Intelligent Messaging Service in an InfoStation-based University Network

Liam Merwick, Ivan Ganchev, M´airt´ın O'Droma
Telecommunications Research Centre.
University of Limerick, Ireland

Abstract: This paper shows how a communication infrastructure consisting of mobile devices, InfoStations and an intelligent gateway can be combined to create a messaging system for a campus sized area. It allows for fast and efficient delivery of messages to a group of users through the provision of two-tier ad-dress space architecture. A particularly novel part is the creation of an intelligent central message processing agent which decides which device, and in what format, the message should be forwarded to base on a user's preferences and the presence (or not) of their registered devices on the network. The benefit of this 'Intelligent Assistant' is the delivery of messages to a user on the device they are most likely to be able to access at any moment in time and thus deliver messages in a timely manner. A system was successfully prototyped which could deliver messages in SMS and email format and was designed so that further message formats could easily be integrated.

Keywords: mobile messaging, InfoStation, Blue-tooth, SMS, two-tier address space

1. Introduction

In recent years communications technology has advanced considerably and people own multiple communication devices (mobile SMS/MMS, PDA, desk-top computer, laptop) and have many means of being contacted (email, phone, Instant Messaging). Often, when trying to get a message to a person, there is no way of knowing the best way of notifying him/her in a timely manner - he/she may be away from their desk or have their phone switched off. When sending a message, some protocols allow for the sender to re-quest to be notified when the message is received or read [Faj98], but the sender is still required to decide which of the recipient's devices/addresses to send the message to (possibly without any knowledge of the recipient's schedule or location). It is of limited benefit to either party if the message is sent to a device that the user has no access to at that time.

The aim of our research was to build a system which allows a person to send a message to another user in the 'best' possible way, i.e. the messaging system can dynamically decide to route that message to the other person based on that person's current location, contact preferences and other criteria (e.g. urgency or price the user is willing to pay). The end result should be that the recipient(s) should get the message in a more timely manner and in a way most suited to them as they will have more control over how and where the messages are received.

The major system components (Fig. 1) are an application that runs on the mobile device (MobileApp), one or more InfoStations through which the mobile devices connect to the messaging system, a central processing application (MessageRedirector) and a web interface which allows a user to update his/her details and addresses (ContactApp).

The system prototype was built upon the service-orientated network architecture described in [Iva06], [Iva07], which is used to deliver lectures, tutorials and tests on a University campus network. However our prototype has implemented software which has a MessageRedirector (cf. Section 4.5) as the central processing component instead of the InfoStation Centre in that architecture.

The remainder of the paper is organised as follows. In Section 2, we introduce some typical use cases of an intelligent messaging system. Section 3 describes related work in this field. The architecture of the messaging system is defined in Section 4 and how it is implemented in Section 5. Finally, we describe our conclusions and future work in Section 6.

2. Typical Use Cases

The operation of the messaging system can be

best described by some typical examples of its use.

2.1. Sending a Message

While parking his car at the university car-park, a lecturer receives an urgent message about a meeting rescheduled for the same day. He notices that the meeting will be held in the same time as a lecture. The lecturer urgently needs to broadcast a message notification to the entire student class about canceling/postponing the lecture. The lecturer types and sends the message on his mobile device which is connected to the nearest base station of the messaging system (e.g. deployed at the car park). The messaging system then decides what is the most appropriate, quickest and cheapest way of delivering this message to each student in the class according to his current individual location (and device in possession) specified in his profile. All registered users (lecturers and students) have profiles containing, among other things, information about the best way of forwarding urgent messages to them at any particular moment, e.g. by SMS/MMS, email, fax, voice mail or other-wise.

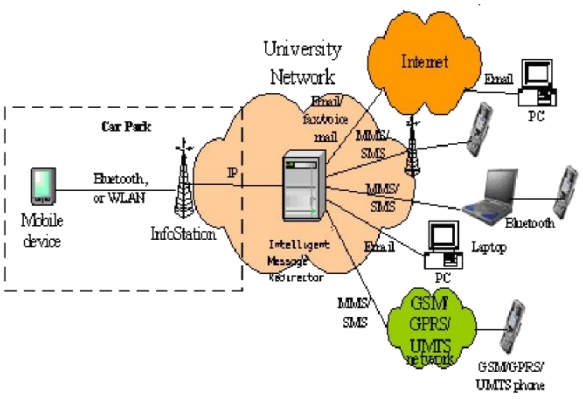

Figure 1: Messaging System Components

2.2. Changing a User's Preferred Contact Address

A student studying in the library or a professor giving a lecture, to avoid distractions, wishes to receive messages via email rather than via SMS. Similarly, a lecturer working at his computer in his office may prefer to handle incoming messages via email. The user logs into the web interface. Following a successful login, the user's details, including a list of addresses, are displayed. The user raises the priority value in the field associated with his preferred email address so that it is the highest priority address. The web interface validates the details before inserting them into the database. The Intelligent Assistant that is part of the Message Redirector will route any messages received to the user's email account until the priorities are re-adjusted.

