client maintains profiles for each
user; one for each operational mode, and an additional profile which applies to all modes. A user can edit her current mode profile, or her universal profile, by tapping the Edit My Profiles item, under the Filtering menu. The editing dialog is shown in Figure 4-11.

Profiles consist of logical rules, constructed of “IF-THEN” statements based on attribute-value pairs. Users edit their profiles by adding or deleting rules from their profiles. The editing dialog allows the user to edit either the profile corresponding with his current mode of operation, or the profile associated with all modes.

To add a rule, the user first selects a message attribute from the dropdown list, and then inserts a corresponding value into the value field, as shown in Figures 4-12 and 4-13.
Multiple attribute-value pairs can be added for any given rule, using the AND button. Next, a consequence should be defined for messages that match all of the stated conditions, by using the THEN button. Finally, the rule can be added to the profile using the Add button.

Existing rules can be deleted from a profile. The user can tap on any rule in the Existing rules list to see its complete definition, such as in Figure 4-15, or delete the rule by tapping the Delete button.

To reduce message volume, the HM client provides the option of filtering all incoming
messages against a *Known Sender* list. If this list is enabled, only messages which originate from a sender on the list may be made active.

To toggle the use of the *Known Sender* list, the user can tap the *Enable/Disable Known Sender List* item, under the *Filtering* option. She can also tap the *Edit Known Sender List*, to edit the senders on the list, through the interface shown in Figure 4-16.

The use of this dialog is fairly intuitive. To add a sender, the user can type the sender’s username in the text field and tap the *Add* button or type `<return>`. To delete a sender, she should tap the username in the *Existing Known Senders* list and then tap the *Delete* button.
4.1.8 Help Manuals

Since some of the client-side features are quite complicated, detailed help manuals are provided for the major functions of the Hanging Messages client. These manuals explain concepts to the user, such as KNOWN SENDER LIST, location modes, and profile rules.

![Help menu](image)

Figure 4-17: Help topics available

To use help, the user can select the Help item from the Help menu, and tap the desired entry on the help topic list. Figure 4-17 displays the list of available topics; there is at least one help manual entry associated with each menu option.

Figure 4-18 shows an example help manual entry, designed to assist the user with composing a new message.

![Compose help manual](image)

Figure 4-18: Help manual entry for “Compose”

53
The other help manual entries provided by the client are similar in language and level of detail.

4.2 Installation and Configuration

4.2.1 Server

The Hanging Messages system was designed to be flexible but simple to use. Server installation requires unpacking the Java class files, and configuring system users and (optionally) the category hierarchy.

The server has three configuration files, all of which are plain ASCII text and can be easily edited:

- .categories
  
  This file stores the updated system category hierarchy. Changes to this file are propagated to any client that logs into this server. The syntax of a line in this file is:
  
  **Parent:** Child1 Child2 Child3
  
  It is assumed that the first line refers to the root node. Categories cannot be removed once they have been added to the server.

- .messages
  
  This file stores some data that allows the server to recover message tracking state if it is stopped and restarted. No configuration is necessary. In fact, this file generally should not be tampered with.

- .users
  
  This file stores user verification data. By default there is one user:

  **testuser testpasswd**

  Unfortunately, at the moment, the data is stored in plain text. This makes it easier to edit but also makes the data very insecure. In the future this should be changed. The syntax of a line in this file is:

  **user password**
Currently the password checking method is case insensitive. A new user can be added by adding a new line to the file.

Once installed, the server can be started with a one line Java command.

![Figure 4-19: Starting the server](image)

The server listens for connections on port 5000; firewalls may need to be modified to allow this behavior.

### 4.2.2 Client

The client also includes Java class files and a few configuration files. Default settings for user profiles and modes can be changed by editing the configuration files. The addition of a new mode requires the creation of a corresponding, appropriately named file.

The client configuration files are described below:

- `.inbox.<user>`

  This file stores the current state of the user’s inbox. It should not be modified.

- `.modes.<user>`

  This file stores the operational modes of this user. Each mode should have an associated configuration file. The modes can be edited by hand, or through the graphical interface.

