Peer2Me - Rapid Application Framework for Mobile Peer-to-Peer Applications

Alf Inge Wang, Tommy Bjørnsgård and Kim Saxlund
Department of Computer and Information Science
Norwegian University of Science and Technology
N-7491 Trondheim, Norway
alfw@idi.ntnu.no, tommybj@idi.ntnu.no,saxlund@idi.ntnu.no

ABSTRACT

This paper presents the Peer2Me framework that enables developers to create mobile peer-to-peer applications. The framework provides an API that is easy to adopt, yet capable of creating advanced peer-to-peer applications. The framework was built to provide applications providing pure peer-to-peer network where all nodes have the same responsibility and services. Further, the framework provides services to transparently manage the detection of new and lost peers. The message component of the framework makes it possible to exchange any kind of data between peers including Java objects. The Peer2Me has been implemented in Java 2 Micro Edition (J2ME) and runs on standard mobile phones. The framework supports management and communication of mobile ad hoc networks (MANETs) like Bluetooth. The paper describes the architecture, the API and some of the applications developed using the Peer2Me framework. Further, we share and discuss experiences from developing mobile peer-to-peer applications.

KEYWORDS: Rapid application development, Mobile ad hoc networks, Peer-to-peer networks, Bluetooth, J2ME.

1. INTRODUCTION

Traditionally, the client-server was the mainstream approach for doing distributed computing. However, the centralisation of services and information in the client-server approach easy becomes a problem in terms of performance bottleneck and reduced availability because of single-point of failure. The peer-to-peer architecture removes this problem by allowing all involved computer act as equals in the network [19]. This makes it possible to balance the network load and provide a more fail-proof network by using alternative routing in case a network connection between two nodes fails. It was programs for music-file sharing like Napster that really showed the potential of peer-to-peer computing [5]. Since Napster, peer-to-peer computing has become mainstream and peer-to-peer applications for chatting and file sharing are now included into operating systems like Microsoft Windows Vista and Mac OS X Tiger.
Currently, most peer-to-peer applications and architectures are designed to work in a fixed and wired infrastructure like the Internet. The development of wireless network technologies, mobile devices and programming environment for mobile devices have made it possible to move the peer-to-peer computing to a wireless environment [11, 14]. By bringing peer-to-peer computing to the mobile and wireless platform, we have to struggle with the classical challenges of mobile computing within the areas of wireless communication (heterogeneous networks, low and variable bandwidth, disconnection, etc.), mobility (find closest server, network handover, where to store data etc.) and portability (limited CPU, memory, battery, displays, keyboards, etc.) [18]. However, mobile peer-to-peer computing also offers new opportunities that can be utilised like providing location-based services [6, 13] and social computing [7, 8] using short-range networks.

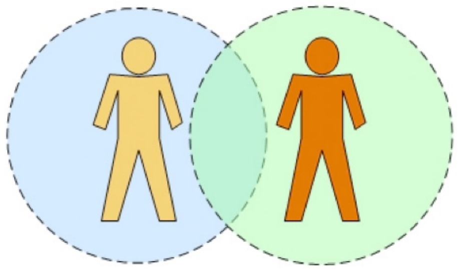


Figure 1. Digital Spheres Intersecting

Most wireless devices support some kind of personal area network (PAN) technologies like irDA and Bluetooth [16]. PAN is often used to transfer data between mobile devices and PCs, or between mobile devices and peripherals like headsets and keyboards. A PAN can be seen as a digital sphere around the mobile device enabling the exchange of data to nearby devices. This digital sphere can also be used to provide mobile computer supported cooperative work (mobile CSCW) [22]. Figure 1 shows how mobile CSCW can be provided by overlapping digital spheres where data can be exchanged between applications run on two user devices. In such an environment, the support for mobile peer-to-peer is essential and the support and establishment of mobile ad hoc networks (MANETs) are necessary. A MANET is a self-configuring network where peers can join and leave the network dynamically making the wireless network topology unstable and unpredictable [17]. MANETs can be utilised in situations where persons with mobile devices meet and there is a need for exchanging data.

MANETs enable mobile users to interact in new ways. The interaction between users can either be explicitly initiated by the users, it can be automatically initiated by the mobile devices, or a hybrid of the two [20]. Such applications can be used for initiating collaboration between users of same interests, e.g., an application for finding people with same research interest at a conference [21]. Further, MANETs can be used to create application for proximity chats and file exchanges, or simply for leisure like games.

The Peer2Me project was initiated to enable rapid development of proximity-based peer-to-peer applications for mobile devices on the Java 2 Micro Edition (J2ME) platform. Our main goal was to develop a high-level programming framework that made it possible for developers to only use simple primitives and methods to manage the complexity of peer-to-peer MANETs. It was also essential that the Peer2Me framework was transparent and hid the network technology used for communication. The Peer2Me framework enables researchers and application developers to explore utilisation of MANETs and how MANET applications can provide collaborative support for mobile users. The main contribution of this paper is to describe the Peer2Me framework and experiences we from building and using the framework.

The rest of the paper is organised as follows. Section 2 describes the architecture and the design of the Peer2Me framework. Section 3 describes how the framework can be used in applications. Section 4 presents experiences we have gained from developing Peer2Me applications. Section 5 relates our framework to similar approaches. Finally, Section 6 concludes the paper.