2.3. Adding an address to a profile

The user logs into the web interface which interacts with the database back-end containing the users' details and addresses. Following a successful login, the user's details are displayed including a list of addresses. The user selects the option to add a new address and fills in the requested details. The web interface validates the details before inserting them into the database.

3. Related Work

There are numerous software applications available which allow for communications via mobile devices. These applications are provided by both the mobile carriers (e.g. standard applications such as SMS and MMS) as well as open source (e.g. numerous down-loadable Java applets) and proprietary (e.g. Nokia Presence) offerings which build upon the platforms provided by the carriers (e.g. Brew, Vodafone Live!) and device manufacturers (e.g. Nokia's Series 40/60).

The research undertaken, and described in this paper, focuses on creating new infrastructure to route messages and where possible leverages existing applications to actually send and receive messages. Many individuals and organisations have put considerable effort into providing tools to send/receive message for the various standard formats (and in many cases multiple high quality applications exist for each message format – each focusing on different usage scenarios). Enabling users to select from pre-existing applications allows users to choose the application that best suits their needs and allows us to focus on extending the capabilities of the messaging system instead of re-implementing the clients.

3.1. State Of The Art

3.1.1. A Unified Messaging System

One instance of a unified messaging system is GlobalCom [Bar02], which is a suite of web-based tools that can be used to send messages in various formats. They used a single message-independent for-mat to store the messages and implemented their own clients (such as mobile text, email, chatrooms, etc.) which access the message store database and convert the messages to the standard format (SMS, IMAP, etc.) for display and transmission [HBN03]. By implementing the system in this way, GlobalCom allows the user to choose what device and application to use to access their messages (as opposed to the system trying to decide the most likely client being used by them at that moment in time). However, a drawback to the system is that it increases the number of applications/clients that a user needs to operate as they will still have to interact with people outside of this closed messaging system and thus continue to have to use their regular email/chat clients, etc.

3.1.2. A Proxy-Based Platform

The iMobile EE [CHJ+03] project aims to hide the complexity of multiple devices and content sources by acting as a message gateway that allows mobile devices using various protocols on different ac-cess networks to relay messages to each other. This is a continuation of the original iMobile [RCCC01] project and as well as implementing a message proxy, it seeks to provide proxies for information such as stock quotes, weather and flight information. The iMobile research approaches the messaging problem in a similar way to our project, where the processing of messages is carried out on a single server which in turn routes the messages to the other clients. iMobile allows devices to connect to the iMobile server[1] over the GSM network. This differs from our re-search, which implemented Bluetooth InfoStations to allow inter device-networking within the campus without depending on commercial providers (part of the project's remit was to be able to minimise the cost of sending the messages by utilising the University network or creating new communications infrastructure over which the operation costs could be controlled).

[1] *the equivalent of the MessageRedirector in our project*

3.2. Innovations

Where our research differs from many of the other projects in this field is in the logic which decides how to deliver the message to the recipient. Other applications have focused on improving the connection between the mobile devices such as improved bandwidth, latency, etc. This has been termed "Always Best Connected (ABC)" [Mat06] and the message is transmitted in a single format (e.g. SMS).

Rather than introduce new message formats or new applications for communicating, this research sought to build upon existing applications in an attempt to create a universal messaging system. This allows the user to continue using their messaging applications of choice and to hide the translation between the message formats in the messaging system infrastructure. The next section describes the architecture of the messaging system and gives a high-level view of how all the components interact (the actual implementation details are discussed in Section 5).

4. System Architecture

This section describes the overall architecture of the messaging system and how each part of the system interacts with all the other components.

A graphical overview of how the system components interact is given in Fig. 2 and by following the arrows in the diagram, some of the possible message paths though the system can be traced. A number of use cases, given in Section 2, describe how a user would interact with the system.