- `<mode>.<user>`
Each mode has a corresponding configuration file, which stores the rules associated with that mode. The naming convention is the mode name in lowercase letters, with all spaces removed. The rules can be edited by hand, or through the graphical interface. The syntax for the rules is as follows:

\[
\text{<attr1> |<value1>| AND <attr2> |<value2>| THEN <status>}
\]

Any number of attribute-value pairs can be listed. The consequent status must be: active, passive, or ignored. The first line of the file must specify the default message status.

- .rules.<user>

This file stores the preference rules that apply regardless of current operational mode. The rules can be edited by hand, or through the graphical interface. The syntax for the profile rules is as follows:

\[
\text{<attr1> |<value1>| THEN <attr2> <value2>}
\]

At this time the only supported consequent attributes are category and status. Again, any number of attribute-value pairs can be listed, and the first line must consist of the default message status.

To add a new user, copy each of the template files (each uses the naming convention .<file>.testuser) and change the suffix to the new user's username. In the future, an “Add User” feature for the graphical interface should be added to automate this process.
Chapter 5

Evaluation

5.1 Significant System Characteristics

The development of a system such as Hanging Messages involves making intelligent assumptions about usage patterns, hardware availability, and other factors, and using them to justify design tradeoffs. At every stage, alternatives are considered and decisions are made based on developer’s priorities. Eventually, the philosophies of the designer become evident in the form of strengths and weaknesses in the final product.

This section discusses a few particularly significant weaknesses, as well as some core strengths, and explores the alternatives and the reasoning behind the associated design decisions.

5.1.1 Connectivity and Bandwidth

The Hanging Messages system relies heavily on a continuous, relatively fast wireless Internet connection to help achieve the goal of relevant Just-In-Time communication. Not only does the client need to communicate with the server to retrieve and send messages, client-server communication is used for user authentication, transmission of the category hierarchy, and other ordinary system tasks.

Obviously, some connectivity is required to allow basic system functionality. However, had this been a major concern during the development process, additional measures might have been implemented to minimize the effects of a lost connection.

For instance, the server could transmit messages to recipients regardless of location
and time. In this case, client agents would monitor the user's whereabouts and determine matches between message attributes and user context; the user would receive the message whether or not the client had connectivity while she was at the designated location. An additional benefit would be increased user privacy, since the server would never receive any client location information.

The downside of this approach is that a large number of messages might be unnecessarily transmitted, and would need to be stored in memory or on disk. The privacy of the sender might also be compromised, since the recipient could potentially gain access to messages that were not meant to be delivered.

The other major issue with current wireless technology is bandwidth. Even with the relatively small amount of data that is contained in a single message, the application takes about ten or fifteen seconds to finish sending a new message. The login process, although shorter, also takes several seconds to complete. While some delay is unavoidable, it is possible that the client-server communication protocol could be streamlined to some extent. This improvement was not included in this version.

This implementation of the HM application assumes good connectivity, primarily because the advantages of added workarounds for poor connectivity were not pressing enough to warrant the extra time and effort they would require. As a prototype, the focus of developing the system was to demonstrate a new channel for effectively communicating relevant messages, that would be useful in certain situations, rather than to guarantee message delivery. It is expected that wireless technologies will continue to advance, resulting in better connectivity and higher bandwidth, and smoother operation of the Hanging Messages system.

5.1.2 Software Concerns

Both the HM client and server applications are written entirely in Java. For the server this was a logical choice, since Java has a simple, comprehensive networking packages, and is easily portable between operating systems. Since server load was expected to be relatively low, the performance issues associated with Java servers were not considered major concerns.

However, for the client, portability was not a consideration; the client application currently can only be used on the Pocket PC (Windows CE 3.0) platform. The obvious choice might have been Visual C++; the resulting applications have a tighter integration into
the user interface, and are generally considered to be faster than Java programs. In this case, it was decided that these qualities were not significant enough to justify the added development time.

Java was used to develop a desktop version of a client, in order to thoroughly test the server. When implementation was begun on the handheld version, the use of Visual C++ including MFC, was found to be unnecessarily complex for this application. In the end, the desktop version was revised and rewritten for use on the Pocket PC, and found to be adequate for the purposes of the system. The development of future implementations with more robust functionality might require significant software revisions.

