Peer2Schedule - An Experimental Peer-to-peer Application to Support Present Collaboration

Alf Inge Wang and Peter Nicolai Motzfeldt
Dept. of Computer and Information Science
Norwegian University of Science and Technology
N-7491 Trondheim, Norway
alfw@idi.ntnu.no, peternic@idi.ntnu.no

***Abstract*—This paper describes experiences from implementing an experimental mobile peer-to-peer application called Peer2Schedule aimed at improving and supporting collaboration where people are collocated. Peer2Schedule was built on top of the Peer2Me framework that provides management of mobile ad hoc networks over Bluetooth. The goal of the Peer2Schedule project can be divided into three main areas. 1) To develop a collaborative peer-to-peer application to evaluate the usefulness of such applications; 2) to evaluate the technical limitations of J2ME and Bluetooth in this domain; and 3) to implement and evaluate a mobile application to improve present ad-hoc collaboration. In this paper we also investigates the social and network issues related to such applications. Our findings indicates that Bluetooth and J2ME are useful technology to implement such applications, but the long time to establish connections and security issues in Bluetooth reduce the usability of such applications. In addition, it is essential that collaborative peer-to-peer applications must be developed as open source projects, to allow the users of such applications to evaluate the source code to assess how the application handles privacy. Finally, the paper elaborates on how the problem domain affects how the control-flow is managed in peer-to-peer applications.**

Keywords: Collaborative applications, Mobile ad hoc networks, Peer-to-peer networks, Privacy.

I. Introduction

The peer-to-peer (P2P) architecture pattern has become more and more used in distributed computing. The centralisation of services and information in the client-server approach often becomes a problem in terms of performance bottleneck and reduced availability because of single-point of failure. The P2P architecture removes this problem by allowing all involved computer act as equals in the network [23]. This makes it possible to balance the network load and provide a more fail-proof network by using alternative routing if a network connection between two nodes fails. It was programs for sharing and exchanging mp3-files like Napster that really showed the potential of P2P computing [6]. Since Napster, P2P computing has become mainstream, and P2P applications for chatting and file sharing are now included into operating systems like Microsoft Windows Vista and Mac OS X Tiger.

Currently, most P2P 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 migrate the P2P computing to a wireless environment [16], [19]. By bringing P2P 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, disconnections, etc.), mobility (find closest server, network handover, where to store data etc.) and portability (limited CPU, memory, battery, displays, keyboards, etc.) [22]. Mobile P2P computing also offers new opportunities that can be utilised like providing location-based services [7], [18] and social computing [8], [11] using short-range networks.

The P2P architecture pattern fits very well to implement collaborative applications, as the nodes in the network represents the users, and the network between the nodes represent the collaborative links between the users. Mobile ad hoc networks (MANETs) with a short radius like irDA, Bluetooth and WiFi enable a new kind of collaborative applications to support users that are physically collocated. The mobile devices can provide the users with services that will ease or improve the collaboration for users present at the same location.

The research within this area can be characterised as mobile computer supported co-operative work (mobile CSCW) [27] where every users is surrounded by a digital sphere that represents the users' personal area networks (PANs). The intersection between these spheres is the basis for mobile collaboration through exchange of data between applications running on the mobile devices involved. In such environments, the support for mobile P2P is essential. Further it requires support for and establishment of MANETs. A MANET is in this context defined as a self-configuring network where peers can join and leave the network dynamically making the wireless network topology unstable and unpredictable [20]. MANETs provide an opportunity for a new kind of applications that can cope with users that come and go. Also, it imposes new challenges to make the applications useful and user-friendly.

MANETs enable mobile users to interact in new ways. The interaction between users can be explicitly initiated by the users, it can be automatically initiated by the mobile devices, or a hybrid of these two [25]. Such applications can be used to initialise collaboration between users of same interest, e.g., an application to find people with same research interest at

a conference [26]. Further, MANETs can be used to create applications for proximity chats and file exchanges, or simply for entertainment like games.

In this paper we describe the Peer2Schedule application that was created to support mobile collaboration and to demonstrate a practical usage of mobile CSCW. The application is used for planning the next meeting by searching the electronic calendars of the mobile devices of all participants for open time-slots. Further, the paper describes an evaluation of the application with emphasis on usability, social issues, and network challenges.

The rest of the paper is organised as follows. Section II briefly describes the Peer2Me framework, which Peer2Schedule is built on. Section III describes some background of the Peer2Schedule project, social issues, and the architecture of the Peer2Schedule application. Section IV presents an evaluation of the Peer2Schedule application related to usability and network issues. Section V relates our approach to similar applications and frameworks. Finally, Section VI concludes the paper.

