Mobile Peer-to-peer Collaborative Framework and Applications

Alf Inge Wang
Dept. of Computer Science and Technology, Norwegian University of Science and Technology
Sem Sælandsv. 7-9, NO-7491 Trondheim, Norway
Phone number: +47 7359 4485, Fax number: +47 7359 4459
Email: alfw@idi.ntnu.no

Mobile Peer-to-peer Collaborative Framework and Applications

This chapter describes the Peer2Me mobile peer-to-peer framework, Peer2Me applications, and experiences from using the Peer2Me framework. Peer2Me supports mobile collaboration utilizing Bluetooth and Java ME. The framework runs on standard Java ME-enabled mobile phones and it enables rapid development of various kinds of collaborative peer-to-peer applications. In the chapter we describe some of these applications as well as experiences from implementing these peer-to-peer applications: a file-sharing application, a chat application, a quiz game, a face-to-face meeting scheduler, a real-time game, an automatic business exchange application, and a find the right person application. All these applications have been analysed to discover limitations of the framework, limitations of the technology and the potential usefulness. Finally, the analysis is summarised to give a more complete picture of the potential and the limitations of using Bluetooth and Java ME to implement mobile peer-to-peer applications.

Keywords: Mobile Technologies, Software Architecture, Collaborative Technologies, Prototyping, and Software Engineering.

INTRODUCTION

Most peer-to-peer applications and architectures today 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 migrate the peer-to-peer computing to a wireless environment (Kortuem et al., 2001) (Maibaum & Mundt, 2002). The downside of bringing peer-to-peer computing to the mobile and wireless platform is that we have to face the classical challenges of mobile computing related to how to handle wireless communication, how to solve issues related to mobility of the user, and how to overcome the limitations introduced by the portability of the mobile device (Satyanarayanan, 1996). Mobile peer-to-peer computing also offers new opportunities that can be utilized like providing location-based services (Davies et al., 2001) (Long et al. 1996) and social computing (Eagle & Pentland, 2005) (Holmquist et al., 1998) using short-range networks.

Most wireless devices support some kind of personal area network (PAN) technologies like irDA and/or Bluetooth (Miller & Bisdikian, 2004). PANs are commonly used for transferring data between two mobile devices. A PAN can be seen as a digital sphere around the mobile device enabling a collaborative network for users within range. The digital sphere opens for mobile computer supported cooperative work (mobile CSCW) (Wiberg & Grönlund, 2000), (Papadopoulos, 2006). In such environments, 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 (Mohapatra, 2004). MANETs can be utilised in situations where persons with mobile devices meet and there is a need for exchange of data.

MANETs opens for new kinds of user-interaction. 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 (Wang et al., 2006). 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 (Wang et al., 2005). Further, MANETs can be used to create application for proximity chats and file exchanges, or simply for leisure like games.

This book chapter describes a framework for implementing mobile peer-to-peer applications, explores and evaluates several mobile peer-to-peer applications and evaluates the limitations of Java ME and Bluetooth in this context.

BACKGROUND

This section gives an introduction to the background and important terms used in our framework, and describes related work.

Mobile Computer Supported Cooperative Work

Research within Computer Supported Cooperative Work (CSCW) has grown to be a mature research area. However, there are still problems concerning the use of computers for cooperation that remain unsolved. Olson et al. list several advantages of collocating a work force to improve cooperation such as efficient communication paths, less ambiguity in communication, more efficient synchronization of work, and better knowledge management (Olson et al., 2002). The advantages from being collocated stem from the fact that collaboration is probably the most complex, advanced, and unstructured form of human-to-human interaction. Current technology is too limited to cope with such complexity and is therefore not sufficient to solve all the problems in the CSCW domain.

Ellis, Gibbs and Rein describe the different types of CSCW systems/applications in the two dimensions time and place (Clarence et al, 1991). The *time dimension* divide CSCW applications either into real time or asynchronous applications, while the *place dimension* divide such applications into same place or different place. An email-application would according to this model typically be characterised as asynchronous and different place, while a chat-application would be characterised as real time and different place. Most CSCW research has focused on applications that fall into the ”Different Place” category where the CSCW application will be the only communication channel used for collaboration. The users' abilities to communicate and cooperate are limited by the insufficiencies in the technologies and applications used. In the ”Same Place” category, especially coupled with ”Real Time”, CSCW becomes more of a support for the collaborative effort to enrich or strengthen the processes and communication paths.

The Peer2Me framework and applications described in this chapter covers both real time and asynchronous applications in the ”Same Place” category in a mobile environment. The targeted platform for our framework is mobile phones or similar mobile devices as the computer devices must be where the user is (not the opposite). Any mobile device like a PDA or an ultra mobile PC can be used as a target platform for such applications. The main benefit of using a mobile phone as a mobile platform instead of PDAs or ultra mobile PCs are the user base (number of users) for mobile phones is much larger. Further, to use mobile phones as the execution platform for CSCW applications has several advantages over traditional CSCW. *Firstly*, the mobile phones are personal devices meaning that can be used to identify a user. *Secondly*, a user can store his profile on the mobile phone because it is a personal device, enabling the mobile phone to function according to the user's specific needs when interacting with other users. *Thirdly*, mobile phones can be considered to be always on, always present. Due to this, someone using his mobile phone for CSCW purposes will achieve a high degree of user availability compare to traditional CSCW systems where the user is not always with his computer.

Current mobile phones support more than one communication network/technology. Still the most important, and the one with the longest range, is the cellular network provided by the telecom operators. In addition, most mobile phones support low-range personal area networks (PANs). Such networks have a typically range from a few meters up to 50 meters depending on the blocking of signals in the environment. Examples of such PAN technologies provided in many mobile phones are infrared (requires line of sight), Bluetooth and WiFi. Such ad hoc network technologies enable devices to detect and connect to devices that are in sufficient proximity in a decentralized manner. The characteristics of such networks resemblance strongly to the nature of human spontaneity, which make PANs suitable for making spontaneous collaborative applications. A PAN creates a *digital sphere* around a person. The communication range of the PAN limits the digital sphere.

