Mobile Instant Messaging

Roland Parviainen, Peter Parnes
Centre for Distance-spanning Technology
Department of Computer Science
Luleå University of Technology
971 87 Luleå, Sweden

Roland.Parviainen@cdt.luth.se, Peter.Parnes@cdt.luth.se

Abstract

In this paper we describe a Mobile Instant Messaging system, MIM, designed for mobile environments. During design of mobile applications several new problems and possibilities have to be considered that do not exist with applications targeted at desktop PCs. One example of an application not designed for a mobile environment is current, very popular, instant messaging systems such as ICQ, AIM and MSN messenger.

We describe why current systems are not suitable in a mobile environment and present our architecture for a new system, MIM, and show various implementations for different mobile devices such as PDAs, wearable computers and mobile phones.

1 Introduction

Today wireless technologies such as GRPS, new 3G networks and different WLAN technologies are rapidly being developed and deployed providing wireless access to the Internet in a much larger scale than before. At the same time, new mobile platforms like PDAs, advanced mobile phones and wearable computers are becoming more and more common.

Together it will mean that the Internet will be accessible from anywhere, on a wide variety of different platforms and connection technologies. A user should be able to expect to access the same applications and services that are available on a fixed, connected PC from a mobile connected unit.

Software applications for these mobile platforms face a number of different problems that applications designed for PCs do not experience, especially for distributed applications that need network connectivity. In addition new possibilities are opened up for mobile applications such as location independence, location and context awareness. Some of the unique conditions mobile distributed applications face are:

0-7803-7661-7/03/$17.00

Wide variety of user interface possibilities This can range from a normal PC environment to a mobile phone or a wearable computer with only audio user interaction capabilities.

Resource limitations Resources available on most mobile devices are usually very restricted. Different limitations such as weight and size mean less CPU power and storage capabilities.

Heterogeneous networks The connectivity can vary from a low latency always connected broadband connection to a low bit rate wireless connection with high packet loss that only supports HTTP connections through proxies and firewalls. Often the network connectivity can be temporary as well, for example when we move in and out of areas without coverage in a wireless network or for devices such as phones that are not always connected.

Mobility and state We should not be restricted by the limited user interface of mobile devices when we are for example in the office and have access the work PC. It should be possible to easily switch the device we work with while always having access to the same data and applications. Long term and short term state, such as program state and configuration details have to be synchronized between several application instances running on different devices at the same time. A new instance should have access the same state as other connected instances.

These are of course not all the problems and unique conditions that exist, but are the problems that we focused on in the system described in this paper.

1.1 Instant Messaging

Instant Messaging (IM) systems such as ICQ, AIM and MSN Messenger have become extremely popular during the last few years. All these systems are designed for desktop PCs with the assumption that a user only uses one computer,

or at least only one computer at a time. New protocols and standards for instant messaging are being developed by e.g. the IETF and the Jabber Software Foundation, but it is unlikely that these protocols are adapted by any of the major systems any time soon; although Jabber has had some success on internal corporate networks and a client is included in certain mobile phones.

If we want to use these systems in a mobile environment we either have to try to adapt to the assumptions about usage these systems have made or try to adapt the IM systems to better suit the needs of a mobile environment.

In this paper we will present a system that can be used to adapt current instant messaging and similar systems to a mobile environment. In the next section we describe in more detail the problems with the current systems, and in section 2 we introduce our new platform. Then, in sections 3 and 4 we present the architecture and implementation. Sections 5 and 6 discusses related and future work and finally we end with a discussion and conclusions in sections 7 and 8.

Most current IM systems are designed in a way that makes it impossible or very hard to use more than one client program per user at the same time; if a user wants to connect both from their office PC and a mobile client at the same time they're in trouble since the system only allows one connected client at a time. In addition, even if more than one client can be connected at the same time, messages are only forwarded to one of the clients.

Configuration state such as the user's contact list, history, settings etc. is often stored by the client locally, making synchronization a big problem. This is mostly due to scalability issues since most systems have millions of users. Some systems such as the latest versions of ICQ supports hosting the user's contact list at the server which makes it easier to run clients from more than one computer at the same time but does not completely solve the problem. For example, the message history is still stored locally at each client. To get a complete view of the history, access to the history from all clients the user has used is needed.

Also, if the network connectivity is lost or the user switches connection method the user is marked as being "Off-line", although there might be a high probability that the connectivity is regained shortly. In a wireless network this will be problematic since short losses of connectivity might occur often when a user for instance move from one cell to another or briefly looses coverage.

Since current systems assume that the user's position is fixed to one PC there is no location or position information. This is not the case for a mobile device; here a user's position dynamically change and can provide important and valuable information to other contacts.