II. The Peer2Me Framework

Peer2Schedule uses the Peer2Me framework to manage P2P network issues, like establishing connections, detecting nodes, and routing. Peer2Me is a framework aimed at simplifying development of mobile P2P applications [24]. It enables developers to focus on the logic of the actual application and takes care of management data exchange and network configuration in the P2P network. The framework consists of these main parts:

- **Node:** A node is a peer representing a mobile device running the framework.
- **Network:** The network is an abstract representation of the actual network used, enabling to use the same API independent on the underlying network technology.
- **Group:** A collection of connected nodes.
- **Service:** A service is an identifier that applications use to recognise that they run and provide the same service.
- **Message:** The container for all data communication between nodes that can contain text, Java-objects, Java datatypes or binary files.
- **Session:** A session keeps track of know peers, groups, available network technologies etc.
- **Application:** A J2ME midlet (application) running the Peer2Me framework.

Figure 1 shows the architecture of the Peer2Me framework. Figure 1 a) illustrates how the framework fits into J2ME architecture, and Figure 1 b) describes how the different parts of the framework is structured.

The Peer2Me framework is implemented in J2ME MIDP 2.0 and uses the Bluetooth API package for J2ME. The framework automatically establishes connections between the mobile devices and provides a high-level API for managing groups of nodes and exchange of data. The framework can also be used to implement mobile agents as the framework support exchange of serialised Java-objects.

III. The Peer2Schedule Application

The Peer2Schedule application was implemented to provide a solution to the recurring problem of planning the next meeting. The process of planning the next meeting usually consists of many people browsing through their calendars (electronic or paper) to find an open time-slot. Peer2Schedule was implemented to support this planning process by utilising the electronic calendar in mobile devices and the PANs of such devices. As the mobile phone is the most used mobile device in Europe (and other continents) and most of mobile phones used for professional purposes provide advanced electronic calendars and Bluetooth networks, J2ME was chosen as the target execution platform. By choosing J2ME and Bluetooth, this application could run on most mobile devices including PDAs. In addition to J2ME and the Peer2Me framework, Peer2Schedule requires two additional J2ME packages to work:

- **JSR82:** The JSR82 package provides an API to Bluetooth in J2ME. Most mobile phones since 2005 have support for JSR82. The Peer2Me framework also requires JSR82 to run.
- **JSR75:** The JSR75 is a package for J2ME to gain access to Personal Information Management (PIM) information like calendar, address book and files stored on flash memory cards on the mobile device. This package ensures a controlled access to such information and is required to overcome the restrictions of J2ME sandbox limitations.

A. Approach for Peer-to-peer Calendar Scheduling

There are two main network topologies that can be used to manage control-flow in P2P applications: *pure P2P* and *master-controlled P2P*. If the pure P2P approach is used, the peers in the network will automatically discover each other and any of the peers can initiate what actions to take. For some application areas like file sharing, this approach works very well. This topology also works well in cases where the application automatically initiates actions on behalf of the user like automatic exchange of electronic business cards for users within network range with a matching professional profile or a dating applications searching for nearby matching personalities that alert the user if one is found.

However, for applications where one user will initiate the actions to take, a master-controlled P2P approach usually works better. In this approach, one of the peers will be the master in the network, managing the control flow of the application for all involved peers. For the application described in this paper, it is clearly that one user initiates the actions to be executed when he or she tries to find an available timeslot for the next meeting. If the master-controlled approach is used, this will significantly reduce the complexity of the scheduling algorithm required, and it is much more likely to find a free timeslot in less time than using the pure P2P approach. Thus, the master-controlled approach reflects situations where one person is in charge of managing the situation e.g., in a meeting (a chairman or a project manager). It would be possible to