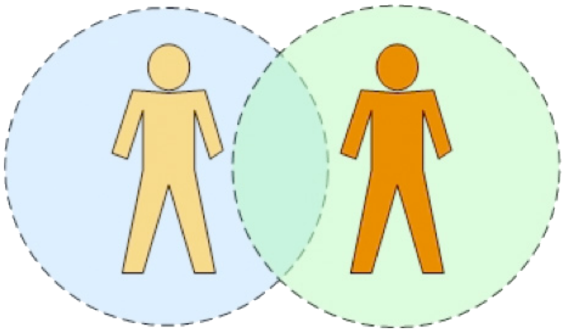


Figure 1 Illustration of digital sphere

From the perspective of mobile CSCW, people are physically collocated when the digital spheres of two or more persons overlaps (see Figure 1). When digital spheres overlap, the mobile devices can start to interact and form a *mobile ad hoc network (MANET)*. The topology of such networks can often be characterised as peer-to-peer. "A distributed network architecture may be called a peer-to-peer network, if the participants share a part of their own hardware resources (processing power, storage capacity, network link capacity, printers, etc). These shared resources are necessary to provide the *service* and *content* offered by the network (e.g. file sharing or shared workspaces for collaboration). They are accessible by other peers directly, without passing intermediary entities. The participants of such a network are thus resource providers as well as resource requestors" (Schollmeier, 2001). The peer-to-peer topology is better suited to cope with dynamic changes compared to the classical client-server topology. This is mainly because client-server has a single point of failure, while in a peer-to-peer network any node can communicate with any other node. Also the peer-to-peer topology better describes and models collaborative patterns between users, as there are direct communication paths between the users. Together, PANs and the P2P topology provide the most suitable basis for building collaborative applications that can be characterised as same place applications.

By using low-range PANs for mobile CSCW applications, the collaborative efforts will have to be either based on chance encounters between peers (impromptu collaboration) or a planned meeting or gathering of peers (formal collaboration). Impromptu collaboration can involve different degrees of user interaction (Wang et al., 2006):

- **Controlled:** In this category the application controls how the users interact through a well-defined protocol where typically one of the peers in the network must be a master controlling the user interaction. This user interaction pattern is typically used for applications that are turn-based or applications that require that the users exchange data according to a predefined pattern. *Example:* A turn-based game like the strategy game Risk.
- **User interaction:** In this category, the users have explicitly to trigger actions that will cause interaction (exchange of data) with other users. The user has to trigger explicitly the collaboration activities, start the information search or request a service. *Example*: Two people at the bus stop that want to exchange MP3 files.
- **Automatic triggered:** In this category the devices automatically trigger collaboration that requires further user interaction. Example: The mobile devices carried by two different people automatically communicate without user interaction and discover that the two persons are sharing the same taste in music. The two people are alerted and are given the possibility to share some MP3 files.
- **Automatic:** In this category the application is responsible for automatically initiating communication between his device and other devices on behalf of the user. The user stores a profile that defines how the application should act with respect to other devices/users and available services. *Example*: A person automatically exchanges MP3 recommendations with other people he or she meets when walking around at the campus.

Formal collaboration can be characterized as being proximity-based, but due to its organized nature it is not opportunistic and spontaneous. This more formal form of using CSCW on mobile phones is more suitable in situations where a collection of users automate parts of their collaborative work process - typically a workflow system (Jing et al., 1999). Our framework focuses mainly on support for impromptu collaborative applications, but can also support formal collaboration like planning the next meeting.

The research within the area of mobile peer-to-peer collaboration has focused on three main areas: Development of frameworks, architectures or technologies to support mobile peer-to-peer applications, development of new innovative applications to support mobile collaborative work, and evaluation of mobile peer-to-peer frameworks, architectures, technologies and applications. The next sections review some of the work in this area.

Mobile Technologies

Mobile technologies have developed rapidly the last couple of years resulting in many different types of mobile devices, a wide spectrum of mobile execution and development platforms, and many types of wireless network technologies.

It is a critical decision for the mobile application developer to choose the appropriate mobile platform for the target audience. The following gives an overview of existing mobile technologies.

The types of mobile devices can be categorised in many ways, but it is hard to find one taxonomy that covers everything - as the functionality and abilities of the mobile devices tend to overlap. The most popular mobile device is the mobile phone with the largest number of users. Previously personal digital assistants (PDAs) (Davids, 1996a) (Davids, 1996b) were popular, but their popularity has fell mainly because most mobile phones now include PDA functionality. Smart phones (Zheng & Ni, 2006) are hybrid devices that combine the functionality of a mobile phone and a PDA but the functionality gap between mobile phones and smart phones are decreasing day by day. Another category of mobile devices is mobile computers, which denotes ultra mobile PCs (UMPCs) (Broll et al., 2008). The main difference between mobile computers and mobile phones and smart phones is that mobile computers are bigger in size and usually runs the same operating system as larger computers. Most mobile phones and mobile computers can install and run mobile applications made for the appropriate mobile platform (see next paragraph). In addition to these multipurpose mobile devices, there are several types of specialized mobile devices like handheld game consoles, mobile media players or recorders, personal navigation devices (like GPS) and others. Some of these mobile devices built for specific purposes are programmable and thus opens for new use of specific purpose mobile devices. In the recent years the trend has been to develop one mobile device that can do everything. One example of such a device is the iPhone from Apple. The iPhone (Macedonia, 2007) is a mobile phone, a smart phone, a GPS, a handheld game console, and a media player. Mobile devices can vary in many respects: size, capacity (CPU, memory, hard drive), screen size, input devices, operating system, support for wireless networks, sensors, special purpose electronics for RFID (Michahelles et al., 2007), GPS (Schreiner, 2007), and smart cards (Husemann, 1999).

Most existing mobile devices allow development of applications that runs on different execution/development platforms (Vaughan-Nichols, 2003). The pioneers of popular mobile operating systems were the Palm OS originally developed for PDAs (later also for smart phones) and Symbian OS for mobile phones. Later the Windows mobile platform and Linux were launched for development on PDAs and later for smart phones and mobile phones. The programming languages used for these platforms are C, C++, and C#. The most popular development platform for mobile devices today is the Java Micro Edition (Java ME) (Helal, 2002), which runs on top of a mobile operating system. The main reason for its popularity is that Java ME applications are device independent and can run on most kind of devices and operating systems. This is only partly true, as the Java ME virtual machines are implemented slightly different on different operating systems and devices, and that the user interface usually must be tailored to the various screen sizes and input devices. There have also been some attempts on developing pure Java-based operating systems for mobile devices, but this platform has not become very popular. Lately two new platforms have received a lot of attention. One is the Apple's iPhone SDK (Apple, 2008) that allows developers to develop applications for iPod Touch and iPhone running a minimized version of Mac OS X. iPhone or iPod Touch applications are written in the programming language objective-C, C or C++. The other platform is Google's open source Android platform (Google, 2008), which runs the Dalvik virtual machine on top of a

Linux mobile operating system. The applications in Android are written in the Java language, but compiled into Dalvik Bytecode.