Implementations of different Instant Messaging systems already exist for platforms such as PocketPC and J2ME MIDP for PDAs and mobile phones, but they still have similar problems.[1] They do not attempt to solve the problem of mobility; it is just implementations of the same applications for different platforms.

[1] See e.g. QSM (http://www.softex-india.com/qsm/) or MiMessenger (http://www.mordax.com/manual/manual_mim_en.html)

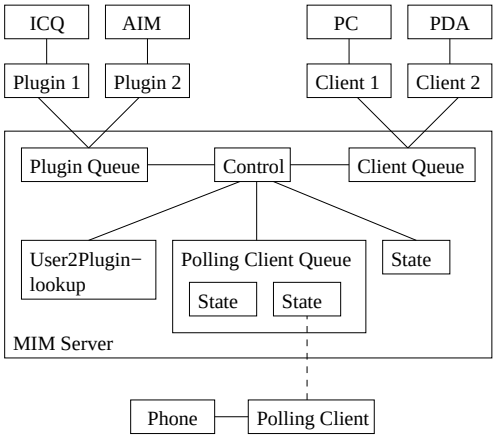
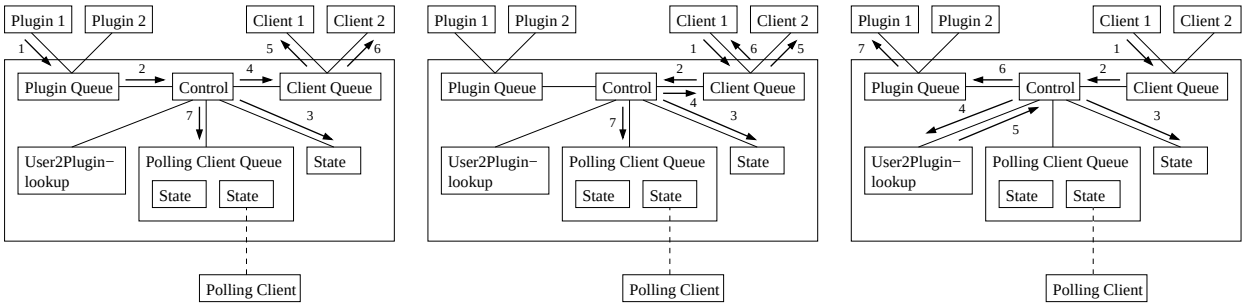


Figure 1: Architecture

2 Mobile Instant Messaging

Designing a new IM system and expecting all users to adopt it is unrealistic; most users simply use the same system all their friends and contacts use. The extremely large user base of the systems makes it hard to see any new better system gaining any large market share. Therefore, a system that can use the current IM systems is highly desirable. It is also desirable to be able to connect to several systems at the same time and aggregate the different user communities and present a single list of contacts to the user. Clients that can connect to different systems already exist, for example EveryBuddy[2], Trillian[3] and GAIM[4].

A user should be able to move from his home PC, to a mobile unit such as a phone while on the road and finally to his work PC without having to remember to manually logout from each unit before switching to a new one. The user status should not switch from "Online" to "Off-line" and back again each time the user switch unit or when the network connectivity is temporarily lost since the user will probably regain connectivity shortly. The contact list should be automatically synchronized among our clients; if a new contact is added while using a mobile phone this new user should also show up in the contact lists in all other clients. If a contact sends a message, we should be able to receive and read it on any of the connected applications at any time. None of the current systems can do this today.



a) Message arrives from a IM system b) User reads the message c) User replies

Figure 2: Example of a message exchange

2.1 Limitations of current solutions

3 Architecture

The goals from the previous section can be achieved by using a hierarchical system with a Mobile Instant Messaging (MIM) server running on the **Home PC**, which is for example the user's actual home PC, office PC or a central network server. Different IM systems are connected by MIM plugins, providing the MIM server with contact lists, users' status, functions for setting our own status, receiving and sending messages etc.

Clients connect to the MIM server, which handles all synchronization and stores state such as received and unread messages, history and status. When a client first connects, the full state is retrieved from the server. Further updates to the state, such as an incoming message, are sent to all connected clients. The MIM server is always connected to the different IM systems so a user will be able to receive messages at any time. If no clients are connected over a duration of time, the status and auto response message can be set to pre-configured values.

If clients can not keep a connection to the MIM server open continuously, it can also run in a polling mode, where the client regularly connects to the server for synchronization: messages that have been written are sent to the server and on to their recipients, new messages are downloaded, etc. This mode is primarily for clients running on phones, for example the MIDP profile as used by many "Java" phones only guarantees network connectivity through HTTP.