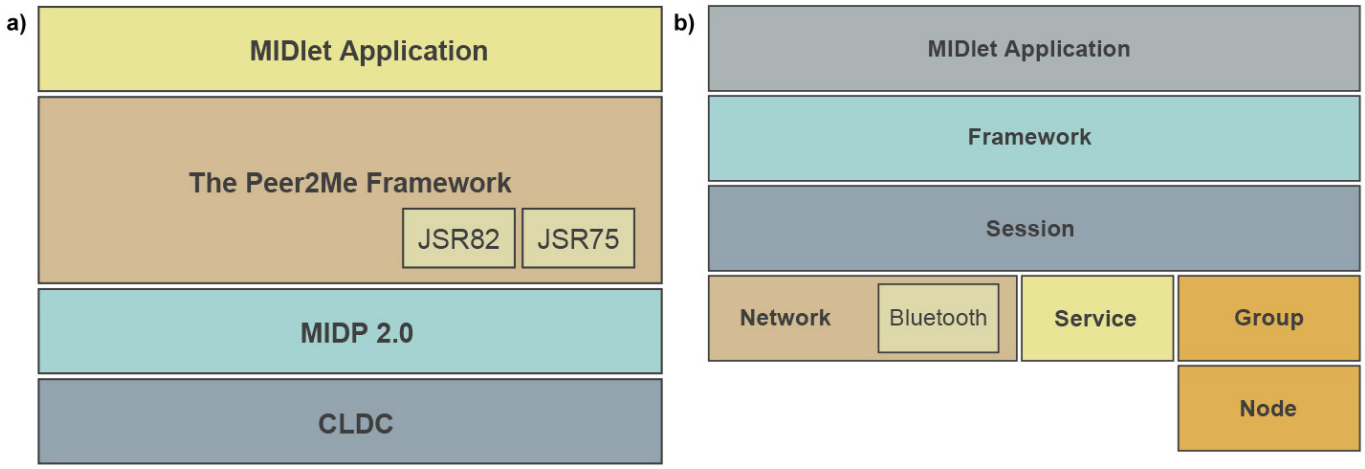

Fig. 1. The Peer2Me Framework architecture

plan next meeting where everyone could propose the time, the result is collected, and several iterations have to be made until a timeslot where all participants can meet can be found. However, it is likely that this will be very time-consuming approach. By using the master-controlled approach, one person can propose a timeframe for when next meeting should be and then collect all the calendar information from the participants to find if there are any open timeslots.

B. Social challenges

Wang et. al have recognised three main social challenges for mobile P2P collaborative applications: *Usage, update* and *thrust*. *First*, the application must have minimum user-base to be useful and all possible users must have the application installed on their mobile device. This also means that it is critical that such applications are based on technology that can run on most mobile devices. By choosing J2ME and Bluetooth, most mobile devices will be the target platform as these technologies are supported by most mobile phones, smartphones, and PDAs. *Second*, it is critical that the information used by the application is updated regularly so that this information is always kept up-to-date. In our case, it is important that the calendar information on the mobile devices reflects the actual calendar of the user including all professional and non-professional activities. *Third*, applications used for collaborative purposes in ad-hoc P2P network must be trusted by the users. The trust can be decomposed into two main parts: technical and privacy trust. Technical trust means that the user must trust that the application will not harm the mobile device in any way, e.g. by introducing viruses or security hazards. The privacy trust means that the user must trust that applications will not misuse or store private information. This issue is particularly relevant for the Peer2Schedule application, as the application requires checking all the users' calendars to find an available timeslot for a meeting. The only way for the user *to be sure* that personal information is not misused, is for the user to check the source code of the application before the application is installed on the mobile device. This implicates that such applications should be developed as open source projects, enabling the users to get insight into how the program works. However, for most users this is not possible because of lack of technical insight (skills in programming and J2ME in particular). But by making such applications open source, it will give people with the sufficient technical skills other than the developers the opportunity to check for privacy violations. If the developers distribute the application over the web, they should provide both the executable Jar-file as well as all the source files in the distribution. In addition, the webpage for the application should provide a user feedback section like a Wiki or a discussion forum, where technical users that have evaluated the source code of the application according to trust violations can report their results. This also opens for giving other useful feedback to the developer to improve future versions of the application.

In our application, only the available timeslots are exchanged between the peers. Each peer will only check its own calendar for available timeslots for a given timeframe specified by the initiator node in the network. In this way, no private information about specific calendar entries will leave the mobile device.

C. Architecture and Design

The architecture of the Peer2Schedule application is shown in Figure 2. The architecture is based on the model-view-controller (MVC) architecture pattern [17] providing flexibility to change the graphical user interface (GUI) without changing the whole application.

The *Controller package* is the control centre of the architecture, where the Peer2Schedule class is the main controller of the application handling events and initiates actions to be taken. Figure 2 shows some of the main methods used in the Peer2Schedule class.