5.1.3 Configurability

All aspects of the client application were designed with maximal user configurability in mind. Reasonable defaults were initialized for all configurable items, but a key goal was to give the user as much flexibility as possible, to adjust system functionality to her own needs and preferences.

The message visibility options are one example of this characteristic. Although the system only recognizes three types of messages: active, passive, and ignored, it allows the user to select the visibility for the message types; both active and passive messages can be made more or less intrusive. Although the current version requires the user to indicate his preferences through the interface, a future implementation might learn user visibility preferences and associate them with particular temporal or spatial attributes. This improvement would achieve a similar result, but would save the user the effort of actively configuring the application.

Another configurable user option is the known sender list. Users who want to stringently restrict incoming messages can enable a filter that compares all message senders against a list of allowed users. Only those messages whose senders are on the list may be made active. This function caters to users who find their message volume is undesirably high, or users who wish to receive messages only from a select group of people.

These and other functions were included in the client application in an effort to allow users greater flexibility. Every individual may have different expectations of the same system, and configurability is one way to fulfill the needs of more users. The end result is an increase the number of situations in which the system serves its function, and consequently,
a more useful application.

5.1.4 User Privacy

Ensuring user privacy is a major concern for emerging location based services. Consumers are uneasy with the idea that companies may be tracking their whereabouts, using the data for their own purposes, or even inadvertently allowing other parties access to this information. Some may feel that the benefits of these new applications are not worth the price of their privacy, and choose not to use the services.

Many people are also hesitant to use personalized services which collect user preferences and usage information. A prime example is Amazon.com, which not only tracks user purchases and clickstream patterns, but stores credit card, address, and other personal information. Although Amazon’s current privacy policy states, “we are not in the business of selling [customer information] to others,” the fact remains that Amazon controls this information, and does not guarantee its security in case of a change in business model or privacy policy.

Since Hanging Messages is both a location aware application and a personalized service that stores user preferences, privacy is an especially important concern, which is reflected in a few key design choices.

All user preferences in the system are stored on the client side, on the user’s personal handheld device. The downside to this decision is that messages cannot be filtered by the server. Consequently, a greater burden is placed on the client, which does all message processing, as well as the communication channels, through which more messages are sent. However, privacy concerns were considered more pressing than performance issues, and locating user data on the client is the only solution that allows each user complete control over his personal information; the information becomes as secure as the device itself.

Similarly, in order to allow users more control over their location information, tracking is done by the client, which then queries the server for messages at a particular location. The server currently discards the queries as soon as matching messages are delivered. Although the user has no guarantee that the server is not storing location information, she does have the ability to manually set the client location. Because the responsibility for determining location rests solely on the client, the server has no way of knowing if the user is actually at a location, or whether she merely wishes to receive messages located there.
To ensure that only the recipient has easy access to a message, no messages are shown in the inbox until a user has been authenticated. This slight security precaution ensures that under regular use, multiple users of the same handheld device will not see each others messages. Currently messages are stored as plain text in files which can be read by any user. To complete the privacy scheme, messages should be encrypted before being stored in memory. This improvement may be included in a future implementation.

5.2 Usability

Hanging Messages was designed for use by the general public; both technical and non-technical users. Given this goal, the client interface should be as simple and intuitive as possible. In order to help achieve this result, the early implementations were shown to a few (5) test users, who were encouraged to provide usability feedback.

5.2.1 User Feedback

Comments included both superficial, layout related issues, and a few process oriented changes. Several specific suggestions were made:

- Users should be able to tap `<return>` at the end of a text field to achieve the same result as tapping the `OK` button.

- Detailed help manuals are necessary to explain features such as Known Sender List and Popup Active, etc.

- The menu divisions were arbitrary and should be reorganized. In addition, it was confusing to have See Passive and Hide Passive both in a menu, but only one of Enable Known Sender List and Disable Known Sender List was shown.

- The `Edit My Profiles` dialog was completely confusing for several users. The buttons should be named more consistently and the layout should match the process flow.

- The user should be able to tap twice on each existing rule, for more details (like the inbox functionality).

- Location coordinates should be translated to names for incoming messages as well, if mappings are available.
• *User* and *Pswd* on the login dialog should be renamed to *Username* and *Password*, even if the corresponding text fields would then need to be shortened.