2. THE Peer2Me FRAMEWORK

This section describes the architecture, and design and implementation issues related to the Peer2Me framework.

2.1. Peer2Me and J2ME

Sun Microsystems developed Java 2 Micro Edition (J2ME) [15] to provide a general execution platform for resource-constrained devices. J2ME consists of various configurations, profiles and optional packages to support various kinds of equipment. For mobile devices like mobile phones and PDAs, the Connected Limited Device Configuration (CLDC) is the most common configuration that is tailored for devices with wireless network capacities. In the same way the Mobile Information Device Profile (MIDP) is the most used profile for such devices. The MIDP provides an environment for creating and managing applications, called MIDlets, including GUI libraries [15]. Most mobile phones sold today supports J2ME and MIDP 2.0. In addition, some mobile phone models support a variety of optional packages that that provides API for various purposes like location, 3D graphics, multimedia support, security, speech etc. Figure 2 shows how our Peer2Me framework fits into the J2ME environment.

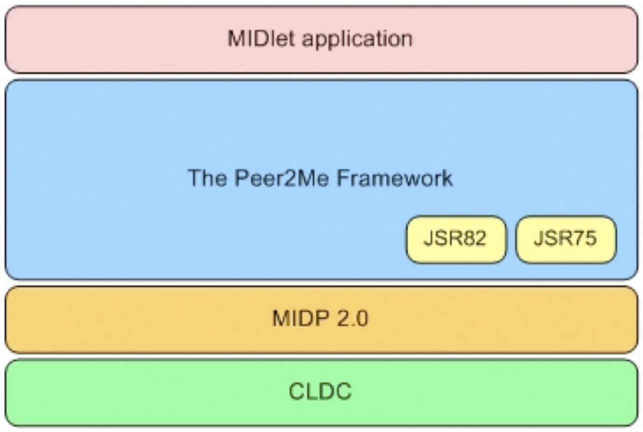


Figure 2. The Peer2Me Framework and J2ME

Our framework is built upon MIDP 2.0. In addition, the Peer2Me framework uses two optional packages:

- **JSR82:** J2ME API to access and manage Bluetooth networks.
- **JSR75:** J2ME API to access Personal Information Management (PIM). The JSR75 makes it possible to read and write to the file system as well as access data (both read and write) in the built-in address book and calendar of the mobile device.

The current Peer2Me implementation only supports Bluetooth networks, but the architecture is made modular to also support other types of networks when they become supported in the J2ME environment.

2.2. The Peer2Me High-level Architecture

The Peer2Me architecture is based on a layered architectural pattern where each layer is assigned with its own responsibility, and one layer is based on the layer below. By using the layered approach, the architecture would gain positive characteristics like modularity and transparency. The

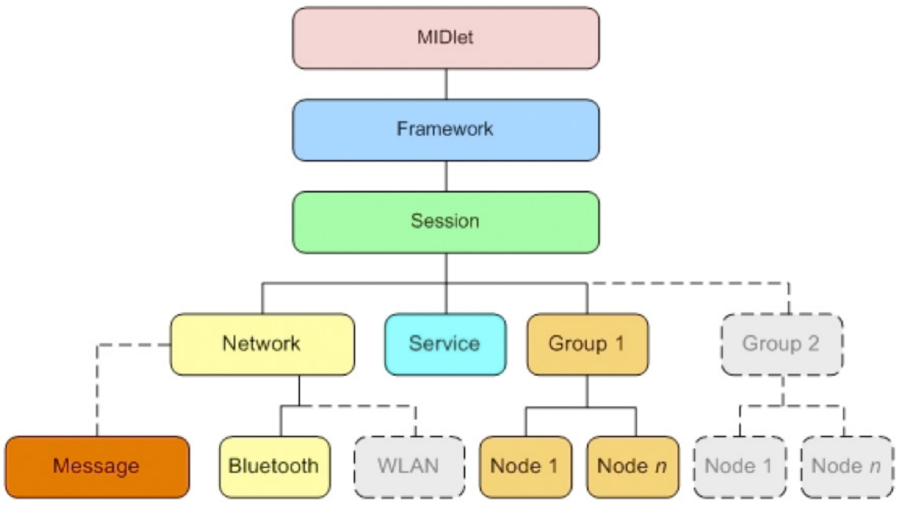


Figure 3. The Peer2Me High-level Architecture

negative effect by using this approach could be slower execution if the applications often have to go up and down several layers to carry out the operations. As the Peer2Me framework should be used on resource constrained execution platform, we decided to use few layers in the architecture. Figure 3 shows the high-level architecture of Peer2Me. Figure 3 shows the main parts of the Peer2Me architecture (note that the MIDlet is not a part of the Peer2Me framework).