See figure 1 for an example of this architecture with two different MIM plugins, one for the ICQ network and one for the AIM network. Three clients are connected to the MIM server: One on a PC, one on a PDA and one on a mobile phone. The clients on the PC and the PDA use a full graphical user interface, while the Phone client use a limited user interface adapted to the possibilities of the unit.

[2]http://www.everybuddy.com/
[3]http://www.trillian.cc/
[4]http://gaim.sf.net/

3.1 Communication Protocol

We use a simple light weight, low bandwidth message passing system that can work over any protocol that supports binary data transfer, although it could be implemented on top of for example a tuplespace system such as JavaSpaces[1] or T Spaces[2], but work would be needed to make these systems work efficiently over the limited network services that are currently provided by e.g. Java MIDP phone. This limitation is one of the reasons that rules out protocols based on IP multicast that would have been more efficient in a system like ours.

3.2 Example of a message exchange

In figure 2 we see an example of a message exchange in MIM. Three clients and two plugins are connected to the server.

In a) a message is received from a IM system by a plugin, then in b) the messages is read by a user using one client and finally in c) a message is sent as a reply back to the plugin which forwards the message to the IM system.

An incoming message is sent through the plugin queue to the server control which sends it on to both the client queue and the polling client queue. It also forwards the message to the state component for storage. The client queue sends the message to all clients that are connected, while the polling client queue stores it in a state component for each connected polling client.

When a user reads the message, the client sends a notification back to the server through the client queue. This notification is broadcasted to all connected clients as described above. Clients can use this notification to change the status of received messages from "Unread" to "read". The notification is also sent to the central state which updates the status of the message in storage.

A message from a client to an IM system is sent through the client queue to the server, which first sends it to the state component. It then uses the "User to plugin" component to look up which plugin that is connected to the target IM system. Then the message is sent to the correct plugin through the plugin queue.

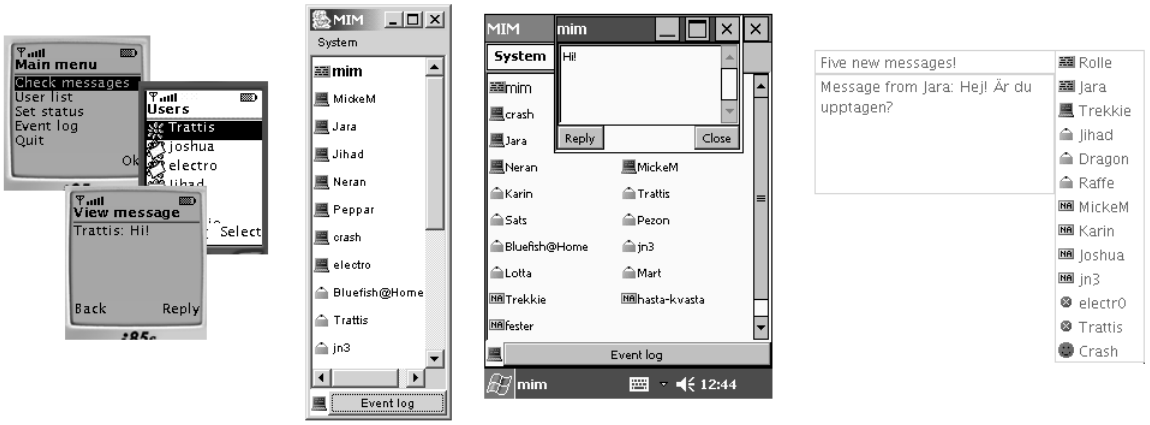


Figure 3: MIM running different platforms; on phones (The screen shots are from the J2ME Wireless Toolkit emulator), on a PC, on an iPAQ Pocket PC PDA and on a wearable computer (the screen shot shows what is displayed in the top right corner of the head up display).

4 Implementation

The current implementation consists of a mix of Java and C++ applications. All communication is based on TCP or HTTP. The communication protocol is a simple message based protocol, where each message such as a status update or user status information usually is between 20 and 30 bytes.

4.1 Clients

All clients are implemented in different versions of the Java platform. They can be delivered to the client platform by regular software installation mechanisms, via applets, Java Webstart or Over the Air (OTA) provisioning for WAP and GPRS. See figures 3 and 4 for some screen shots of different clients.

The interface for PDAs and PCs is created by the same application, while it had to be completely rewritten due to the different nature of the user interface API in J2ME MIDP. A user interface description language such as UIML[3] could be used instead to provide different user interfaces for different platforms automatically from one specification.