• There should be an outbox, where users can see what messages they have sent in the past.

• A directory structure or some way for users to organize received messages would be useful.

Several of the users initially wondered how Hanging Messages would be useful. After being presented with a few examples, they generated scenarios of their own. One user suggested that the system could be used to distribute rental car assignments; messages could be hung in the rental parking area for each customer, to inform them of the location of their designated vehicle.

### 5.2.2 Implementation Changes

All superficial layout suggestions made by users were incorporated into the interface. For example, the menus and menu items were reorganized into a more comprehensible format, with message related choices under a *Messages* menu, filtering items under a *Filtering* menu, and other user configurable options under an *Options* menu.

Functional changes were implemented if they were supported by the existing data structures in the client; location translation for incoming messages was easily added, as were detail dialogs for existing profile rules. Adding help manuals involved quite a bit more effort, but as the application was not easily understood without their aid, they were made a priority and quickly implemented.

The *Edit My Profiles* dialog was completely redesigned.

Several possible layouts were considered and presented to test users for further input. The objective was to create an interface which required only as much explanation as the help manual provided. The resulting dialog, shown in Figure 5-1, is a vast improvement on the initial layout, as it matches the user thought process much more closely.

The idea of having multiple message storage locations, in the form of a directory structure or at least an outbox, was considered briefly, but it was decided not to implement this functionality in the current version. Because the system was designed for real-time messag-
Figure 5-1: The evolution of the Edit My Profiles dialog

...ing, users should not often feel the need to examine very old messages, whether incoming or outgoing. The inbox was considered enough storage space for typical use.
Chapter 6

Related Work

Hanging Messages is a system that enables Just-In-Time user communication, with an emphasis on using context to reduce message volume. This chapter describes several related projects; other “location-aware” applications which use spatial information in different ways, and a few agents which perform message filtering.

6.1 Find-The-Nearest

As Global Positioning System (GPS) and other location technologies have emerged in recent years, location-based applications have been introduced as well. Commercial wireless providers, viewing location based services as the “killer application” for wireless Internet, have introduced several services of the “find-the-nearest” variety. These applications allow customers to use their wireless mobile devices to search for the nearest retailers, restaurants and other services. In the United States, Vicinity’s BrandFinder[23] is a popular choice; it is provided by Verizon, OmniSky, AT&T Digital PocketNet, Palm VII, and Nextel Online. BeVocal adds a voice recognition twist, allowing customers to speak the name of a business, to find the nearest office.

6.2 Location and Customization

Location monitoring also allows information customization. In Germany, CT Motion and T-Mobil have teamed together to allow companies to monitor their employees, and intelligently send them relevant information based on location[24]. InfoMove has taken a different
approach, focusing on drivers rather than individuals moving by foot. Their services include “real time personalized traffic updates, real-time vehicle monitoring, profiled location-based advertising, [and] emergency assistance and concierge services[25].”

Several projects at the Media Lab have involved location based services. In Wearable Computing, a project entitled “Memory Glasses” served as a short-term memory aid and reminder system[26]. Location, time, and user activity were all considered when triggering reminders. This is similar to the Rhodes’ Remembrance Agent[27], which in its wearable form, uses location as a context for suggesting reminders.

6.3 Location and Information

At Rutgers University, researchers have developed a system for marking web sites with specific locations, and subsequently retrieving those sites that are nearest in location, or accessing pages whose contents depend on user location[28]. This system is intended for use with mobile clients, and indeed location-based content may improve the wireless web experience significantly.

Impulse[10] is an ongoing project in the Software Agents Group that combines e-commerce with “brick-and-mortar” commerce by using agents on wireless mobile devices to negotiate purchases with retailers near the user’s location. A typical scenario involves a consumer who passes a shop which carries a product that she wants. The handheld alerts the user, who proceeds to examine the product at the store itself. Price comparison or other information can be accessed through the handheld, before making a decision to purchase. Various implementations of the Impulse scenario deal with different uses of location information.

Wherehoo, developed by Jim Youll, is a search engine optimized for location-specific searches, used as an infrastructure element in Impulse implementations. Periscope[29], a related project, includes a telescope-like device that feeds direction and distance parameters to Wherehoo, and retrieves data placed at the specified location.