- **Node:** A *node* is a logical representation of a peer, i.e., a mobile phone running the framework. Two or more nodes can form a mobile ad hoc network.
- **Group:** A *group* is a collection of nodes that know of each other's existence. All the nodes in a group can communicate with each other.
- **Service:** A *service* is a description and an identifier of an application running the framework. To enable Peer2Me applications running on mobile devices to interact, they all have to run the same service. This means that the service provides the ability for applications to see that the provide the same service.
- **Network:** A *network* is an abstraction of the network layer representing the communication medium accessed by the framework instance. The network layer can consists of several network implementations that also can be run simultaneously. Note that if some functionality is specific to the network technology used, it must be directly accessed in the specific network implementation (e.g., Bluetooth).
- **Message:** A *message* is an entity that can be exchanged between the nodes. A message can be sent to single nodes or to groups and can contain text, serialised objects, or any data type or binary data such as pictures, video, documents etc.
- **Session:** A *session* represents the lifetime of all the communication between the nodes in a group. A session keeps track of known nodes, groups and available network mediums.
- **Framework:** A *framework* represents the core entity between the application and the rest of the system. The framework hides all the complexity for the application developer and provides the interface to Peer2Me.
- **Application:** A Peer2Me application will be implemented as a *MIDlet* running on top of the Peer2Me framework.

2.3. Design and Implementation Decisions

This section describes design and implementation decisions made in Peer2Me to provide a transparent high-level API.

Pure versus hybrid peer-to-peer model: Pure peer-to-peer approach where all the peers have the same responsibility for managing resources and communication is not supported directly in Bluetooth. The Bluetooth technology is based on a master-slave communication protocol, where the Bluetooth master will search for nearby Bluetooth devices that will become slaves when communicating. The master-slave configuration is essential the same as client-server and suffers from the same problems of single point for failure. A possible solution to avoid this problem could

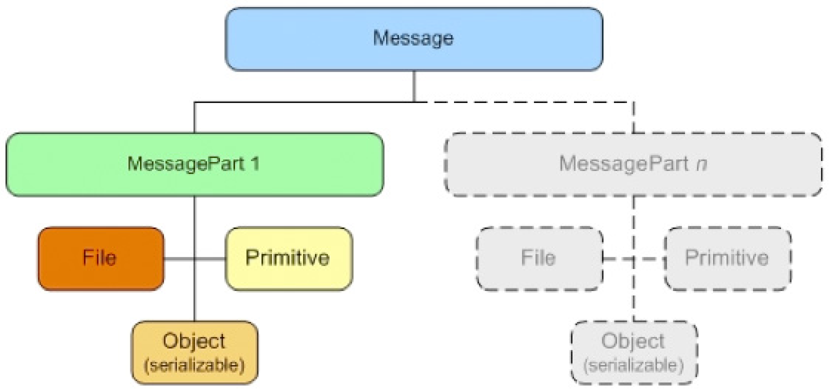


Figure 4. Message structure

be to make the master node delegate the server responsibility to a new node in case of a failure, but this would require a very complex protocol without getting a high availability of the system. In Peer2Me we implemented a pure peer-to-peer protocol avoiding the difficulties of the master-slave paradigm. The master-slave connections are established dynamically when there is a need for one node to send a message to another node or nodes. This means that in Peer2Me all the nodes know all other nodes, and every node have the same responsibility.

Communication protocol: Bluetooth provides two different protocols for implementing communication between peers: RFCOMM and OBEX. RFCOMM emulates a RS-232 serial connection, which provides a stream-based interface. The advantage using RFCOMM is that it is very straightforward to implement in J2ME, but it has some major disadvantages. RFCOMM only supports one session at a time between two devices, and the maximum amount of active RFCOMM services a Bluetooth device can have is 30. When RFCOMM is used, the receiving Bluetooth device must read the stream and later parse the stream. OBEX is much more versatile than RFCOMM and provides support for setting up a session of two communicating devices. The information is sent between the devices in the form of a put or a request pattern. In contrast to RFCOMM, OBEX also fully supports headers (also user defined) to describe the context of the message. Peer2Me uses the OBEX protocol that enables the support for a variety of types of messages including serialisable objects. It also makes it possible to only send the headers, making it possible for the client to decide to receive the data or not.

Messages: Peer2Me provides a very flexible message handling. Figure 4 shows the structure of a message in Peer2Me. A message consists of one or more message parts. Each message part can be of one out of three main types: A file, a primitive data type and a serialisable object. All the data types supported in J2ME (int, double, float, string, boolean, char, long and short) can be used in a message type. This means that Peer2Me should be capable of managing and sending any information between peers. Since Peer2Me supports transfer of objects between peers, the framework could also be used to create a mobile peer-to-peer agent system. In order for the framework to send objects over the network, the object must be serialised. Currently, there are no serialisable interfaces available for J2ME, so we had to create our own. Our interface provides two methods: one for serialising and one for deserialising.

Detecting new nodes: MANETs are characterised by a very dynamic network topology where nodes continually are added and removed from the network. The discovery of new nodes was in Peer2Me implemented using the Bluetooth discovery protocol provided in J2ME that searches for all nearby Bluetooth devices. It then filters and performs a *service search* for all mobile phones detecting all devices running a specific Peer2Me service. If the discovery detects any new nodes, references to the new nodes are created so that a connection can be established later on. After a completed search, the node shares the result with all the notes it has found. This process is illustrated in Figure 5. In 5A, node A searchers and discovers B and C. In 5B, node A sends messages to B and C, which contain information about all the nodes in the vicinity. In 5C, all the nodes can start to establish connections and exchange data. A search for new nodes is initiated when a Peer2Me application is run. How often the search for new nodes is run during the execution of an application is up to the application developer to decide.