4.1. Two-Tier Address Space

One of the central innovations in this project is the use of two-tier address space architecture. Each user in the messaging system is identified by a unique userID (more details in Section 4.2). This userID can be considered a 'virtual address' which the messaging system maps to a specific contact address ('real address') in the list of contact addresses that the

recipient has provided.

An analogy for this dynamic one-to-many address translation would be a demultiplexer (a digital switching device that has one input and several outputs). The sender uses the userID to identify the recipient; the messaging system demultiplexes the userID input and outputs the required contact address (the Intelligent Assistant provides the 'control bits' to select the contact address). A graphical representation of the operation is depicted in Fig. 3.

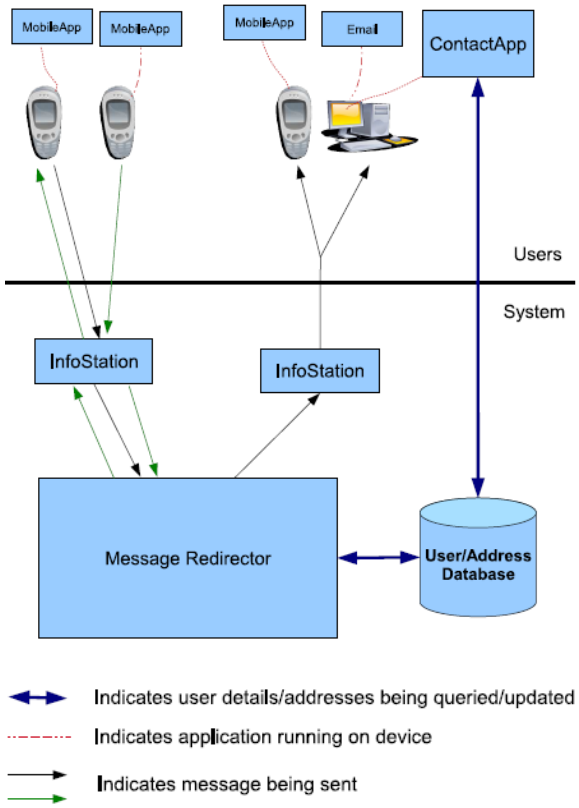

Figure 2: Messaging System Components

Benefits of this address space architecture include:

- The recipient gets the message delivered to the most suitable address/device.
- The sender need only maintain a single contact address for the recipient.
- If the recipient does not want to be disturbed, they can forward messages to an address that won't result in an interruption.
- The recipient can add new contact addresses without having to inform all their contacts. Once added to the ContactApp, the new address will be used if it meets the criteria decided by the Intelligent Assistant.

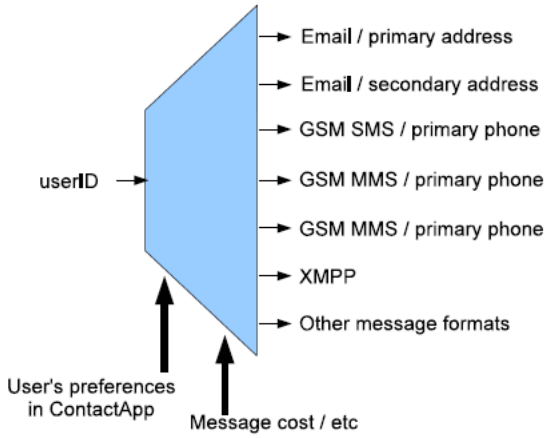

Figure 3: Two-tier Address Space - an address demultiplexer

4.2. Database

Each user of the messaging system has a unique identifier in the format of an RFC2822 [Res01] address (i.e. like the format of email addresses) with a userID part and a domain part (userid@domain).

Each user profile has one or more addresses associated with it. Each address entry consists of three pieces of information: (1) the contact address (e.g. email address in the RFC2822 format, mobile phone number), (2) the type of address (email, SMS, etc.), and (3) a user assigned priority to allow the user indicate to the system the ordering of the formats that they would prefer to receive messages (e.g. via email rather than via Instant Messaging (IM)). In the absence of an ordering specified by the user, a default ordering of the formats is provided by the system, which favours formats that have a lower cost per transaction (e.g. send as email rather than SMS, which would incur a network provider charge).

The user list and address list are stored in a database accessible to all the infrastructure components. Each address has a link to the associated user details via a "foreign key".

Multiple user profiles can also be grouped in hierarchies (in the same way email addresses can be added to mailing lists) so that messages can easily be sent to multiple users with each one receiving the message in their preferred format as described in

Section 2.