Annotate Space[30], a project at New York University’s Tisch School of the Arts, focuses on associating “thought-provoking [audio-video] experiences” with locations, to educate people about the places they visit. Her premise is that the use of find-the-nearest and other directory services reduces the user’s knowledge of the physical world. To restore some of this
experience, site-specific educational or entertaining information should also be associated with locations.

6.4 Location and Messaging

Although location based applications have been developed for many purposes, there are only a few which address person-to-person communication. Some recent projects which incorporate location into notes or messages are described below.

6.4.1 Locust Swarm

The Media Lab’s Wearable Computing group recently worked on a project called “Locust Swarm” which included the development of a location-based messaging system, which had a sophisticated location detection scheme that could determine location within a building[31]. The Locust Swarm emphasized privacy by allowing the user to control how much information is shared with others.

6.4.2 HaikuHaiku

A commercial application called HaikuHaiku by a company called Neoko allows people to post 3-line haikus at specific locations[32]. Currently the haikus can be read through wireless devices but must be input through a browser. Users must register through the company’s website in order to view or input haiku messages. No filtering services are available yet.

6.4.3 Geocorder

The concept of location-based messaging has also been presented on the “halfbakery” website, which is catalogs new ideas and reader responses. The “Geocorder” proposal suggests leaving audio messages through small devices with embedded GPS, wireless, and voice I/O. Since halfbakery discusses ideas rather than actual implementations, no specifics are mentioned, but reminders, graffiti, recommendations and virtual messages boards, are all mentioned, as are filtering and categorization[33].

67
6.4.4 comMotion

Another Media Lab project, comMotion[34], is "a location-aware computing environment" which allows a user to associate reminders with locations, so that she will be reminded of her grocery list while at the supermarket, or of her book wishlist while near the bookstore. comMotion allows any user to send any other user a reminder at some specific location, determined by GPS.

Interesting features of comMotion include location learning, a scheme through which the system tracks user travel patterns, and asks the user to identify frequented locations by name, and a speech interface, which is perhaps more useful than a graphical interface to users on the go.

Hanging Messages is like comMotion in basic message placing functionality. Both systems allow users to place reminders or messages at a certain location, but have other unique features and enhancements. The most significant difference is that while comMotion focuses on improving the interface and the user experience, Hanging Messages focuses on using location, time, and other context to constrain this new channel of communication to useful level.

6.4.5 GeoNotes

GeoNotes[35] is another system that allows "annotation of physical locations" with virtual messages or notes. A GeoNote can be placed at the current location, or any location specified in a directory. The creator can perform access control for the note, by specifying a list of approved "buddies" or allowing the note to be universally visible.

Users can view GeoNotes in one of three modes: push, push/pull, and strict pull, where push indicates that notes can actively alert the user, and pull means that the user must perform a search before she receives any notes. GeoNotes allows users to filter notes by usage; users may choose to view notes “put by friends” or “read by friends” or “most popular.” Users are further encouraged to categorize notes as “private” “commercial” or “information” for the benefit of other users.

GeoNotes and Hanging Messages both allow users to communicate with location specific messages. The access control feature of GeoNotes is essentially equivalent to the recipient functionality in Hanging Messages, and both systems use categorization to different degrees.
While GeoNotes focuses on filtering notes by usage patterns, determining relevancy by examining the reading and posting behavior of other users, Hanging Messages uses personal profiles, a more complex categorization scheme, and some other personalization mechanisms.

### 6.5 Just-In-Time Filtering

The other major emphasis of Hanging Messages is on message filtering. New communication channels are advantageous only if the resulting communication is helpful and relevant to the users. Without effective filtering mechanisms, increased messaging capability may add to information overload. This is especially true when messages are delivered in real-time, taking the user’s attention away from her current task. This section discusses several existing agents that attempt to sift out extraneous messages, and analyzes their filtering methods.

#### 6.5.1 Pigeon

One example of a filtering agent is demonstrated in the Pigeon project[36], a wireless two-way messaging system. Pigeon focuses on several aspects of wireless messaging, including mitigation of mobile device constraints, greater and more improved messaging functionality, and architecture issues. Agent filtering is used by Pigeon to provide message processing and personalization services.