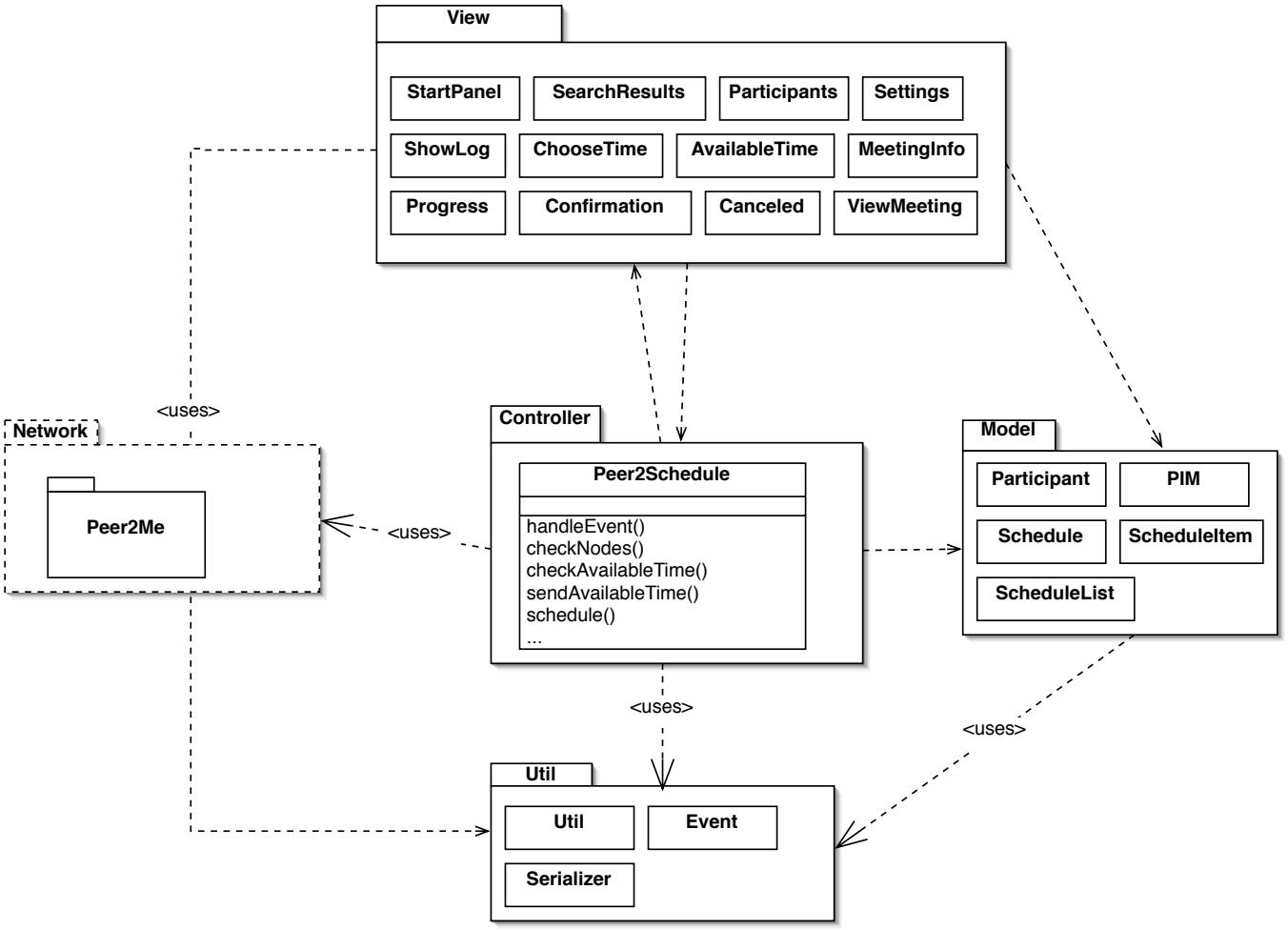

Fig. 2. The Architecture of the Peer2Schedule Application

The *View package* consists of twelve classes representing all the GUI panels used in the application. These classes extend the J2ME GUI class Screen (subclass of Displayable). These GUI panels use the standard GUI items provided in J2ME like Forms, TextFields, DateFields, Gauges, StringItems, and ImageItems.

The *Model package* contains of classes managing the business logic and representing the problem domain of the application: *Participant* manages up-to-date information about the participant in the meeting planning process, *PIM* is responsible for accessing and managing the calendar information, *ScheduleItem* contains a timeframe and a Boolean attribute stating if the timeframe is available or not, *ScheduleList* holds a list of ScheduleItems and keeps information about several timeslots, and *Schedule* manages all the ScheduleList from all participants and calculates the available timeslots when participants are available.

The *Network package* is responsible for managing all P2P communication between the mobile devices, including establishing connections, managing new and lost nodes, managing grouping and setting up sessions, and managing all message routing and the actual data exchange. All the network issues are being taken care of by the Peer2Me framework that is initiated from the Peer2Schedule class.