Most mobile devices have built-in support for at least one wireless network technology. Since this book chapter is about peer-to-peer applications for users being co-located, the telecom wireless networks such GSM will not be described here. Most of the network technologies for close-range communication are based on radio, but there are also alternatives using infrared or ultrasound. The most common alternative to radio-based networks today is infrared data association standard (IrDA) (Ashok & Agrawal, 2003). IrDA can provide data transmission speed from 2.4Kbit/s up to 16Mbit/s and the range can be up to 5 meters. IrDA transmitters and receivers are cheap to implement and does not consume much battery. The main disadvantage is that IrDA transmissions require line of sight to work. The most common radio-based wireless network technologies used today is Bluetooth (Reynolds, 2008) and WiFi (Kapp, 2002). Bluetooth was design for smaller devices, requires less battery, but suffers from lower transmission speeds (max 3Mbit/s for version 2.0) and a max range of up to 100 meters. The IEEE 802.11 standards (a, b, g, n and y) for WiFi comes in various configurations with different speeds and range from 2Mbit/s and range of 100 meters outdoors and up to 248Mbit/s and range of 250m outdoors. An alternative is ZigBee (Geer, 2005), which is a radio-based wireless network technology intended for simple lightweight devices and provides data transmission speeds from 20Kbit/s to 250Kbit/s with range form 10 to 75 meters. In 2008, two new wireless network standards mainly to be used instead of cables between devices have been defined. The Wireless USB (WUSB) (Leavitt, 2007) standard is a short-range, high-bandwidth radio communication standard with data transmission speeds up to 480Mbit/s at 3 meters and 110Mbit/s at 10 meters. The competing standard is the Wireless firewire (Zhang et al., 2001) standard that should give data transmission speeds up to 480Mbit/s at the range up to 9 meters.

To be able to develop mobile peer-to-peer applications that are highly available and useable, the underlying mobile technology needs to be:

- **Widely available and supported by most mobile devices:** This would make ensure a huge number of potential users.
- **The mobile device must be highly portable:** Users must be able to utilise the peer-to-peer services wherever they are. In practise this means that the mobile device must be small and lightweight.
- **The range and the speed of the mobile network technology must be sufficient to establish networks:** The digital spheres must have some radius to ensure that people in the same area can connect and a data transmission speed sufficient for normal data exchanging data.

The underlying mobile technology for Peer2Me was chosen based on the requirements described above. When we considered the target mobile device for the Peer2Me framework, it was rather obvious that the mobile phone would give the highest potential number of users. In Norway where the Peer2Me framework was developed, nearly 100% of the population has a mobile phone. In addition, we chose to implement Peer2Me in Java ME. The Java ME platform makes it possible to run Peer2Me application on most mobile phones as well as many PDAs and smart phones that have a Java ME virtual machine installed. In addition, the choice of Java ME also enables Peer2Me applications to run on mobile computers. Since the mobile phone and Java ME was chosen as the device and software platform, Bluetooth was the only viable choice for a wireless network technology as Bluetooth is the network technology supported by most mobile phones, PDAs and smart phones. Bluetooth has sufficient range (about 50 meters in open air for mobile phones) and sufficient transfer speed (about 320Kbit/s for average mobile phones). Another reason for choosing Bluetooth was that it requires less battery than e.g. WiFi. In practice this means that the Peer2Me applications can be used for longer period of time before a recharge.

Other Peer-to-peer Platforms

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 (Maibaum & Mundt, 2002) 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 (Kawulok et al., 2005) 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 (Auvinen et al., 2006) 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 are 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.

MOBY (Horozov, 2002) 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.

Proem (Kortuem, 2002) 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.

PnPAP (Harjula et al., 2004) is a middleware developed at the University of Oulu in Finland. PnPAP is a plug-and-play application platform that enables dynamic selection between diverse peer-to-peer networks and session management protocols while preserving the best available network connectivity. The architecture of PnPAP consists of an API-layer, a PnPAP engine layer and a layer for handling the actual connection. The PnPAP platform has been developed for Symbian S60 platform and can support the UMTS, Bluetooth, GPRS and WLAN networks.

The JMobiPeer (Bisignano et al., 2005) 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.

Peer2Me – A Peer-to-peer Framework

The Peer2Me project was initiated to enable rapid development of proximity-based peer-to-peer applications for mobile devices built on top of the Java Micro Edition (Java ME) platform (Helal, 2002). Our main goal was to develop a high-level programming framework enabling developers to use simple primitives and methods to manage the complexity of peer-to-peer mobile ad hoc networks. It was also essential that the Peer2Me framework should be transparent and hide the network technology used for communication.

Our current implementation of the framework is based on the CLDC 1.1 and MIDP 2.0. In addition, the Peer2Me framework uses two optional packages (Java ME APIs):

- **JSR82** to access and manage Bluetooth networks
- **JSR75** to access Personal Information Management (PIM).

The current Peer2Me implementation only supports Bluetooth networks, but the architecture is made modular to also support other types of networks such as WiFi when they are supported in the Java ME environment and/or by the devices.

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 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 2 shows the high-level architecture and the main parts of Peer2Me framework (note that the MIDlet is not a part of the Peer2Me framework).

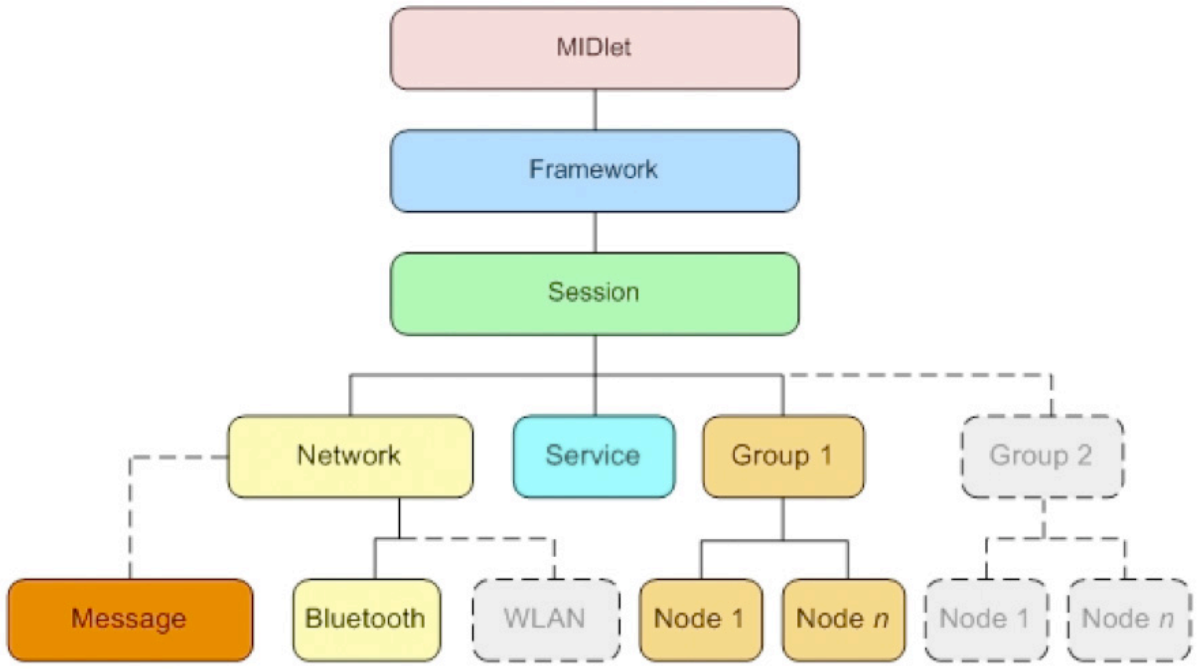


Figure 2 The Peer2Me High-level architecture

The Peer2Me high-level architecture consists of the following parts:

- **Node:** A node is a logical representation of a peer, i.e., a mobile phone running the framework. 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 acts as a unique ID for an application running the framework. Only Peer2Me applications *sharing the same service* can interact.
- **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 accessed directly 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, and 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.