Although Java is platform independent this is not enough to create one user interface for different clients for two reasons; the same user interface APIs are not available on all platforms that runs Java, and the limitations and possibilities of the available platforms varies widely. It does not help that the user interface is written in Java if it is based on a graphical user interface and our platform only have audio and text user interface capabilities. A user interface that works well on a PC does most often not work well on a PDA due to the limited screen size.

PocketPC 2002 The implementation for Pocket PC PDAs uses the Jeode Personal Java runtime which is almost a full JDK1.1.8 implementation.

PC The PC version uses the same implementation as for PocketPC, but the different platforms are detected at runtime and the user interface is adapted to better suit the particular platform.

Phone The implementation for mobile phones uses the Java 2 Platform, Micro Edition (J2ME)[4] as provided by the Connected Limited Device Configuration (CLDC)[5] and Mobile Information Device Profile (MIDP)[6]. Only HTTP communication is used, as this is the only network connectivity that is guaranteed by the MIDP standard. The big differences between J2ME and other Java platforms mean that we unfortunately can not use the same implementation although many classes can still be reused. For example, the user interface had to be completely rewritten to suit the limited interfaces of MIDP. This system also provides a much cheaper way to send messages from phones than the, at least in Europe, extremely popular SMS services (see section 7 for a comparison).

Wearable computer This implementation is still experimental since the wearable computer it is designed for is still being built. Currently it uses an M2 head up display and different small keyboards, connected either to a laptop or an iPAQ PDA. The user interface uses low level Java graphics primitives to create a custom interface more suitable for a head up display.

4.2 Plugins

Three plugins exist today, an ICQ plugin, an IRC plugin and a plugin for the chat component in the MarratechPro E-meeting program. The ICQ plugin is written using the Licq[5] plugin API in C++. The basic instant messaging fea-

[5] http://www.licq.org/

Figure 4: MIM running on a Nokia 7650 phone

tures such sending and receiving messages, changing status, retrieving user information etc. are implemented.

The IRC[7] plugin is implemented as a Java program that shows the different IRC channels as users in the aggregated user list in MIM. Currently it is not possible to dynamically join and leave channels. Only the basic functions such as messages to channels and private messages to users are implemented.

MarratechPro is an e-meeting product by Marratech AB[8] which is designed for IP multicast but it also works with unicast. It has components such as audio, video, shared whiteboard and chat and is mostly written in Java. The chat plugin for MIM uses the chat agent from the MarratechPro application, so it is fully compatible. Different e-meeting groups are represented as different users in the MIM user list.

5 Related work

Frameworks developed in the fields of ubiquitous and pervasive computing are designed to solve many of the problems mobile applications face. For example in [9], approaches for systems for pervasive computing are presented: data and functionality need to be separated, force applications to explicitly acquire all resources, and to be prepared to reacquire them, a common system platform allowing applications to run on wide variety of devices. These approaches are similar to the ones used in our system. The system built on these principles, *one.world*[10] could have been used as platform for our system, but like many other similar architectures it relies functionality (such as object serialization) which is not available on all of our target platforms.

Instead of having multiple concurrently running clients, we could use one client that migrates to the current device such as in the Sprite[11] or Emerald[12] systems, but this does not work if a client becomes disconnected from the network since we then do not have any way of migrating to another device. There are also problems with for example migrating a client using the Java AWT API to a device only supporting the Java MIDP API.

A remote desktop solution such as VNC[13] or Microsoft RDP[14] would solve some of our problems but would also introduce new ones. The available bandwidth can be very limited and the latency high, which would not provide a good user interface experience. Also, the user interface can not adapt to the new platform and we have no way of using the application with out network connectivity. For some platforms such as mobile phones these solutions are not practical at all.

6 Future work

Plugins for other systems will be implemented. One way to achieve this is to use one of the applications that already have support for all the major instant messaging systems. Not only instant messaging systems can be plugged in, other chat and messaging services can be used as well.

Since most wireless communication technologies can be used to position the user, positioning services can be plugged in to provide valuable information to other users, such as adding the user's current position or the last known position to the auto response message in ICQ. The Alipes platform [15] provides an architecture for location aware applications and can use several different positioning technologies and combine different positioning sources to one more accurate position.

One form of mobile platforms that need special user interface design is wearable computers, where the user interface can be limited to a low-resolution head mounted display and a chord keyboard or even only audio as the only means of input or output. See e.g. [16] for an example of the problems of user interface design for wearable computers.