The *Util package* is a helper and support package to provide services that are used by classes in other packages. This package consists of the *Util class* which holds a set of static methods the other classes can access, the *Event class* which is a helper class for the MVC architecture pattern managing incoming events from the view which is sent to the handleEvent method in the Peer2Schedule class, and the *Serializer class* used to serialise objects before they are sent between nodes. Serialisation of objects is not natively supported in J2ME.

Figure 3 shows the steps taken to schedule a meeting in Peer2Schedule. When the user starts the application, he or she has three menu choices: Log, Settings and Start. The *Log*

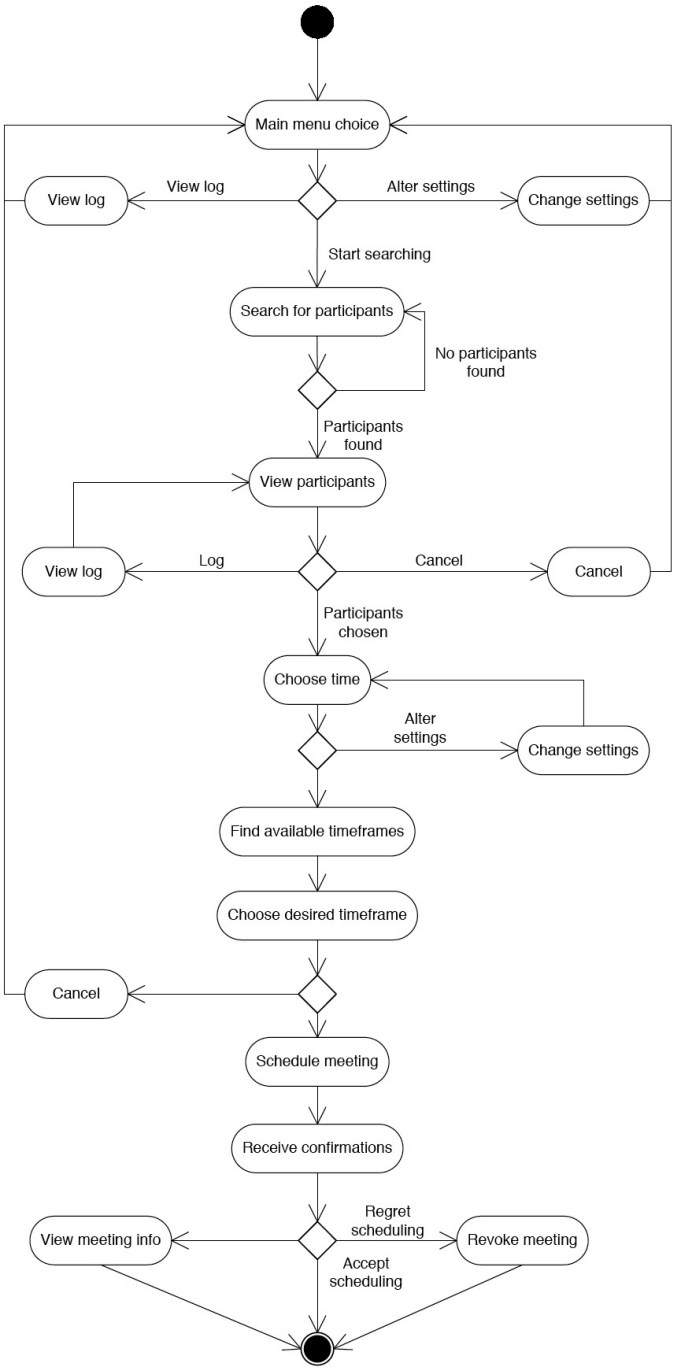

Fig. 3. The Peer2Schedule program flow

selection displays the log of previous attempts of scheduling meetings. This functionality is mainly for testing purposes and is useful in testing that the application is running as it should and that the P2P network data exchange is working as expected. The *Setting* selection is used to set the timeframe for which the application should search within (e.g., from 08:00am to 6:00pm). The *Start* selection starts the scheduling process and begins with a search for other participants. If no participants or not all are found, the initiator can initiate a new search. The participants are listed, and the initiator should then select the ones to be included in the scheduling process. Usually, all participants found will be selected. The only case it could be necessary to not select everyone in the list is if two parties are scheduling meetings using Peer2Schedule at the same time in the same area. Some Bluetooth devices can discover other Bluetooth devices in a within a radius of 50 meters (open air). Note that Peer2Schedule will only list the mobile devices that run the same service in the Peer2Me framework (not all nearby Bluetooth devices). The next step is to choose start- and end-date of the period in which the meeting is supposed to take place. At this point it is also possible to change the Settings for the timeframe. Then the mobile devices of the participants exchange information, and the mobile device of the initiator collects a list of the available timeslots. The next step is for the initiator to choose a timeslot and enter some meeting details like description of the meeting and location where the meeting will be held. This information is then sent to all the participants' mobile devices, and the event is inserted into their electronic calendars if confirmed. The initiator will then receive confirmations from all the participants. Based on these confirmations, the initiator can then either accept the scheduled meeting or revoke it (removes this event from all the participants electronic calendars).