To implement a Peer2Me application, the developer needs to import several parts of the framework: The *framework interface* itself, a *subscriber interface* acting as a listener for nearby peers, a *message interface* handling message exchange between peers, the *network interface* for setting the network, and the *node interface* giving a local representation of the nodes in the network. After the Peer2Me framework has been initiated, the Peer2Me application will automatically set a peer-to-peer mobile ad hoc network and search for other nearby devices running the same Peer2Me service. The Peer2Me framework will dynamically detect coming and going peers, which is handled through events. It is up to the developer of the Peer2Me application to define how to react to events in the peer-to-peer network. Through the subscriber interface, the developer must implement four methods that defines how the application should react to events:

- *searchCompleted*: This method is called when the Peer2Me framework has completed a search for nearby Peer2Me peers running the same service.
- *nodeDiscovered*: This method is called when a new node has been detected in the network.
- *nodeLost*: This method is called when a node can now longer be detected. The node detection mechanism is running in a separate thread and it sends out a ping to all nearby peers and waits for an echo from all of them.
- *messageReceived*: This method is called when a message has been sent from another peer.

The source code for the initialisation of a Peer2Me application is shown in Listing 1.

Listing 1. Initialisation of Peer2Me

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

In line 1, the MIDlet Chat2Me must implement the FrameworkSubscriber interface from the Peer2Me framework. The CommandListener is a Java ME interface to catch events from the user interface. Then a framework variable must be created (line 2). The lines 3 and 4 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 5 searches for nearby devices running the same Peer2Me service.

The Peer2Me framework has been downloaded and used at other institutions than Norwegian University of Science and Technology (NTNU), but we do not have a complete overview of the usage and application developed in Peer2Me by external institutions. For more details on the Peer2Me framework, see (Wang et al., 2007).

Framework to Characterize and Evaluate Mobile Peer-to-Peer Applications

The motivation behind the evaluation of the Peer2Me applications was to discover the usefulness and the usability of the applications, to assess how well the Peer2Me supported the implementation of the applications, and if there were any limitations of the application due to the underlying technology used (Java ME, mobile phone and Bluetooth). The framework consists of 14 characteristics, which are measured either as a number or in selection of a few specified textual short descriptions. Table 1 shows the framework composed of 14 characteristics and how each characteristic is measured. The characterisation is based on what has been described in the previous section Mobile Computer Supported Cooperative Work. The framework can be used to compare various mobile peer-to-peer applications, but also as a checklist of issues the developer should consider when developing a mobile peer-to-peer application.

Table 1 Framework for characterising mobile peer-to-peer applications

ID	Evaluation criteria/Characteristics	Measure
1	Number of users typically involved	(1-100 users)
2	Classification according to the place	(Same place, Different place, Both)
3	Classification according to time	(Asynchronous, Real time, Both)
4	Classification according to planning	(Impromptu collaboration, Formal collaboration)
5	Classification according to user interaction	(Controlled, User interaction, Automatic triggered, Automatic)
6	Classification according to collaboration pattern	(Master controlled, True peer-to-peer)
7	Classification according to how collaboration is improved	(Initiate collaboration, Improve coordination, Improve negotiation, Improve exchange, Improve communication)
8	The degree of usefulness of application	(Very low, Low, Medium, High, Very high)
9	The degree of replacing manual collaboration	(Very low, Low, Medium, High, Very high)
10	The degree of replacing existing collaboration support	(Very low, Low, Medium, High, Very high)
11	Limitations in the application due to wireless technology (Bluetooth)	(None, Some limitations, Severe limitations)
12	Limitations in the application due to development platform (Java ME)	(None, Some limitations, Severe limitations)
13	Limitations in the application due to the device (mobile phone)	(None, Some limitations, Severe limitations)
14	Limitations in the application due to framework (Peer2ME)	(None, Some limitations, Severe limitations)

Evaluation and Description of Mobile Peer-to-peer Collaborative Applications

This section describes various peer-to-peer applications that have been developed using the Peer2Me framework. The different applications have been developed to explore the possibilities of mobile face-to-face collaborative applications, and to discover the usefulness and limitations of such applications. This section also describes an evaluation and characterisation of the applications in terms of usefulness, usability, how difficult it was to realise the applications in Peer2Me, and how well the underlying technology can support the applications using the Peer2Me framework.

The Peer2Me developers performed the evaluation of the framework and applications them selves. The results of the following evaluation are based on more than four years of experimentation with development and usage of peer-to-peer applications. The subjects used in the usability tests were students at NTNU that volunteered to test Peer2Me applications, and we collected the usability data through a combination of observation, interviews and evaluation forms.

Peer2Share – A File Sharing Application

Peer2Share is a simple application for easily exchanging files between mobile devices using Bluetooth. Any kind of files like mp3-files, ring-tones, pictures, and movie clips can be exchanged. The application searches for all nearby devices running Peer2Share. The user must initiate the file exchange himself, and choose who to share files with and what files to share. The main difference with this application compared to the native file exchange support in most mobile phones is that the network connections and discovery of users are set up automatically and a user-interface specific for file exchange is provided. Figure 3 shows two screenshots from setting up the Peer2Share application.

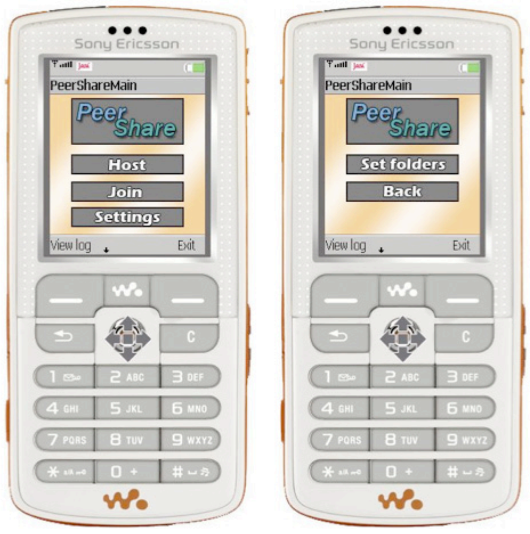


Figure 3 Screenshots from the Peer2Share application

The characterisation of Peer2Share is shown in Table 2. The characterisation shows that this application is master-controlled meaning that one mobile device is in charge for managing the communication. Two users typically use the Peer2Share application, although more users are supported. The application does not suffer from limitations in Bluetooth, Java ME, the mobile phone or Peer2Me. The Peer2Share application does not bring any new functionality compare to existing functionality on mobile phones, and the main contribution is an easier set-up and a more convenient user interface.