It is also possible for a user to access the database via the web interface (c.f. Section 4.6) and update their profile. The operations supported include:

- Add/delete contact methods ("addresses") by which messages can be sent to the user.
- Disable/enable specific contact methods in the database.
- Select a default method of contact.
- Set the 'priority' of an address in order to give user a hint to the Intelligent Assistant (c.f. Section 4.5.2) as to which address to select.

4.3. Mobile Application

The mobile application (MobileApp) runs on the users' mobile devices (such as a mobile phone or PDA). This is what is known as the Mobile Station in GSM parlance [Sco96], [ETS95]. The application allows the user to send messages in various standard formats such as SMS [ETS98] and interfaces with external messaging systems such as email.

4.3.1 Functionality

The MobileApp provides a message editor which al-lows a user to compose text messages.

The MobileApp can also save received messages and has an application-specific AddressBook to store the userID/domain tuple of contacts (This address data (userID/domain tuple) is also sent as part of the message format and the MessageRedirector can query the database (c.f. Section 4.2) to get the recipient's contact details).

The application on the mobile device controls when the messages are pushed from the mobile device and pulled from the InfoStation. It polls the InfoStation on a regular basis while it is within range to indicate that it is still within the cell and to check to see if any messages are available for the user.

4.3.2. Network Access

The messaging system has to be able to cope with having intermittent network access as the user will not always be within range of an InfoStation (c.f. Section 4.4).

Messages which fail to be sent will not be automatically retried asynchronously (this is a common design, e.g. [Hos02] and implementations of the SMS specification [ETS98] do this). If an attempt to send a message fails for whatever reason, the sender is given the option of retrying immediately or saving the message to persistent storage. Implementing asynchronous retrying would be too complex, may not be what the user wants, as the information contained in the message may have a limited useful lifespan and repeated attempts at sending the message may use too much power and reduce the battery life of the mobile device.

The MobileApp only connects to one InfoStation at a time and there is no 'hand-off' (when a mobile device switches seamlessly to a new InfoStation, discussed in [TWB96], [AMH+99], [BB95]). This will not be an issue as sending a message is an instantaneous event (unlike a voice call, which can last for a considerable amount of time, during which the user may be in motion and move out of range of the InfoStation to which they were connected).

4.4. InfoStation

The messaging system contains one or more InfoStations which provide the coverage necessary for the users to connect with the messaging system. The requirements for the InfoStation hardware are ruggedness, reliability and low cost as there could be quite a number of them in remote and outdoor locations in a large university campus.

A graphical representation of the major components in the InfoStation is depicted in Fig. 4

The InfoStation provides a service for the mobile device to connect and upload any messages that the user has created and wishes to send to other users of the messaging system. It then places these messages on its Send (Message) Queue, which the MessageRedirector will be watching and will read from when a new message arrives.

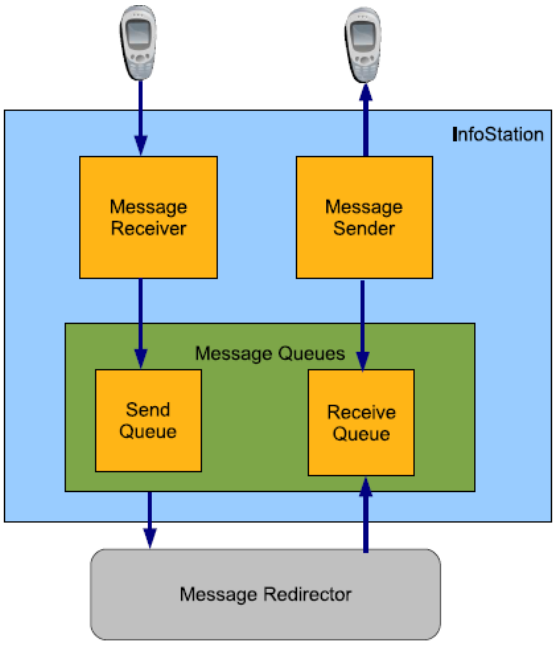

Figure 4: Diagram of the InfoStation internals

The InfoStation also provides a service for the mobile device to connect and download messages intended for its user. The InfoStation will have been polling the Receive (Message) Queue and will down-load any messages that the MessageRedirector has placed on it to be routed to a mobile device, which is known to be connected to that InfoStation. The mobile device checks on a regular basis and downloads any messages which are outstanding for that user.