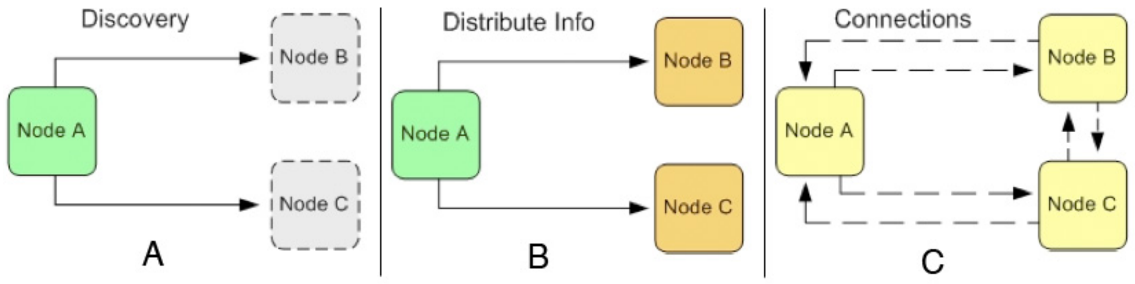


Figure 5. Detection of new nodes

Detecting lost nodes: In our Peer2Me framework, it was important to provide a mechanism to effectively detect when nodes were lost. The two traditional approaches for detecting that parts of the system are (not) available are to use the *ping/echo* and the *heartbeat* approach. We chose to implement the ping/echo approach, as this approach is more reliable. While for heartbeat, the nodes send out a message at a given interval, it requires the other nodes to constantly listen for specific messages containing the heartbeat, without the necessity to acknowledge the message. Under normal circumstances, the heartbeat approach would generate fewer messages than ping/echo. The ping/echo approach would normally generate $m = 2(n-1)$ while heartbeat would generate $m = (n-1)$, where m = number of messages and n = number of peers. The heartbeat approach generates half the messages compared to ping-echo. However, by using the OBEX protocol (in contrast to RFCOMM), the acknowledgement is unnecessary. In OBEX, the peer can simply try to establish a connection to check if the peer is available and alive. A failed connection will cause the node to remove the destination peer from its list. The cost of performing a connection test in OBEX is very low, as only a small header is needed to be transferred. A heartbeat approach would have required sending a message and would have required more data traffic over the network. The ping/echo mechanism can run constantly on the devices, as it uses a separate channel in OBEX, not interrupting the normal communication. The nodes are themselves responsible for detecting losses of other nodes and losses of nodes are not broadcasted. This makes it also possible to change the interval of performing the ping/echo depending on the available bandwidth of the network.

Initialisation of the framework: The first time a Peer2Me application is run, the framework has to be initiated from the MIDlet (the application) by calling the method *initialize()*. The initialisation will be performed through the layers of the framework by initialising the session layer, the network layer and the Bluetooth implementation. The last step of the initialisation process is to create a network listener (in current version a Bluetooth listener). The network listener makes it now possible for other nodes running the application to perform a service discovery and find this node. After the framework has been initiated, the MIDlet can start a search for other nodes as shown in Figure 6. The search is performed through the layers of the framework and the MIDlet is alerted when a node is found.

Other design considerations: When testing J2ME applications in the actual execution environment, there was no support for a console for test output on the mobile devices. Even though this is available on simulators running on PCs, it is essential to be able to track errors on the mobile devices. From our experience of developing mobile applications for over 5 years, we know that applications running fine on simulators do not necessarily behave the same way on the physical mobile phone. To catch and manage runtime errors, the Peer2Me framework provides a strong handling of exceptions that catches typical Peer2Me errors like lack of initialisation of framework, file, group and node not found etc. In addition to the exception handling, we also provide a log package that can be used by all classes in the framework that stores and manages runtime-messages. The log is useful for debugging as well as getting real-time information of the framework's progress and well-being.

3. Peer2Me APPLICATION DEVELOPMENT

As mentioned in the beginning of this paper, a main goal of the Peer2Me project was to create a framework for rapid application development of mobile peer-to-peer applications. In this section, we will describe through code examples the main parts of the Peer2Me API that an application developer has to use.

3.1. Initialisation of a Peer2Me application

Before you start to create any Peer2Me application, you need to import all the necessary parts of the framework:

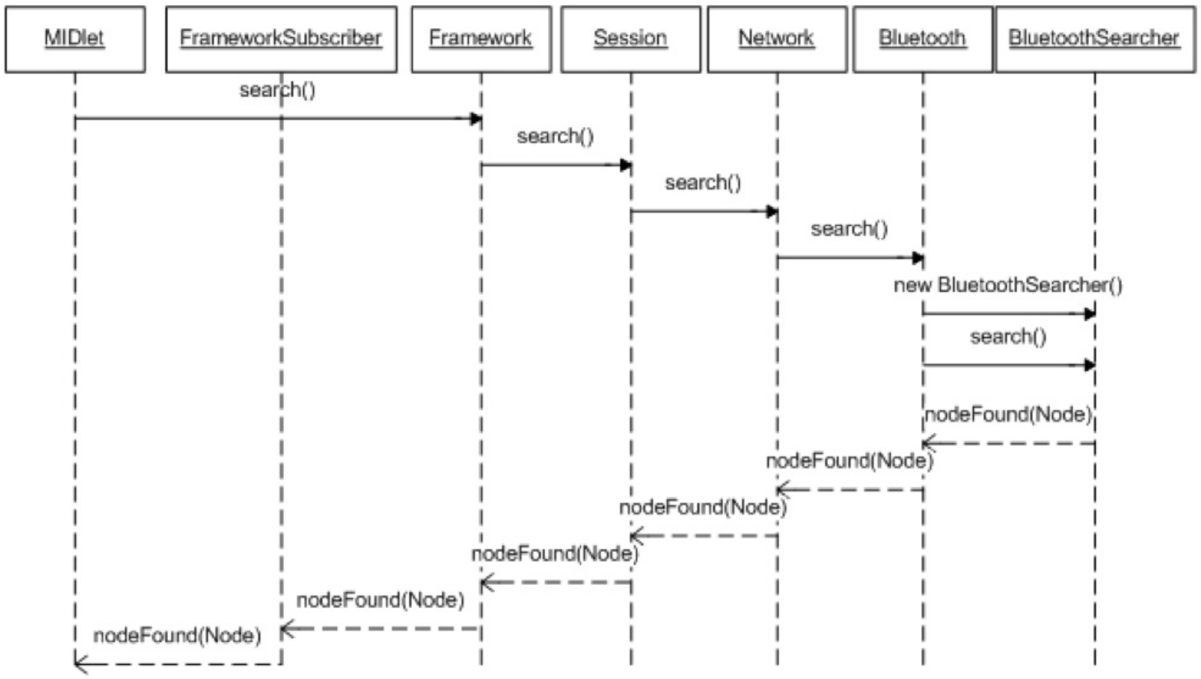


Figure 6. A sequence diagram for the initial search for nodes

Framework, FrameworkSubscriber, Message, Bluetooth and Node. The rest of the initiation of the Peer2Me framework is shown in Listing 1.

Listing 1. Initialisation of Peer2Me

```
public class Chat2Me extends MIDlet implements        1
    FrameworkSubscriber, Commandlistener {
 private Framework framework;                         2
                                                      3
 framework = Framework.getInstance(”MyGroup”,”        4
     Chat2Me”,new Bluetooth(),this);
 framework.initialize();                              5
 framework.search();                                  6
```

In line 1, the MIDlet created must implement the *FrameworkSubscriber* interface from the Peer2Me framework. The *CommandListener* is a J2ME interface to catch events from the user interface. Then a framework variable must be created (line 2). The lines 4 and 5 initiate the framework with the parameters for name of group, name of service, the network used and a reference to the MIDlet itself. The device running the application is now available for service discovery from other devices running Peer2Me. Line 6 searches for nearby devices running the same Peer2Me service.

3.2. Event-handling in Peer2Me

After the framework has been initiated, the Peer2Me application need to implement the four methods, *nodeDiscovered, nodeLost, searchCompleted* and *messagePart* provided by the FrameworkSubscriber interface. The *nodeDiscovered* is invoked when a new node is found in the network. Note that there is no automatic search for new nodes in Peer2Me after the initial search (line 6 in Listing 1) has been completed. The application programmer must include a search for new nodes in the application herself.

The methods *nodeLost* is invoked when the background ping/echo mechanism has detected that one of the nodes in the group is not reachable anymore. Detection for lost nodes is run in background by the framework in a separate thread.

The *searchCompleted* method is called when the initial search (line 6 in Listing 1) has completed.

Listing 2 show an example of how *nodeDiscovered*, *nodeLost* and *searchCompleted* can be implemented. In the implementation of the first two methods, the *node.getNodename()* method is used to notify the user through the GUI that the node has joined or left. For the latter method, the application sets the GUI in focus and refreshes the GUI.

Listing 2. Example of use of nodeDiscovered

```
public void nodeDiscovered(Node node) {
 dialog.append(node.getNodename()+”␣has␣joined”
     ,null);
}

public void nodeLost(Node node) {
  dialog.append(node.getNodename()+"␣has␣left",
       null);
  }

public void searchCompleted() {
          display.setCurrent(dialog);
}
```

Another method that can be invoked by the framework is the *messageReceived* method. This method is invoked whenever the node receives a message from another node. This methods is further described in Section 3.3.

3.3. Manage Messages in Peer2Me

As described in Section 2.3, the management of messages in Peer2Me is flexible in terms of what data can be sent between peers. In addition, the management of message interface is easy to use and understand for the application programmer. Listing 3 shows an example of how two different types of messages can be created and sent to another node. Note that every element added to a message uses a key string as an identifier (the second parameter).

Listing 3. Create and send message

```
Message message = new Message();
message.addElement(1,"type");
message.addElement("Hello␣world!","info");
message.addReceipt(node);
framework.sendMessage(message);

Message message2 = new Message();
message2.addElement(2,"type");
message2.addElement(true,"info");
message2.addReceipt(node);
framework.sendMessage(message2);
```

Listing 4 shows how a message can be received. In this example, the message element is used to identify what type of data is sent in the message element called *"info"*.

Listing 4. Receive message

```
public void messageReceived(Message message) {
 int type = message.getInt("type");
 switch (type) {
  case 1:
   String info = message.getString("info");
   break;
  case 2:
   boolean info = message.getBoolean("info");
   break;
  default:
   break;
 }
}
```

As seen from the listing, the message handling is flexible, but yet easy to use.

As described in this section, the Peer2Me API is high-level and requires few concepts to deal with the complexity of peer-to-peer computing. In the following section, we will describe experiences from using the framework to create applications.