Table 2 Characterisation of Peer2Share

ID	Evaluation criteria/Characteristics	Result
1	Number of users typically involved	2 users
2	Classification according to the place	Same place
3	Classification according to time	Asynchronous
4	Classification according to planning	Impromptu collaboration
5	Classification according to user interaction	User interaction
6	Classification according to collaboration pattern	Master controlled
7	Classification according to how collaboration is improved	Improve exchange
8	The degree of usefulness of application	Medium
9	The degree of replacing manual collaboration	High
10	The degree of replacing existing collaboration support	Low
11	Limitations in the application due to Bluetooth	None
12	Limitations in the application due to Java ME	None
13	Limitations in the application due to the mobile phone	None
14	Limitations in the application due to Peer2ME	None

Peer2Chat – A Chat Application

Peer2Chat is a simple chat application for people being co-located. The application works like any other chat application and can be used to communicate with people in areas where you are not allow to talk like at a library or in a class room. In addition, the chat application can be used to start communicating with people you do not know at waiting areas like bus-stops, train stations and airports or at public transportation like in busses, trains etc. You can also use the chat application to play text-based games, like guessing riddles, quiz, etc.

The characterisation of Peer2Chat is given in Table 3. The characterisation below shows that from two to seven users are typically involved. The Peer2ME framework and how Bluetooth connections are established limits the maximum number of users. More than seven users can be supported if dynamic establishment of Bluetooth connections is added to the Peer2ME framework. Although this application is regarded as real time, it is not critical that the messages between the users are exchanged with less delay than 1 second. This application is regarded as a true peer-to-peer application as there is no central node managing the network traffic.

Table 3 Characterisation of Peer2Chat

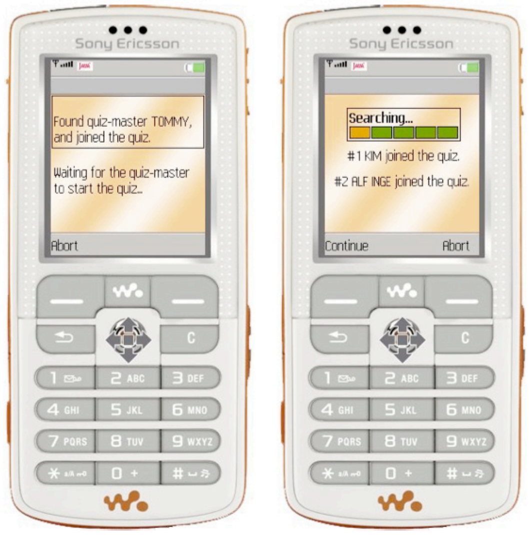
ID	Evaluation criteria/Characteristics	Result
1	Number of users typically involved	2-7 users
2	Classification according to the place	Same place
3	Classification according to time	Real time
4	Classification according to planning	Impromptu collaboration
5	Classification according to user interaction	User interaction
6	Classification according to collaboration pattern	True peer-to-peer
7	Classification according to how collaboration is improved	Improve communication
8	The degree of usefulness of application	Medium

9	The degree of replacing manual collaboration	Medium
10	The degree of replacing existing collaboration support	Medium
11	Limitations in the application due to Bluetooth	Some limitation (no scatternet support)
12	Limitations in the application due to Java ME	None
13	Limitations in the application due to the mobile phone	None
14	Limitations in the application due to Peer2ME	Some limitations (maximum 7 users)

Compared to traditional chat applications, Peer2Chat is a bit limited as you can only chat with people within your digital sphere (up to 50 meters). However, this application was developed just for this purpose. This limitation can be removed if a scatternet (a network consisting of several linked PANs) is established. The most useful usage of Peer2Chat is for communicating in areas where you are not allowed to talk.

PeerQuiz – A Quiz Game

PeerQuiz is a quiz-game for mobile phones for co-located players. 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 the given alternatives, and a winner will be declared based on most correct answers. This application requires that one user acts as a master and sets up the game before other players can join in. The master will decide the number of questions to be played and all communication in the game goes through the master device. Figure 4 shows screenshots from setting up a PeerQuiz game.



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 4 Screenshots from the PeerQuiz application

A characterisation of PeerQuiz is given in Table 4. The PeerQuiz application is master-based, where the device of the initiator of the game will be the coordinator of the communication and control flow. The communication between the devices must be carried out in real time, to give the fast user response required for games. As this is a quiz game, the real time requirements are not so high that the lag and limited bandwidth in Bluetooth introduce any problem.

Table 4 Characterisation of PeerQuiz

ID	Evaluation criteria/Characteristics	Result
1	Number of users typically involved	2-7 users
2	Classification according to the place	Same place
3	Classification according to time	Real time
4	Classification according to planning	Impromptu collaboration
5	Classification according to user interaction	Controlled
6	Classification according to collaboration pattern	Master controlled
7	Classification according to how collaboration is improved	Improve communication
8	The degree of usefulness of application	Low
9	The degree of replacing manual collaboration	Low
10	The degree of replacing existing collaboration support	Low
11	Limitations in the application due to Bluetooth	None
12	Limitations in the application due to Java ME	None
13	Limitations in the application due to the mobile phone	None
14	Limitations in the application due to Peer2ME	None

Peer2Schedule – A Face-to-Face Meeting Scheduler

Peer2Schedule is an application made for making planning of meeting easier. The person that initiates the meeting planning will input a time frame for when the meeting should be. This mobile device will connect to all nearby mobile devices and check their calendar entries for availability. The initiator can then choose an open time spot for next meeting and the calendar of all the mobile devices involved will be updated with a new calendar entry with all necessary meeting information. This application is very useful at meetings to find the time for next meeting. To make this meeting planning work, all mobile devices must run the Peer2Schedule application. The main goal of this application is to provide workflow automation. Figure 5 shows the process of scheduling a meeting using Peer2Schedule.