7 Discussion

Using a system like MIM on a mobile phone to send short text messages can be a lot cheaper than using the traditional SMS service. Comparing current prices for the largest Swedish phone operator, we see that the difference is huge. A SMS message has a maximum length of 160 characters and sending a MIM message of 160 characters over GPRS from a phone uses 392 bytes, including HTTP headers and data for both the HTTP request and response. If this data exceeds free monthly included data volume the price is 0.0069 - 0.048 SEK depending on the subscription, while the price for one SMS message is 1.50 SEK. This is about 31-217 times cheaper.

We can also evaluate how our system relates to the different problems we stated in section 1. It can be seen that the MIM system handles all of the problems:

Wide variety of user interface possibilities Some platforms can be detected at runtime and the interface

adapts to the special characteristics, but other platforms need specialized user interfaces.

Resource limitations State is kept at a server, not on each client reducing the storage needed. An IM client is not a CPU intensive application so there is no problem with the low CPU power of e.g. mobile phones.

Heterogeneous networks We use a low bandwidth protocol that can be used over HTTP, so it can easily be used on devices with limited network capabilities. Clients that can not maintain a continuous connection with the server can run in a polling mode instead. If we loose connection for some time we will regain all lost information when we reconnect again.

Mobility and state Any number of clients can be connected at the same time, and any client has access to the same information and messages. If messages are received and a new client later connects, it will receive these messages as well.

8 Conclusions

In this paper we have described a new instant messaging system, MIM, designed for mobile environments that do not suffer from problems current solutions do. The system supports any number of concurrently connect clients, which are synchronized so that a client can connect and disconnect at any time with out losing any messages. The system is designed to inter-operate with different current instant messaging systems such as ICQ or MSN Messenger. It can run on different platforms with different capabilities such as PCs, PDAs, mobile phones and wearable computers and can work over temporary, low bandwidth connections.

It illustrates some of the design principles that are necessary in a mobile environment: minimize state kept at the clients, adapt the user interface to the different platforms and allow concurrently running clients.

Acknowledgements

This work was done within the RadioSphere project, which is supported by the Swedish Foundation for Strategic Research and the Objective 1 Norra Norrland - EU structural fund programme for Norra Norrland. Support was also provided by the Centre for Distance-spanning Technology (CDT).

References

[1] The JavaSpaces Specification. Sun Microsystems, Inc.

[2] T.J. Lehman, S.W. McLaughry, and P. Wyckoff. TSpaces: The Next Wave. In *Proceedings of the Hawaii International Conference on System Sciences (HICSS-32)*, 1999.

[3] C. Phanouriou. *UIML: A Device-Independent User Interface Markup Language*. PhD thesis, Virginia Tech, 2000.

[4] Java^{TM} 2 Platform: Micro Edition, A White Paper. Sun Microsystems, Inc.

[5] Connected, Limited Device Configuration Specification 1.0a. JSR-000030, Sun Microsystems, Inc.

[6] Mobile Information Device Profile Final Specification 1.0a. JSR-000037, Sun Microsystems, Inc.

[7] C. Kalt. Internet Relay Chat: Client Protocol, 2000. IETF RFC2812.

[8] Marratech AB. <URL:http://www.marratech.com/>.

[9] R.Grimm, J.Davis, B.Hendrickson, E.Lemar, T.Anderson, B.Bershad, G.Borriello, S.Gribble, and D.Wetherall. Systems Directions for Pervasive Computing. In *Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, pages 128–132, 2001.

[10] R. Grimm, T. Anderson, B. Bershad, and D. Wetherall. A System Architecture for Pervasive Computing. In *Proceedings of the 9th ACM SIGOPS European Workshop*, pages 177–182, 2000.

[11] F. Douglis and H. Ousterhout. Transparant process migration: Design alternatives and the Sprite implementation. *Software – Practice and Experience*, 21(8):757–785, 1991.

[12] E. Jul, H. Levy, N. Hutchinson, and A. Black. Fine-grained mobility in the Emerald system. *ACM Transactions on Computer Systems*, 6(1):109–133, 1988.

[13] T. Richardson, Q. Stafford-Fraser, K.R. Wood, and A. Hopper. Virtual Network Computing. *IEEE Internet Computing*, 2(1), 1998.

[14] Remote Desktop Protocol (RDP) Features and Performance, White Paper. Microsoft Corporation.

[15] J. Nord, K. Synnes, and P. Parnes. An Architecture for Location Aware Applications. In *Proceedings of the Hawaii International Conference on System Sciences (HICSS-35)*, 2002.

[16] A. Schmidt, H.W. Gellersen, M. Beigl, and O. Thate. Developing user interfaces for wearable computers: Don't stop to point and click. In *International Workshop on Interactive Applications of Mobile Computing (IMC2000)*, 2000.