Pigeon uses the concept of send-side and receive-side processing. Messages are “flexible” and consist of combinations of basic building blocks, which may be plain text, rich text, or in notation format. “Care for \choice{lunch|dinner}\optional{soon}?” is an example. The sender puts together building blocks of a request, which are send-side processed before being sent. The recipient can perform receive-side processing, evaluating the reply components of a message.

In the Pigeon project, agents are used for two purposes; they expand and compress encoded messages, reducing uplink bandwidth usage, and they provide personalization services. These agents are located on the “home network” rather than with the users, in order to handle disconnected devices. Instead of providing real-time services, Pigeon strives to conceal the deficiencies of current mobile wireless technology using agents.
6.5.2 E-mail

Existing commercial e-mail systems such as Qualcomm’s Eudora and Microsoft Outlook also provide ways for the user to filter messages. Like in Hanging Messages, these filters generally consist of rules which examine message attributes such as sender, recipient, subject, or body. However, mainstream e-mail filters do not allow multiple profiles for a user, and have fewer attributes to use in defining filtering rules.
Concluding Remarks

7.1 Future Implementation

Several feature extensions which would have contributed significantly to the utility of Hanging Messages were not implemented in this version, due to time constraints. Since the focus of the project was on building the basic messaging functionality and adding those core filtering mechanisms necessary to make the system practically usable, these features were deferred, to perhaps be implemented in the future.

7.1.1 Message Modification

Currently, messages can only be added, and not deleted or modified. In some cases, it might be useful to annotate or entirely remove an existing Hanging Message. For example, if a request were sent to a group of people, and one member of the group fulfilled the request, she could annotate that message to indicate that she had done so, or delete the message altogether, reducing message volume.

However, allowing message modification introduces the concept of access control. For example, in this case, only members of the group to which the request was sent should be able to remove or modify the message. The sender would have to specify access control permissions while composing the message, if she wished to allow message modification. On the server side, modifications would have to be written to the messages on the server, or perhaps new message would be inserted and the old message would be invalidated.
7.1.2 Message Reactivation

When a message is received, it can be deleted or kept in the user’s inbox, but will not be delivered to the user again. Originally it was intended that the user should have the option of “reactivating” the message, so that it would be re-delivered if the user returned to the same location. This behavior would be useful if the user received a reminder or request, could not complete the task at that time, but intended to do it later. He could reactivate the message, so that the message would be delivered the next time he approached the same location.

Implementation of this feature would require changes to the message manager. Instead of just tracking which recipients had received which messages, the message tracker would have to allow the client to change the delivery status of a message. Although this is not a difficult change to make, it does require adding functionality to the message server, as well as to the server communication protocol.

7.1.3 Multiple Locations

It would be a relatively simple task to add the ability to send a message to multiple locations. However, extensive use of this feature might reduce the location-specificity of messages, defeating the purpose of the system. An alternative would be to allow sending a message to locations of the same type; for instance, a message could be sent to all supermarkets in a 5 mile radius. Implementation of this feature would require adding translation capacities to the client, and the compilation of a directory of places, classified by location type.

7.1.4 Complex Filtering Rules

Profile rules must be flexible in order to accommodate wide variation in user preferences. Although the existing scheme of IF-THEN statements with AND clauses does allow a fairly large amount of customization, it does not capture all of the configurations that could be specified if OR and NOT statements were allowed. Adding these types of clauses would involve re-tooling the message filters, to do more sophisticated parsing, and a slight revision to the user interface.
7.1.5 Recognition Groups

The existing *Known Sender List* is intended to be similar to "buddy lists" such as those in instant messaging systems. Each individual can indicate which users she knows, and can optionally receive messages only from those users. A similar potential feature is the idea of recognition groups. Users can join groups of users, with the result that all members of the group are known to each other and accept messages from each other.

This would be useful in a community such as a workplace, where all employees of a company can belong to a group and thus become known to each other. A recognition group must be owned by one or more users, who administer group membership. This change would require fairly significant modification on both the client and server sides of the application.