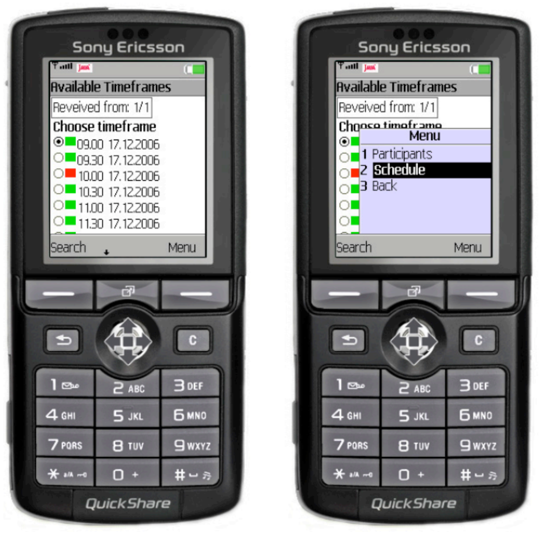


Figure 5 Screenshots from the Peer2Schedule application

The characterisation of Peer2Schedule is given in Table 5. The Peer2Schedule application is a good example of an application where the user interaction is controlled by the application through one master. The device of the initiator will be the master, and the workflow is controlled through the master peer. Peer2Schedule can also be classified as an application to support formal collaboration, as it is a mobile workflow application. The initiator will also have a different role than the other users, as he needs to configure the meeting schedule before the process can start.

Table 5 Characterisation of Peer2Schedule

ID	Evaluation criteria/Characteristics	Result
1	Number of users typically involved	2-7 users
2	Classification according to the place	Same place
3	Classification according to time	Asynchronous
4	Classification according to planning	Formal collaboration
5	Classification according to user interaction	Controlled
6	Classification according to collaboration pattern	Master controlled
7	Classification according to how collaboration is improved	Improve coordination
8	The degree of usefulness of application	High
9	The degree of replacing manual collaboration	High
10	The degree of replacing existing collaboration support	High
11	Limitations in the application due to Bluetooth	Severe limitations
12	Limitations in the application due to Java ME	Some limitations
13	Limitations in the application due to the mobile phone	Some limitations
14	Limitations in the application due to Peer2ME	None

Although this application is a very useful collaborative tool, it is limited by restrictions introduced by Bluetooth and how current mobile phone run Java ME applications. The Bluetooth technology reduces the usability of the application by long time to establish connection between all devices and the search for open times slots cannot be performed automatically without user intervention, as all users involved must accept a security prompt before the search can be performed.

Peer2BrickBlock – A Peer-to-Peer Real-time Game

The Peer2BrickBlock game is a mobile real-time peer-to-peer game where to goal is for the player to push other player into traps. Every player controls a brick, which can be moved around in a 2D playfield. The playfield is an open area where you have a trap and several power-ups to increase the size of your brick, make your brick move faster and make your brick stronger (easier to push other bricks around). The trap and the power-ups are randomly placed on the screen. When a player has been pushed into a trap, he will loose one life and the brick will re-spawn after some seconds on the playfield. All the user's screens should reflect all players movements in real-time.

Table 6 shows the characterisation of Peer2BrickBlock. Due to the small screen on mobile phones, this game is best suited for few players (four or less). This is a real time game where it is important that the game events are distributed without any long delays to all players. The user interaction is user-driven, meaning that the network traffic between the devices depends on how the users interact. This is also a pure peer-to-peer application where no player is the master, e.g.

any player can change the game preferences and the game starts when one of the players pushes the start button.

Table 6 Characterisation of the Peer2BrickBlock applications

ID	Evaluation criteria/Characteristics	Result
1	Number of users typically involved	2-4
2	Classification according to the place	Same place
3	Classification according to time	Real time
4	Classification according to planning	Impromptu collaboration
5	Classification according to user interaction	User interaction
6	Classification according to collaboration pattern	True peer-to-peer
7	Classification according to how collaboration is improved	Initiate collaboration
8	The degree of usefulness of application	Low
9	The degree of replacing manual collaboration	Low
10	The degree of replacing existing collaboration support	Low
11	Limitations in the application due to Bluetooth	Severe limitations (slow discovery)
12	Limitations in the application due to Java ME	None
13	Limitations in the application due to the mobile phone	Some limitations (screen)
14	Limitations in the application due to Peer2ME	Severe limitations (slow connection)

The Peer2BrickBlock application caused us some major headache, and we discovered major limitations in our Peer2ME framework and in Bluetooth. The main problem was that in Peer2ME Bluetooth connections are established when needed between the devices. For a real time game this takes too much time, making the network lag ruining the game play. Also we found that the performance of the OBEX protocol in Bluetooth that Peer2ME uses is not sufficient for real time updates with minimum lag.

PeerCardExchange – An Automatic Business Card Exchange Application

The PeerCardExchange is an application used to automatically exchange digital business card stored on a mobile device with people with the same interests. The user must first enter information like name, contact information, company, position, picture, URLs, etc to complete his own digital business card. The next step is to enter the domains his is working in using a pre-defined ontology mapping the existing domains, e.g. computer graphics, mobile computing, software engineering, etc. The ontology is hierarchically defining high-level domains at the top and more specific domains further down in the tree structure. The final step is to enter the domains of the persons he wants to receive business cards from. After this initialisation process has been completed, the users can let the mobile device search for other mobile devices running the same service. If a match of domain is found, digital business cards are exchanged between the mobile devices. This application is useful for instance at conferences with many people where the mobile device will collect business cards from people with the same interests automatically on behalf of the users. After the user has initiated the application, he can just walk around to automatically collect business cards without any user intervention. However, this application requires that most people in the same area run the same application to be useful (must have a critical mass). The application can also be used for non-professional services like dating by using other domain models.

Table 7 shows the characterisation of PeerCardExchange. This is an application where a lot of users can be involved in a big area. However, the user interaction is mostly sequential in that two mobile devices check for matching domains and then continues for a new search for another device. This application is after initialisation automatic and requires no user interaction. The user can simply look at the result (collected business cards) after walking around in an area with other users for a time. The usefulness of this application is limited both by the slow discovery time in Bluetooth (20+ seconds for Bluetooth 1.x and 10+ seconds for Bluetooth 2.x) and that few mobile phones allow Java ME applications run as background process.

Table 7 Characterisation of the PeerCardExchange application

ID	Evaluation criteria/Characteristics	Result
1	Number of users typically involved	2-100
2	Classification according to the place	Same place
3	Classification according to time	Asynchronous
4	Classification according to planning	Impromptu collaboration
5	Classification according to user interaction	Automatic
6	Classification according to collaboration pattern	True peer-to-peer
7	Classification according to how collaboration is improved	Improve exchange
8	The degree of usefulness of application	High
9	The degree of replacing manual collaboration	High
10	The degree of replacing existing collaboration support	High
11	Limitations in the application due to Bluetooth	Severe limitations
12	Limitations in the application due to Java ME	Some limitations
13	Limitations in the application due to the mobile phone	None
14	Limitations in the application due to Peer2ME	None

Peer2FindPerson – A Find-the-Right-Person Application

This application is very similar to PeerCardExchange, but it will initiate direct contact between two users. The initialisation process is the same as for PeerCardExchange, where the user must enter his own domain and enter the domain of the person he is interested in meeting. After the initialisation process has been completed, the mobile device will search all nearby mobile devices for a match in domains. If a match is found, the mobile device will notify both users by vibrating or making a sound and showing the picture of the matching person found. Both persons can then find each other and start talking. As with the PeerCardExchange, this application is useful where many people that do not know each other are meeting e.g. at a conference. The application makes it easier to find people interested in the same topics and that would have mutual benefits of collaborating. This application can also be used for searching for persons with specific skills to solve a problem. A picture of the Peer2FindPerson in use is shown in Figure 6.