4.5. MessageRedirector

The MessageRedirector (MR) is the central controlling component of the messaging system and contains some of the novel innovations referred to in Section 3.2. Each InfoStation will pass any message it receives to the MessageRedirector for processing as can be seen in Fig. 5.

The next subsections describe the major modules that make up the MessageRedirector and how the modules work in unison.

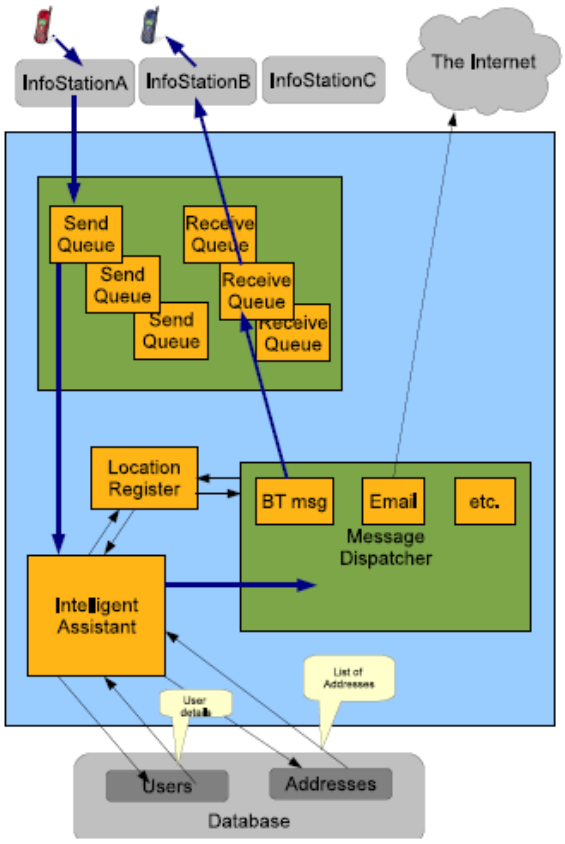

Figure 5: Diagram of the MessageRedirector internals

4.5.1. Overview of operations

When the MessageRedirector receives a message, it parses the message format and gets the sender's and recipient's userIDs as well as decoding the actual message payload. It provides the recipient's userID to the Intelligent Assistant, which provides the MessageRedirector with their preferred contact address. The MessageRedirector then inputs the sender's userID, the recipient's contact address and the message payload to the Message Dispatcher, which sends the message to the recipient.

4.5.2. Intelligent Assistant

It is the Intelligent Assistant that decides what ad-dress the message is sent to. The Assistant queries the database for all the addresses associated with the recipient's userID and decides the user's preferred contact address. The Assistant takes into account

factors such as the urgency of the message, the cost associated with sending the message to the various addresses and passes back to the MessageRedirector the address to which it thinks the message should be forwarded.

4.5.3. Message Dispatcher

Once the preferred address has been discovered, the MessageRedirector passes the address and the mes-sage text to the Message Dispatcher module, which sends the message in the required format (which can be decoded from the preferred address). Knowledge of the various message formats is limited to the Message Dispatcher module and this encapsulation of message processing allows the MessageRedirector itself to have no knowledge of the message formats. Adding support for a new format is a matter of plugging in an extra format implementation to the Message Dispatcher without having to modify the Mes-sageRedirector itself.

4.5.4. Location Register

The messaging system keeps track of each user's current location using a Location Register, which contains the address of the InfoStation to which the user's mobile device is connected. The MessageRedirector uses this information to route the messages to the correct InfoStation, which in turn sends the message to the mobile device. This is based on the same principle as the Home Location Register (HLR) in GSM [Sco96].

4.6. Web Interface

A web interface to the messaging system is provided so that users can modify their profile, which contains their address details.

The messaging system administrator can create an account/password for a user using the web interface (called ContactApp).

When the user logs in and is successfully authenticated, the application displays the user's details and the list of addresses that they have registered. The user can then add or remove addresses, modify their details or change the preferred priority of the addresses (i.e. control the order in which the MessageRedirector selects an address in order to forward a message to the user).

4.7. Privacy Issues

Depending on how this project was implemented, there could have been privacy concerns with the information that may have been available. For example, if delivery notification was enabled, it could be possible to track a user's presence depending on how a message was delivered to them (e.g. a person would have to be on campus to receive a Bluetooth message).