IV. Evaluation of Peer2Schedule

This section presents an evaluation of the Peer2Schedule application.

A. Usability Test of Peer2Schedule

The evaluation of the Peer2Schedule application was performed as a usability-test where we observed and interviewed five users. The goal of this test was to investigate the usability of the application and look for potential problems in actual usage of mobile collaborative applications for collocated users.

The test-session was started with a quick introduction of the application and a description of how it worked. Then the test-panel was asked to plan some meeting appointments using the application. After a half hour of observed testing, the test-panel was asked to fill out a questionnaire consisting of five questions. Here is a summary of the answers of the six questions:

1) What is the degree of usability of the application? (1 not good, 2 acceptable, fairly good, 4 good, 5 very good)
Result:

AVERGAGE	MEDIAN	STDEV	MAX	MIN
3.6	4	0.55	4	3

Spoken or observed comments: All in the test-panel agreed that the application was usable, but some parts could have been better explained. After a couple of runs, all in the test-panel said the application was easy to use. The most noticeable negative effect on the usability was

that all the users had to agree to a Bluetooth security message prompt before any data could be sent from the device.

2) Are there unnecessary features of the application?
   Results: All members of the test-panel pointed out that it was only the Bluetooth security prompt that made the application a bit cumbersome to use and was unnecessary. The panel agreed that the application did not have any confusing features, which matched well with our intention of making the application as simple as possible.
3) What features is missing from the application?
   Results: The test-panel suggested that it would be nice to have a feature to share events directly from the application (this can be done from the native calendar application on the mobile devices). It could be useful to include a warning about upcoming meetings and it would also be nice to support other network technologies than Bluetooth, e.g. Infrared and WiFi. It was also pointed out that the application could list all the appointments that have been added and a list of all revoked meetings. Another feature could be to include a list of all the participants in the meeting entry.
4) What kind of improvements do the application need?
   Results: The improvement that all in the panel mentioned was to make it possible to automate the process by getting rid of Bluetooth security message prompts. In addition, the test-panel suggested to provide more information in the application about the result of the meeting scheduling and to automatically include relevant information in the calendar entries (like a list of the participants).
5) Did you run into any problems? If yes, please describe the problems.
   Results: One of the phones was running in power save mode, which limited the connections to only one node. After disabling the power-save, everything worked properly. Another problem was related to connection and communication issues. The discovery time to establish connection was very long and the bandwidth between the phones was low. Another problem was the detection of new participants when two phones search simultaneously, which resulted in very long discovery times and an error message that data could not be sent to the intended recipient. If two nodes initiate a search simultaneously, they will not discover each other. This is a known issue when using OBEX protocol in Bluetooth; a node cannot accept incoming requests when trying to connect to other nodes.
6) Is this an application you might use?
   Results: 80% of the test-panel said that they might use this application. However, the test-panel pointed out that there were several issues that needed to be solved for it to be used regularly. The issues pointed out were to solve the connection issues (long discovery time and detection of new nodes) and to get rid of the Bluetooth security prompt.

B. Issues of Mobile Collaborative Applications

The development of mobile collaborative applications is challenging because of the complexity of management of MANETs. To be able to create a MANET, the nodes need to establish connection in relatively short time. At the same time, the established connections need to be robust and cope with new and lost nodes. This means that the characteristics of the underlying network technology are critical for a mobile collaborative application to work. By implementing the Peer2Schedule application using the Peer2Me framework, multiple network technologies can be used without changing the application source code. As of today, the only network technology supported by the Peer2Me framework is Bluetooth. Bluetooth is a widely used and a robust network technology, but suffers from long discovery times. The Bluetooth specification defines the discovery time to be around 10.24 seconds, at its best. From our experience the average discovery time is approximately 13 seconds. From a usability perspective, 13 seconds is a long time to wait for an application to respond. A possible solution to this problem could be to hide this problem by performing a discovery while the initiator of the application is entering information about the meeting. This could cause problems in terms of low responsiveness in the application because of the CPU overhead of running several threads on a mobile device. Another limitation of using Bluetooth technology for mobile collaboration is the way it handles security issues. When a connection with a peer is initiated, the user needs to accept this connection. It is possible to set the permission of the application manually on the phone, but we have experienced that the *"Ask Once"* option does not work for J2ME applications. The security requirement to always press *"I accept"* when sending messages is conflicting with the usability of mobile collaborative applications.