4. EXPERIENCES AND EVALUATION

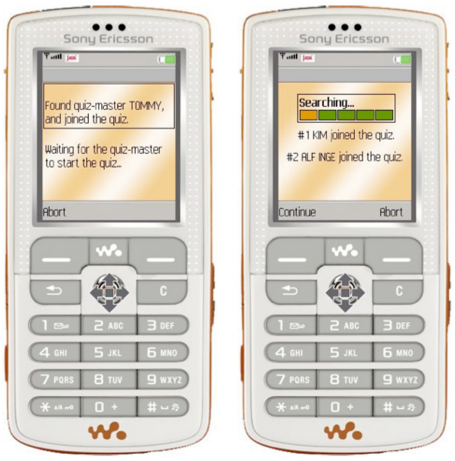
In this section we will share some experiences from creating Peer2Me application and evaluate the framework.

4.1. Peer2Me Applications

The work with the Peer2Me framework started in 2003 by analysing several mobile peer-to-peer scenarios that involved collaborative aspects. At this stage of the project we investigated scenarios like multi-player strategy games like risk, real-time Bluetooth games, synchronisation of information on mobile devices when users pass each other, support for planning the next meeting, etc. All these scenarios were characterised by users that were at the same location where the mobile phones needed to exchange some data.

Through the history of Peer2Me (current version is 2.0), we have also developed several Peer2Me applications. Here is a short description of some of them:

- **PeerQuiz:** A quiz-game for users with mobile phones being in the same area. PeerQuiz is initiated by one user sending a set of questions to all surrounding users. All users that accept the challenge will have to choose between possible answers for a set of questions, and a winner will be declared based on most correct answers. Figure 7 shows the screenshots of the PeerQuiz application and Figure 8 shows a picture taken where 4 students played PeerQuiz in a public area.


Found quiz-master TOMMY, and joined the quiz.

Waiting for the quiz-master to start the quiz...

Searching...

#1 KIM joined the quiz.

#2 ALF INGE joined the quiz.

Figure 7. Screenshots of the PeerQuiz application

- **PeerShare:** This is a file-sharing application for mobile phones that uses Bluetooth to transfer files between the devices. Any kind of files can be exchanged, but usually mp3-files and ring-tones are exchanged.

Screenshots from the PeerShare application is shown in Figure 9.

Figure 8. User test of the PeerQuiz application

- **Peer2MeAnalyzer:** This application is used to test performance of the Bluetooth network in a peer-to-peer topology. The application can measure discovery time, connection time, data transfer speed and enable tests of network range of Bluetooth devices.

- **Chat2Me:** A proximity-based multi-user chat application for mobile phones.

- **Converging top ten list:** An application for automatically updating prioritised lists when mobile users pass each other. This application can e.g. be used to find the top-ten beer price list of the campus area.

- **Business card exchange [21]:** This application makes it possible for the mobile devices to people in an area to search for people with the same interests and exchange electronic business cards if any are found (or beep the user).

- **Find next meeting:** This application is used for people in the same room or area to find possible dates and times for a meeting within a certain time slot by searching each of the users calendar on the mobile phone.

- **Peer voice message exchange:** This application makes it possible to exchange voice messages directly between mobile devices.

The various applications developed have different characteristics. Some of the applications uses asynchronous data exchange, while others requires real-time data exchange. In some of the applications, the user will initiate interaction with other users, some applications interaction without any user intervention, while others are hybrids [20]. Although we have developed a few applications, we believe that we have only scratched the surface of possible Peer2Me applications.

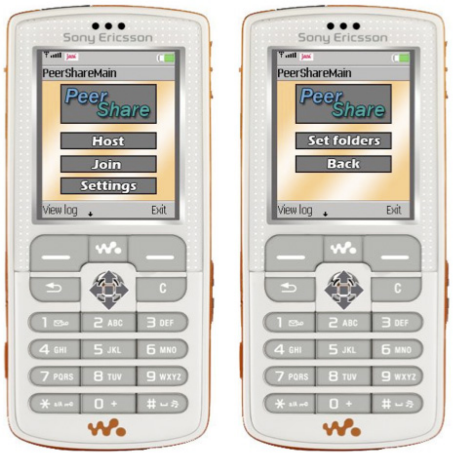


Figure 9. Screenshots of the PeerShare application

4.2. Emulators vs. Mobile phones

During the development of the Peer2Me framework and Peer2Me applications we used both emulator environments and mobile phones for testing. One might think that if the software runs smoothly on an emulator, it will do the same one a mobile phone. However, this is not the case. One main difference between running J2ME applications on an emulator and on an actual phone is performance. When running on an emulator, you do not have to consider performance issues. When running on a mobile phone, this becomes essential. We discovered this problem from testing early versions of the framework running several threads on the mobile phones. Running from two up to four threads would not cause any problems on most phone models we tested. However, running up to ten threads resulted the phone to stop responding or simply turning itself off. This experience resulted in an implementation of the framework with few threads and careful synchronisation of the threads. Another important limitation is the memory consumption. When running on an emulator, we do not have to think about this problem. When running on a mobile phone, memory is essential.