This is an issue that other projects have also encountered: [JPB05] (mentioned in Section 3) designed a system where the sender provides the context in which a message should be delivered - the sender does not have to know the recipient's current presence status or indeed should not find out that status without their permission. As in this project the DeDe team also chose not to provide delivery notifications.

5. Implementation

This section describes how the messaging system is implemented. Section 5.1 describes the messaging system components that the user interacts with and Section 5.2 details the necessary background components required to distribute the messages.

5.1. User Interface

The user interacts with the messaging system through the mobile application (MobileApp, running on the mobile device) and the web interface (ContactApp, accessible via any web browser). Typically a user would use the MobileApp more frequently than the ContactApp; once addresses are entered in the ContactApp, a user would probably only log in to change their preferred contact address once or twice a day whereas they would send and receive messages via the MobileApp throughout the day.

5.1.1. Mobile Application

The MobileApp application was written in

the Java programming language which allows it to run on the wide range of phones and mobile devices that run J2ME (a specialised virtual machine specifically for low-powered mobile devices). Details of the Mobile-App were provided in Section 4.3.

In order to guarantee a usable response and a satisfactory user experience, the MobileApp User Inter-face (UI) and the message sending/receiving components have separate software threads of execution. This multithreaded approach means that the user can navigate the application's menus while messages are being sent and received in the background and there is no risk of the mobile device display locking up if a user goes out of range of an InfoStation, etc.

A screenshot of the application's main menu is shown in Fig. 6. This menu shows a list of the features available in the application, which will be explained in further sections.

Figure 6: Menu of features of MobileApp

Message Editor

A central component of the MobileApp is the message editor which enables the user to compose the messages they wish to send. It utilises the comprehensive editing features of the J2ME platform such as predictive text entry.

Address Book

Users of MobileApp can save the addresses (userid/domain tuples) of people they frequently send messages to. The address book saves and retrieves the contact details to/from persistent storage on the mobile device using the J2ME *RecordStore* class. This maintains the address list on the mobile device across sessions and after the device has been powered OFF.

Message Stores

Other features of the MobileApp include the ability to save received and unsent messages (e.g. messages composed while out of range of an InfoStation). The messages are also saved to persistent storage on the mobile device using the J2ME *RecordStore* class.

5.1.2. Web Interface

The web interface allows a user to control how messages are routed to them and is accessible using any web browser (c.f. Section 4.6).

The web application's main requirement is that it can access the database containing the user details and that a web server is running on the machine on which the web application is hosted (the open-source Apache web server (httpd) is used in this prototype).

The first page presented to the user when they connect to the website is an authentication page with an option to jump to a sign-up page, which provides the opportunity for a user to subscribe to the service, where they could provide their username and password, if this is their first time using it.

A screenshot of the key part of the main menu that is presented to the user upon login is shown in Fig. 7. The upper part of the page shows the userID of the person currently logged in and lists the operations that the user can perform on the addresses. Other parts of the ContactApp, not shown here, permit operations such as logging out of the session, adding a new address or modifying their details.

The upper part of the page shows the user's details and the lower part shows the addresses that are associated with this user. The addresses are displayed in the order of the priority that the user assigned to

them. Addresses can be deleted, disabled or modified from this page.

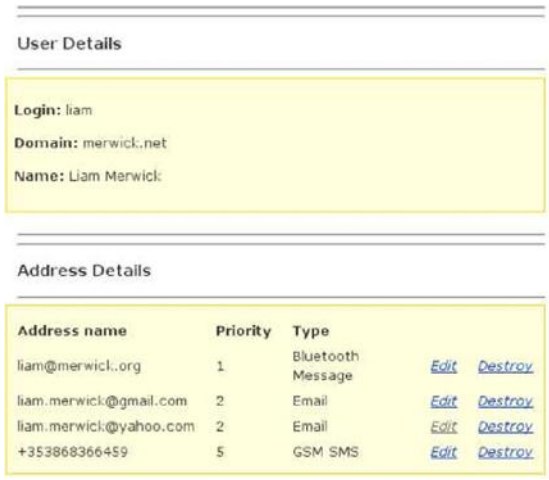

Figure 7: Menu of features of ContactApp

The format of these addresses was explained in Section 4.2. In addition to the system not delivering messages to addresses it knows are not available (e.g. the device is out of range), the user can also manually disable an address so that the messaging system does not consider it when it decides which address to select.

When a user adds a new address by which they can be contacted, he/she enters the address, a priority relative to their other addresses and then selects the address type from a drop down list of all the address types supported.