Figure 6 A picture of two users that have found that they have matching interests

Table 8 shows the characterisation of Peer2FindPerson. Although this application resembles PeerCardExchange it has some noticeable differences. Peer2FindPerson is intended to initialise collaboration between two persons and the involved users are notified in real time if a match is found. Also the user interaction in Peer2FindPerson must be regarded as automatic triggered as the application searches for a match and notifies the users if a match is found.

Table 8 Characterisation of the Peer2FindPerson application

ID	Evaluation criteria/Characteristics	Result
1	Number of users typically involved	2
2	Classification according to the place	Same place
3	Classification according to time	Real time
4	Classification according to planning	Impromptu collaboration
5	Classification according to user interaction	Automatic triggered
6	Classification according to collaboration pattern	True peer-to-peer
7	Classification according to how collaboration is improved	Initiate collaboration
8	The degree of usefulness of application	High
9	The degree of replacing manual collaboration	High
10	The degree of replacing existing collaboration support	High
11	Limitations in the application due to Bluetooth	Severe limitations
12	Limitations in the application due to Java ME	Some limitations
13	Limitations in the application due to the mobile phone	None
14	Limitations in the application due to Peer2ME	None

As with the PeerCardExchange application, Peer2FindPerson suffers from the same problems with long discovery time in Bluetooth and that JaveME applications normally cannot be run in background.

Evaluation Summary

In the previous sections we have presented seven different mobile collaborative applications with different characteristics. The most noticeable differences is in how collaboration is improved through applications covering functionality to initiate collaboration, improve coordination, improve exchange and improve communication. The only area missing identified in evaluation framework is improvement of negotiation that could typically be an application to negotiate about desired resources on behalf of the users. The described Peer2Me applications had variations in being asynchronous or real time, impromptu collaboration or formal, and the full range of variation in how user interaction was managed by the application. Most applications typically involved between 2 and 7 users, but some involved fewer due to the limited screens on mobile phones and some more due to the fact that interaction between lots of devices are handled sequentially. None of the applications supported collaboration at different place although this could be provided if e.g. Peer2Chat supported scatternet (not currently supported in Bluetooth). Some applications were limited by the underlying technology and suffered from the slow device discovery process in Bluetooth, the low bandwidth of Bluetooth, the lack of support for running Java ME applications in background, the usage of the OBEX protocol in Peer2ME, and how security is handled in Bluetooth.

The choice of technology of using Java ME and Bluetooth to implement Peer2Me was both a blessing and a curse. The main benefit is that the Peer2Me framework can run on most mobile phones, PDAs and mobile computers as Java ME and Bluetooth are supported on most mobile devices. For peer2peer applications that do not require short discovery and connection time between the devices, the Peer2Me works fine. This problem has been minimised with the Bluetooth 2.x standard where the discovery time has been reduced from about 20 seconds to 10 seconds. The problems of running Java ME applications in background and the security prompt issues in Bluetooth are device-dependent and are solved for most new mobile devices. These issues would not have been any problems if Peer2Me has been implemented as native code. However, this would have limited the usefulness of the framework. To use WiFi instead of Bluetooth as a wireless network technology will solve problems related to slow discovery and low bandwidth, but this will also limit the framework to be supported by fewer devices.

The roster of Peer2Me applications presented in this chapter consists of rather simple applications. More complicated applications can be implemented by combining the functionality of two or more existing Peer2Me applications. One example would be to combine the Peer2Schedule and Peer2Share applications to implement a more complete meeting application that supports both planning of meetings and events as well as sharing documents. Another example would be to combine the Peer2Chat and Peer2Quiz applications to implement a more versatile application to kill time in waiting areas. The final example is to combine the Peer2Share, PeerCardExchange and Peer2FindPerson applications into one making it up to the user if he just wants to exchange business cards, to get a notification when a matching person is close by and give the opportunity to exchange documents with this person. The main challenge in implementing more advanced peer-to-peer applications is that some mobile devices have problems with large Java ME applications in terms of footprint or memory usage.

Future Trends

As seen from the conclusion of this book chapter, mobile network technologies like Bluetooth and Java ME still have some shortcomings to prevent a real break-through for mobile peer-to-peer applications. However, as the network technologies improve, new opportunities arise and new applications appear. Today, there are PAN-technologies that do not suffer from the same problems you have to deal with in Bluetooth. The main problem is that most mobile devices do not support these technologies. There is a need to define standard technologies and frameworks for mobile peer-to-peer enabling all kinds of applications and devices to collaborate using pre-defined well-proven protocols and architectures.

The mobile technology changes rapidly and it is hard to predict the future. The two most interesting recent mobile implementation platforms are iPhone SDK from Apple and Android from Google. None of these platforms have built-in support for peer-to-peer applications, but there are some on-going projects to develop peer-to-peer frameworks for these platforms. However, none of the currently on-going projects provide support for co-located peer-to-peer (directly between the devices). The introduction of sensors and activators brings new opportunities for mobile peer-to-peer applications. Up till now, the introduction of sensors in mobile environment has not been utilized due to the size and power consumption of existing sensors. However, the sensors in the future can be seen as smart dust or brilliant rocks where the sensors are so small that they can be integrated into any material and any device (Satyanarayanan, 2003). This makes it possible for mobile applications to sense location, proximity, temperature, pressure, etc., which can be used to give the application the required input to give only the most relevant services to the users. To make such context-aware systems, it is required to have an architecture that can handle sensor networks, and retrieve and manage the sensor information in a sufficient manner (Anagnostopoulos, 2007). A combination of sensor technology, a strong integration with web-services like provided in the iPhone SDK and Android, and support for co-located peer-to-peer data transfers will open to a new range of applications that can enrich, support, simplify and automate human-to-human collaboration.

CONCLUSION

In this book chapter we have presented the Peer2ME framework for developing mobile peer-to-peer application to support collaboration. Further, we have presented several Peer2ME applications and classified them according to an evaluation framework for such applications. The evaluation framework presented is useful for characterising and evaluating mobile peer-to-peer applications as well as it can be used as a checklist when developing new mobile peer-to-peer applications. We have also discovered some limitations in Bluetooth, Java ME and Peer2ME that will limit the usability of such applications.