C. Proposed Improvements

Based on the feedback from testing the Peer2Schedule application, we have identified some improvements that could improve the usability of the application. One way of improving the usability of Peer2Schedule is to change the discovery of new devices to only be performed once. This means that new mobile devices cannot be detected during the scheduling of meeting. If another participant wants to join, the whole meeting scheduling process can be restarted. A Bluetooth search is a time consuming activity.

Another way of improving the usability of the application is to provide support for WiFi. The use of WiFi would improve the time to establish connections and data transfer bandwidth and make the application much more responsive. However, today only few mobile devices support WiFi networks. By using WiFi, the security issues of Bluetooth would also be easier to solve.

Finally, to solve the problem of simultaneously detection of nodes in Bluetooth, the RFCOMM protocol, which supports simultaneously communication, could be used instead of

OBEX. This change must be made in the Peer2Me framework (not the Peer2Schedule application).

V. Related Work

This section gives an overview of research on mobile P2P applications, and also gives a short introduction to some alternatives P2P frameworks for mobile applications.

JXTA [19], [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 [2] 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.

Proem [15] is another framework for developing and deploying P2P collaborative applications in a MANET 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. Proem requires a Java Standard Edition to run, limiting the devices to run Proem to powerful PDAs.

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 likely due to high requirements on CPU and memory from running the framework.

MOBY [12] provides a network for mobile P2P exchange of services and data. MOBY offers a dynamic service location and client mapping to achieve an adaptive network optimising performance and reliability. MOBY uses heavily JINI functionality and can thus not run in a J2ME environment.

Nützel and Kubek describe a mobile P2P application for distributed recommendation and re-sale of music, which they call Mobile Music Messenger (MMM) [21]. This application was designed as an extension for the PotatoSystem [9] and was implemented for mobile phones in J2ME using JXME to manage the P2P communication. The purpose of this application is to allow users to find other users with similar musical taste to sell own songs or to find new music for themselves. MMM does not support proximity-based communication networks like Bluetooth, as all communication is done over WANs using telecom networks.

In [14], Kirdal et. al describe the MOTION P2P platform for mobile teamwork that addresses the needs to support virtual communities of employees that need to share artefacts. MOTION is implemented on top of PeerWare that supports distributed search, and publish and subscribe of services. This system provides a Java Swing user-interface as well as a traditional web-interface limiting the system to run on powerful PDAs or laptop PCs. MOTION does not support direct P2P communication between the mobile devices using PAN, but relies on WAN-technology.

In [10], Hayes and Wilson describe experiences from implementing a P2P file-sharing application for MANETs. This application uses the Gnutella Protocol to provide decentralised file sharing. The application was implemented to run on laptops and powerful PDAs, and Bluetooth was used for communication between the devices. A performance test shows that the average time to set up a connection between devices was 7.8 seconds. This was notably lower than our experiences, but it is likely the difference is because more powerful devices are used.

Other related work worth mentioning is the project LightPeers, which is a P2P framework on mobile devices [5], Antoniadis and Courcoubetis' work on issues related to the design of mobile social software in a multi-hop P2P environment [1], and Kellerer et. al that outline the requirements and building blocks for a P2P based service platform for mobile environments from a telecom perspective [13].

As seen from this related work session, there is currently little research with focus on mobile P2P applications supporting collaboration for collocated users. This is mainly due to few existing P2P frameworks for MANETs, making it hard to implement such applications.

VI. Conclusion

In this paper we have presented a mobile P2P application to support present collaboration. Such applications utilise MANETs to ease collaborative tasks for collocated people equipped with mobile devices. In our case, we developed an application to making it easier to find an available timeslot for the next meeting. The critical question to ask is then: Is it useful to use a P2P application running on mobile phones along with MANETs to find a suitable time for a meeting? In order to find an answer to this question, we arranged a user-test. We provided a group of people with the application, and let them use it without any help. After they had used Peer2Schedule, they were asked to do the same all over again using only the calendar on the phone. The positive effect one would expect of Peer2Schedule was the limited manpower it used to arrange the meeting. Only one person should be needed to use the mobile phone, as opposed to everyone checking their calendar. However, due to how security for data transfer is implemented in Bluetooth, all users running the Peer2Schedule have to accept the connections to all participants during the scheduling. Due to this problem, the effectiveness of Peer2Schedule was limited, and lowered the usability. The whole idea of the Peer2Schedule application was to automate the scheduling process without too much user intervention. If the security issue in Bluetooth is solved, such collaborative P2P applications will be very useful and automate several collaborative tasks. In addition, we found that it is essential that applications like Peer2Schedule is developed

as open source project, as the source code is the only way to check if privacy issues are handled in a sufficient manner.