5.2. Prototype System Infrastructure

This section details the implementation of the infrastructure used by the messaging system. The user does not directly interface with these components but they are essential to the operation of the system. The infrastructure is made up of the MessageRedirector and InfoStation applications provided by this project along with externally provided supporting applications such as the Message Queue.

5.2.1. InfoStation

In the prototype implementation for this project the InfoStation Java application runs on basic PC hard-ware but when deployed in a production messaging system in a University, a low cost ruggedised system could be built and used. A USB dongle was added to the PC to provide Bluetooth connectivity.

The InfoStation application is a multithreaded application, with two threads waiting for Bluetooth connections. The first thread receives messages users wish to send and another thread pushes messages to the users' mobile device when it periodically connects to check for messages. In order to communicate with the MessageRedirector, two message queues are created for each InfoStation in the system (c.f. Figure 4).

5.2.2. MessageRedirector

For this project, the MessageRedirector also runs on basic PC hardware and is implemented in Java. In addition to the architecture and requirements listed in Section 4.5, the MessageRedirector initialises some of the general infrastructural components such as the Java Message Queues which are used by the InfoStations to communicate with the MessageRedirector.

Intelligent Assistant

The 'Intelligent Assistant' is the "brains" of the MessageRedirector. It is the module within the MessageRedirector which parses the message received from an InfoStation, decodes the message receiver's address and decides what format/device belonging to that user should receive the message.

Each message received by the Message Redirector is parsed to decode the sender's and recipient's addresses as well as the message payload.

Message Dispatcher

Given that we have a range of message formats (and possibly more in the future), we wish to have an extensible way of sending messages once we know the format they are to be sent in. For this reason, the component in the MessageRedirector that

processes and routes the messages is implemented using the Factory software pattern [EGV95]. A generic MsgDispatcher class is subclassed by the specific dispatcher implementation for each message type (e.g. BluetoothMsgDispatcher, EmailMsgDispatcher, etc.). When the Intelligent Assistant returns the recipient's contact address, it also informs the MessageRedirector of the address type and the message payload is input to the Message Dispatcher class of that type.

Location Register

As described in Section 4.5.4, the Location Register is a module within the MessageRedirector which keeps track of the InfoStation to which a user's mobile device is currently connected. Each time a message is received by the MessageRedirector from an InfoStation, the MessageRedirector calls the setUserLocation method in the Location Register with the sender's details and information on the location of the InfoStation to which the sender is connected. The Location Register stores this in an in-memory HashMap so that the MessageRedirector can later find out what InfoStation to send a message to in order for the InfoStation to push it to the user's mobile device.

If a mobile device has not been in contact with an InfoStation within the previous two minutes, it is deemed to be out of range and the entry is considered dormant in the Location Register. However, it is not evicted from the cache so that a record is kept of a user's last known location. Instead, any requests to the Location Register for that user via the isUserConnected API routine return an 'out of range' error until contact is made by the device again.

Naturally, a support application to provide selflearning, smart userlocation management could be developed which would be campus- and userspecific. For instance knowing that certain users can be within range of specific InfoStations (e.g. restaurants, meeting rooms, etc.) for extended periods of time (greater than two minutes) would allow the Intelligent Assistant to correlate a location returned by the Location Register with a list of delivery preferences (controlled by the user) for specific locations and to choose a delivery method based on the user's preferences (with the user having to explicitly change their address preferences when they entered the area).

6. Conclusions

This paper has described the realisation of an Intelligent Messaging Service in an InfoStation based University Network along with a detailed explanation of the underlying components which make up the system. A functional messaging system was developed consisting of all the necessary components to allow end-to-end message delivery including the mobile device application, InfoStations, central processing unit as well as the web infrastructure to manage the users' profile and contact information. The sys-tem was implemented within a single building, rather than campus wide, but we were successful in sending messages between mobile devices as well as to external messaging systems (e.g. to email servers hosted on the Internet). Delivery times were measured as being comparable with existing formats (e.g. an email between two users) which demonstrated that inserting an Intelligent Assistant into the communication's critical path and performing delivery decisions based on a user's location was feasible.

Future work includes extending the number of message formats supported, implementing Wi-Fi connectivity support on both application running on the mobile devices (MobileApp) as well as the InfoStation, improving the interaction with standard applications provided by the phone platforms (e.g. augmenting the standard contact list to include the address de-tails for the unified messaging system) and adding a billing solution for the message formats which incur a charge when being sent. Also an important augmentation of the Location Register support for the Message Redirector would be the development of self-learning smart location management functionality.