Acknowledgements

We would like to thank Thomas Fossum, Lars Kirkhus, Anders R. Sveen, Michael Sars Norum, Carl-Henrik Wolf Lund, Steinar Hestnes, Torbjørn Vatn, Tommy Bjørnsgård, Kim Saxlund, Martin Jarrett, Eivind Sorteberg, and Hassan Syed Shah for their invaluable effort in the Peer2Me project. This work has been sponsored by the Leiv Eriksson mobility program offered by the Research Council of Norway.

REFERENCES

Anagnostopoulos, C. B., Tsounis, A., and Hadjiefthymiades, S. (2007), *Context Awareness in Mobile Computing Environments*, Wireless Personal Communications, 42(3), August, pages 445-464.

Apple Computer (2008), *iPhone Dev Center*, web: http://developer.apple.com/iphone/, visited Sept. 9th 2008.

Ashok, R. L, and Agrawal, D. P. (2003), *Next-Generation Wearable Networks*, Computer 36(11): 31-39.

Auvinen, A., Vapa, M., Weber, M., Kotilainen, N., and Vuori, J. (2006), *Chedar: peer-to-peer middleware*, Parallel and Distributed Processing Symposium, pp. 7, 25-29 April.

Bisignano, M., Di Modica, G., and Tomarchio, O. (2005), *JMobiPeer: A Middleware for Mobile Peer-to-Peer Computing in MANETs*, In Proceedings of the First international Workshop on Mobility in Peer-To-Peer Systems (MPPS), June 6 – 10.

Broll, W., Lindt, I., Herbst, I., Ohlenburg, J., Braun, A-K., and Wetzel, R.(2008), *Toward Next-Gen Mobile AR Games*, IEEE Computer Graphics and Applications 28(4): 40-48.

Clarence A. Ellis, Simon J. Gibbs, and Gail Rein (1991), *Groupware: some issues and experiences*, Communications of the ACM, 34(1):39–58.

Davids, N. (1996a), *Personal digital assistants: Part 1*, IEEE Computer 29(9): 96-99.

Davids, N. (1996b), *Personal digital assistants: Part 2*, IEEE Computer 29(11): 100-104.

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

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

Geer, D. (2005), *Users Make a Beeline for ZigBee Technology*, Computer 38(12): 16-19.

Google (2008), *Android – An Open Handset Alliance Project*, web: http://code.google.com/android/, visited Sept. 18th 2008.

Harjula, E., Ylianttila, M., Ala-Kurikka, J., Riekki, J., and Sauvola, J. (2004), *Plug-and-play application platform: towards mobile peer-to-peer*, In Proc. of the 3rd international conference on Mobile and ubiquitous multimedia: 63-69.

Helal, S. (2002), *Pervasive Java*, IEEE Pervasive Computing 1(1), pages 82-85.

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

Horozov, T., Grama, A., Vasudevan, V., and Landis, S. (2002), *MOBY-a mobile peer-to-peer service and data network*, Parallel Processing, pp. 437-444.

Husemann, D. (1999), *The Smart Card: Don't Leave Home Without It*, IEEE Concurrency 7(2): 24-27.

Jing, J., Huff, K., Sinha, H., Hurwitz, B., and Robinson, B. (1999), *Workflow and Application Adaptations in Mobile Environments*, Second IEEE Workshop on Mobile Computer Systems and Applications: 62-69.

Kapp, S. (2002), *802.11: Leaving the Wire Behind*, IEEE Internet Computing 6(1): 82-85.

Kawulok, L., Zielinski, K., and Jaeschke, M. (2005), *Trusted group membership service for JXME (JXTA4J2ME)*, Wireless And Mobile Computing, Networking And Communications, 2005. (WiMob'2005), pp. 116-121 Vol. 4, 22-24 Aug.

Kortuem, G. (2002), *Proem: a middleware platform for mobile peer-to-peer computing*, ACM SIGMOBILE Mobile Computing and Communications Review, 6(4), pages 62-64.

Kortuem, G., Schneider, J. , Thaddeus, D. P. , Thompson, G. C. , Fickas, S., and Segall Z. (2001), *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, Sweden, 27-29 August.

Leavitt, N. (2007), *For Wireless USB, the Future Starts Now*, Computer 40(7): 14-16.

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

Macedonia, M. (2007), *iPhones Target the Tech Elite*, Computer 40(6): 94-95.

Maibaum, N. and Mundt, T. (2002), *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.

Michahelles, F., Thiesse, F., Schmidt, A., and Williams J. R. (2007), *Pervasive RFID and Near Field Communication Technology*, IEEE Pervasive Computing 6(3): 94-96.

Miller , B. A. and Bisdikian, C. (2004), *Bluetooth Revealed*, Addison-Wesley, 2 edition.

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

Olson, J.S., Teasley, S. , Covi, L., and Olson, G. (2002), *The (currently) unique advantages of collocated work*, MIT Press.

Papadopoulos, C (2006), *Improving Awareness in Mobile CSCW*, IEEE Transactions on Mobile Computing, vol. 5, no. 10, pp. 1331-1346, Oct.

Reynolds, F. (2008), *Whither Bluetooth?*, IEEE Pervasive Computing 7(3):6-8.

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

Satyanarayanan, M. (2003), *From the Editor in Chief: Of Smart Dust and Brilliant Rocks*, IEEE Pervasive Computing, vol. 02, no. 4, pp. 2-4, Oct-Dec.

Schollmeier, R. (2001), *A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications.* In Proceedings of the First international Conference on Peer-To-Peer Computing (P2P'01), August 27 - 29, 2001). P2P. IEEE Computer Society, Washington, DC, 101.

Schreiner, K. (2007), *Where We At? Mobile Phones Bring GPS to the Masses*, IEEE Computer Graphics and Applications 27(3): 6-11.

Vaughan-Nichols, S. J. (2003), *OSs Battle in the Smart-Phone Market*, Computer 36(6): 10-12.

Wang, A. I. , Bjørnsgård, T. , and Saxlund , K. (2007), *Peer2Me - Rapid Application Framework for Mobile Peer-to-Peer Applications*, In The 2007 International Symposium on Collaborative Technologies and Systems (CTS 2007), page 10, Orlando, Florida, USA, May 21-25.

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

Wang, A. I. , Sørensen, C.-F. , and Fossum, T. (2005), *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.

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

Zhang, H., Udagawa, T., Arita, T., Tsuji, J., Okada, K., Sasase, I. and Nakagawa, M., (2001) *Wireless 1394: a new standard for integrated wireless broadband home networking*, Vehicular Technology Conference (VTC 2001), IEEE VTS 53(2): 1124-1128.

Zheng, P. and Ni, L. M. (2006), *Spotlight: The Rise of the Smart Phone*, IEEE Distributed Systems Online, 7(3), pages 13, 2006.