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] Panayotis Antoniadis and Costas Courcoubetis. The Case of Multi-hop Peer-to-Peer Implementation of Mobile Social Applications. In *ICSNC '06: Proceedings of the International Conference on Systems and Networks Communication*, page 5, Tahiti, French Polynesia, 2006. IEEE Computer Society.

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

[3] Mario Bisignano, Giuseppe Di Modica, and Orazio 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] Bent Guldbjerg Christensen. Lightpeers: A lightweight mobile p2p platform. In *PERCOM '07: Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 132–136, New York, USA, 2007. IEEE Computer Society.

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

[7] Nigel Davies, Keith Cheverst, Keith Mitchell, and Alon Efrat. Using and determining location in a context-sensitive tour guide. *Computer*, 34(8):35–41, 2001.

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

[9] Fraunhofer IDMT. Website of the PotatoSystem. web: http://www.potatosystem.com, 2007.

[10] Anna Hayes and David Wilson. Peer-to-Peer Information Sharing in a Mobile Ad Hoc Environment. In *WMCSA 2004: Sixth IEEE Workshop on Mobile Computing Systems and Applications*, pages 154–162, English Lake District, UK, 2004. IEEE Computer Society.

[11] Lars Erik Holmquist, Joakim Wigstrom, and Jennica Falk. The Hummingbird: Mobile Support for Group Awareness. In *Demonstration at ACM 1998 Conference on Computer Supported Cooperative Work*, 1998.

[12] Tzvetan Horozov, Ananth Grama, Venu Vasudevan, and Sean Landis. MOBY - A Mobile Peer-to-Peer Service and Data Network. In *2002 International Conference on Parallel Processing (ICPP'02)*, pages 437–444, 2002.

[13] Wolfgang Kellerer, Zoran Despotovic, Maximilian Michel, Quirin Hofstätter, and Stefan Zöls. Towards a Mobile Peer-to-Peer Service Platform. In *SAINTW'07: 2007 International Symposium on Applications and the Internet Workshops*, page 2, Hiroshima, Japan, 2007. IEEE Computer Society.

[14] Engin Kirda, Harald Gall, Pascal Fenkam, and Gerald Reif. MOTION: A Peer-to-Peer Platform for Mobile Teamwork Support. In *COMPSAC02: 26 th Annual International Computer Software and Applications Conference*, pages 1115–1117, Oxford, UK, 2002. IEEE Computer Society.

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

[16] Gerd Kortuem, Jay Schneider, Dustin Preuitt Thaddeus, G. C. Thompson, Stephen Fickas, and Zary 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.

[17] G. Krasner and S. Pope. A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 system. *Journal of Object Oriented Programming*, 1(3):26–49, 1988.

[18] Sue Long, Rob Kooper, Gregory D. Abowd, and Christopher G. Atkeson. Rapid prototyping of mobile context-aware applications: The cyberguide case study. In *Mobile Computing and Networking*, pages 97–107, 1996.

[19] Nico Maibaum and Thomas 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.

[20] Prasant Mohapatra, Chao Gui, and Jian Li. Group communications in mobile ad hoc networks. *Computer*, 37(2):52–59, 2004.

[21] Jürgen Nützel and Mario Kubek. A Mobile Peer-To-Peer Application for Distributed Recommendation and Re-sale of Music. In *AXMEDIS'06: Second International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution*, pages 93–98, Leeds, UK, 2006. IEEE Computer Society.

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

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

[24] Alf Inge Wang, Tommy Bjørnsgård, and Kim Saxlund. Peer2Me - Rapid Application Framework for Mobile Peer-to-Peer Applications. In *CTS 2007: The 2007 International Symposium on Collaborative Technologies and Systems*, Orlando, Florida, USA, 2007. IEEE Press.

[25] Alf Inge Wang, Michael Sars Norum, and Carl-Henrik Wolf 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.

[26] Alf Inge Wang, Carl-Fredrik Sørensen, and Thomas 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.

[27] Mikael 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.