4.3. Peer2Me Technical Data and Performance

The Peer2Me framework has been optimised to provide a small footprint to reduce use of memory and storage capacity. The footprint of Peer2Me version 2.0 is only 37,5kBytes. The typical footprint of a simple MIDlet application running Peer2Me is about 10kBytes. The footprint of the version 1.0 of Peer2Me was 47.2kBytes even it did not provide all the functionality and not the same high-level API as in version 2.0. We have been able to provide a more advanced framework, with higher abstraction level and reducing the footprint by almost 10kBytes. We also carried out some performance tests on the v1.0 version and v2.0 version of the framework. The v1.0 used the RFCOMM protocol, while the v2.0 used OBEX. In v1.0, the average transfer rate of data between nodes was 7kBytes/sec. In v2.0, the average transfer rate of data between nodes was 18kBytes/sec (over 250% improvement). We believe that the improved performance using OBEX is because RFCOMM divides the data-stream into many small packets that all will be sent with overhead. In OBEX, it is possible to send data up to 10kBytes as one packet with minimal overhead.

4.4. Is Peer2Me suited for Rapid Development?

The main goal of the Peer2Me project was to provide a high-level API for J2ME for creating mobile peer-to-peer applications that hides the underlying complexity. We have achieved this goal by providing the application programmers with only few concepts to learn: A simple initiation, search of nodes, easy to use yet powerful message handling and few event interfaces to manage node dynamics. To compare how our framework compares to using the Bluetooth API in J2ME directly, we implemented a Bluetooth chat application, Chat2Me, and compared it to the BlueChat application developed by Ben Hui [2]. The applications provided the same level of functionality, but Chat2Me provided a more advanced GUI. If we compare only the total number of lines, the BlueChat consisted of 242 while the Chat2Me consisted of 138 lines of code. This means that the BlueChat required 175% more lines of code. We also compared the network specific code, and here the BlueChat application consisted of 42 lines while Chat2Me consisted of 16. The reduction was over 250% lines of code. We have also arranged workshops with several programmers to see how much time it was required to program the network code for a mobile peer-to-peer application using Peer2Me compared by using the Bluetooth API provided in J2ME directly. The results showed that by using the Peer2Me framework, the application programmer could create mobile peer-to-peer application within an hour. From our own experiences using the standard Bluetooth API in J2ME, it would require from 8 to 16 hours to do the same thing.

5. RELATED WORK

There are several projects that have developed frameworks for developing peer-to-peer application in MANETs. We will in this section present the most prominent projects.
JXTA [14, 4] is an open-source framework for developing P2P applications. JXTA provides a set of protocols and APIs for general-purpose, computer-to-computer communication and is platform and network independent. JXME [1] is JXTA for Java 2 Micro Edition (J2ME) and is a lightweight implementation of JXTA for mobile devices. It is specifically aimed at devices without sufficient computation and/or communication resources to participate in the network on their own. The JXME implementation provides full JXTA functionality through the use of a relay host. There is also a JXME proxiless initiative, but there is currently no stable implementation. As JXTA does not have a pure peer-to-peer version working for J2ME, it cannot be compared to Peer2Me.

Mobile Chedar [12] is a middleware being an extension to the Chedar peer-to-peer network allowing mobile devices to access the Chedar network and communicate to other Mobile Chedars. The goal of the Chedar software is similar to Peer2Me: To provide a convenient API for peer-to-peer application developers. The Mobile Chedar is implemented in J2ME and Bluetooth is used for communication. In contrast to Peer2Me, the Mobile Chedar is based on a hybrid peer-to-peer model that uses a Mobile Chedar gateway node as the master in the network. The Mobile Chedar gateway node is run on a PC that also provides an Internet gateway for the mobile nodes. However, this approach suffers from having a single point of failure like client-server solutions.

Proem [10] is a framework for developing and deploying P2P collaborative applications in a mobile ad-hoc networking environment. The main objective of Proem is to provide a common framework for rapid development of applications for ad-hoc network environments. The framework is implemented in Java, and can be run on various wireless mobile devices. Proem is designed to be independent of underlying network transport protocols, and can be implemented on top of TCP/IP, HTTP, Bluetooth and others. The original Proem was based on a Java Standard Edition, limiting the devices to run Proem to powerful PDAs. There have been attempts to create a J2ME version of Proem that have not succeeded.

The JMobiPeer [3] framework is very similar to Peer2Me in many respects. It provides support for discovery, group management and peer management. In addition JMobiPeer offers interoperability with JXTA. The implementation of JMobiPeer is based on J2ME. However, the actual execution of JMobiPeer has only been tested on emulators on standard PCs. This is probably because the framework has high requirements on CPU and memory. In addition, the framework does not reveal any details on the API or if they provide a pure or hybrid peer-to-peer solution.

MOBY [9] provides a network for mobile peer-to-peer exchange of services and data. MOBY offer a dynamic service location and client mapping to achieve an adaptive network optimising performance and reliability. MOBY uses heavily JINI functionality and can there for not be run in a J2ME environment.