7.1.6 Agent Learning

An extremely useful addition to Hanging Message would be learning capability for the user agents. Currently users edit their preferences manually, for locations, profile rules and the known sender list. If users could edit their preferences incrementally, as they use the system, it would be a vast improvement.

For example, if a user sends messages to the same recipient several times, a learning agent might ask the user if she wants to add the recipient to her known user list. Similarly, if the user never seems to receive messages before 9am, the agent might inquire if the user wants to add a rule to her profile to screen out messages arriving before 9am.

Although the usability of the system would be greatly improved, adding this feature might require substantial alterations to the client agent.

7.1.7 Encryption

For a system that enables person-to-person communication, security is a major concern. Users must entrust important and potentially confidential data to the system, making data encryption a key issue. This concern was not addressed at all in the existing application. However, a more mature implementation would certainly have to allow client-side message encryption and decryption, so that the messages stored on the server would not be accessible should the server be compromised.
7.2 Conclusion

Wireless technology, mobile computing, and other recent innovations will continue to advance and develop. However, to effectively take advantage of these technologies, new applications must provide a real, useful service. Users must feel that the benefits of the new services far outweigh any disadvantages. Hanging Messages addresses this issue by emphasizing message filtering, which is the key to making Just-In-Time messaging a viable, widely used communication channel.
Appendix A

Server: Class Dependencies

Legend: module A depends on Const
module B depends on module C
Appendix B

Client: Class Dependencies

Legend: module A depends on Const module B depends on module C
# Appendix C

## Server Communication Protocol

<table>
<thead>
<tr>
<th>client command</th>
<th>effect</th>
<th>initial state</th>
<th>final state</th>
<th>server response</th>
</tr>
</thead>
<tbody>
<tr>
<td>RST</td>
<td>reset server state</td>
<td>(any)</td>
<td>LISTENING</td>
<td>RESET</td>
</tr>
<tr>
<td>IDT&lt;user&gt;</td>
<td>identify user</td>
<td>LISTENING</td>
<td>IDT</td>
<td>server version ERR&lt;error&gt;</td>
</tr>
<tr>
<td>ACT&lt;action&gt;</td>
<td>perform action</td>
<td>IDT</td>
<td>VERIFY</td>
<td>none ERR&lt;error&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CATEGORIES</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>STORE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>QUERY</td>
<td></td>
</tr>
<tr>
<td>SNDR&lt;sender&gt;</td>
<td>specify sender</td>
<td>STORE</td>
<td>STORE</td>
<td>none</td>
</tr>
<tr>
<td>RCPT&lt;recipient&gt;</td>
<td>specify recipient</td>
<td>STORE QUERY</td>
<td>STORE QUERY</td>
<td>none</td>
</tr>
<tr>
<td>SUBJ&lt;subject&gt;</td>
<td>specify subject</td>
<td>STORE</td>
<td>STORE</td>
<td>none</td>
</tr>
<tr>
<td>CAT&lt;category&gt;</td>
<td>specify category</td>
<td>STORE</td>
<td>STORE</td>
<td>none</td>
</tr>
<tr>
<td>BEG&lt;begin time&gt;</td>
<td>specify beginning of delivery time</td>
<td>STORE</td>
<td>STORE</td>
<td>none</td>
</tr>
<tr>
<td>END&lt;end time&gt;</td>
<td>specify end of delivery time</td>
<td>STORE</td>
<td>STORE</td>
<td>none</td>
</tr>
<tr>
<td>LOC&lt;location&gt;</td>
<td>specify location</td>
<td>STORE QUERY</td>
<td>STORE QUERY</td>
<td>none</td>
</tr>
<tr>
<td>BODY&lt;body&gt;</td>
<td>specify body of message</td>
<td>STORE</td>
<td>STORE</td>
<td>none</td>
</tr>
<tr>
<td></td>
<td>end of store or query action</td>
<td>STORE QUERY</td>
<td>DELIVERING</td>
<td>OK NAK OK</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DELIVERING</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LISTENING</td>
<td>ERR&lt;error&gt; ABORT</td>
</tr>
<tr>
<td>NEXT</td>
<td>ask server for next message</td>
<td>DELIVERING</td>
<td>DELIVERING</td>
<td>message DONE</td>
</tr>
</tbody>
</table>

79
Bibliography