References

1. [AMH+99] I.F. Akyildiz, J. McNair, J.S.M. Ho, H. Uzunalioglu, and W. Wang. Mobility Management in Next-Generation Wireless Systems, 1999.

2. [Bar02] Declan Barber. Globalcom: a unified messaging system using synchronous and asynchronous forms. In *PPPJ '02/IRE '02: Proceedings of the inaugural conference on the Principles and Practice of programming*, 2002, pages 141–144, Maynooth, County Kildare, Ireland, Ireland, 2002. National University of Ireland.

3. [BB95] Ajay V. Bakre and B. R. Badrinath. Handoff and systems support for indirect tcp/ip. In *MLICS '95: Proceedings of the 2nd Symposium on Mobile and Location Independent Computing*, pages 11–24, Berkeley, CA, USA, 1995. USENIX Association.

4. [CHJ+03] Yih-Farn Chen, Huale Huang, R. Jana, T. Jim, M. Hiltunen, S. John, S. Jora, R. Muthumanickam, and Bin Wei. Immobile ee: an enterprise mobile service platform. *Wirel. Netw.*, 9(4):283–297, 2003.

5. [EGV95] Ralph Johnson Erich Gamma, Richard Helm and John Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Professional, 1995.

6. [ETS95] ETSI. *ETSI GSM Technical Specification 04.22*, 1995.

7. [ETS98] ETSI. *SMS specification: ETSI TS 100 901 (GSM 03.40 version 7.3.0 Release 1998)*, 1998.

8. [Faj98] R. Fajman. *RFC 2298 - An xtensible Message Format for Message Disposition Notifications*, 1998.

9. [HBN03] Paul Healy, Declan Barber, and Brian Nolan. Developing unified messaging system apps in java. In *PPPJ '03: Proceedings of the 2nd international conference on Principles and practice of program ming in Java*, pages 137–138, New York, NY, USA, 2003. Computer Science Press, Inc.

10. [Hos02] Ashima Hosalkar. *Building Mobile Applications with J2EE, J2EE-J2ME and J2EE Extended Application Servers*. Pace University, White Plains, NY, USA, 2002.

11. [Iva06] Ivan Ganchev, Stanimir Stojanov, Mairtin O'Droma and Damien Meere. An InfoStation-Based University Campus System for the Provision of mLearning Services. In *ICALT '06: Proceedings of the Sixth IEEE International Conference on Advanced Learning Technologies*, pages 195–199, Washington, DC, USA, 2006. IEEE Computer Society.

12. [Iva07] Ivan Ganchev, Stanimir Stojanov, Mairtin O'Droma and Damien Meere. An InfoStation-Based University Campus System Supporting Intelligent Mobile Services. In *Journal of Computers*, volume 3, pages 21–33. Academy Publisher, 2007.

13. [Jon04] Jonna H¨akkil¨a and Jani ¨antyj¨arvi. User experiences on combining location sensitive mobile phone applications and multimedia messaging. In *MUM '04: Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, pages 179–185, New York, NY, USA, 2004. ACM Press.

14. [JPB05] Younghee Jung, Per Persson, and Jan Blom. Dede: design and evaluation of a context-enhanced mobile messaging System. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 351–360, New York, NY, USA, 2005. ACM Press.

15. [Mat06] Matthias Siebert, Ivan Ganchev, Mairtin O'Droma, F. Bader, H. Chaouchi, I. Armuelles, I. Demeure and Fintan McEvoy.A 4G generic ANWIRE system and service integration architecture. *SIGMOBILE Mob. Comput. Commun. Rev.*,10(1):13–30, 2006.

16. [RCCC01] Chung-Hwa Herman Rao, Yih Fam Robin Chen, Ming-Feng Chen, and Di-Fa Chang. iMobile: a proxybased platform for mobile services. In *WMI '01: Proceedings of the first workshop on Wireless mobile internet*, pages 3–10, New York, NY, USA, 2001. ACM Press.

17. [Res01] P. Resnick. *RFC 2822 - Internet Message Format*, 2001.

18. [Sco96] John Scourias. Overview: The global system for mobile communications, 1996.

19. [TWB96] Mark S. Taylor, William Waung, and Mohsen Banan. *Internetwork Mobility the CDPD Approach*, 1996.