6. CONCLUSION

In this paper we have presented the Peer2Me framework for rapid development of mobile peer-to-peer applications. Despite the limitations in J2ME with a stripped down Java class-library and the limited resources on mobile phones in terms of CPU and memory we have managed to develop a full-fledged framework that transparently manages a MANET. The Peer2Me API is very simple to learn and consists only of a few lines for framework initialisation and implementation of four methods that will be invoked for events (like when a node is unavailable). Our message interface is easy to use but powerful and flexible enough to exchange any data type including serialisable objects. The latter makes it possible to use Peer2Me as the foundation of a mobile agent system for MANETs. In order to do so, the state of the objects need to be extracted before sending the agent, and the object must be re-initiated with the same state on the receiving node.
The main contribution of the Peer2Me project is to provide a high-level API for peer-to-peer computation for mobile phones using Bluetooth. The Peer2Me framework is very useful for exploring computer support for spontaneous collaboration (when users occasionally meet) and for mobile collaboration in general. Through our applications we have demonstrated some areas where Peer2Me can be used. We will continue to explore various ways to utilise the Peer2Me framework to support mobile collaboration.

ACKNOWLEDGEMENT

We would like to thank Thomas Fossum, Lars Kirkhus, Anders R. Sveen, Michael Sars Norum, Carl-Henrik Wolf Lund, Steinar Hestnes and Torbjørn Vatn for their invaluable effort in the Peer2Me project.

REFERENCES

[1] C. W. A. Arora and K. S. Pabla. jxme: JXTA Platform Project. Web: http:
www.jxme.org, February 2005.

[2] Benhui.net. Connecting PC and Phone with Java Bluetooth API - Part 1. Web: http://benhui.net/modules.php?name=Bluetooth, 2006.

[3] M. Bisignano, G. D. Modica, and O. Tomarchio. JMobiPeer: A Middleware for Mobile Peer-to-Peer Computing in MANETs. In *First International Workshop on Mobility in Peer-to-Peer Systems (MPPS) (ICDCSW'05)*, pages 785–791, 2005.

[4] J. Brendon and J. Wilson. *JXTA*. New Riders Publishing, 2002.

[5] D. Clark. Face-to-face with peer-to-peer networking. *Computer*, 34(1):18–21, 2001.

[6] N. Davies, K. Cheverst, K. Mitchell, and A. Efrat. Using and determining location in a context-sensitive tour guide. *Computer*, 34(8):35–41, 2001.

[7] N. Eagle and A. Pentland. Social Serendipity: Mobilizing Social Software. *IEEE Pervasive Computing*, 04(2):28–34, 2005.

[8] L. E. Holmquist, J. Wigstrom, and J. Falk. The Hummingbird: Mobile Support for Group Awareness. In *Demonstration at ACM 1998 Conference on Computer Supported Cooperative Work*, 1998.

[9] T. Horozov, A. Grama, V. Vasudevan, and S. Landis. MOBY - A Mobile Peer-to-Peer Service and Data Network. In *2002 International Conference on Parallel Processing (ICPP'02)*, pages 437–444, 2002.

[10] G. Kortuem. *A methodology and software platform for building wearable communities*. PhD thesis, University of Oregon, December 2002.

[11] G. Kortuem, J. Schneider, D. P. Thaddeus, G. C. Thompson, S. Fickas, and Z. Segall. When Peer-to-Peer comes Face-to-Face: Collaborative Peer-to-Peer Computing in Mobile Ad hoc Networks. In *First International Conference on Peer-to-Peer Computing*, Linköping, Sweeden, 27-29 August 2001.

[12] N. Kotilainen, M. Weber, M. Vapa, and J. Vuori. Mobile Chedar A Peer-to-Peer Middleware for Mobile Devices. In *Third IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'05)*, pages 86–90, 2005.

[13] S. Long, R. Kooper, G. D. Abowd, and C. G. Atkeson. Rapid prototyping of mobile context-aware applications: The cyberguide case study. In *Mobile Computing and Networking*, pages 97–107, 1996.

[14] N. Maibaum and T. Mundt. JXTA: A Technology Facilitating Mobile Peer-To-Peer Networks. In *International Mobility and Wireless Access Workshop (MobiWac'02)*, pages 7–13, Fort Worth, Texas, USA, 12 October 2002.

[15] S. Microsystems. Java 2 Platform, Micro Edition (J2ME). Web: http://java.sun.com/j2me/, 2005.

[16] B. A. Miller and C. Bisdikian. *Bluetooth Revealed*. Addison-Wesley, 2 edition, 2002.

[17] P. Mohapatra, C. Gui, and J. Li. Group communications in mobile ad hoc networks. *Computer*, 37(2):52–59, 2004.

[18] M. Satyanarayanan. Fundamental Challenges in Mobile Computing. In *Fifteenth ACM Symposium on Principles of Distributed Computing*, Philadelphia, PA, 1996.

[19] M. P. Singh. Peering at peer-to-peer computing. *IEEE Internet Computing*, 05(1):4–5, 2001.

[20] A. I. Wang, M. S. Norum, and C.-H. W. Lund. Issues related to Development of Wireless Peer-to-Peer Games in J2ME. In *First Conference on Entertainment Systems (ENSYS 2006)*, page 6, Guadeloupe, French Caribbean, February 23-25 2006.

[21] A. I. Wang, C.-F. Sørensen, and T. Fossum. Mobile Peer-to-Peer Technology used to Promote Spontaneous Collaboration. In *The 2005 International Symposium on Collaborative Technologies and Systems (CTS 2005)*, page 8, Saint Louis, Missouri, USA, May 15-19 2005.

[22] M. Wiberg and Åke Grönlund. Exploring Mobile CSCW: Five areas of questions for further research. In *Proceedings of IRIS23 (Information Research in Scandinavia)*, Trollhättan, Sweden, 2000.